Fortschritt-Berichte VDI

VDI

Reihe 20

Rechnerunterstützte Verfahren

Nr. 475

Daniel Marolt,
Reutlingen

SWARM: A Novel Methodology for Integrated Circuit Layout Automation Based on Principles of Self-organization

Berichte aus dem **Electronics & Drives Lab** der Hochschule Reutlingen · Prof. Dr.-Ing. Jürgen Scheible (Hrsg.)

Hochschule Reutlingen
Reutlingen University

SWARM: A Novel Methodology for Integrated Circuit Layout Automation Based on Principles of Self-organization

Daniel Marolt

Fortschritt-Berichte VDI

Reihe 20

Rechnerunterstützte Verfahren

Dr.-Ing. Daniel Marolt, Reutlingen

Nr. 475

SWARM: A Novel Methodology for Integrated Circuit Layout Automation Based on Principles of Self-organization

Berichte aus dem **Electronics & Drives Lab** der Hochschule Reutlingen · Prof. Dr.-Ing. Jürgen Scheible (Hrsg.)

Hochschule Reutlingen
Reutlingen University

Marolt, Daniel
SWARM: A Novel Methodology for Integrated Circuit Layout Automation Based on Principles of Self-organization
Fortschr.-Ber. VDI Reihe 20 Nr. 475. Düsseldorf: VDI Verlag 2020.
256 Seiten, 112 Bilder, 26 Tabellen.
ISBN 978-3-18-347520-9, ISSN 0178-9473,
€ 85,00/VDI-Mitgliederpreis € 76,50.

Keywords: Integrated Circuits – Analog Layout – Floorplanning – Placement – Routing – Constraints – Electronic Design Automation – Optimization Algorithms – Procedural Generators – Multi-Agent Systems

After more than three decades of electronic design automation, most layouts for analog integrated circuits are still handcrafted in a laborious manual fashion today. This book presents *Self-organized Wiring and Arrangement of Responsive Modules* (SWARM), a novel interdisciplinary methodology addressing the design problem with a decentralized multi-agent system. Its basic approach, similar to the roundup of a sheep herd, is to let autonomous layout modules interact with each other inside a successively tightened layout zone. Considering various principles of self-organization, remarkable overall solutions can result from the individual, local, selfish actions of the modules. Displaying this fascinating phenomenon of *emergence*, examples demonstrate SWARM's suitability for floorplanning purposes and its application to practical place-and-route problems. From an academic point of view, SWARM combines the strengths of procedural generators with the assets of optimization algorithms, thus paving the way for a new automation paradigm called *bottom-up meets top-down*.

Bibliographische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet unter www.dnb.de abrufbar.

Bibliographic information published by the Deutsche Bibliothek
(German National Library)
The Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie (German National Bibliography); detailed bibliographic data is available via Internet at www.dnb.de.

Als Manuskript gedruckt. Printed in Germany.

ISSN 0178-9473

ISBN 978-3-18-347520-9

Preface

It is necessary sometimes to take one
step backward to take two steps forward.
Vladimir Ilyich Ulyanov (Russian revolutionary)

On January 9, 2007, in a keynote address at the Macworld Conference & Expo held in San Francisco, Apple CEO Steve Jobs told his audience that "every once in a while, a revolutionary product comes along that changes everything". With these words, Jobs introduced Apple's first-generation iPhone. And indeed, a revolutionary product it was.

How did Apple achieve that? Instead of trying to merely improve the available concepts of smartphones, the developers set out to –as Jobs put it– reinvent the phone. But before inventing something new by exploring promising ideas, they first took a step back and thoroughly investigated the existing problems. In particular, they focused on the downsides of the traditional smartphones' keyboards, which were cumbersome to use, took valuable space away from the display, had fixed control buttons regardless of the application, and stayed always there whether you needed them or not. Based on this examination, Apple came to the conclusion that they had to provide something entirely new: a triple-layered capacitive multi-touch screen. Above everything else, it was this invention that turned the iPhone into a breakthrough device and unleashed a skyrocketing market segment. By 2019, worldwide smartphone revenues have reached an incredible amount of 404.6 billion U.S. dollars.

Below any modern smartphone's surface, the user's finger gestures are processed by an *integrated circuit* (IC) which translates the fingers' actions into discrete digital data. Now, the global pursuit of *digitalization* suggests that analog technology is antiquated, but the opposite is the case. Any machine's external environment –including the user's finger– is (and will always be) analog. With the rising interconnectedness of the world and the increasing interaction between a device and its surroundings due to a sophistication of analog-digital interfaces (for example, a smartphone nowadays has up to a dozen different types of sensory elements), analog circuitry remains indispensable and even gains more and more importance for the functional diversification of microelectronics and therefore the digital transformation of our time.

Before an IC can be fabricated, the circuit has to be designed and must be turned into a physical layout defining the geometries of the photolithographical masks needed for the fabrication. Both steps can be facilitated via methods and tools from the field of *electronic design automation* (EDA). In fact, modern ICs would –because of their tremendous complexity– be impossible to realize without the aid of EDA, which can thus be considered a key technology for today's unparalleled advancement of semiconductor products and their prominence in every domain of our lives. Unfortunately, in contrast to the digital domain, the circuit design and especially the layout creation of analog circuits still suffers from a low degree of automation because most analog automation approaches keep on struggling to find evident industrial acceptance – despite continuing efforts of improvement. A revolution is required.

The task of analog layout automation is addressed by the dissertation *Layout Automation in Analog IC Design with Formalized and Nonformalized Expert Knowledge* [1], which the book at hand is based on. But reflecting Jobs' attitude, the intent of that thesis is not just to enhance existing automation approaches. Instead, Part I of the dissertation takes a step back and first of all performs a profound examination of the underlying problems. Then, a fundamentally new and highly interdisciplinary automation methodology is conceived, denoted as *Self-organized Wiring and Arrangement of Responsive Modules* (SWARM). Part II of the dissertation conveys several ideas from other disciplines which inspired the conception of SWARM, before describing the new methodology in detail. Finally, Part III covers the first-generation implementation of SWARM and discusses its results.

The latter two parts of the dissertation –which concentrate on the novel approach of the SWARM methodology– are reprinted in the book at hand, thereby encapsulating the practical product of the first part's academic ruminations in tangible form. The book's first three chapters 1 to 3 correspond to the dissertation's chapters 5 to 7 (originally making up Part II). The last three chapters 4 to 6 here match the dissertation chapters 8 to 10 (which constitute the original Part III). Regarding the omitted Part I, the reader is kindly referred to the original dissertation, which is publicly available in electronic form and can be downloaded[1] from OPUS, the publication server of the University of Stuttgart. In the following text, concrete references to the first part of the dissertation have been supplemented with a citation of [1]. Otherwise, the original contents have been adopted without any modifications.

[1] https://dx.doi.org/10.18419/opus-10231

Contents

Chapter 1

Clarification of the Task

When people will not weed their own minds,
they are apt to be overrun by nettles.
Horace Walpole (British politician)

This chapter reprises the considerations of Part I from the original dissertation [1] in a nutshell to capture the essence of the task that is to be treated in this thesis. As was discussed in the previous three chapters, that task exhibits three different tenors: (1) the technical aim to achieve, (2) the scientific challenge behind that aim, and (3) the practical ambition of this work.

1.1 Technical Aim

Chapter 2 of the original dissertation [1] explicated that the problem of *analog layout* is a severe bottleneck in the design flow of *integrated circuits* because automation attempts have persistently failed to satisfy the high quality demands of practical analog ICs or are too narrowly focused on specific applications. In industrial environments, analog layout design is still done in a largely manual fashion today and heavily relies on human expertise to cope with the *More than Moore* character of the analog domain. That invaluable *expert knowledge* of seasoned layout engineers is pivotal for satisfying the diverse host of intricate *design constraints* that need to be taken into account to obtain the desired *functionality*. While some design constraints can be concisely expressed through *formalized* constraint representations, many requirements are simply communicated in a *nonformalized* way due to the difficulty and effort for describing them formally.

Existing automation approaches can be basically divided into *optimization-based* approaches and *generator-based* approaches. The former automation strategy works algorithmically and is capable of considering formally expressed design constraints *explicitly*. In contrast, the latter

automation strategy works procedurally and can consider design constraints only *implicitly*, but without the need to formalize them. So far, no automation approach supports a well-balanced consideration of *formalized and nonformalized* constraints, although both kinds of constraint representation are equally indispensable to cope with the *qualitative complexity* of the analog design problem. For that reason, the technical aim of this thesis is to combine the advantages of the two automation strategies into a novel *layout methodology* able to consider design constraints both *explicitly and implicitly*.

1.2 Scientific Challenge

In Chapter 3 of [1], the working principles of optimization algorithms and procedural generators have been illustrated. An *optimization algorithm* works by iterating through a *repetitive loop* of solution space exploration and candidate evaluation. Thereby, a proposed candidate layout is successively refined with respect to explicitly expressed optimization goals as well as confinements of the solution space. Due to the problem complexity, optimization algorithms usually focus on one specific design task (floorplanning, placement, or routing). A *procedural generator* follows a *straight sequence* of commands to create a customizable layout, depending on the generator's input parameter values. Procedural generators for layout design are commonly implemented as *parameterized cells* (PCells) that can be used by being *instantiated*. In contrast to an optimization algorithm, a procedural generator has the natural ability to perform both placement and routing simultaneously.

A characteristic trait of optimization algorithms is that they translate the layout problem into an abstract mathematical model and optimize exactly the modeled aspects – not one thing more. Examining that strategy, this thesis identifies a distinct underlying paradigm denoted as *top-down automation*. In contrast to optimization algorithms, which self-intelligently *find* a layout *solution* at runtime, a procedural generator merely (re-)produces a layout *result* while the actual solution is *preconceived* by the human expert who implemented the generator. So, procedural generators follow a fundamentally different automation paradigm here denoted as *bottom-up automation*. While optimization algorithms rely on an *abstraction* of the problem (to *lessen* the degrees of freedom) and repeatedly *re-invent* the solution anew, procedural generators pursue the *generalization* of a known solution (to *increase* the degrees of freedom) which facilitates a parameterized way of layout *re-use*.

Relying on a complete formalization of the problem at hand, optimization algorithms are *universalists*: they can be flexibly applied to a wide range of circuits, but concede losses in layout quality (due to the inconvenience of having to *formalize all* design requirements, and also because of the algorithmic inability to take all these design requirements into consideration). In contrast, procedural generators are *specialists*: they produce layouts in full-custom quality but are limited to specific design tasks like automating a certain device type or circuit

class (and may still require significant predevelopment effort for covering sufficient parametrical variability and to *anticipate all* design requirements within the range of the respective design task in advance). Hence, the strengths and weaknesses of top-down and bottom-up automation precisely complement each other. This insight suggests, that converging the two paradigms towards a new *bottom-up meets top-down* flow may not only facilitate a consideration of both formalized and nonformalized constraints, but could be the key to *close the gap* of analog layout automation one day. However, the heterogeneity of top-down and bottom-up automatisms poses the scientific challenge of investigating how these two *fundamentally different paradigms* can be combined at all.

1.3 Practical Ambition

Apart from establishing the theoretical foundations for a substantially new *layout methodology*, this thesis also intends to underline its practical usefulness. For that purpose, the field is plowed in two regards: not only to prepare the ground for the philosophy of an innovative *bottom-up meets top-down* flow, but also to straighten out some misconceptions from the past 30 years of analog EDA. In particular, this adverts to the mistake of comparing analog automation approaches in a More Moore style, as is traditionally done in literature. Chapter 4 in [1] argued that a thorough assessment adheres to many diverse criteria and is therefore a nontrivial More than Moore issue. In short, evaluating a layout methodology is supposed to consider both its impact on a customer's *design productivity* (which may not only yield an *efficiency gain* but also demands some *effort* and *expense*) as well as the degree of achievable *layout quality* (which has two facets referred to as *functionality* and *consistency*).

Based on these ruminations, the practical ambition of this work is to demonstrate its implementation in practice and to assess its true value for industrial application. Knowing about the tremendous complexity of analog layout design (and the unfortunate habit of previous EDA achievements to overestimate oneself), it is only fair not just to point at the assets of the new methodology but also to give an honest account of its shortcomings and to concede for which scope of problems it is really suitable. While providing an objective comparison of analog layout approaches is difficult in general, this is particularly true for the thesis at hand because its merit is not merely to extend some existing automation technique in one specific regard, but to pioneer an innovative, rather unconvential, and quite interdisciplinary multi-agent approach. Building upon the basic idea of distributing the layout design tasks in a decentralized way, Chapter 2 introduces the phenomena and the concepts from which that systemic approach can be deduced, before Chapter 3 describes the complete methodology in detail.

Chapter 2

An Interdisciplinary Approach – Preliminary Considerations

The future is already here – it's
just not very evenly distributed.
William Gibson (American-Canadian writer)

This chapter provides some preliminary considerations for an understanding of the interdisciplinary approach that will be taken by the layout methodology proposed in this thesis. First of all, Section 2.1 articulates the basic idea of tackling the analog layout design problem by *distributing* the design tasks across autonomously interacting, parameterized layout components. Next, Section 2.2 calls attention to three seminal terms in that context: (1) *decentralization*, (2) *self-organization*, and (3) *emergence*. For historical and didactical reasons, these three terms are then elucidated in reverse order: Section 2.3 illustrates the phenomenon of emergent behavior, while Section 2.4 discusses certain principles of self-organization, until Section 2.5 presents a relevant spectrum of existing models and implementations of decentralized systems. Finally, Section 2.6 alludes to the adaptation of these topics to the analog layout design problem, thus directly segueing into the description of the novel methodology in Chapter 3.

2.1 Divide and Conquer – Distribute and Conquer

As already discussed in Section 3.2 of [1], optimization algorithms and procedural generators follow two fundamentally different automation paradigms. But in one respect, it can be said that they share a common drawback: the determination of the layout solution is effected in a totally *centralized* way. Either, the algorithm has to solve the entire design problem all alone, or the layout solution must be completely preconceived by the design expert who develops the generator.

In the case of an optimization algorithm, one central optimization engine takes care of every design decision, thus performing the solution finding in a "dictatorial" fashion. However, the burden of having to consider all design constraints explicitly, leads to the *quality gap* shown in Figure 3.18 (page 61 of [1]). Dividing the design problem *top-down* into separate placement and routing steps –as it is done in the digital domain– can surely reduce the problem complexity, but is obviously not a feasible pragmatism in the analog domain due to the strong interdependence of the different design tasks (as underlined by the example in Figure 3.12 on page 44 of [1]).

In the case of a procedural generator, trying to realize a high-level circuit module with a powerful layout PCell, the developer of the PCell must be "prophetic" enough to foresee all design eventualities in advance. This, and the difficulty of covering the entire variability with a reasonable amount of development effort leads to the hunched-down curve and the *automation gap* in Figure 3.19 (page 63 of [1]). Of course, the design problem can be unraveled hierarchically, using generators as building blocks to form more complex modules in a *bottom-up* fashion. Still, such an approach has not yet shown to be profitable enough above the level of *simple modules* (as defined in Table 2.1 of [1]).

This thesis proposes a more concerted divide-and-conquer approach: a system, where layout design tasks are *distributed* across multiple autonomous layout components which are able to interact with each other to solve the design problem by themselves. The idea is motivated by the observation that such *decentralized systems* can outperform "classical" approaches when targeting problems of high complexity [2]. While it has been criticized that the term *complexity* is often used without giving a clear definition [3], one widespread view in *complexity theory* is that the complexity of a system is associated with the number of system components and the connections by which they are interrelated [4].

In [5], that view is phrased as: "in order to have a complex you need: 1) two or more distinct parts, 2) that are joined in such a way that it is difficult to separate them". This notion of a *complex system* is quite reminiscent of the *qualitative complexity* in analog layout design, where the design components are mutually so perturbing and the design constraints so heavily entwined, that it is not generally possible to decompose the problem in the conventional, purely reductionist *top-down* fashion, nor to assemble the solution in a constitutive, building-block-based *bottom-up* style. Instead, a *decentralization* approach is much more promising to tackle the problem – especially when taking a look at how remarkably interesting and seemingly intelligent decentralized systems can behave (Section 2.2).

2.2 Decentralization, Self-organization, Emergence

Nature demonstrates some striking examples of decentralized organisms, such as the flocking behavior exhibited by a group of birds: although each bird chooses its own course itself, their aggregate activity can lead to impressive cohesive motions [6]. Most prominently, as illustrated

in Figure 2.1 (taken from [7]), large birds can often be observed to migrate in a V-formation, which is not only fascinating to watch but has some vitally important benefits such as reduced predation risk [8] and aerodynamic gains [9] (which also led to an adoption of such behavior in air forces: the so-called echelon flight formation for military aircraft [10]). Again, it should be emphasized that the collective flocking behavior arises –without central guidance– from the local actions of the flock's individual members. The birds manage to *self-organize*, wherein the joint achievement of the flock goes beyond the capabilities and the grasp of each individual animal. This form of *suprasummativity*[1] exemplifies an intriguing phenomenon called *emergence* [13], which can not only be encountered in nature but also in many other fields, as will be illustrated in Section 2.3.

Figure 2.1: Flock of blue geese in V-formation, photographed at the Mississippi Delta [7].

If emergence occurs in a decentralized system, it can lead that system from an initially entropic state into a form of overall order. Such a process represents one acknowledged notion of *self-organization* [14] (although, providing a universal definition of self-organization is again a debate of its own [15]). An interesting question is, under which circumstances self-organization can occur – not only to explain certain observable occurrences of self-organization but also to learn how self-organizing systems can be built to solve practical problems [16]. The latter intention is also pivotal for the work in this thesis, because the interaction of the layout components has to be engineered such that it evolves into a flow of self-organization, turning an initially disordered candidate layout into a "perfectly ordered" (DRC-clean, LVS-correct, and constraint-compliant) layout result. Research on the principles of self-organization is still rather juvenile, but many appealing theories have been formulated since the 1940s, particularly in the

[1]The term *suprasummativity* originates from *Gestalt psychology* [11], where it denotes the principle, that the human mind acquires meaningful perceptions by forming a global whole that is independent of its parts in the perceptual system [12].

field of *cybernetics*. Notable works that are relevant to the topic of this thesis will be presented in Section 2.4.

A characteristic trait of decentralized systems is *distributed control* [17], which means that authority is allotted to multiple subsidiary entities of the system. With respect to layout automation, the benefit of suchlike decentralization is not just to share the workload, but rather to break down the complexity of the design problem – or, in other words, to meet the problem complexity with a system that is itself very complex (in the sense of [4] and [5] above). To achieve this, an important preparation is to decouple the system entities from the total complexity of the overall problem by making them autarkic (i.e., self-sufficient) and letting them interact with each other using only local information, without knowledge about the global state of the system [18]. In general, this also implies that the behavior of each entity is to be based on very few and very simple rules [19]. The relationship between such rules on the *microscopic* level and the resulting *macroscopic* behavior (terms used by [20]) are investigated in many existing models and implementations, some of which can be subsumed under the term *artificial life* (A-Life) [21]. Relevant achievements, from simulators of biological organisms to artificial systems that have been specifically designed for solving mathematical problems, will be covered in Section 2.5.

Interestingly, the close relatedness of the three topics emergence, self-organization, and artificial life (which will be detailed in the subsequent three sections), becomes especially apparent in the following formulation: “The theoretical focus of *A-Life* is the central feature of living things: *self-organization*. This involves the spontaneous *emergence*, and maintenance, of order out of an origin that is ordered to a lesser degree.” [22]. The following Section 2.3 shows some illustrious examples of emergence.

2.3 Emergence: A Natural Phenomenon

The idea of a whole that is greater than the sum of its parts, dates back to Aristotle and the time of the ancient Greeks [23], but it was not until 1875 that the term *emergent* was coined by the English philosopher G. H. Lewes regarding certain effects of a chemical reaction [24]. These two perceptions already indicate that the definition of emergence has an *epistemological* aspect and an *ontological* aspect [25]. Epistemologically speaking, emergence is an attribute denoting that the global properties of a system cannot be reduced to the properties of its lower-order parts [26]. For example, the color of gold cannot be derived from examining a single gold atom [27]. In ontological terms, emergence is a process whereby new, collective structures evolve over time through interactions of simpler subunits [28]. This phenomenon accounts for the observation that intelligent behavior can arise from interacting entities that are individually not “terribly smart” [29]. The two aspects of emergence have lead to a distinction between different forms of emergence, as will be discussed in Section 2.3.1.

2.3.1 Forms of Emergence

In taxonomies for emergence, the epistemological meaning above relates to what has been denoted as *nominal emergence* [30] (Figure 2.2, left). Nominal emergence can further be divided into *intentional* and *unintentional* emergence [31]. As an example for intentional emergence, the function of a machine is an emergent property of its components (e.g., the ability to tell the time is an emergent property of a clock's inner gears and wheels). Unintentional emergence mainly refers to statistical quantities of aggregations with many identical particles (e.g., the volume of a gas). While the epistemological meaning of emergence has, above all, incited a philosophical debate about the overuse of reductionism in physics research [32], the ontological notion is not only more relevant for the work in this thesis, but also more spectacular (as the examples in this section are about to show).

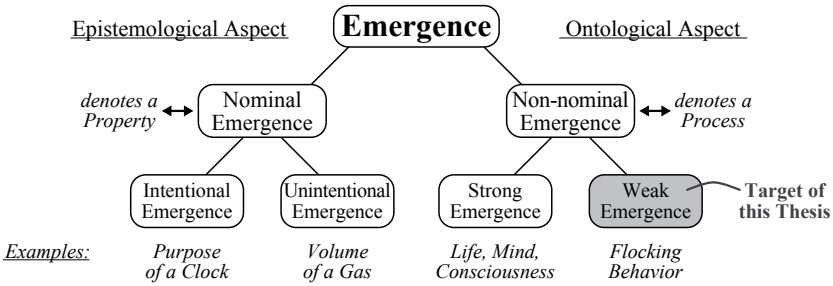


Figure 2.2: Overview of the different forms of emergence as commonly classified in literature.

The ontological, i.e., *non-nominal*, form of emergence is characterized by a dynamic behavior which is unexpected and unpredictable, whereat philosophers like to distinguish between *strong* and *weak* forms of non-nominal emergence [33] (Figure 2.2, right). Strong emergence (also denoted as *radical* emergence [34]) refers to processes that cannot even *in principle* be derived from lower-level processes [35]. Often-cited examples are life, mind, consciousness, and culture (also see [36]). However, the existence of strong emergence is contentious, with critics pointing to the obvious lack of evidence [37]. Weak emergence refers to processes where the macroscopic behavior is completely determined by the causal dynamics of its parts, but can only be predicted by performing a complete simulation [38] (which is often not possible because not all details about the parts and their dynamics are known [33]).

While the idea of strong emergence remains subject to philosophical discourses (surrounded by a lot of controversy), it is the understated notion of weak emergence that attracts scientific interest, mainly in complex systems theory [39], and also represents the target of this thesis. Speaking of weak emergence, the following real-life examples are by definition "weak" – but phenomenologically quite astonishing nonetheless. For that reason, the remainder of this thesis

simply uses the term *emergence* to denote that form of non-nominal emergence which is so relevant for the presented work.

2.3.2 Emergence in Biology

Emergent cohesive motions of animate beings are not only exhibited by flocks of birds (as in Figure 2.1), but can also be observed in many other aggregations of congeneric creatures such as shoals/schools[2] of fish [40], herds of mammals [41], swarms of insects [42], or crowds of people [43] (see Figure 2.3, with pictures from [44], [45], [46], and [47]).[3] Behavior of that kind is collectively denoted as *swarm behavior* or *swarming*. As such, these two terms do not only apply to swarms of insects, but to animals in general and also to inanimate entities. A highly topical example for the latter case is swarm robotics [49], a field of interest that is predicated on biological studies of natural swarm behavior. Likewise, swarming has also been the inspiration for the population-based Particle Swarm Optimization technique discussed in Section 3.1.3 of [1].

Figure 2.3: Emergent cohesive motions of animate beings: a school of jacks [44], a herd of deer [45], a swarm of bees [46], and a crowd of humans [47].

[2]Shoaling means that some fish stay together and form a social group, swimming around arbitrarily. Schooling means that the fish are all swimming in the same direction.

[3]This kind of emergent behavior cannot be attributed to groups of canids because they involve dominance, e.g., an alpha wolf among a pack of wolves [48].

A captivating example of intelligent swarm behavior is given by ant colonies such as in Figure 2.4 (image taken from [50]), especially in respect of the ants' collective ability to find the shortest path between their nest and a nearby food source [51]. As explained in [52], every wandering ant deposits a pheromone trail which in turn attracts other ants that intensify the pheromone trail. Since pheromones evaporate over time, the amplification of the pheromone density correlates inversely with the distance that the ants need to travel down a path and back again. Thus, the discovery of the shortest path *emerges* automatically from the individual movements of the ants. This pheromone-guided ant behavior has also found its way into computational optimization: it served as the blueprint for the Ant Colony Optimization algorithm (again, see Section 3.1.3 in [1]).

Figure 2.4: Safari ants foraging for food on the Chogoria route of Mount Kenya [50]. The shortest path to the food source emerges from the aggregate pheromone deposition of the ants.

2.3.3 Emergence in Physics

Emergent behavior can not only be found among sentient beings, but also on atomic and molecular levels. One shining example is crystallization, the process where microscopic particles aggregate in a thermodynamic solid state and form highly ordered lattice structures. In nature, crystallization is nicely demonstrated by snowflakes, where due to subtle differences in crystal growth conditions –depending on temperature and humidity– polymorphic crystal structures with different geometries abound [53]. Despite the immensely wide variety of sizes and shapes, the aggregation of the ice molecules often leads to an emergence of dihedral symmetry in symmetry group D_6 (i.e., six-fold radial symmetry) on the macroscopic scale [54]. This

phenomenon is also shown by the magnified snowflake exemplar in Figure 2.5 (photography by [55]), which has six variously shaped ice arms grown, thus constituting the characteristic stellate appearance well known from snowflakes.

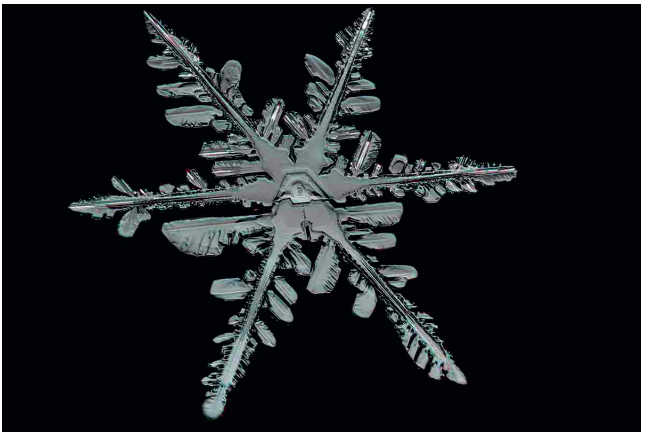


Figure 2.5: Stellate snowflake with dihedral symmetry [55] – an example of emergence in physics.

While crystals are structures of solid matter, emergence can also be encountered in liquids. For instance, one of the most commonly studied emergent phenomena in fluid mechanics is Rayleigh-Bénard convection [56]: in a thin layer of liquid which is slightly heated from the bottom, the temperature gradient leads to an upward conduction of thermal energy and thus an upwelling of lesser density fluid from the bottom layer, which –at a certain point– spontaneously becomes ordered on the macroscopic level (referred to as *spontaneous order*). This results in the appearance of rotating convection cells with regular patterns such as hexagons, eyes, and traveling waves (see samples in Figure 2.6, extracted from [57]). The formation of these so-called Bénard cells cannot be predicted due to their high sensitivity to the system's microscopic initial conditions. Suchlike behavior is denoted as *deterministic chaos* [58] (also known as the *butterfly effect*) and is subject to *chaos theory*. Another popular example of deterministic but chaotic behavior can be seen in the response of a double-rod pendulum [59].

2.3.4 Emergence in Mathematics

Some geometric figures exert fascination for being self-similar at all scales. Such patterns are referred to as *fractals* and can also be considered as examples of emergence [60]. A fractal's infinite self-similarity emerges from a simple rule or operation that is applied *recursively*. While

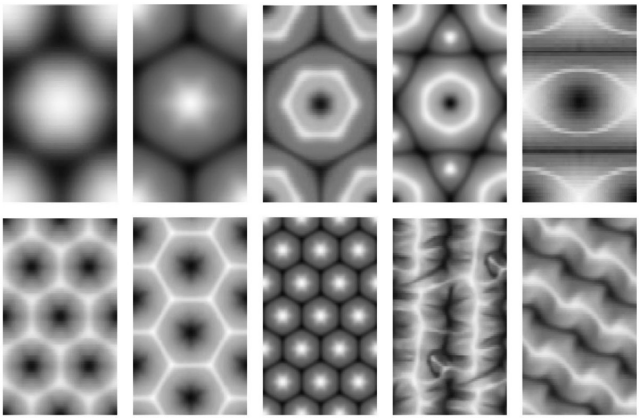


Figure 2.6: Samples of spontaneous order in liquids, emerging via Rayleigh-Bénard convection [57].

fractals can be readily found in nature (see, for instance, a Romanesco broccoli), they are a particular topic of scientific interest in mathematics – namely, in the branch of fractal geometry. One prominent fractal shape is the so-called Koch snowflake [61], which is constructed by drawing an equilateral triangle and then recursively modifying each side of the triangle by dividing the side into three segments and buckling the middle segment into another triangle. Figure 2.7 (created by [62]) shows a colored example of such a Koch snowflake (exhibiting the same radial symmetry as the real snowflake encountered in Figure 2.5).

Apart from attaining geometric beauty, fractal structures have also been used for engineering purposes. For instance, they find practical application in the design of fractal antennas as employed in telephone and microwave communications [63]. Due to their recursive pattern generation, fractals are commonly said to be self-replicative. Interestingly, it was work on self-replicating systems in the 1940s that led to the origination of a concept known as the *cellular automaton*. As will be described in Section 2.5.1, cellular automata are discrete models used in mathematics –but also in many other fields of science– to simulate dynamical systems and investigate emergent phenomena. The thematic relatedness of these topics is further confirmed by the fact that *recursivity* is considered to be one principle of self-organization, as Section 2.4 is about to touch upon.

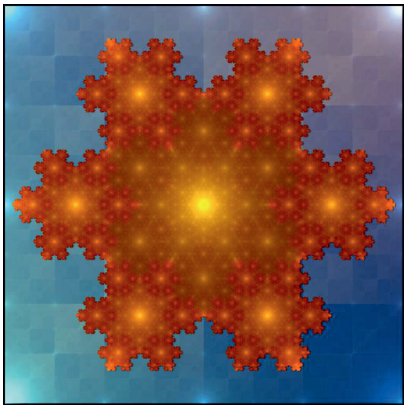


Figure 2.7: Fractal shapes like this Koch snowflake [62] can be regarded as examples of emergence.

2.4 Principles of Self-organization

The terms *emergence* and *self-organization* are sometimes used synonymously (e.g., see the debate in [64]), but they actually denote two different concepts and can exist in isolation [65] (although, as [66] puts it, this can be considered a matter of definition). As an example, a tornado is an emergent weather phenomenon but it would be presumptuous to call its appearance a product of self-organization. Vice versa, a group of castaways on a deserted island might democratically self-organize themselves, without any ranking of leadership, in order to survive – in that case, the ultimate purpose (i.e., survival) is only *nominally* emergent. The crucial question is, what ingredients are in general required –or helpful– to achieve a flow of self-organization, especially with the intention to provoke a form of emergence which is not merely nominal. Section 2.4.1 first introduces four basic constituents of self-organizing systems before Section 2.4.2 to Section 2.4.8 elaborate on seven self-organization fundamentals discussed in literature. Inhowfar all these ingredients can also be found in the layout methodology worked out by this thesis will be subject to Section 3.5.

2.4.1 The Basic Constituents of Self-organization

Referring to [67], which contemplates self-organization from a managerial point of view, a company team involves –apart from (1) the team itself, i.e., a group of interacting *workers*– three more constituents in order to self-organize: (2) a set of behavioral *rules* by which the workers may act, (3) *pressure* to get the group going, and (4) clear *goals* that are made known to the group. Urged on by these goals, one can say that there is a perpetual cycle of pressure being ex-

erted on the workers and thus stimulating them, with the workers constantly trying to satisfy that pressure by acting according to the given rules (while each worker may simultaneously attempt to pursue his or her own individual goals). These four constituents, illustrated in Figure 2.8 (a), can in general be found in any self-organizing system. And as will be shown in Section 3.1.1, they also represent the pillars of the designated layout methodology – however, with one subtle reservation, because if the goals are completely known to the workers, achieving these goals is only an intentional, *nominally* emergent property.

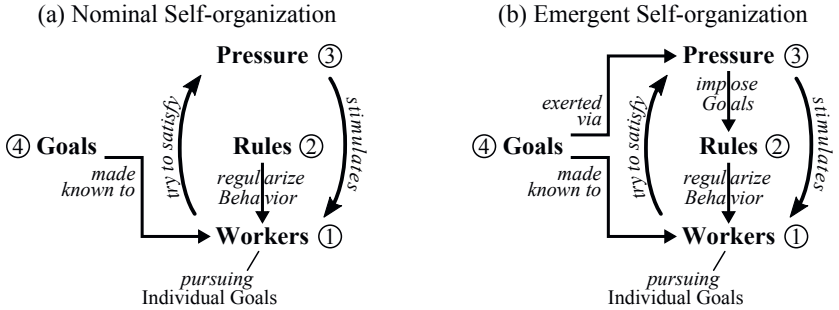


Figure 2.8: The basic constituents of self-organization: (1) workers, (2) rules, (3) pressure, (4) goals.

The situation is different in insect societies such as ant colonies (see Section 2.3.2), where labor is said to be *distributed*: such a system is accredited with the advantage that individual workers do not need to share information about how to achieve the desired collective aim [68]. Although shared knowledge can be necessary in a cooperating company team, the ant colony is better off without it: assessing the global state of the system would even be disadvantageous for the colony because "seeing the whole" is both perceptually and conceptually overburdening for any single ant [69]. In that case, detaching the workers from the colony's overall goal is even beneficial and yields an emergent form of self-organization which is not merely nominal. This poses the dilemmatic question, how suchlike emergent behavior can be aroused to solve a specific problem without plainly setting the ultimate goals. As will be covered in Section 3.1.1, this thesis takes an approach –indicated in Figure 2.8 (b)– where the achievement of the goals is not effectuated all directly (i.e., by making them completely known to all workers) but also by imposing the goals via the pressure and thereby steering the workers' actions in an indirect fashion.

2.4.2 Operational Closure and Structural Coupling

Exerting pressure to evoke emergent behavior means to put a self-organizing system "on the spot" by changing its environmental conditions. If the system is unable to influence these conditions, its only possible response is to accommodate itself to the new situation by changing its own internal structure. A system capable of maintaining itself in this way, can be called *autopoietic*.[4] According to [71], autopoietic systems are *operationally closed*, i.e., neither does the system alter its environment nor does the environment directly participate in the systems internal operations. For that reason, *operational closure* entails that the system must rely on self-organization (at least from the environment's perspective).

In contrast to *allopoietic* systems, the connection between an autopoietic system and its environment is not a causal chain with inputs and outputs, but a latent relationship referred to as *structural coupling* [72]: the environment cannot directly manipulate but only irritate the system to spur a self-produced change (which, over time, is denoted as *structural drift*). At the same time, an autopoietic system is said to be *cognitively open*. This *cognitive openness* denotes that the system does not exist in isolation but is receptive to events in its environment and able to adapt itself to environmental perturbations [73]. Such a system, which is capable of increasing its chance of survivability in a turbulent world by accommodating itself to a changing environment, is also called a *complex adaptive system*.

2.4.3 The Edge of Chaos

The work of [74], which examined the behavior of cellular automata (see Section 2.5.1) in 1990, identified a phase transition from highly ordered to highly disordered dynamics which was likely to produce *emergent computation* (because near that phase transition, the cellular automaton had the greatest potential for the support of information storage, transmission, and modification). The transition region, located in the vicinity of the border between stable and unstable automaton behavior, became known as the *edge of chaos* (in [75] initially denoted as *onset of chaos*). Figure 2.9 (adapted from [76]) illustrates the edge of chaos as the order parameter regime where high-level structures of a system appear on the macroscopic scale. The *spontaneous order* of these structures is said to be *metastable* because it is usually short-lived and can be rapidly replaced by disorder due to disturbances of that balanced state.

Until today, the phrase *edge of chaos* has also been adopted in various other fields of science (including physics, biology, and sociology) to describe the observation that many systems operate in a region between order and chaos, where the capabilities of the system are maximal [77]. For example, regarding creativity in cognitive systems, it is alleged that innovation occurs right at the edge of chaos [78]. On this note, but in more general terms, complexity theorists note that

[4] The term *autopoiesis* was coined in cognitive biology to explain the nature of living organisms [70].

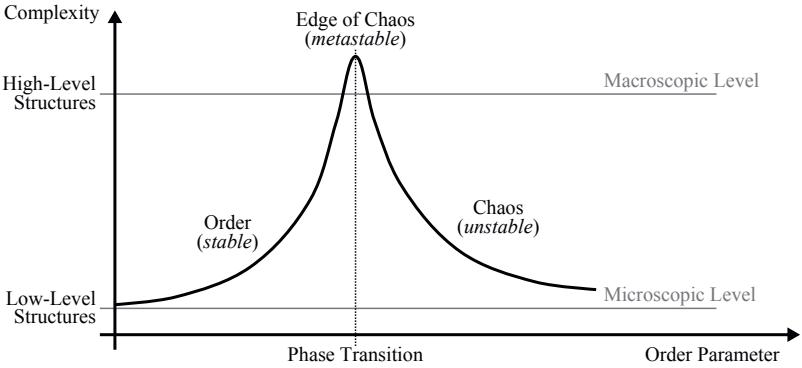


Figure 2.9: Operating at the edge of chaos [76] is considered as one principle of self-organization.

the proximity of a system's operating point to the edge of chaos is decisive for the evocation of emergent behavior and self-organization [79].

2.4.4 Recursivity and Feedback

A constant background theme of self-organization is *recursion* [80]. It is often put on record that a system's ability to reach a stable –or metastable– configuration (also denoted as an *attractor*) involves some kind of *recursivity*. In the context of *systems theory*, recursivity means that the output of some operation serves as the input for its next operation, i.e., cause and effect mutually influence each other [81]. In [82], such a recursive system is called a *nontrivial machine* and described as a *finite state machine*: it has the property that the response for a certain stimulus is not necessarily the same when the same stimulus is applied at a later time. Thus, in contrast to a *trivial machine* (which effectively denotes a combinatorial circuit, i.e., a system whose output depends only on its input), the behavior of a nontrivial machine is history-dependent and analytically unpredictable even though it is synthetically deterministic. This analytical unpredictability means that an experimenter –without knowledge about the inner workings of the machine– is not able to "crack the code" just by observing sequences of input-output pairs. According to [83], all autopoietic systems (Section 2.4.2) are organized recursively.

Recursivity is commonly understood as *feedback*, whereat two types of feedback are discerned: *positive feedback* and *negative feedback*. Regarding the latter, control cycles for error-controlled regulation are based on a negative feedback loop which counteracts a system variable's deviation from the target value (the so-called setpoint). In contrast, positive feedback (sometimes also referred to as self-enhancement, amplification, or autocatalysis) reinforces fluctuations of a variable and thus promotes changes in a system. For that reason, positive feedback

can result in a snowballing effect that may even destroy the system (*resonance disaster*), but interestingly it can also lead to an equilibrium instead [84]. As an example, one may consider the foraging behavior of ants (see Section 2.3.2): the pheromone deposition is continuously increased by attracting more ants that deposit further pheromones. However, suchlike stabilization in fact also involves negative feedback in the end: an exhaustion of resources [85]. In this example, when the entire ant colony ultimately follows the pheromone trail, there are no more ants left to amplify the pheromone density any further. In complex self-organizing systems, there usually are several interlocking feedback loops, both positive and negative.

2.4.5 Stigmergic Interaction

While *operational closure* and *structural coupling* (Section 2.4.2) designate a form of oblique correlation between a system and its environment, a similar relationship called *stigmergy* [86] can also be encountered among the entities that interact within the system. Stigmergy is a mechanism of indirect coordination whereby entities in a decentralized system communicate with each other not directly but via modifications of their environment [87]. Stigmergy was first observed in social insects, with one example (already given in Section 2.3.2) being the way ants exchange information by laying down traces of pheromone. Accordingly, algorithmic Ant Colony Optimization techniques involve a deposition of "virtual pheromones" [88].

Another –nonbiological– example is provided in [89], where self-organizing traffic lights stigmergically co-control each other via the car density of the traffic that they regulate. This example, displayed in Figure 2.10 (adapted from [89]), illustrates that stigmergy is not only an integral part of collective intelligence in natural organisms, but that it is also considered to be an expedient principle for designing self-organizing systems [90]. For instance, the work of [91] implements stigmergic interaction between a group of mobile robots whose movements are coordinated through local configurations of their environment.

2.4.6 Reducing Friction and Promoting Synergy

One term that regularly appears in the context of emergence and self-organization is *synergy* [92]. There even exists a dedicated scientific branch called *synergetics* [93] (the science of cooperation), but still, the term *synergy* is ambiguous. Often, synergy is paraphrased with the Aristotelian formulation that the whole is greater than the sum of its parts [94]. In that case, the term is synonymous to the *epistemological* understanding of emergence (see Section 2.3.1). In [95], synergy is referred to as the combined effect of an interaction, so the term is used in the *ontological* sense of emergence. Sometimes, synergy does not denote the effect of an interaction but the interaction itself (e.g., in [96]). Despite this equivocality, there seems to be consensus in two regards. First, synergy always indicates a form of cooperation. Second, while

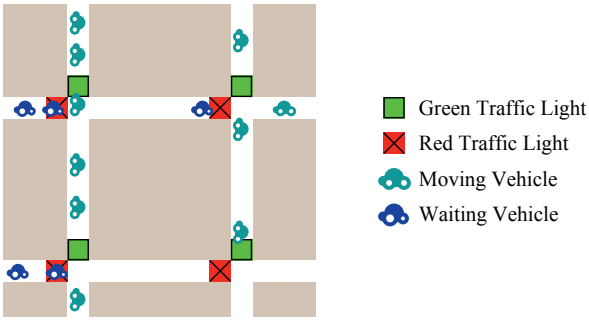


Figure 2.10: Stigmergy: the traffic lights do not communicate directly, but via the regulated traffic [89].

emergence rather represents a long-term phenomenon resulting from many interacting entities, the term synergy can apply to a single (inter-)action.

Mirroring both of these aspects, the author of [97] understands synergy as a correlation or concourse of action. In this spirit, but with a focus on the design of self-organizing systems, the author of [98] speaks of synergy when the action of an entity is not only beneficial for the entity itself but also for the other entities (and thus for the entire system). So, a *synergistic* action amounts to a "win-win" situation. If an entity increases its own satisfaction at the cost of the other entities, this is denoted as *friction*. The main premise of the work in [98] is that reducing friction and promoting synergy in a self-organizing system will result in better performance. This is backed by the observation, that evolutionary processes also tend towards synergistic relationships on the level of genes and genomes [99].

2.4.7 The Virtue of Selfishness

As already mentioned in Section 2.4.1, it can be advantageous if the interacting entities of a self-organizing system have only local information and elide any global knowledge about the system's overall state. Carrying this notion a bit farther, one might suggest to design the individual entities not only as *ignorant*, but even as *selfish* [100], so as to dumb down their decision-making. Intuitively, it may appear doubtful to assume that a group of such egoistic individuals could interact in a way that is beneficial for the system as a whole. However, a strong case for this idea of egoism can be made by studying nature once again. In particular, the *selfish herd theory* [101] states that –in contrast to previous beliefs– the gregarious behavior of animal species is not owing to mutual benefits of the animals but results from selfishness: in the face of predators, each animal tries to put other conspecifics between itself and the predator, which inevitably leads to an aggregation of the herd –as for example seen in Figure 2.11 (photo

by [102])– and to a reduction of predation risk for the entire group. Thus, the aggregate behavior of that population is not rooted in cooperation but in competition.

Figure 2.11: According to the selfish herd theory, animals aggregate out of selfishness (photo: [102]).

The characteristics of such competitive settings are mathematically investigated in *noncooperative game theory* (see Section 2.5.2), where egoistical *players* are called *self-interested*. However, as [103] points out, self-interested does not necessarily mean that the players want to cause harm to each other, but that they act according to their individual preferences and within their own scope of action. Regarding the contemplations of Section 2.4.6 about *synergy*, this remark connotes that –although synergy implies cooperation instead of competition– the selfishness of an entity can still allow for synergistic actions. An equivalent notion can also be found in economics, where the metaphor of Adam Smith's *invisible hand* describes that society can benefit from the self-interested efforts of individuals [104].

2.4.8 Law of Requisite Variety

A herd's reaction to predatory danger (as discussed in Section 2.4.7) exemplifies the pivotal question which escorted all of the preceding ruminations in this Section 2.4: how apt is a self-organizing system in responding to disturbances that emanate from its environment? One fundamental answer to that question is found in Ashby's *law of requisite variety* [105]. By Ashby's definition, *variety* denotes the total number of potential distinct states of a system. In a more general sense, variety is the repertoire of actions available to a system. Ashby's law (also

known as the first law of cybernetics [106]) says: the larger the repertoire of actions available to a system, the larger the variety of environmental disruption the system is able to compensate.

Ashby's law of requisite variety can be understood as a condition for dynamic stability under external perturbations, and is also considered as a simpler version of Shannon's Tenth Theorem [107]. Two further notions shall be brought up here: first, according to Ashby, variety is a measure for the complexity of a system [108], and second, Ashby's law has also been phrased as "variety absorbs variety" [109]. These two notions elucidate the statement that was made in Section 2.2 about the approach taken by the methodology in this thesis: to address the complex problem of analog layout with a system that is itself very complex. In other words, variety of action is required by the methodology to cope with the many degrees of freedom and the diversity of constraints that are so characteristic of the analog IC domain.

2.5 Models of Decentralized Systems: A Form of Artificial Life

Inspired by the phenomenological occurrences of emergence (see Section 2.3) and with respect to the oft-enunciated principles of self-organization (Section 2.4), various models of *decentralized systems* have been proposed since the middle of the 20$^{\text{th}}$ century, targeting the examination of emergent behavior as well as the practical utilization of self-organizing structures. And although no approach specifically addresses the problem of layout automation in analog IC design, it is worthwhile to survey a couple of relevant achievements. Therefore, this Section 2.5 presents an overview of existing concepts in that field, particularly concentrating on those ideas that have influenced the work of this thesis (as will be indicated in each of the following subsections and comprehensively summarized in Section 3.5). It has already been mentioned that a couple of implemented (i.e., computer-based) models of decentralized systems imitate natural biological processes and are, for that reason, often referred to as *artificial life* [110]. A famous example –the *Game of Life*– will be given in the following Section 2.5.1, which covers the concept of *cellular automata*.

2.5.1 Cellular Automata

The concept of cellular automata, suggested by Stanislaw Ulam and taken on by John von Neumann for his work on self-reproducing systems in the 1940s [111], can be considered the first exemplification of artificial life. A cellular automaton is a space- and time-discrete mathematical model, given as an n-dimensional lattice of cells each of which behaves like a *finite state machine*. At any point in time, all cells are in one of a finite number of states, and with every time step, each cell changes or keeps its state according to a simple *transition function* that

involves the state of the cell itself and the states of its neighboring cells [112]. In their most illustrative form, cellular automata have two dimensions (represented as an orthogonal grid of adjacent squares), each cell switches between two states, and a cell's neighborhood is comprised of the cell's eight surrounding cells.

One example of such a cellular automaton is *Conway's Game of Life* [113], which was devised in 1970 and attracted much interest in and beyond academia. In Conway's Game of Life, every cell is in a state of being "alive" or "dead" and –depending on its neighboring cells– either dies (due to isolation or overpopulation), lives on (to the next generation) or becomes alive (as if by reproduction) at the next time step. The fascination with the Game of Life emanates from its visualization of emergent behavior, because –based on this simple set of rules– quite interesting patterns arise after several generations. These include static patterns (still lifes), periodic patterns (oscillators), moving patterns (gliders, spaceships), self-copying patterns (replicators), patterns that emit other patterns (guns, rakes, breeders), and patterns that take a large number of generations to stabilize or vanish (methuselahs). Figure 2.12 shows some examples of such patterns as well as a Garden of Eden (an orphan pattern that has no predecessor and can therefore not appear beyond the automaton's initial configuration). Apart from the surprising ways, in which such complex configurations evolve over time, a scientifically interesting aspect of the cellular automaton is its computational power: since it is possible to construct counters and logic gates by letting gliders interact with each other, Conway's Game of Life is *Turing complete* [114].[5]

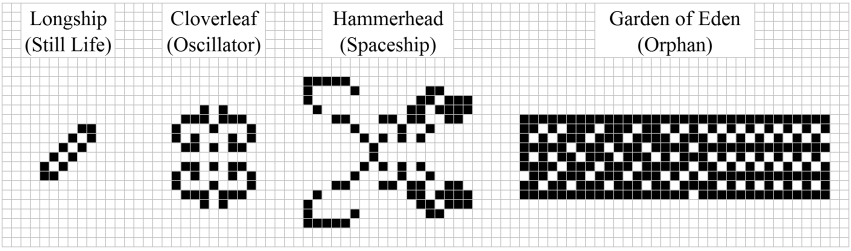


Figure 2.12: Examples of patterns in Conway's Game of Life, a two-dimensional cellular automaton.

Another Turing complete system that can be described as a cellular automaton is *Langton's Ant* from 1986 [115]. It simulates step-wise orthogonal movements of a single virtual ant whose decisions are based on two simple rules. Over thousands of steps, these rules lead to complex behavior, as the ant can be observed to first create plain symmetric patterns, which later become

[5] In computability theory, Turing completeness signifies that a system can be used to simulate a Turing machine (which is an abstract mathematical computation model). This means, that such a system is generally capable of performing all data manipulation tasks which are also accomplishable by real-world computers.

chaotic and irregular, until the ant's motion eventually leads to the emergence of a straight, diagonal, indefinitely repeating ant trail. Similar to Conway's Game of Life, Langton's Ant is not only interesting for its capability of universal computation (proven in [116]), but for the evolvement of captivating patterns such as spirals and snowflake-like, hexagonal structures (in a generalized concept known as *Turmites*). Langton's Ant can be implemented within *Wireworld*, a cellular automaton first presented in 1987 [117]. The cells in Wireworld cover four different states and behave with respect to four very trivial rules. Like the previously mentioned examples of cellular automata, Wireworld is computationally universal, despite the simplicity of its state transitions. It is particularly designed to simulate electronic circuits and has even been used to implement a Turing complete computer.

Until today, many types of cellular automata have been developed and are being investigated for various practical applications such as image recognition, task scheduling, and cryptography [118]. Apart from that, their computational capabilities and the diversity of emergent patterns soon inspired scientists to draw certain parallels between cellular automata and the complexity of natural processes. Already in 1969, Konrad Zuse proposed that even the evolution of the universe itself is, at heart, digitally computed by some sort of cellular automaton, presuming that physical laws are not continuous by nature but discrete [119]. In 2002, Stephen Wolfram published an extensive study of cellular automata to pronounce their importance for science in general, supposing huge potential for providing insight into the behavior of complex systems in all kinds of fields [120].

The bearing of cellular automata for this thesis is plain: just like the simple interactions of an automaton's cells leads to the emergence of "living" higher-level structures, the designated layout methodology attempts to let the overall layout solution emerge from the dynamics of interacting lower-level layout components. And as will be seen in Section 3.5, some particular aspects of cellular automata (e.g., including an entity's local areal neighborhood into its decision-making, or changing the "state" of an entity) can also be identified in the novel layout methodology.

It is interesting to note that the main inventor of cellular automata, von Neumann, was also one of the founders of modern *game theory*, as follows in Section 2.5.2 (by the way, Conway's Game of Life is also referred to as a zero-player game).

2.5.2 Game Theory

One key discipline to model the self-interested decision-making of entities in a decentralized system is game theory [121], which is the mathematical study of strategic interaction among independent rational *players* who try to maximize their so-called *payoff*. Game theory primarily serves the examination of *coalitional* or *competitive* behavior in such situations and the prediction of probable *outcomes*. Commonly, this is achieved by identifying so-called *solution*

concepts which allow game theorists to deduce *equilibrium strategies* for each player. The most influential solution concept in game theory is the *Nash equilibrium* [122], where no player can increase his or her own payoff by unilaterally choosing another strategy. The payoff is usually represented by a *utility function* which –in short– maps a player's preference for a possible outcome to a real number.

Game theory distinguishes between many different types of games. Considering the criteria that can also be recognized in the work of this thesis, a game can be

- a *cooperative game* (where groups of players can form coalitions) or a *noncooperative game* (where each player makes individual choices),
- a *discrete game* (where all players choose from a finite set of strategies) or a *continuous game* (where strategies are chosen from an infinite continuous set),
- a *symmetric game* (where the payoffs for a strategy are independent of the player, as in Figure 2.13) or an *asymmetric game* (where changing the identities of the players changes their payoffs, as if the players had taken on different roles),
- a *constant-sum game* (where each player's gain/loss comes at the expense/benefit of the other players) or a *non-constant-sum game* (where the aggregate gains and losses are not the same for every outcome),
- a *simultaneous game* (where all players choose their strategy without knowing about the other players' choices) or a *sequential game* (where players decide one after another, having –at least some– knowledge of the other players' prior choices),
- a *perfect-information game* (a sequential game where a player always knows all choices previously made by the other players) or an *imperfect-information game* (a sequential game where a player may have only partial or no knowledge of earlier choices).

If a game is played multiple times, the game being repeated is called the *stage game*. A game, that is played a finite and known number of times, is denoted as a *finitely-repeated game*, whereas a game that is played infinitely often or a finite but unknown number of times is referred to as an *infinitely-repeated game*.

Games in game theory can be represented in various ways. The most fundamental game representation is the *normal form* because many game-theoretic settings can be reduced to normal-form games. Simply put, the normal form allows to represent the payoffs for a set of n players, depending on the vector of chosen actions (called an *action profile*), in an n-dimensional payoff matrix. A textbook example of a game that can be represented in normal form is the well-known *Prisoner's Dilemma* [123] (depicted in Figure 2.13). Since the normal form does not support any notion of sequence, sequential games are often reasoned about in the so-called *extensive form* representation. An example of such an extensive form game, where the successive actions of the players are represented in a rooted game tree, is *Centipede* [124]. Both the Prisoner's

Dilemma and the Centipede game illustrate the paradoxical case that two players might choose not to cooperate although both would be better off by doing so.

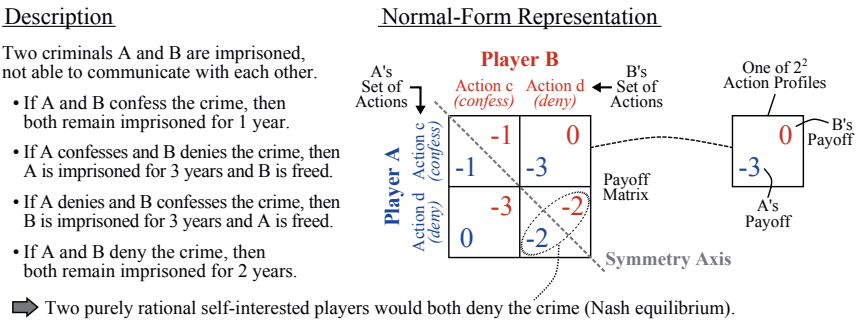


Figure 2.13: The Prisoner's Dilemma – a noncooperative, discrete, symmetric game in normal form.

Game theory is used to model real-life situations of conflict and coordination. In the 1940s, game theory was initially developed to understand economic behavior [125], but today it also plays an important role in disciplines as diverse as political science, biology, psychology, philosophy, and computer science. With respect to this thesis, game theory is interesting since the decision-making of the interacting layout components in the novel layout methodology can also be regarded from a game-theoretical point of view. However, these layout components are no mathematical conceptions of abstract players but concrete computational entities – such as the entities used to build *multi-agent systems*, which will be the topic of the subsequent Section 2.5.3.

2.5.3 Multi-Agent Systems

Multi-agent systems [126] are a relatively new field of computer science that followed a 1980s trend from classically centralized approaches of artificial intelligence towards distributed control. A multi-agent system is defined as a network of loosely coupled *agents* that interact with each other in a given environment. Therein, an agent [127] is a computational entity such as a software program or a robot (with the corresponding environment being a virtual space or a real physical setting, respectively). Such an agent is commonly accredited with the following characteristics:

- the agent is able to perform actions upon its environment,
- the agent is autonomous in its decisions about how to act,
- the agent has a particular degree of intelligence (and often mobility),

- the agent has a limited viewpoint with only incomplete information about its environment,
- the agent is rational in that it adheres to a goal-directed behavior following clear preferences.

According to [128], agents can be grouped into the following five classes: simple reflex agents, model-based reflex agents, goal-based agents, utility-based agents, and learning agents. As a side note in reference to Section 2.5.1, multi-agent systems are related to cellular automata with respect to the consideration that both implement a *complex system* (in the sense of [4] and [5], as mentioned in Section 2.1), albeit at two ends of the spectrum: compared to each other, cellular automata are complex systems with simple agents, whereas multi-agent systems are simple systems with complex agents [129].

In [130], multi-agent systems are denoted as concepts of distributed artificial intelligence that complement artificial intelligence and artificial life. Although multi-agent systems can (and are usually meant to) exhibit some form of collectively intelligent behavior, they must be distinguished from certain methods of *swarm intelligence*, which are investigated in the context of mathematical optimization and thus represent an algorithmic sub-branch of artificial intelligence. As already discussed in Section 3.1.3 of [1], a method of swarm intelligence such as Particle Swarm Optimization (for which a lot of different applications exist [131]) is a population-based optimization technique where the solution space is explored by multiple *particles*, each of which represents one candidate solution to the optimization problem. This is a fundamental difference towards a multi-agent system, in which every agent contributes to solving a global problem but is not itself a solution to the problem.

While the idea of multi-agent systems has not gained widespread recognition up until the mid-1990s, interest in the field is now growing significantly, to some extent spurred by the initiated technical evolution towards the Internet of Things and Industry 4.0 [132]. Today, the usefulness of multi-agent systems is actively examined in a large host of diverse applications, such as aircraft maintenance [133], supply chain management [134], and energy resource scheduling [135], just to name a few. With the thesis at hand, the problem of layout automation is added to that palette since the interacting layout components of the new layout methodology can –according to the five characteristics above– also be understood as agents. However, an extraordinary trait therein is that each of these agents is not only meant to help in solving the actual layout problem but in fact represents a part of its solution.

It should be mentioned that the term *multi-agent system* is sometimes confused with the notion of an *agent-based model*, although the two concepts are not the same – as will be explained in Section 2.5.4.

2.5.4 Agent-based Models of Collective Motion

There is a considerable overlap between multi-agent systems and agent-based models, but a subtle distinction can be made [136]: multi-agent systems are understood as an engineering approach (i.e., to solve particular problems), whereas agent-based models rather have a share in analytical science (such as in simulating the behavior of biological organisms). While agent-based modeling can be used to imitate a wide variety of decentralized systems in numerous domains, this section is particularly focused on models of collective motion, as exhibited by ensembles whose members perform spatial moves. Such models, in which the environment represents a geographical space, are called *spatially explicit* [137]. That conception also applies to this thesis, regarding the intention to let autonomous layout components interact with each other inside a layout cell.

The first suchlike agent-based simulation model was *Reynolds' Boids* from 1987 [138], an artificial life program that animates the flocking behavior of birds (or "creatures" in general). As illustrated in Figure 2.14, Boids implements a so-called *distributed behavioral model* where the collective, polarized motion of the flock emerges from three simple *steering rules* obeyed by each individual flockmate (referred to as a *boid*: a bird-like "bird-oid" object). In [139], these steering rules are denoted as *separation* (steer to avoid crowding flockmates), *alignment* (attempt to match the heading and speed of flockmates), and *cohesion* (steer to move towards the average position of flockmates). As in a cellular automaton, the steering rules involve a boid's local-only neighborhood, such that each flockmate reacts solely to nearby neighbors in its immediate vicinity.

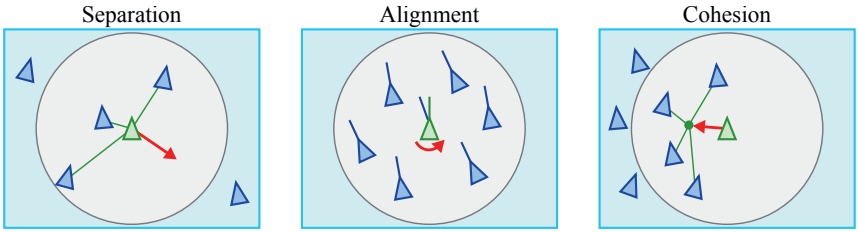


Figure 2.14: Depiction of the three simple steering rules in Reynolds' Boids (adapted from [139]).

Concerning the definition of such a neighborhood, a basic question is whether the environment of the boids' flock is two- or three-dimensional (as in [140]). Apart from this dimensionality, the neighborhood is further defined by its extent – a property for which two prevalent models can be consulted [141]. In the so-called *metric distance model*, a boid is only affected by those of its neighbors that are located within a certain range. This accounts for three different, concentric, circular zones (zone of repulsion, zone of alignment, zone of attraction) which

correspond to the three steering rules above. In contrast, the *topological distance model* lets a boid pay attention to a fixed number of its closest neighbors, regardless of their respective distance. A hybrid metric-topological composition model is presented in [142].

Several behavioral extensions of Reynolds' original Boids program have been proposed by others. For example, the authors of [143] complement the alignment rule (see above) with a force that facilitates a change of leadership. The work of [144] provides an additional rule called escape and incorporates the effect of fear. This idea can also be found in [145], where the conception of Boids is adopted for the emulation of artificial fishes which are driven by three mental state variables: fear, hunger, and libido. Emotions and sensations of this kind have also inspired certain facets of the work in this thesis. Another interesting aspect in that regard is memory, which has for example been worked into the animal group formation model of [146] and will also be an element of the layout methodology that Chapter 3 is about to present.

Encouraged by Boids, spatially explicit agent-based models can also be found in other disciplines. Examples include the cellular-automaton-based Nagel-Schreckenberg model for freeway traffic simulation [147], as well as the Vicsek model of nonbiological *self-propelled particles* (also referred to as self-driven particles) in physics [148]. A perseverative issue in suchlike agent-based modeling is the question of what the simplest possible sets of rules are that lead to the emergence of synchronized, collective motion [149]. However, it is important to note that in contrast to a flock of birds or a convoy of cars, the interacting components in the layout methodology provided by this thesis are not supposed to exhibit a unidirectional laminar movement but an overall flow that can rather be described as a motion of centripetal yet turbulent convergence.

For simulating the behavior of decentralized systems, a plethora of general-purpose agent-based simulation toolkits –both proprietary and open source– are readily available (with a comprehensive survey being given in [150]). As an example, the Java Agent Development Framework (JADE) [151] is a software framework for the development of intelligent agents in Java, based on the Foundation for Intelligent Physical Agents (FIPA) standards for hetereogeneous and interacting agents. However, tools such as these are of no further interest here due to the specialized topic of this thesis, for which a tailor-made implementation had to be provided (as will be covered and demonstrated in Chapter 4).

2.6 Adaptation to the Problem of Analog Layout Design

Incited by the ideas portrayed in this chapter, the thesis at hand strives to address the task of analog layout automation by implementing a system of interacting layout components with distributed control. The conceptual approach behind this methodology is highly interdisciplinary, drawing from (A) the developments of *decentralized systems*, (B) the investigations of *self-organizing structures*, and (C) the observations of *emergent behavior* discussed in the preceding sections:

(A) The composition of the proposed system follows the idea of *decentralization*. Hence, the setup of the taken approach is in line with other models of artificial life as covered in Section 2.5.

(B) The interaction demeanor of the system's individual layout components is carefully worked out in an endeavor to pay respect to the principles of *self-organization* described in Section 2.4.

(C) The behavior of the overall system is supposed to let full-custom quality layouts emerge from its execution, mirroring the natural phenomenon of *emergence* as illustrated in Section 2.3.

The detailed elaboration of the new layout methodology is about to follow in Chapter 3. Therein, the overview of Section 3.1 describes the composition of the system (A), while the interaction demeanor of the system's layout components (B) is subject to Section 3.2, Section 3.3, and Section 3.4. As already mentioned, parallels to existing models of decentralized systems as well as the incorporation of the presented principles of self-organization into the work of this thesis are summarized in Section 3.5. The emergent behavior of the system (C) will be depicted in the examples of Chapter 4.

Chapter 3

The Methodology: Self-organized Wiring and Arrangement of Responsive Modules

In our narrow house of boards, bestride
the whole creation, far and wide;
move thoughtfully, but fast as well,
from heaven through the world to hell.
Johann Wolfgang von Goethe: Faust
(translated by Walter Kaufmann)

This thesis proposes an analog layout automation methodology where layout components interact with each other in a *decentralized* flow of *self-organization* that leads to *emergent* behavior. As will be discussed in this chapter, the layout components –denoted as *responsive modules*– are autonomous, mobile, rational, computational entities (and thus *agents*, in accordance with the definition in Section 2.5.3). On that basis, the presented methodology neither implements a purely optimization-based *top-down* approach nor a purely generator-based *bottom-up* approach (as defined in Section 3.1.1 and Section 3.1.2 of [1]), but represents a multi-agent system which teams the two automation strategies. The new methodology is henceforth referred to as *Self-organized Wiring and Arrangement of Responsive Modules* (SWARM) [152], wherein the terms arrangement and wiring denominate the tasks of placement and routing.[1] SWARM is meant to be used by layout designers for the creation of analog layouts that fit within a given, sufficiently large, rectilinear outline. As will be discussed here (and demonstrated in Chapter 4), the SWARM methodology is able to include both *formalized* and *nonformalized* expert

[1] Semantically, arrangement and wiring are thus used as synonyms for placement and routing here, but the choice of diction is deliberate. This is not only for the sake of the SWARM acronym but mainly because –in the parlance of EDA– the terms placement and routing are commonly understood as optimization-based automation techniques.

knowledge into the automation by effectuating an *explicit* and *implicit* consideration of design constraints.

3.1 Overview of the SWARM Methodology

As will be outlined in Section 3.1.1, SWARM is composed of three core concepts that are dexterously correlated with each other. To give an impression of how these concepts are supposed to engage in a purposive interaction, Section 3.1.2 sketches the designated self-organization flow in a superficial example. In Section 3.2, Section 3.3, and Section 3.4, each of the three concepts will be discussed in a section of its own. Section 3.5 completes this chapter with some final remarks about the conception of SWARM, discerning it from optimization algorithms and then also taking a retrospective look back at Chapter 2.

3.1.1 The Three Core Concepts of SWARM

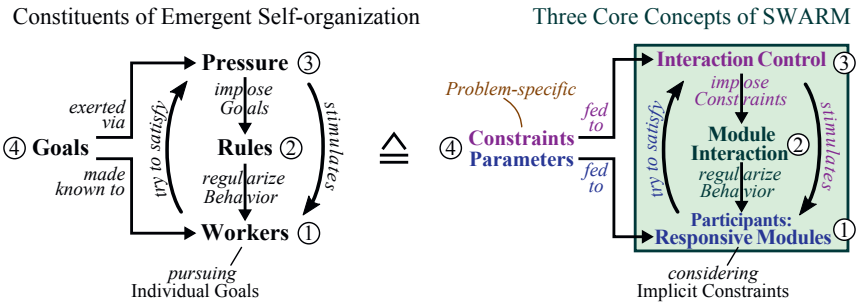


Figure 3.1: The constituents of emergent self-organization and the corresponding SWARM concepts.

The left part of Figure 3.1 reprises the depiction of Figure 2.8 (b) from Section 2.4.1, stating that a truly emergent self-organizing system requires four basic constituents: (1) a group of interacting *workers*, (2) a set of behavioral *rules*, (3) the exertion of *pressure*, and (4) overall *goals* that can be directly communicated to the workers and indirectly inducted via the pressure. As illustrated in the right part of Figure 3.1, all these constituents can also be found in SWARM. Excluding the *goals*, which are always specific to the particular layout problem (i.e., the respective design restrictions and design objectives), the SWARM system itself is thus composed of three problem-independent core concepts:

Responsive Modules: The concept of the already mentioned *responsive modules* is SWARM's analogon to the *workers* in Figure 2.8. As Section 3.2 will discuss in greater detail, a

responsive module is a *context-aware* procedural generator that can act on its own behalf within its design environment.

Module Interaction: SWARM allows a set of responsive modules to interact with each other and to arrange themselves inside a given layout territory. During this *module interaction*, each module acts according to a set of relatively simple behavioral *rules*, which will be the topic of Section 3.3.

Interaction Control: To steer the module interaction towards a compact and constraint-compliant layout arrangement, SWARM implements a supervising *interaction control* organ that exerts *pressure* by stimulating the modules and by imposing the design restrictions and objectives (i.e., the *goals*) onto the module interaction. The interaction control organ will be addressed in Section 3.4.

Before going into the three core concepts in detail, Figure 3.2 shows the layout automation approach of SWARM, as opposed to the working principles of optimization algorithms and procedural generators (see Figure 3.1 and Figure 3.13 in [1] respectively). SWARM combines these two automation strategies, since each responsive module has the *bottom-up* capabilities of a procedural generator, while the interaction control organ provides a *top-down* perspective on the design problem. Consequently, the design restrictions and objectives that can be committed to SWARM comprise both module parameters as well as formalized constraints.

On that basis, high-level constraints are imposed by the interaction control organ to be then *explicitly* considered during the module interaction, while each individual module simultaneously takes care of its innate low-level constraints *implicitly*. To underline the decentralized character of this module interaction approach, the layout that emerges from SWARM via self-organization is denoted as the layout *outcome* (taken from game theory and contrasting the terms layout *solution* in Figure 3.1 [1] and layout *result* in Figure 3.13 [1]). Section 3.1.2 provides an exemplary depiction of SWARM's self-organization flow.

3.1.2 Depiction of SWARM's Self-organization Flow

Although various kinds of decentralized systems have been developed in the past decades (see Section 2.5), SWARM is the first suchlike system whose interacting entities are layout modules. As already done in Figure 3.1 and Figure 3.2, these interacting modules are subsequently denoted as *participants* in order to discern them from the miscellaneous notions of entities found in other existing systems, some of which are listed in Table 3.1.

The self-organization in SWARM is structured as a so-called *run* which, in its most elementary form, follows the flow of control depicted in Figure 3.3. Given a set of participants (responsive modules), SWARM takes an initial module *constellation* that may for example be template-based, handcrafted, or randomly created. Next, SWARM takes the user-defined *zone*

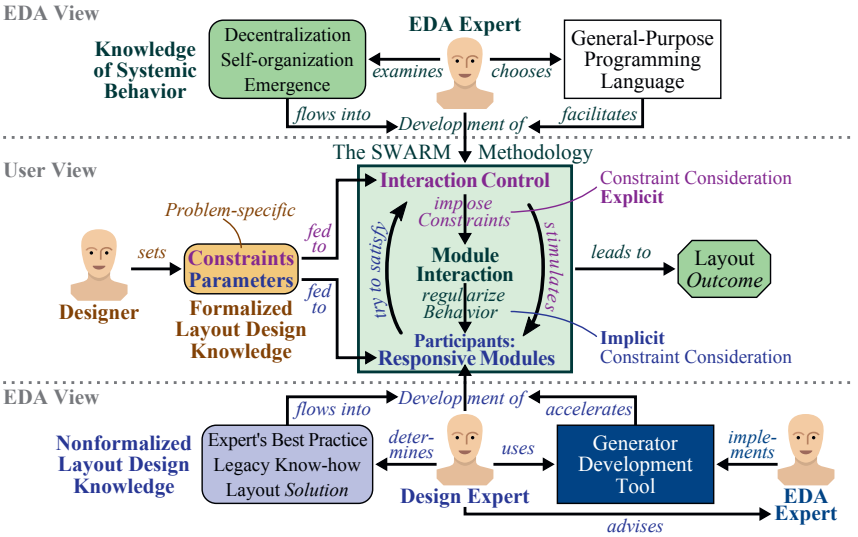


Figure 3.2: Working principle of the SWARM system, as opposed to Figure 3.1 and Figure 3.13 in [1].

Table 3.1: Denomination of interacting entities in selected examples of decentralized systems.

Name	Year	Description	Entities
Conway's Game of Life [113]	1970	Two-dimensional Cellular Automaton	Cells
Centipede [124]	1981	Extensive Form Game (Game Theory)	Players
Reynolds' Boids [138]	1987	Agent-based Simulation Model	Boids
Particle Swarm Optimization [131]	1995	Population-based Optimization Algorithm	Particles
JADE [151]	2001	Java Agent Development Environment	Agents
SWARM [152]	2015	Multi-Agent Layout Automation Approach	Participants

Z (which demarcates the available layout territory) and centers the outline of Z on the bounding box of the initial constellation. After this *initialization phase*, the supervising control organ enlarges Z such that its area is significantly greater than the sum of the participants' individual areas (see Section 3.4). Then, the *self-organization phase* starts with a so-called *round* of interaction, where each participant is allowed to perform an *action* (see Section 3.3). By performing an action, a participant undergoes a *transformation*: it may move inside the layout zone Z, may also rotate around its center, and may even deform itself into a different layout variant with

nominally equal electrical behavior (see Section 3.2). The actions in one round are meant to emulate a concurrent behavior but are in fact executed sequentially.

Multiple rounds of interaction are repeated until no action at all is performed within one round. This situation is denoted as a *settlement* because each participant has *settled* at a definite position. If the module constellation is *viable*, which means that all participants are in a legal and satisfying position, then Z is slightly tightened to induce further rounds of interaction and thus another settlement (otherwise, the participants failed in finding an appropriate arrangement, and the SWARM run gets aborted). This tightening-settlement cycle continues until Z is *tight*, which means that Z has reached its original, user-defined size that marks the available layout space. The last settlement ends the SWARM run and represents the layout *outcome* of the overall self-organization. In the end, a *finalization phase* gives the opportunity to perform some dedicated post-processing on the obtained layout. This may involve auxiliary tools or can be accomplished manually.

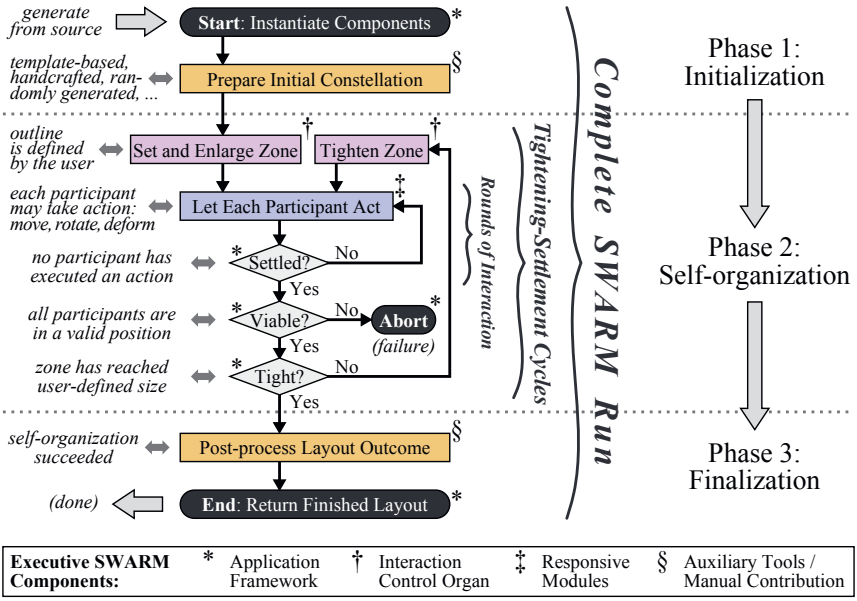


Figure 3.3: Flow of control in a SWARM run (initialization, self-organization, and finalization phase).

Figure 3.4 provides an exemplary depiction of a SWARM run. The example shows seven participants being successively goaded towards a compact arrangement inside a rectangular layout zone. This illustration brings a vivid biological analogon to mind, because SWARM's idea

of autonomous layout modules that perform areal movements across the plane of an increasingly tightened layout territory, imitates the roundup of livestock. For instance, a shepherd drives a group of sheep together just be encircling them (affecting the group like a predator – also see Figure 2.11), thereby leaving it up to every single animal to find its individual place among the herd. In SWARM, the participating layout modules can be thought of as sheep, with the supervising control organ taking on the role of the shepherd.

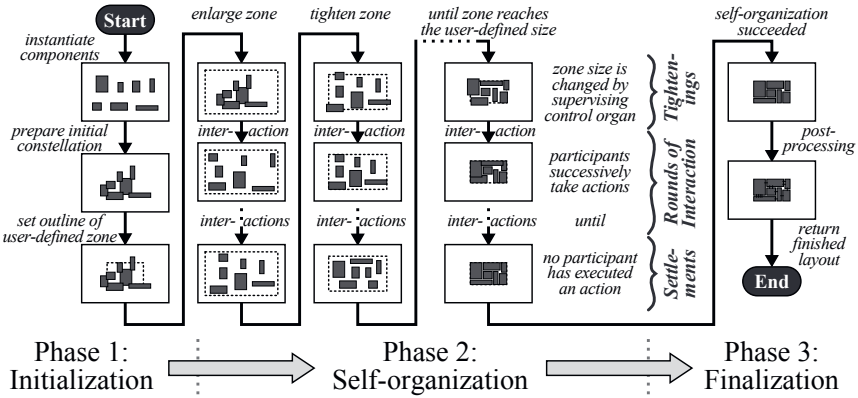


Figure 3.4: Exemplary depiction of a SWARM run, visualizing the analogy to the roundup of livestock.

During the module interaction, the contour of Z is employed as an explicit Fixed Outline (see Section 3.1.1.2 in [1]) constraint. In that regard, the outline of the layout zone represents a strict confinement, but with the approach of making Z progressively smaller, it is thus used to pursue a basic optimization goal: minimizing the total layout area. In a similar way, SWARM is also able to address the other major optimization goal commonly found in analog layout design: minimizing the total wirelength. This issue is not depicted in Figure 3.4, but will be broached in Section 3.3.1. Also not shown in the example above is the fact that a participant is not necessarily a single module, but may be an *association* of multiple components. This aspect is the central topic of the following Section 3.2 about the first core concept of SWARM.

3.2 Responsive Modules

The *responsive modules* in SWARM are realized as procedural generators, enhanced with a couple of additional capabilities: they can react to changes of their environment (Section 3.2.1), they can be used to administrate a group of design components (Section 3.2.2), they can be

imposed onto each other in a hierarchical fashion (Section 3.2.3), and they are able to assume a different layout variant from an exhaustive set of possible alternatives (Section 3.2.4).

3.2.1 Context Awareness

As already said, the layout modules in SWARM interact with each other as *agents*. This presupposes an ability to sense, reason, and act. Since a procedural generators is a piece of software, it can be readily implemented in a way that facilitates intelligent reasoning. But in their common form, procedural generators are conceptually not able to access their environment. Therefore, sensing and acting requires to enhance the conventional concept of procedural generators with a specific feature, subsequently referred to as *context awareness*. Context awareness allows an *instance* of a procedural generator (1) to *read* information from its design context and (2) to *modify* that context, including –self-referentially– the instance itself. With these abilities, the instance can realize a *responsive* design entity that reacts to environmental changes like a *reflex agent* does as shown in Figure 3.5 (a).

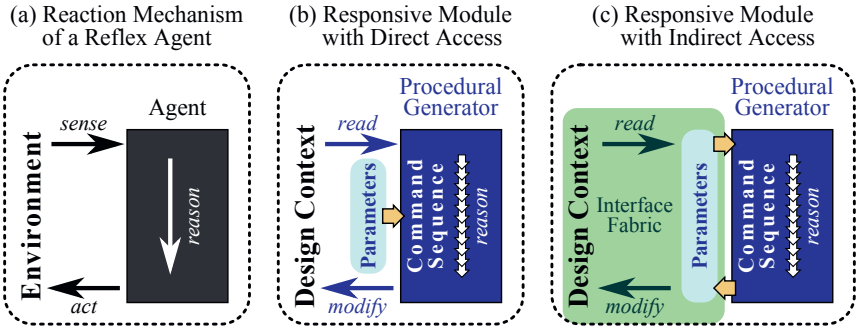


Figure 3.5: Via context awareness, procedural generators can react to environmental changes as agents.

The most straightforward way of implementing context awareness is to equip a procedural generator with *direct access* to its design context, as illustrated in Figure 3.5 (b). However, such an endeavor requires a fundamental extension to the IC design framework itself, which cannot be accomplished without additional efforts on the vendor's side. An initial implementation of that approach has led to the concept of *FR PCells* [153], but this concept is still in its infancy and no official release for productive use is available so far.

Therefore, as shown in Figure 3.5 (c), SWARM takes a different approach which facilitates *indirect access*, abusing a generator's parameters to communicate information between an instance and its design context, as also done in [154] and [155] (see Section 3.1.2.5 of [1]). For

that purpose, SWARM implements a dedicated *interface fabric* which reads data from the design context, encapsulates it in a single parameter value, and passes that value to the instance. After extracting and processing the context data, the instance also writes its response into a single value and stores it in a parameter (which is thus not used as an input parameter but as an output parameter). The interface fabric parses that response value and –if necessary– modifies the design context on behalf of the instance (as will be seen in Section 3.2.2). In contrast to a FR PCell, the instance must be actively re-evaluated if its design context is changed.

3.2.2 Governing Modules

Based on context awareness, a procedural generator for a *simple module* (in the sense of Table 2.1 from [1]) can be easily realized as a single (i.e., self-contained) responsive layout module. This is feasible, if a corresponding generator such as a *circuit PCell* is available on the schematic side. If instead, the schematic circuit is flat, i.e., comprised of *primitive devices*, then the *Generate from Source* step (see Section 2.1.4 in [1]) places corresponding devices in the layout.

One possibility to get to the module level on the layout side is presented in [156], which performs a circuit structure recognition in the schematic diagram to identify functional units and replace the respective devices in the layout with parameterized modules. A drawback of that technique is the problem to maintain SDL-conformity since the 1:1 device correspondence between schematic and layout has to be sacrificed for this purpose. A better method in that regard is made possible by the powers of context awareness, as it allows for using a responsive module in the layout to "manage" a group of devices. A responsive module of that kind is referred to as a *governing module*.

3.2.2.1 Temporary Context Duplication

If the context awareness of a governing module is implemented via indirect access, the need to gather a large regiment of detailed geometrical information (e.g., the individual pin positions, gate finger shapes, and bounding boxes of multiple transistors that need to be "managed") and encapsulate that information in a single parameter value may be cumbersome. A more elegant idea, which can be understood as *temporary context duplication* is presented in [157] (and is also pursued in this thesis): the module re-instantiates the devices internally so it can then determine all relevant geometries from these "clones" (which may be deleted afterwards). For that approach, the module requires only the type, the parameter values, the location, and the orientation of each external device. Based on that information, the utilization of a governing module generally follows an *adoption process* –written as a sequence $A = (Ab, Ad, Am, As)$– which is basically comprised of four operations (although not all of these operations are mandatory in every case):

(1) *Absorption*: "create an image of the outside" (symbol Ab)
The module reads the *external* device information (in the case of direct access) or receives it from the interface fabric (indirect access), and re-instantiates the devices internally.

(2) *Adaptation*: "measure and calculate inside" (symbol Ad)
The module determines all relevant device geometries and creates its *internal* layout as per these geometries, thus computing certain correcting quantities. The internal devices can then be deleted.

(3) *Amendment*: "modify the outside" (symbol Am)
The module transforms (moves and/or rotates) each *external* device according to the computed correcting quantities. If the context access is indirect, the transformation data is encapsulated in a parameter value and then put into effect by the interface fabric.

(4) *Assimilation*: "harmonize the inside with the outside" (symbol As)
The module repeats operations (1) and (2) in order to adjust its *internal* layout with respect to the new arrangement of its *external* devices (again, the internal devices may then be deleted). Now, the module and its context are supposed to be in perfect conformance with each other.

As an example, Figure 3.6 illustrates the utilization of a governing module which is based on the Differential Pair procedural generator from Figure 3.14 in [1]. Initially, there are four transistors in the layout, as column **Context** shows. In (1), the module absorbs these transistors and instantiates them internally. Column **Module** displays the inner layout of the module instance. In the design, the module with its internal transistors lies precisely on top of the external transistors, as can be seen in column **Design**.

Next (2), the module determines the pin positions of all internal transistors, connects them by creating wires and vias following a preconceived routing scheme –thereby also computing the correcting quantities mentioned above–, and then deletes the internal transistors such that only the *wiring* remains inside the module. Afterwards (3), the module pushes the external transistors together according to the correcting quantities computed before.[2] In that *positioning*, the correcting quantities reserve sufficient vertical space to accommodate the wiring between the transistors, which clarifies why the correcting quantities were calculated in conjunction with the previous creation of the wiring. Finally (4), the module generates its layout anew to harmonize the wiring with the modified design context. In the end, the external transistors and the wiring inside the module constitute an optimal Quad layout.

[2] In this example, the basic interdigitation of the transistors –specified by their initial relative locations (1)– is not changed through the amendment operation. Instead, the transistors are only compacted (according to a module-specific, preconceived, simple, row-based compaction scheme).

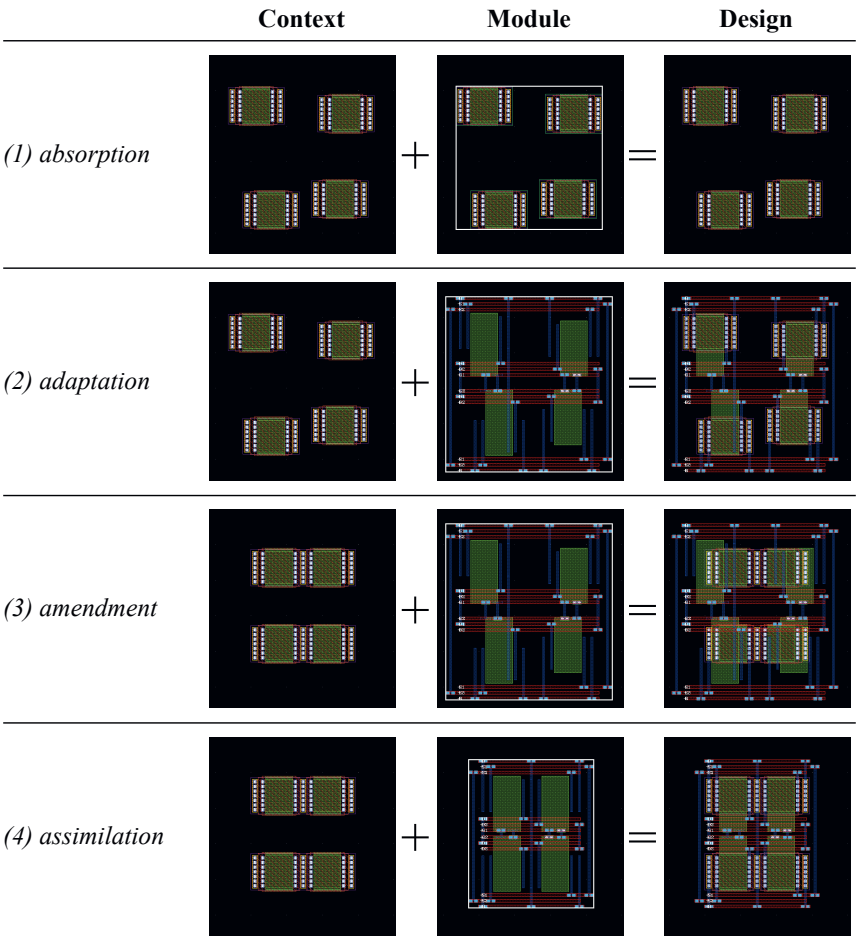


Figure 3.6: Adoption process of a governing module, exemplified for a Differential Pair (Quad layout).

3.2.2.2 Co-transformations in a Governing Module

With temporary context duplication, context awareness is not only utilized during the self-organization phase, but already in the initialization phase of a SWARM run: if the given schematic is flat, a governing module can be imposed on each group of devices that needs to be managed as a compound functional unit. From then on, it is the governing module's duty to care for these "children" it has adopted. That is to say, a transformation of the entire module during the self-organization is in fact a set of correlated transformations (*co-transformations*), which involves the transformation of the governing module itself and a corresponding transformation if its adopted children. So, if the governing module moves or rotates when interacting with other modules, it has to make sure that its children are also moved and rotated accordingly. To deform itself into another layout variant, the governing module repeats the adoption process and moves/rotates its children thereby. If the deformation includes a deformation of the children, the children are deformed first, and then the governing module repeats the adoption process.

Figure 3.7 gives an example of such a deformation, pertaining to the Differential Pair module above. Starting off with the previous module variant from Figure 3.6, the co-transformation of the adopted children consists of –first– deforming each of the four transistors from a 1-finger variant into a 2-finger variant (0), and –second– moving the transistors according to their new layout geometries, which is done in (1)–(3) as part of the adoption process. Then, the co-transformation of the governing module is to deform itself by assimilating itself in (4), which represents the last operation of the adoption process and sees the overall module layout turn into a 2-finger variant of the Quad.

3.2.3 Module Associations

Like in the examples of Figure 3.6 and Figure 3.7, it is usually the case that the positioning of a module's adopted devices strongly depends on the designated wiring. In that situation, it is suitable to perform both of these tasks with one single, *panfunctional* governing module, as it is done by the Differential Pair module above. If there is not such a strong dependence, two separate modules can be employed for the two different tasks.

Figure 3.8 shows a Current Mirror that involves three governing modules. One module is a pure Positioning module which does not create any physically relevant layout shapes and is therefore denoted as a *meta-module*. So, the adoption process A_{Pos} of that module includes only an absorption and an amendment operation (for which the correcting quantities are already computed during the absorption). This adoption process can be written as $A_{\text{Pos}} = (Ab_{\text{Pos}}, -, Am_{\text{Pos}}, -)$ to indicate that there is no adaptation and no assimilation involved (as is characteristic of a meta-module). Another module –referred to as a Wiring module– provides the connectivity. The third module is an Info module that writes device identifiers onto the

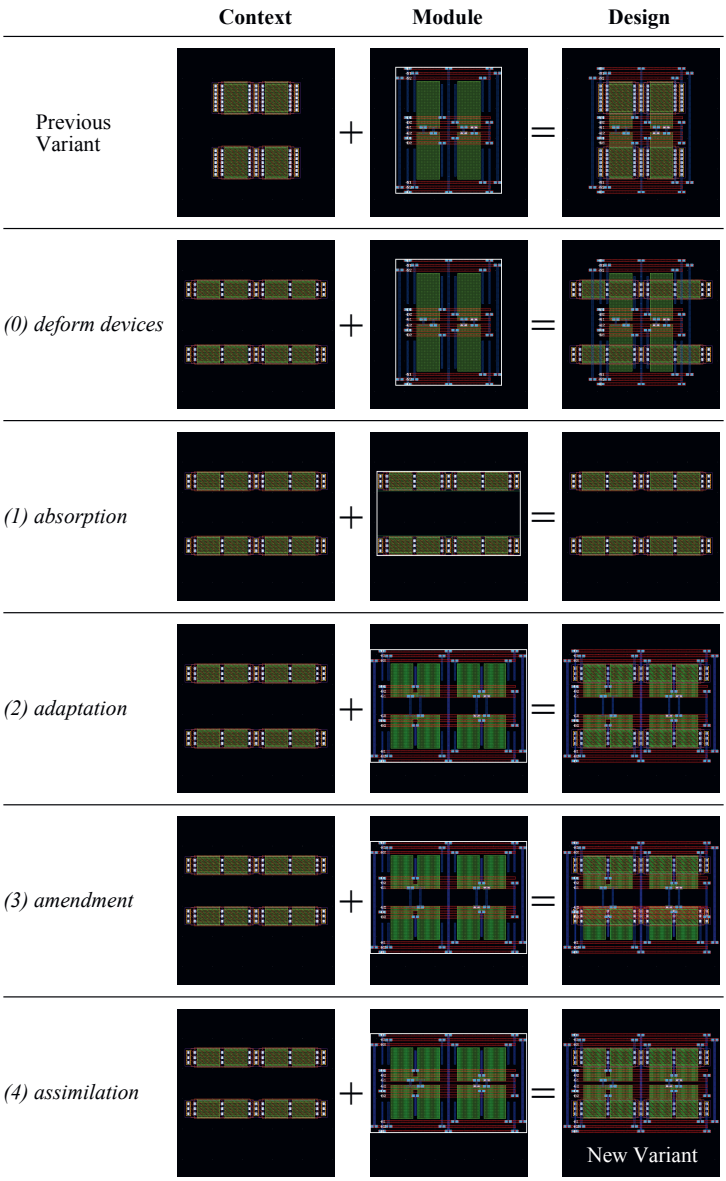


Figure 3.7: Exemplary deformation of a governing module, involving a set of co-transformations.

transistors to display how they have been interdigitated. The latter two modules do not perform amendments (which eliminates the need for assimilations), so their respective adoption process A_{Wir} and A_{Inf} only consists of an absorption and an adaptation operation (see Figure 3.8). The entire construct, for the reason of involving more than one governing module, is called a *module association*.

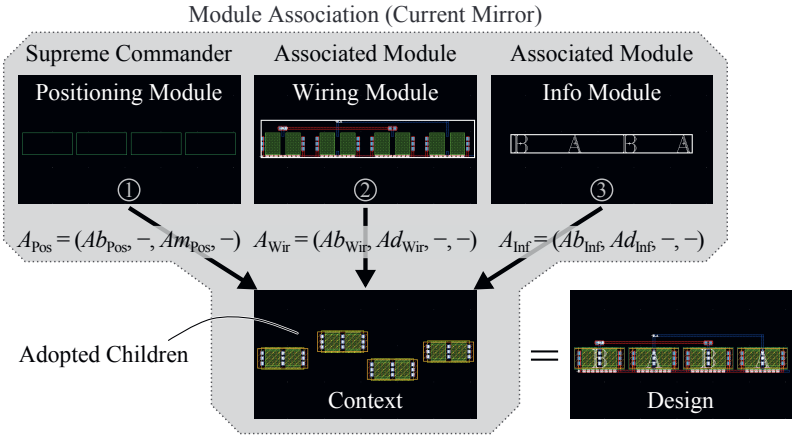


Figure 3.8: Example of a Current Mirror module association consisting of three governing modules.

3.2.3.1 Supreme Commanders

In every module association, one of its governing modules has to be implemented as a *supreme commander* which rules over the other modules (denoted as *associated modules*) and maintains the consistency of the entire construct. Thus, if the module association is about to perform a transformation, it is in the supreme commander's authority to apply the respective co-transformation to its adopted children and associated modules. If the transformation includes a deformation, the supreme commander also has to trigger the adoption process for each associated module once again. As it is the case in Figure 3.8, these are not supposed to perform amendments, but if they do, then they need to trigger the adoption process for the supreme commander anew (which again affects the entire module association). Attention must be paid in order to avoid an infinity loop here. This responsibility is left to the module developer, as well as the question which module is destined to act as supreme commander: normally, it is the module that performs the primary design task (as defined by the developer). If –instead of a module association– only one panfunctional governing module is involved, that module is by definition a supreme commander.

The Current Mirror layout from Figure 3.8 is reprised in Figure 3.9 to illustrate a deformation of the module association. In this example, the deformation of the module association is based on a rotation of its adopted children. Initially (a), the context of the module association is made up of the four aligned transistors from the design of the previous variant. To perform a deformation, the supreme commander –here: the Positioning module– first applies the appropriate co-transformation to its adopted children, which in this case means to rotate each device by $90°$. Then (b), the Positioning module adopts the devices anew (via A_{Pos}) to move them closer together and align them. This represents the co-transformation of the supreme commander, who furthermore cares for the co-transformation of its associated modules. That means, the Positioning module triggers their respective adoption process again (A_{Wir} and A_{Inf}), which leads to the new layout variant: a Current Mirror design whose transistors are rotated "upright", as opposed to the previous layout variant of the module association, where the transistor channels run "across" the design. These two transistor orientations will again be picked up in Section 3.2.4.2.

3.2.3.2 Hierarchical Module Associations

A module association may not only be comprised of primitive devices and governing modules, but can also incorporate other module associations. Thus, module associations can be hierarchically imposed onto one another to build larger design components, as Figure 3.10 illustrates with an example. Initially (a), there are two groups of transistors in the design, with seven devices per group. Next (b), two Current Mirrors are created from each group of transistors according to the adoption process known from Figure 3.8. As indicated in column **Module Hierarchy** (Figure 3.10 again), there are now two module associations, each of which contains three governing modules. All three of these governing modules adopt the seven transistors, with the Positioning module representing the supreme commander of its module association. In (c), the two module associations are adopted by the Positioning module of a Symmetric Current Mirror Pair (marked with an asterisk to distinguish it from the other Positioning modules), which has the purpose of keeping two Current Mirrors next to each other side by side.

The amendment operation Am^{*}_{Pos} of this adoption process A^{*}_{Pos} effectuates a movement and a rotation of the two Current Mirrors, which is applied to the two supreme commanders as their respective co-transformation. Each of the supreme commanders then also cares for the corresponding co-transformation of its adopted children and of its associated modules. Since the amendment operation does not include a deformation, the entire set of co-transformations involves only translational moves and rotations (such that –in contrast to Figure 3.9– the transistors on device level need not be adopted once again by the governing modules in the module association). The obtained layout is a symmetric arrangement of the two Current Mirrors as shown in column **Design** of Figure 3.10.

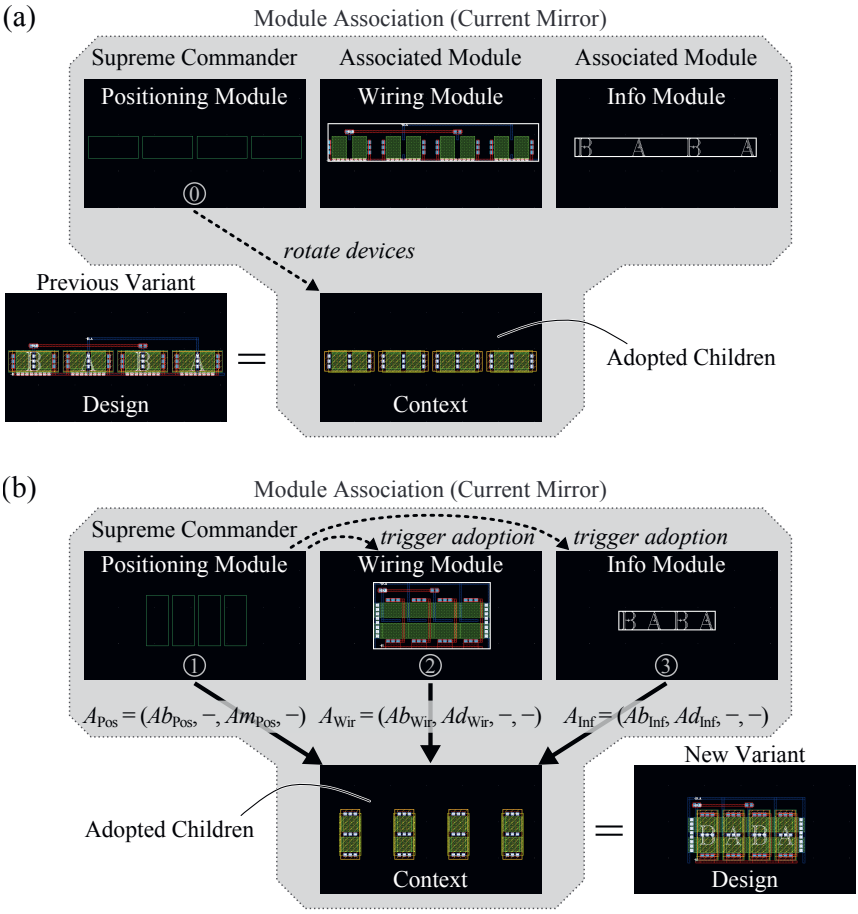


Figure 3.9: Exemplary deformation of a module association via (a) device rotation and (b) re-adoption.

3.2.3.3 Co-transformations in a Module Association

Table 3.2 lists the possible transformations that a module association can undertake by performing a *movement* M, a *rotation* R, and/or a *deformation* D. These transformation elements are subsequently referred to as *morphisms* Φ. Further shown in the table are the involved co-transformations that the adopted children (indexed I), the supreme commander (indexed II), and the associated modules (indexed III) need to undergo. As should be clear from the discussion so

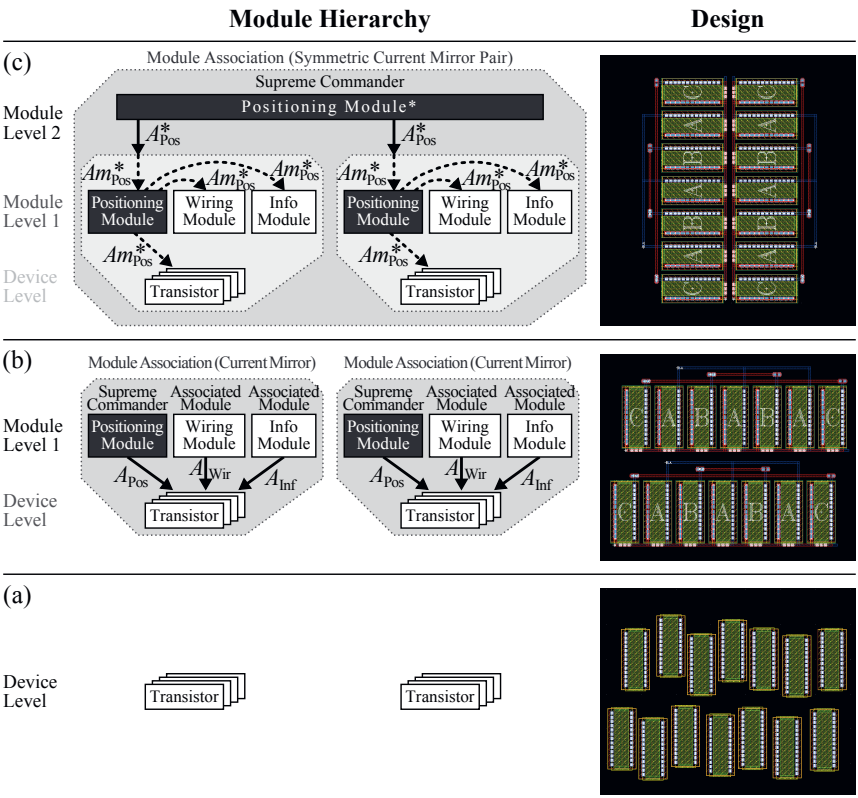


Figure 3.10: Example of a hierarchical module association with (a) 14 transistors on device level, which are adopted by (b) two Current Mirror module associations, which are in turn adopted by a (c) Symmetric Current Mirror Pair.

far, each of these co-transformations can also consist of different morphisms. To mark the order, in which the individual morphisms are performed, two symbols are introduced. A diamond $\diamond$ indicates that two morphisms are commutative, i.e., a morphism Φ_a can precede or succeed another morphism Φ_b. In contrast, a triangle $\triangleright$ denotes a noncommutative order: $\Phi_a \triangleright \Phi_b$ says that Φ_a cannot succeed but only precede Φ_b.

It it important to note that each rotation R in a co-transformation has the same pivotal point as the rotation of the module association. This is to say that the components of the module association are not rotated around their individual points of origin. Instead, the pivotal point is always the center point of the entire module association. By the same token, that center point

Table 3.2: Possible transformations of a module association with corresponding co-transformations. Items in square brackets are optional morphisms, while A denotes a re-adoption. A diamond $\diamond$ denotes commutativity, whereas a triangle $\triangleright$ signifies noncommutativity.

Transformation of the Module Association			**Co-transformations**		
Movement	Rotation	Deformation	Adopted Children (I)	Supreme Commander (II)	Associated Modules (III)
M	-	-	$M_I \diamond M_{II} \diamond M_{III}$		
-	R	-	$R_I \diamond R_{II} \diamond R_{III}$		
M	R	-	$(M_I \diamond R_I) \diamond (M_{II} \diamond R_{II}) \diamond (M_{III} \diamond R_{III})$		
-	-	D	$(([R'_I] \diamond [D'_I]) \triangleright M'_I) \triangleright A_{II} \triangleright A_{III}$		
M	-	D	$(M_I \diamond (([R'_I] \diamond [D'_I]) \triangleright M'_I)) \triangleright A_{II} \triangleright A_{III}$		
-	R	D	$(R_I \diamond (([R'_I] \diamond [D'_I]) \triangleright M'_I)) \triangleright (R_{II} \triangleright A_{II}) \triangleright (R_{III} \triangleright A_{III})$		
M	R	D	$(M_I \diamond R_I \diamond (([R'_I] \diamond [D'_I]) \triangleright M'_I)) \triangleright (R_{II} \triangleright A_{II}) \triangleright (R_{III} \triangleright A_{III})$		

is also supposed to be identical before and after a deformation D. Items in square brackets represent an optional morphism, while A denotes a re-adoption for the purpose of deforming the module association. Deformations are also reflected by the appearance of M', R', and D': these are morphisms of the adopted children that serve the deformation of the module association, as was exemplified in Figure 3.9. So, a movement M' of the adopted children is not equal to the movement M of the module association.

At first glance, it might be surprising to see that in some cases a movement M of the module association leads to a corresponding movement M_{II} of the supreme commander and a corresponding movement M_{III} of its associated modules, while in other cases it does not. The latter can be observed in those cases where the transformation of the module association also contains a deformation D. It is in those cases, that the supreme commander and its associated modules perform a re-adoption A_{II} and A_{III} to account for the deformation. And since this re-adoption occurs *after* movement M has already been applied to the children that are to be adopted, the new locations of these children immanently manifest themselves in the supreme commander and the associated modules without the need to perform explicit movements M_{II} and M_{III}.

Using the notation $\Phi_a \succ \Phi_b$ to express that morphism Φ_a is followed by morphism Φ_b, the deformation of the Differential Pair module in Section 3.2.2.2 and the deformation of the Current Mirror module in Section 3.2.3.1 can be formally expressed in the style of Table 3.2. Concerning the first of these two examples, the visualization in Figure 3.7 helps to see which co-transformations are involved in the deformation of the Differential Pair:

$$D'_I \succ M'_I \succ A_{II}. \tag{3.1}$$

Since the Differential Pair is realized with a single governing module, it does not feature any associated modules and therefore the deformation does not involve a re-adoption A_{III}. Regarding the second example, the deformation of the Current Mirror in Figure 3.9 can be expressed as:

$$R'_I \succ M'_I \succ A_{II} \succ A_{III}. \tag{3.2}$$

In both examples, M'_I is in fact performed by the amendment operation of the supreme commander's adoption process, while the important aspect about A_{II} is its assimilation operation. In the latter example though, A_{II} is moot because the supreme commander is a pure Positioning module (i.e., a meta-module). Still, both expressions are in line with the sequence of co-transformations articulated in Table 3.2.

3.2.3.4 Coordinate System Issues

The preceding ruminations reveal that amendments do not only play an important role during the initialization phase of a SWARM run, but whenever a module deforms itself into another layout variant. Now, one should remember that for every amendment, the respective modifications are determined *inside* the module but are applied to its *outside* design context. On that account, one particular issue has to be kept in mind: the internal coordinate system of a module instance is not necessarily equal to the coordinate system of its context (unless the instance's point of origin is $(0, 0)$ and the instance is not rotated). For that reason, some detailed thoughts need to be spent on the adoption process, and –first of all– on the absorption operation therein. As already mentioned in the beginning of Section 3.2.2, the absorption operation requires the type, the parameter values, the location, and the orientation of each external component. The former two items can be neglected for the following discussion, because only the location and the orientation depend on the coordinate system.

The *location* L of a design component is subsequently expressed as a pair of coordinates $L = (x, y)$, where x denotes the horizontal coordinate and y denotes the vertical coordinate of the location. Regarding the *orientation* O of a design component, it must be acknowledged that the component may not only be rotated by a certain angle but that it can also be flipped around a certain axis. As illustrated in Figure 3.11, this results in a total of eight possible orientations each of which can be formally denoted via $O_{\bar{h}} = (\alpha, \bar{h})$. The *angle* α of a component's orientation is to be interpreted counterclockwise and can assume one of four possible values: $\alpha \in \{0°, 90°, 180°, 270°\}$. Flipping can occur horizontally (around a vertical axis) or vertically (around a horizontal axis). As Figure 3.11 shows, the value for $\bar{h}$ indicates *horizontal flipping*, with $\bar{h} \in \{0, 1\}$. Value $\bar{h} = 1$ means that the design component is horizontally flipped, whereas $\bar{h} = 0$ signifies that the component is not flipped. Vertical flipping does not provide additional orientations because any orientation that involves a vertical flip, can also be obtained with an appropriate rotation and a horizontal flip. The equivalence between the *horizontal no-*

tation $O_{\bar{h}} = (\alpha, \bar{h})$ for orientations based on horizontal flipping, and the *vertical notation* $O_{\upsilon} = (\alpha, \upsilon)$ for orientations based on *vertical flipping* υ, is given in Table 3.3.

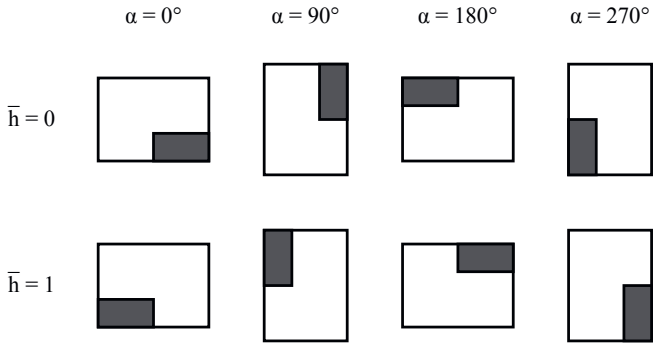


Figure 3.11: Eight orientations of a design component, depending on angle α and horizontal flipping $\bar{h}$.

Table 3.3: Equivalence between horizontal notation and vertical notation for a component's orientation.

Horizontal Notation $O_{\bar{h}} = (\alpha, \bar{h})$	**Vertical Notation** $O_{\upsilon} = (\alpha, \upsilon)$
$(0°, 1)$	$(180°, 1)$
$(90°, 1)$	$(270°, 1)$
$(180°, 1)$	$(0°, 1)$
$(270°, 1)$	$(90°, 1)$

In the initialization phase of a SWARM run, the absorption operation of a module instance works straightforward because the module can be instantiated in default orientation, i.e., with $O_{\bar{h}} = (0°, 0)$. For every component that is to be absorbed, the following location coordinates x_i and y_i, as well as the orientation specifiers α_i and $\bar{h}_i$ must be passed to the module instance for internal context duplication:

$$x_i = x_A - x_P, \tag{3.3a}$$

$$y_i = y_A - y_P, \tag{3.3b}$$

$$\alpha_i = \alpha_A, \tag{3.3c}$$

$$\bar{h}_i = \bar{h}_A, \tag{3.3d}$$

where x_A and y_A are the coordinates of the component to Absorb, while x_P and y_P are the coordinates of the Procedural module instance, with α_A and $\bar{h}_A$ representing the orientation specifiers of the component to Absorb. During the self-organization phase of a SWARM run, the orientation of a module instance can become $O_{\bar{h}} \neq (0°, 0)$, which must be taken into account by the absorption operation. This is achieved by transforming the location coordinates and the orientation specifiers converse to the orientation of the module instance, as will now be explained in greater detail.

Regarding the location coordinates, these need to be rotated by $-\alpha_P$ in order to compensate the angle α_P in the orientation of the module instance. From planar trigonometry, the rotation of a point (x, y) around an angle α can be calculated via:

$$x' = x \cdot \cos(\alpha) - y \cdot \sin(\alpha), \tag{3.4a}$$

$$y' = x \cdot \sin(\alpha) + y \cdot \cos(\alpha). \tag{3.4b}$$

For the absorption operation, this rotation must proceed around the origin of the module instance. This means, that $x = x_A - x_P$ and $y = y_A - y_P$. To take flipping into account, the location coordinates must first be flipped in reverse before the rotation is applied. In this thesis, the possible orientations of a design component (see Figure 3.11) are expressed via the notation $(\alpha, \bar{h})$, which means that the component is ***first*** rotated by α and ***then*** horizontally flipped in case $\bar{h} = 1$. Concerning the absorption operation, this is why the reverse flipping of the location coordinates must be performed ***prior*** to the reverse rotation.[3] This can be conveniently achieved with the term $1 - 2\bar{h}_P$, which evaluates either to 1 (in case $\bar{h}_P = 0$) or to -1 (in case $\bar{h}_P = 1$) and can thus be used to negate (or not negate) the horizontal coordinate $x_A - x_P$. The full formulas for the calculation of x_i and y_i are shown in equation 3.6.

Regarding the orientation specifiers $(\alpha_A, \bar{h}_A)$ of the component to absorb, these must –like the location coordinates– also be "reversed" to equalize the orientation of the module instance. For each possible component orientation $(\alpha_A, \bar{h}_A)$ and for all possible module orientations $(\alpha_P, \bar{h}_P)$, Table 3.4 lists the reversed component orientation $(\alpha_i, \bar{h}_i)$ that must be passed to the module instance for internal context duplication.

As can be seen in that table, the component to absorb has to be rotated into a flipped orientation if and only if either the component is in a flipped orientation or if the module instance is in a flipped orientation. This relation is an exclusive or (XOR), which in Boolean logic means that $\bar{h}_i = \bar{h}_A \oplus \bar{h}_P$. In arithmetic terms, this can be expressed as:

$$\bar{h}_i = 1 - |\bar{h}_A + \bar{h}_P - 1|. \tag{3.5}$$

Equivalent to the reverse rotation of the component location (x_A, y_A) (see above), the angle α_A of the component has to be rotated by $-\alpha_P$. Intuitively, this is expected to be achievable

[3] The order "first rotate, then flip" in the notation $O_{\bar{h}} = (\alpha, \bar{h})$ is just a question of definition. It could just as well be defined vice-versa, for both horizontal flipping and vertical flipping. However, it is important that this order is respected by the absorption operation.

Table 3.4: Reversed orientation of a component that is to be absorbed by a rotated governing module.

	reversed component orientation $(\alpha_i, \bar{h}_i)$ **for all possible module orientations** $(\alpha_P, \bar{h}_P)$							
$(\alpha_A, \bar{h}_A)$	$(0°, 0)$	$(90°, 0)$	$(180°, 0)$	$(270°, 0)$	$(0°, 1)$	$(90°, 1)$	$(180°, 1)$	$(270°, 1)$
$(0°, 0)$	$(0°, 0)$	$(270°, 0)$	$(180°, 0)$	$(90°, 0)$	$(0°, 1)$	$(90°, 1)$	$(180°, 1)$	$(270°, 1)$
$(90°, 0)$	$(90°, 0)$	$(0°, 0)$	$(270°, 0)$	$(180°, 0)$	$(90°, 1)$	$(180°, 1)$	$(270°, 1)$	$(0°, 1)$
$(180°, 0)$	$(180°, 0)$	$(90°, 0)$	$(0°, 0)$	$(270°, 0)$	$(180°, 1)$	$(270°, 1)$	$(0°, 1)$	$(90°, 1)$
$(270°, 0)$	$(270°, 0)$	$(180°, 0)$	$(90°, 0)$	$(0°, 0)$	$(270°, 1)$	$(0°, 1)$	$(90°, 1)$	$(180°, 1)$
$(0°, 1)$	$(0°, 1)$	$(90°, 1)$	$(180°, 1)$	$(270°, 1)$	$(0°, 0)$	$(270°, 0)$	$(180°, 0)$	$(90°, 0)$
$(90°, 1)$	$(90°, 1)$	$(180°, 1)$	$(270°, 1)$	$(0°, 1)$	$(90°, 0)$	$(0°, 0)$	$(270°, 0)$	$(180°, 0)$
$(180°, 1)$	$(180°, 1)$	$(270°, 1)$	$(0°, 1)$	$(90°, 1)$	$(180°, 0)$	$(90°, 0)$	$(0°, 0)$	$(270°, 0)$
$(270°, 1)$	$(270°, 1)$	$(0°, 1)$	$(90°, 1)$	$(180°, 1)$	$(270°, 0)$	$(180°, 0)$	$(90°, 0)$	$(0°, 0)$

by simply calculating $\alpha_i = \alpha_A - \alpha_P$. But, as Table 3.4 shows (see top-left and bottom-right quadrants), this is only true in case $\bar{h}_i = 0$. If $\bar{h}_i = 1$ (see top-right and bottom-left quadrants, marked gray) then the reversed angle must be calculated as $\alpha_i = \alpha_A + \alpha_P$. The reason for this subtlety is illustrated in Figure 3.12: a positive rotation of a component that is in a regular (i.e., not flipped) orientation, entails an *increase* of the component's angle (a), but rotating a component in flipped orientation causes a *decrease* of the angle (b). The correct sign can be included by putting a factor before α_P. That factor is calculated as $2\bar{h}_i - 1$, which evaluates to $1 - 2 \cdot |\bar{h}_A + \bar{h}_P - 1|$.

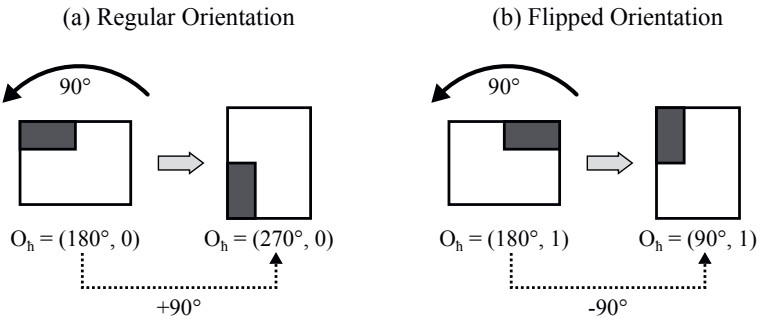


Figure 3.12: Angle change of a rotated component in (a) regular orientation and (b) flipped orientation.

Putting it all together, the advanced expressions for calculating the location coordinates and orientation specifiers from equation 3.3 can now be written in closed form as follows:

$$x_i = (1 - 2\bar{h}_P) \cdot (x_A - x_P) \cdot \cos(-\alpha_P) - (y_A - y_P) \cdot \sin(-\alpha_P), \tag{3.6a}$$

$$y_i = (1 - 2\bar{h}_P) \cdot (x_A - x_P) \cdot \sin(-\alpha_P) + (y_A - y_P) \cdot \cos(-\alpha_P), \tag{3.6b}$$

$$\alpha_i = \alpha_A + (1 - 2 \cdot |\bar{h}_A + \bar{h}_P - 1|) \cdot \alpha_P, \tag{3.6c}$$

$$\bar{h}_i = 1 - |\bar{h}_A + \bar{h}_P - 1|. \tag{3.6d}$$

With these formulas, the expressions from equation 3.3 become obsolete, because the trivial case that the module instance is in default orientation $(0°, 0)$ is of course also covered by the calculations in equation 3.6: setting $\alpha_P = 0°$ and $\bar{h}_P = 0$ in equation 3.6 leads to the expressions given in equation 3.3. Thus, the formulas of equation 3.6 are valid for a module's adoption process in the initialization phase of a SWARM run, and also when the module rotates and deforms itself during the self-organization phase. Pertaining to the initial remark of this Section 3.2.3.4, the coordinate system issues that have been discussed above with regard to *reading* a module's design context, must equivalently be taken into account when the module is *modifying* its design context.

3.2.4 Layout Variability

With the ability to deform itself into a different layout variant, a governing module can cover various aspect ratios without altering its nominal electrical behavior. For that purpose, a governing module must –in addition to the layout operations discussed in Section 3.2.2 and Section 3.2.3– first of all be able to *tell* the respective degrees of freedom for the circuit it implements. These degrees of freedom define the *variability* $\mathcal{V}$ of the module, i.e., the set of all feasible layout variants that the module can assume. Depending on the degrees of freedom, a distinction can be drawn between *discrete variability*, where the set of assumable layout variants is limited, and *full variability*, where the module is able to change its dimensions in a continuous range.

SWARM supports both kinds of variability. However, if a module features full variability, possible deformations cannot be set out in advance but are determined ad hoc during the module interaction. For that reason, the topic of full variability will be deferred to Section 3.3. Thus, the subsequent discussion concentrates on discrete variability, which is the prevalent kind of variability on the level of *simple modules*.

3.2.4.1 Intrinsic Variability

Every design component has an *intrinsic variability* $\acute{\mathcal{V}}$. For example, in the case of a parameterized MOS transistor (referred to as type `MT`), the degree of freedom for potential deformations

is its *number of fingers* (see Figure 2.1 in [1]), here denoted as $\mathtt{f}$. Formally, the possible values for $\mathtt{f}$ are given by $\mathbb{D}_{\mathtt{f}}$, which represents the *domain* of that parameter. In the particular case of $\mathtt{f}$, which is defined as an integer parameter, this domain is in general a subset of the set $\mathbb{N}^+$ of positive natural numbers: $\mathbb{D}_{\mathtt{f}} \subset \mathbb{N}^+$. For a particular instance ***MT*** of the transistor, the domain is $\mathbb{D}_{MT,\,\mathtt{f}} = \{1, 2, \ldots, \mathtt{f}_{\text{max}}\}$, where the parameter value $\mathtt{f}_{\text{max}}$ for the maximum number of fingers depends on the transistor's total channel width.

Simply put, $\mathbb{D}_{MT,\,\mathtt{f}}$ defines the intrinsic variability $\acute{\mathcal{V}}_{MT}$ of the transistor instance. More formally, the intrinsic variability can be expressed as a function $g_{\mathtt{MT}}$ of $\mathbb{D}_{MT,\,\mathtt{f}}$ where $g_{\mathtt{MT}}$ represents the behavior[4] of the transistor generator:

$$\acute{\mathcal{V}}_{MT} = g_{\mathtt{MT}}\big(\mathbb{D}_{MT,\,\mathtt{f}}\big). \tag{3.7}$$

As an alternative notation, the intrinsic variability is henceforth written as a mapping from the domain of a parameter to the set of corresponding layout variants. In the case of the MOS transistor instance, the mapping is thus written as:

$$\acute{\mathcal{V}}_{MT} \Leftarrow \mathbb{D}_{MT,\,\mathtt{f}}. \tag{3.8}$$

The use of this notation is encouraged due to its compactness. This becomes even more evident when multiple parameters are involved, as the following Section 3.2.4.2 is about to show.

3.2.4.2 Cumulative Variability

If transistors are adopted by a governing module, the intrinsic variability of the transistors contributes to the variability of the module. An example is the Differential Pair module in Figure 3.7, which deforms itself from a 1-finger variant into a 2-finger variant by incrementing the number of fingers in each of its transistors. However, a module may further provide its own intrinsic variability. Regarding the deformation in Figure 3.9, the Current Mirror module association deforms itself into a different variant without modifying the layout of its adopted transistors, but simply by rotating them. So, the orientation of these transistors, a parameter here denoted as $\mathtt{o}$ is one degree of freedom of the Current Mirror $\mathtt{CM}$ with the domain $\mathbb{D}_{CM,\,\mathtt{o}} = \{across, upright\}$. Another degree of freedom is given by the possibility to break the positioning from the depicted single-row variant into a dual-row variant. Hence, the domain of this positioning parameter $\mathtt{p}$ is $\mathbb{D}_{CM,\,\mathtt{p}} = \{single, dual\}$. With these degrees of freedom, the intrinsic variability $\acute{\mathcal{V}}_{CM}$ of a Current Mirror instance ***CM*** is defined by the Cartesian product $\mathbb{D}_{CM,\,\mathtt{o}} \times \mathbb{D}_{CM,\,\mathtt{p}}$.

Apart from this, the Current Mirror can also change the number of fingers in its adopted transistors. Thus, the total variability of the module instance, referred to as its *cumulative*

[4]The behavior $g_{\mathtt{MT}}$ corresponds to the behavior g mentioned in Section 3.1.2.1 of [1], which is an "introversive" generator behavior and should not be mistaken for the "extroversive" interaction behavior that will be subject to Section 3.3.

variability $\widetilde{\mathcal{V}}$, is given by $\mathbb{D}_{CM,\circ} \times \mathbb{D}_{CM,\mathtt{p}} \times \mathbb{D}_{MT,\mathtt{f}}$. In general terms, the *cumulative variability* $\widetilde{\mathcal{V}}_P$ of a Parameterized design component P is the product of its own *intrinsic variability* $\acute{\mathcal{V}}_P$ and the *cumulative variability* $\widetilde{\mathcal{V}}_A$ of its Adopted children:

$$\widetilde{\mathcal{V}}_P = \acute{\mathcal{V}}_P \times \widetilde{\mathcal{V}}_A. \tag{3.9}$$

Regarding the Current Mirror module above, please note that $\widetilde{\mathcal{V}}_A = \acute{\mathcal{V}}_A$ because the cumulative variability of a transistor is identical to its intrinsic variability: $\widetilde{\mathcal{V}}_{MT} = \acute{\mathcal{V}}_{MT}$. This equation holds true for all primitive devices (since they do not have a subhierarchy) and allows to calculate a module's cumulative variability from the bottom up. On that basis, inserting $\widetilde{\mathcal{V}}_P$ for $\widetilde{\mathcal{V}}_A$ on the next higher module level is the key to determine the cumulative variability of a hierarchical module association. For example, the two Current Mirrors in a Symmetric Current Mirror Pair module `SCMP` (as in Section 3.2.3) can either be positioned in their default orientation (see Figure 3.10 (b)) or be rotated (as done in Figure 3.10 (c)). Thus, comparable to the transistors in a Current Mirror module, this orientation $\circ$ of the Current Mirrors in a Symmetric Current Mirror Pair is a degree of freedom with $\mathbb{D}_{SCMP,\circ} = \{across, upright\}$, which defines the intrinsic variability $\acute{\mathcal{V}}_{SCMP}$. Then, the cumulative variability $\widetilde{\mathcal{V}}_{SCMP}$ of an instance ***SCMP*** evaluates to

$$\widetilde{\mathcal{V}}_{SCMP} \tag{3.10}$$

$$= \overbrace{\acute{\mathcal{V}}_{SCMP} \times \quad \widetilde{\mathcal{V}}_{CM}} \tag{3.11}$$

$$= \acute{\mathcal{V}}_{SCMP} \times \overbrace{\acute{\mathcal{V}}_{CM} \times \widetilde{\mathcal{V}}_{MT}} \tag{3.12}$$

$$= \acute{\mathcal{V}}_{SCMP} \times \acute{\mathcal{V}}_{CM} \times \overbrace{\acute{\mathcal{V}}_{MT}} \tag{3.13}$$

and can be obtained from

$$\widetilde{\mathcal{V}}_{SCMP} = \overbrace{g_{\mathtt{SCMP}}\left(\mathbb{D}_{SCMP,\circ}\right)}^{\acute{\mathcal{V}}_{SCMP}} \times \overbrace{g_{\mathtt{CM}}\left(\mathbb{D}_{CM,\circ}\right) \times g_{\mathtt{CM}}\left(\mathbb{D}_{CM,\mathtt{p}}\right)}^{\acute{\mathcal{V}}_{CM}} \times \overbrace{g_{\mathtt{MT}}\left(\mathbb{D}_{MT,\mathtt{f}}\right)}^{\acute{\mathcal{V}}_{MT}} \tag{3.14}$$

which can be expressed more compactly using the notation introduced in equation 3.8:

$$\widetilde{\mathcal{V}}_{SCMP} \Leftarrow \mathbb{D}_{SCMP,\circ} \times \mathbb{D}_{CM,\circ} \times \mathbb{D}_{CM,\mathtt{p}} \times \mathbb{D}_{MT,\mathtt{f}}. \tag{3.15}$$

Based on this calculation principle, all governing modules and module associations are able to determine the total variability that they may exploit to perform deformations. This is a crucial qualification for Section 3.3, because the more exhaustive the variability of a module is, the more choices of action the module will have during a self-organization run.

3.2.4.3 Variability of Primitive Devices

Talking about the self-organization in SWARM, one remark should be made about primitive devices. Primitive devices are *native* devices that come with the semiconductor technology as

part of the PDK. Thus, they are not responsive and are therefore not able to perform actions by themselves, nor are they meant to be capable of answering questions about their variability. To remedy these deficiencies, a *variator module* can be employed. A variator module is a meta-module with the sole purpose to adopt and manage a single, primitive device instance. To perform deformations, it is the variator module's job to determine the variability of that instance.

As should become apparent from the discussion above, the variability of a primitive device is (in contrast to that of the modules introduced so far) usually instance-specific.[5] In the case of a MOS transistor, as already mentioned in Section 3.2.4.1, the variability is defined by $\acute{\mathcal{V}}_{MT} \Leftarrow \mathbb{D}_{MT,\,\mathfrak{f}}$ with

$$\mathbb{D}_{MT,\,\mathfrak{f}} = \{1, 2, \ldots, \mathfrak{f}_{\max}\} = \{1, 2, \ldots, \lfloor \tfrac{\mathfrak{w}_{\mathrm{ch}}}{\mathfrak{w}_{\min}} \rfloor\}, \tag{3.16}$$

where $\mathfrak{w}_{\mathrm{ch}}$ is the total channel width of the transistor instance and $\mathfrak{w}_{\min}$ is the minimum channel width (and thus also the minimum finger width) for that type of transistor.

3.2.4.4 Variability of Simple Modules

To explicate the variability of common, analog basic circuits that belong to the level of *simple modules*, it is apposite to expose which of these circuits have been implemented in SWARM, and what components these circuits are made up of in this implementation.

Showing device identifiers as done by the Current Mirror in Figure 3.8 is a largely circuit-independent thing to do. Similarly, the style of device positioning is quite comparable among several circuits, and even the wiring in these circuits shares a lot of common ground. For that reason, it is feasible to address these three tasks by implementing three generic governing modules, as seen in the bottom rows of Table 3.5. The Info module `I` is so generic, that it can be employed as-is for every basic circuit which allows for custom device interdigitation. The Positioning module `P` and the Wiring module `W` provide parameters for switching between slightly different positioning and wiring behaviors in order to cover several circuit types as well. Thus, the Wiring module `W` and the Wiring modules $\mathtt{W}_{\mathrm{CM}}$, $\mathtt{W}_{\mathrm{CS}}$, and $\mathtt{W}_{\mathrm{WS}}$ (which target different types of Current Mirror circuits) are implemented as one single procedural generator, providing a parameter to choose the respective topology. The situation is the same with the Positioning module `P` and its offshoots $\mathtt{P}_{\mathrm{BK}}$ (which implements a so-called current mirror Bank), $\mathtt{P}_{\mathrm{CS}}$ (which represents a Cascode module), and $\mathtt{P}_{\mathrm{SP}}$ (which realizes a Symmetric Pair module). The Differential Pair module $\mathtt{PW}_{\mathrm{DP}}$ is a panfunctional module since it performs both positioning and wiring.

Table 3.6 lists the analog basic circuits that have been implemented in SWARM utilizing the governing modules presented in Table 3.5. Also included is the MOS Transistor primitive

[5] However, it may also be the case with a module that it has to concede an instance-specific curtailment in its variability, e.g., to satisfy a certain design constraint.

Table 3.5: Generic governing modules and their topological offshoots, as implemented in SWARM.

Module	Purpose	Comment	Based on
$\mathtt{W}_{\mathrm{WS}}$	wiring	for Wide-swing Current Mirror	W
$\mathtt{W}_{\mathrm{CS}}$	wiring	for Cascode Current Mirror	W
$\mathtt{W}_{\mathrm{CM}}$	wiring	for Current Mirror	W
$\mathtt{P}_{\mathrm{SP}}$	positioning	Symmetric Pair module	P
$\mathtt{P}_{\mathrm{CS}}$	positioning	Cascode module	P
$\mathtt{P}_{\mathrm{BK}}$	positioning	Bank module	P
$\mathtt{PW}_{\mathrm{DP}}$	positioning and wiring	for Differential Pair	–
I	show interdigitation	generic Info module	–
W	wiring	generic Wiring module	–
P	positioning	generic Positioning module	–

device, as well as two kinds of circuits that have not been discussed so far: the Cascode Current Mirror CCM and the Wide-swing Current Mirror WCM. Layout examples of these circuits are given by the module instances in Figure 3.13 and Figure 3.14 respectively.

Table 3.6: Analog basic circuits covered by SWARM, based on its implemented governing modules.

Circuit	Abbr.	Components	Ductility	Examples
Symmetric Current Mirror Pair	SCMP	$\mathtt{CM} + \mathtt{P}_{\mathrm{SP}}$	$8 \cdot \mathfrak{f}_{max}$	Figure 3.10
Wide-swing Current Mirror	WCM	$\mathtt{MT} + \mathtt{P}_{\mathrm{BK}} + \mathtt{P}_{\mathrm{CS}} + \mathtt{W}_{\mathrm{WS}}$	$2 \cdot \mathfrak{f}_{max}$	Figure 3.14
Cascode Current Mirror	CCM	$\mathtt{MT} + \mathtt{P}_{\mathrm{BK}} + \mathtt{P}_{\mathrm{CS}} + \mathtt{W}_{\mathrm{CS}}$	$2 \cdot \mathfrak{f}_{max}$	Figure 3.13
Current Mirror	CM	$\mathtt{MT} + \mathtt{P}_{\mathrm{BK}} + \mathtt{W}_{\mathrm{CM}}$	$4 \cdot \mathfrak{f}_{max}$	Figure 3.9
Differential Pair	DP	$\mathtt{MT} + \mathtt{PW}_{\mathrm{DP}}$	$2 \cdot \mathfrak{f}_{max}$	Figure 3.7
MOS Transistor	MT	–	$\mathfrak{f}_{max}$	Figure 2.1 [1]

For every kind of circuit listed in Table 3.6, column **Components** shows which types of components (governing modules and primitive devices) the circuit has been realized with. Also given for each circuit is the total number of its possible deformation variants, denoted as its *ductility* d. For a circuit realized as a module (or module association) P, the module's ductility d_P is simply the mathematical cardinality of the module's cumulative variability:

$$d_P = |\widetilde{\mathcal{V}}_P|. \tag{3.17}$$

If $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ is a set denoting the types of components that a circuit is realized with, and if (for every P in $\mathcal{P}$) the set $\mathcal{I}_P = \{I_{P,1}, I_{P,2}, \ldots, I_{P,n}\}$ denotes the input parameters that can be varied as degrees of freedom to perform deformations, then the ductility d can be formally calculated as the product

$$d = \prod_{i=1}^{k} \prod_{j=1}^{n} |\mathbb{D}_{P_i, I_{P_i,j}}|. \tag{3.18}$$

However, one should take note that in some cases the ductility may actually be smaller than what equation 3.18 suggests. This situation occurs either (1) if the variability of a module is reduced for layout reasons, or (2) when there is a dependence between at least two degrees of freedom. Both aspects can be observed in the `CCM` circuit and the `WCM` circuit (as is reflected by column **Ductility** in Table 3.6):

- Aspect (1) is showcased by the practice to place all transistors in each of these circuits in an upright orientation only. From a layout perspective, this is quite convenient because it allows to connect the transistor gates in a straightforward way just by drawing stripes of poly across the transistors. The evident downside is –of course– the reduction in variability that stems from this confinement.
- Concerning aspect (2), two such dependencies can be encountered in both the `CCM` circuit and the `WCM` circuit. The first dependence pertains to the intrinsic variability of the Bank module $\mathtt{P}_{\mathtt{BK}}$ and the Cascode module $\mathtt{P}_{\mathtt{CS}}$: if, and only if, the Bank module breaks its positioning from a single-row layout into a dual-row variant, then the Cascode module must do so as well (such a deformation is depicted in Figure 3.13). So, the combined variability in these degrees of freedom $\mathtt{p}_{BK}$ and $\mathtt{p}_{CS}$ is not the Cartesian product $\mathbb{D}_{BK,\mathtt{p}} \times \mathbb{D}_{CS,\mathtt{p}}$ but only the subset $\{(single, single), (dual, dual)\}$. The second dependence can be found in the cumulative variability of $\mathtt{P}_{\mathtt{BK}}$ and $\mathtt{P}_{\mathtt{CS}}$: the number of fingers is supposed to be equal among the transistors of both modules. The deformation in Figure 3.14 illustrates how both modules change from a 2-finger variant into a 3-finger variant. Setting the number of fingers to different values is not only disadvantageous in regard of matching, but also leads to a significant amount of dead space, since the total channel width of the Bank transistors is usually identical to that of the Cascode transistors. Therefore, keeping the number of fingers identical represents a constraint that is implicitly considered by SWARM's modules (which comes up to the motivation of this thesis).

For each of the circuits in Table 3.6, column **Examples** as a summary again refers to those figures in this thesis which illustrate the respective module layout and the module's layout variability. Regarding layout variability, the thoughts in this work are supposed to be universally valid, independent of the semiconductor technology at hand. In contrast, the layout details of the implemented modules are not strictly part of the SWARM methodology itself, but always

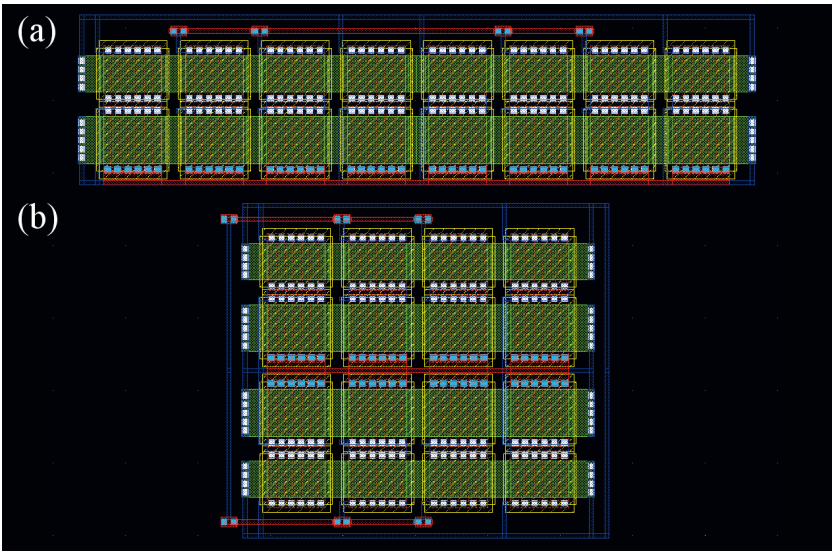


Figure 3.13: Cascode Current Mirror deformation from (a) single-row variant into (b) dual-row variant.

depend on the chosen semiconductor technology, the intended application, and other factors. Thus, the concrete module implementation in practice always lies under the authority of the respective design team (i.e., the **Design Expert** in Figure 3.2). But although the implementations in this thesis can be regarded as being only exemplary, they demonstrate how governing modules and module associations allow to automate the layout creation of basic circuits on the level of *simple modules*. At higher levels, additional and more groundbreaking automation concepts need to be embraced. One such automation concept is the module interaction that will now be described in Section 3.3 and represents the second core concept of SWARM.

3.3 Module Interaction

To form the desired layout block, the governing modules and module associations from Section 3.2 are to be arranged in a constellation that fits within a user-defined zone and explicitly satisfies all design constraints that are not yet implicitly covered by the modules themselves. At this level, irregular (non-matrix) constellations and arbitrary aspect ratios of the zone outline need to be served. Based on the considerations of Section 3.2.4, the modules may provide sufficient variability to achieve that goal, but the enormous amount of possible variations and

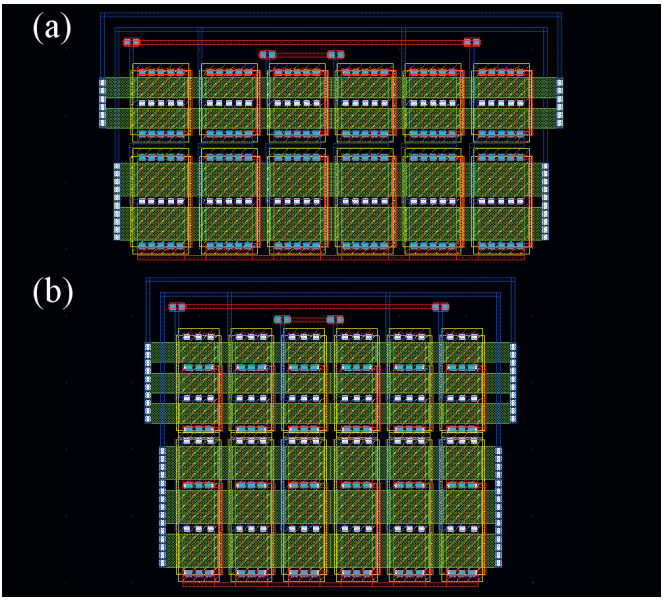


Figure 3.14: Wide-swing Current Mirror deformation from (a) 2-finger variant into (b) 3-finger variant.

constellations raises a combinatorial challenge. Instead of exploring this immense solution space in a top-down manner, the modules are –as outlined in Section 3.1– impelled into a flow of *module interaction* to let them find a suitable arrangement on their own.

Following the idea of decentralization, the modules in SWARM interact upon a maxim of self-determination: in a selfish pursuit of its own personal well-being, each participating module repeatedly chooses its individual course itself, always based on some simple utilitaristic rules and just a local assessment of its current situation. For that purpose, the modules' layout-generating abilites, which can be considered their *introversive behavior* (see Section 3.2.4.1), are extended by an *extroversive behavior*, which dictates a module's reaction to changes of its environment. The extroversive behavior is not necessarily identical among all modules (which are now, in the context of interaction, again referred to as *participants*), but it always abides by a common *action scheme* comprised of the following four *measures*:

(1) assessing the participant's *condition* (see Section 3.3.1),
(2) perceiving its *free peripheral space* (see Section 3.3.2),
(3) exploring and evaluating possible *actions* (see Section 3.3.3),
(4) executing the *preferred* action or staying idle (see Section 3.3.4).

These four measures are exemplarily illustrated in Figure 3.15. Considering a constellation of six participants as in (1), assume that it is participant P's turn to take an action. Following the action scheme, P begins by assessing its condition. Since P detects interference with another participant, P strives for an action that improves its condition. To do so, P perceives the vacant area around it, because most of the possible actions are based on this so-called free peripheral space. As shown in (2), SWARM determines the free peripheral space by extending each of P's four edges it its respective direction until another participant or the zone outline is encountered. Then, as indicated in (3), P explores and evaluates all actions available in the current situation. Some of these actions only affect the participant itself, while the other actions also involve other participants. Next, the actions are compared such that the one which improves P's condition the most can be chosen and executed. In (4), the executed action is even synergistic (see Section 2.4.6): P trades places with another participant and *both* get rid of their interference.

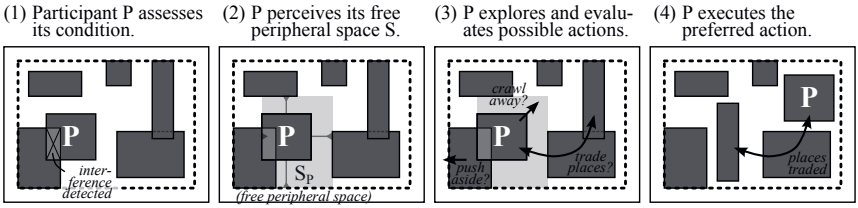


Figure 3.15: A participant's actions follow a common action scheme consisting of four measures.

In the remainder of this section, every single measure of the action scheme will now be covered in a subsection of its own (see Section 3.3.1, Section 3.3.2, Section 3.3.3, and Section 3.3.4, respectively).

3.3.1 Assessment of the Participant's Condition

In terms of game theory (see Section 2.5.2), SWARM can be considered an infinitely-repeated, noncooperative, discrete, asymmetric, non-constant-sum, sequential, perfect-information game in extensive form, with an unknown number of stage games. In each stage game (here: a *round* of interaction), every participant acts in a self-interested way to improve its personal situation. This *utility-theoretic* attitude is a characteristic trait of noncooperative game theory, but unlike typical *utility functions* which map a player's preferences to some real numbers (thus quantifying the player's favored "states of the world" [103]), SWARM implements a more sophisticated decision-making model built around a participant's *condition*.

The condition of a participant P decides whether there is a need to take action or not. Action is provoked when the condition sustains negative influence, which can be exerted by

five different *influencing factors*. These influencing factors are largely based on the fact that all participants are geometric objects with a rectangular bounding box and thus have an area, in contrast to dot-like entities found in some other systems.

The two major influencing factors are denoted as *interference* (repulsion of participants due to overlaps, as already indicated in Figure 3.15) and *turmoil* (attraction of participants due to connectivity), which *can* –depending on their magnitude– provoke an action. In contrast, a participant is *forced* to take an action if it suffers at least one of the other three influencing factors: *protrusion* (overhang beyond the zone outline), *wounds* (regions on P which repeatedly overlapped with other participants in previous rounds of interaction), or *noncompliance* (violation of constraints).[6]

For each kind of influencing factor, every participant has a particular *desire* to reduce and eliminate the factor's negative influence on its condition. This will be subsequently discussed in detail for each individual kind of influencing factor.

3.3.1.1 Interference

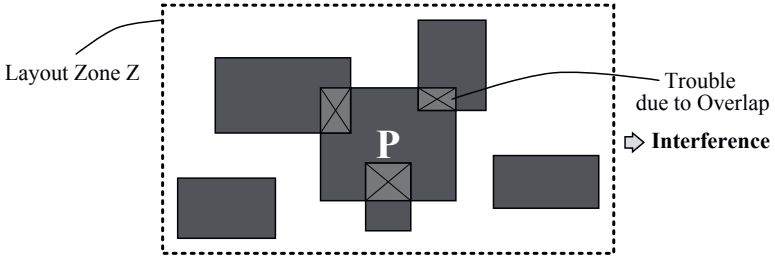


Figure 3.16: Illustration of interference for a participant P.

Interference is when a participant overlaps one or multiple other participants, as illustrated in Figure 3.16. The interference (written Υ) of a participant P is a sum of *troubles* τ between P and n overlapping participants $P_1, P_2, \ldots, P_n$:

$$\Upsilon_P = \sum_{i=1}^{n} \tau_i. \tag{3.19}$$

Therein, each individual trouble τ_i is defined as

$$\tau_i = \omega_i + \vartheta_i \tag{3.20}$$

[6]However, there are situations, where the participant may temporarily tolerate even wounds and noncompliance, for example when performing a so-called *Re-entering* action (see Section 3.3.3.1).

where both ω_i and ϑ_i are scalar values: ω_i is called the *overlap* of P and P_i, while ϑ_i is the *aversion* of P towards P_i.

The overlap ω_i of P and P_i is their intersection area, multiplied with the area of P_i. The first multiplicand is denoted as the *intensity* of the overlap, while the second multiplicand is called *tenacity*. Using the geometrical operators from page 227, the calculation of the overlap can be expressed as

$$\omega_i = \underbrace{\left(\square P \sqcap \square P_i\right)}_{\text{intensity}} \cdot \underbrace{\left(\square P_i\right)}_{\text{tenacity}} . \tag{3.21}$$

One remark should be made about the $\square$ operator, which determines the bounding box of a geometrical object. Since a participant P is not a rudimentary geometrical object (i.e., a mere *shape* such as a polygon) but a hierarchical design component, the bounding box of P must be determined by examining all physically relevant shapes throughout the entire subhierarchy of P.

Definition 3.1. *Consider a participant P, given as an instance of a procedural generator. Let P contain a set of geometrical shapes $\mathcal{G}$ and a set of subinstances $\mathcal{P}$. For the semiconductor technology at hand, let Λ be the set of layout layers that are physically relevant (e.g., excluding logical layers, auxiliary layers and text layers). Then, the set $\mathcal{G}'$ of physically relevant shapes inside P is a subset of $\mathcal{G}$ containing every geometrical shape G whose layout layer ℓ_G is a physically relevant layer:*

$$\mathcal{G}' = \{G \mid G \in \mathcal{G} \wedge \ell_G \in \Lambda\}. \tag{3.22}$$

To determine the bounding box of a hierarchical design component, the $\square$ operator can now be defined in a recursive fashion[7] as follows:

$$\begin{aligned}\square P = \Bigg(&\Big(\min\Big(\min_{\forall G \in \mathcal{G}'}\left(\vdash G\right), \min_{\forall P' \in \mathcal{P}}\left(\vdash(\square P')\right)\Big), \min\Big(\min_{\forall G \in \mathcal{G}'}\left(\perp G\right), \min_{\forall P' \in \mathcal{P}}\left(\perp(\square P')\right)\Big)\Big), \\ &\Big(\max\Big(\max_{\forall G \in \mathcal{G}'}\left(\dashv G\right), \max_{\forall P' \in \mathcal{P}}\left(\dashv(\square P')\right)\Big), \max\Big(\max_{\forall G \in \mathcal{G}'}\left(\top G\right), \max_{\forall P' \in \mathcal{P}}\left(\top(\square P')\right)\Big)\Big) \Bigg).\end{aligned} \tag{3.23}$$

This expression returns the bounding box of P in rectangle notation *$((\check{x}, \check{y}), (\hat{x}, \hat{y}))$, where $(\check{x}, \check{y})$ represents the south-western vertex of the bounding box and $(\hat{x}, \hat{y})$ represents the north-eastern vertex of the bounding box.*

With the inclusion of the tenacity $\left(\square P_i\right)$ into equation 3.21, the repulsion against P is correlated with the size of the overlapping participant P_i. This idea comes to the fore in the calculation of the *prospective*[8] interference that P can achieve by performing an action: if P

[7] The formula is recursive since $\square P$ also appears on the right-hand side of the expression as $\square P'$. This recursive formulation is valid because the deepest component in each branch of P's subhierarchy has no subinstances, i.e., $\mathcal{P} = \emptyset$ for every leaf component of the subhierarchy tree.

[8] See Section 3.3.4.

is compelled to move into a new location where it also overlaps with another participant P', P thus prefers a smaller participant over a larger one (because the prospective interference is smaller). Since P' then in turn has to perform an action to get rid of the interference, being rather small than large is advantageous for the progress of self-organization because finding a new location is easier for smaller participants. In other words, including $\lceil P_i \rceil$ into equation 3.21 achieves that the necessity to move away from situations of interference is inclined to propagate from larger participants to smaller participants. Therefore, the actions to perform become less and less disruptive which benefits the convergence of the overall interaction flow.

Participant's Desire 1 (Interference):
If a participant does not overlap with other participants, then the participant is said to be clear. *During the interaction, every participant strives to act in a way such that it becomes clear. If $\mathcal{P}$ denotes the set of all participants, then the desire of a participant P regarding interference is to achieve that*

$$\forall P' \in \mathcal{P} \setminus P,\ \lceil P \rceil \sqcap \lceil P' \rceil = \varnothing \tag{3.24}$$

which means that

$$\Upsilon_P = 0. \tag{3.25}$$

In the beginning of the interaction, there is no aversion among the participants. So, the initial aversion is zero:

$$\vartheta_{i,0} = 0. \tag{3.26}$$

The same is true for the initial number of *clashes* γ between the participants. Hence:

$$\gamma_{i,0} = 0. \tag{3.27}$$

An overlap ω_i of P and P_i increases P's aversion towards P_i such that the aversion changes from the current value $\vartheta_{i,j}$ to the new value $\vartheta_{i,j+1}$ according to the formula

$$\vartheta_{i,j+1} = (\vartheta_{i,j} + \omega_i) \cdot (\gamma_{i,j} + 1). \tag{3.28}$$

Here, $\gamma_{i,j}$ is the previous number of clashes between P and P_i. After the calculation of $\vartheta_{i,j+1}$, the number of clashes is incremented to the value

$$\gamma_{i,j+1} = \gamma_{i,j} + 1 \tag{3.29}$$

due to the overlap. If there is no overlap between participants P and P_i within one round of interaction, the aversion drops to the value

$$\vartheta_{i,j+1} = \vartheta_{i,j} \cdot \varphi \tag{3.30}$$

where φ is a *conciliation quota* chosen within the interval $[0, 1]$. If $\varphi = 1$, there is no conciliation. In this case, the aversion is never reduced but can only become larger, which means that

$$\vartheta_{i,j+1} \geq \vartheta_{i,j}. \tag{3.31}$$

As soon as the interacting participants yield a *settlement* that is *viable*, the aversion and the number of clashes are both reset to zero for each pair of participants.

3.3.1.2 Turmoil

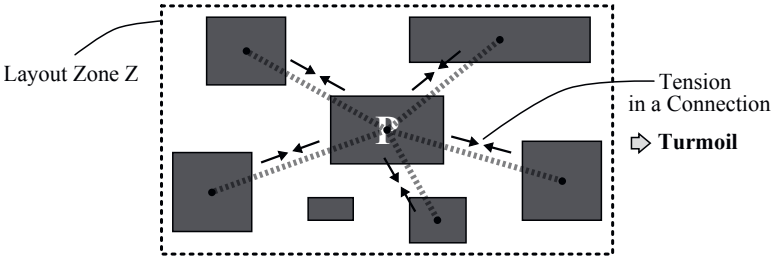


Figure 3.17: Illustration of turmoil for a participant P.

Turmoil is used to take distances between participants into account during the interaction. As shown in Figure 3.17, the distance between two participants is modeled as a straight *connection* between their centerpoints. Formally, every connection C is a quadruple (L_1, L_2, e, s) where L_1 and L_2 are the two endpoints of C, while e is called the *emphasis* and s represents the *strength* of the connection. Both e and s are scalar values that remain constant during a SWARM run.

The emphasis $e \in \mathbb{R}$ is 1 by default and can –depending on the particular requirements of the problem at hand– be individually set by the user within the interval $(0, 1]$ to downgrade the importance of a connection in relation to the other connections (there being a distinction between SWARM's idea of emphasis and the algorithmic notion of a *weight*, as will be discussed in Section 3.5.1). The strength $s \in \mathbb{N}^+$ of a connection between two participants P_1 and P_2 is automatically calculated as the sum

$$s = \eta_1 + \eta_2 - 1 \tag{3.32}$$

where η_1 and η_2 represent the total *number of connections* of P_1 and P_2 respectively (the subtrahend -1 effectuates that the common connection of P_1 and P_2 is not counted twice). Similar to the idea of tenacity in equation 3.21, the consideration of strength has the purpose of streamlining the self-organization. As illustrated in Figure 3.18, a participant is rather drawn to

another one having many connections than to one having fewer connections, which reduces the number of participants that need to follow this motion due to connectivity.

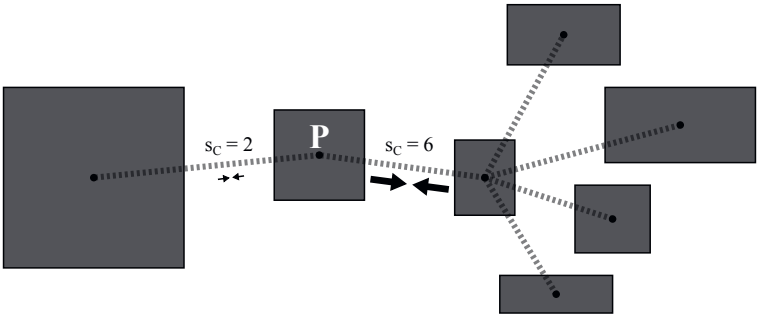


Figure 3.18: Effect of the strength s_C of a connection C on a participant's action.

The condition of a participant sustains negative influence, if its distance to a connected participant exceeds a certain threshold. For any connection, this so-called *relaxation threshold* ϱ is defined as follows.

Definition 3.2. *Given a set $\mathcal{P}$ of participants inside the layout zone Z, consider a connection C between two participants P_1 and P_2. Let Q_1 be a* square *with an area equal to the area of P_1, and let r_1 be the* radius *of the smallest possible circle around Q_1. Furthermore, let λ be a* leeway coefficient *calculated as*

$$\lambda = \sqrt{\frac{[Z]}{[\mathcal{P}]}} = \sqrt{\frac{[Z]}{\sum_{P \in \mathcal{P}} [P]}} \tag{3.33}$$

where $[Z]$ is the area of Z and $[\mathcal{P}]$ is the sum of the participants' individual areas. Then, the value

$$\varrho_{C,1} = \frac{\lambda \cdot r_1}{e_C} \tag{3.34}$$

where e_C represents the emphasis of C, is denoted as the "personal" relaxation threshold *of P_1 for connection C. With $\varrho_{C,2}$ as the personal relaxation threshold of P_2, the sum*

$$\varrho_C = \varrho_{C,1} + \varrho_{C,2} \tag{3.35}$$

is defined as the "total" relaxation threshold *of connection C.*

To illustrate the idea of relaxation threshold, Figure 3.19 shows the connection C of two participants P_1 and P_2, each of which is circumfered by its personal relaxation threshold $\varrho_{C,1}$

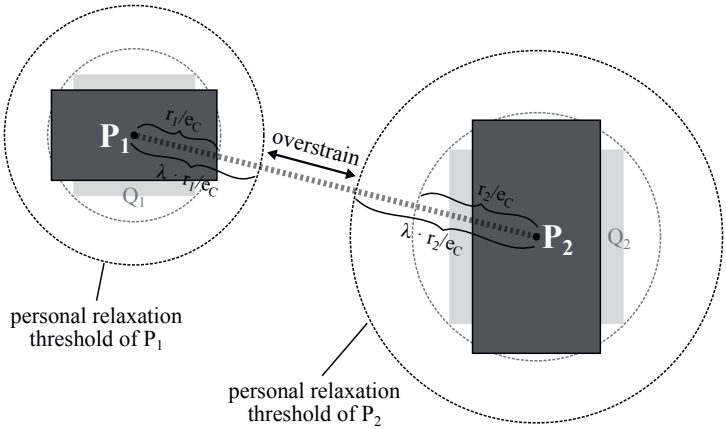


Figure 3.19: Personal relaxation thresholds and overstrain between two connected participants.

and $\varrho_{C,2}$ (for $\lambda = 1.5$ and $e_C = 1$). The difference $l_C - (\varrho_{C,1} + \varrho_{C,2})$ between the length l_C of the connection and the total relaxation threshold is denoted as *overstrain* in the figure.

The radius r of the smallest possible circle around a square Q is half the diagonal of Q and can be determined via

$$r = \frac{1}{2}\sqrt{2\,[Q]}. \tag{3.36}$$

According to the definition above, the area of Q_1 is equal to the area of P_1. This is also true for the area of Q_2 and P_2:

$$[Q_1] = [P_1],\ [Q_2] = [P_2]. \tag{3.37}$$

Using equation 3.36 and equation 3.37, the relaxation threshold can thus be calculated in closed form according to the following formula:

$$\begin{aligned}
\varrho_C &= \frac{\lambda \cdot r_1}{e_C} + \frac{\lambda \cdot r_2}{e_C} && (3.38)\\
&= \frac{\lambda}{e_C} \cdot (r_1 + r_2) && (3.39)\\
&= \frac{\lambda}{e_C} \cdot \left(\frac{1}{2}\sqrt{2\,[P_1]} + \frac{1}{2}\sqrt{2\,[P_2]} \right) && (3.40)\\
&= \frac{\lambda}{\sqrt{2} \cdot e_C} \cdot \left(\sqrt{[P_1]} + \sqrt{[P_2]} \right). && (3.41)
\end{aligned}$$

The length l_C of a connection C is simply the Euclidean distance between its endpoints L_1 and L_2. With $L_1 = (x_1, y_1)$ and $L_2 = (x_2, y_2)$, the length is given by Pythagoras' theorem:

$$l_C = \overline{L_1 L_2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{3.42}$$

Participant's Desire 2 (Turmoil):
If the length of a connection C is below its relaxation threshold ϱ_C, the connection is said to be relaxed *(otherwise* unrelaxed*). During the interaction, every participant strives to act in a way such that all of its connections become relaxed, in which case the participant itself is said to be relaxed. If $\mathcal{C}_P$ denotes the set of a participant P's connections, then the participant's desire regarding turmoil is to achieve that*

$$\forall C \in \mathcal{C}_P,\ l_C \leq \varrho_C. \tag{3.43}$$

Using $\zeta_P \in \mathbb{N}^0$ to denote the number of P's unrelaxed connections, the desire of P to become relaxed can be shortly written as

$$\zeta_P = 0. \tag{3.44}$$

The leeway coefficient λ prevents the participants from clumping together when there is still much space left in the beginning of a SWARM run. With every zone tightening, λ becomes smaller and approaches 1 since Z is successively downsized towards a minimal outline Z' whose area equals $\lceil \mathcal{P} \rceil$:

$$\lim_{Z \to Z'} \lambda(Z) = \lim_{Z \to Z'} \sqrt{\frac{\lceil Z \rceil}{\lceil \mathcal{P} \rceil}} = 1\ . \tag{3.45}$$

Assuming that the area of a participant is roughly the same among all layout variants the participant can deform into, the relaxation threshold of a connection C only depends on λ since e_C is a constant value. Thus, the relaxation threshold ϱ_C can also be written as a function of Z, which successively converges towards

$$\lim_{Z \to Z'} \varrho_C(Z) = \underbrace{\lim_{Z \to Z'} \lambda(Z)}_{=1} \cdot \frac{1}{e_C} \cdot (r_1 + r_2) = \frac{r_1 + r_2}{e_C} \tag{3.46}$$

throughout the course of a SWARM run. In this fashion, the participants are carefully drawn into their final arrangement without depriving them of the leeway they need to organize themselves. If the emphasis of a connection C is set to the default value $e_C = 1$, the relaxation threshold during the last tightening-settlement cycle is $\varrho_C = r_1 + r_2$, so the two participants P_1 and P_2 are expected to move into immediate vicinity of each other.

The above considerations deal with the modeling of a single connection C. But since a participant can be connected to multiple other participants, multiple connections must be taken into account during the interaction. For that purpose, the turmoil Θ of a participant P is calculated by adding up the so-called *tension* θ in each of P's n connections $C_1, C_2, \ldots, C_n$:

$$\Theta_P = \sum_{i=1}^{n} \theta_i. \tag{3.47}$$

The concept of tension elucidates the notion of the term *turmoil*: a participant virtually is "in a turmoil" for being "torn" apart by its omnidirected connections, all of which influence the actions of the participant like tensely strained rubber straps. The respective tension θ in a connection C is given by

$$\theta(l_C) = \begin{cases} l_C \cdot s_C e_C & \Leftrightarrow \quad l_C \leq \varrho_C e_C \\ \big((l_C + \frac{1}{2} - \varrho_C e_C)^2 - \frac{1}{4} + \varrho_C e_C\big) \cdot s_C e_C & \Leftrightarrow \quad l_C > \varrho_C e_C \end{cases} \tag{3.48}$$

where the only nonconstant values are the connection's relaxation threshold ϱ_C (which is changed with each tightening of the layout zone) and the connection's length l_C (which is meant to be reduced via moves of the participants). For

$$l_C \leq \underbrace{\varrho_C e_C}_{=\lambda\cdot(r_1+r_2)} \tag{3.49}$$

the relation of tension to length is linear, otherwise it is quadratic to penalize longer distances more severely und thus vivify the interaction. As illustrated by the graph in Figure 3.20, the constituent formulas of equation 3.48 are chosen such that the entire function is continuous and differentiable. Therefore, using $\theta_\dagger(l_C)$ and $\theta_\ddagger(l_C)$ to denote the constituent formulas for $l_C \leq \varrho_C e_C$ and $l_C > \varrho_C e_C$ respectively, the tension $\theta_\dagger(\varrho_C e_C)$ equals the tension $\theta_\ddagger(\varrho_C e_C)$ as proven by:

$$\begin{aligned} \overbrace{\theta_\dagger(\varrho_C e_C)}^{\theta(l_C)\text{ for } l_C \leq \varrho_C e_C} &= \overbrace{\theta_\ddagger(\varrho_C e_C)}^{\theta(l_C)\text{ for } l_C > \varrho_C e_C} \\ \varrho_C e_C \cdot s_C e_C &= \big((\varrho_C e_C + \tfrac{1}{2} - \varrho_C e_C)^2 - \tfrac{1}{4} + \varrho_C e_C\big) \cdot s_C e_C \\ \varrho_C\, s_C\, e_C^2 &= \big((\tfrac{1}{2})^2 - \tfrac{1}{4} + \varrho_C e_C\big) \cdot s_C e_C \\ \varrho_C\, s_C\, e_C^2 &= \varrho_C e_C \cdot s_C e_C \\ \varrho_C\, s_C\, e_C^2 &= \varrho_C\, s_C\, e_C^2. \quad \square \end{aligned}$$

Equivalently, the slope of $\theta_\dagger(l_C)$ is identical to the slope of $\theta_\ddagger(l_C)$ for $l_C = \varrho_C e_C$. Using $\theta'_\dagger(l_C) = s_C e_C$ and $\theta'_\ddagger(l_C) = (2l_C + 1 - 2\varrho_C e_C) \cdot s_C e_C$ to notate the derivatives of $\theta_\dagger(l_C)$ and $\theta_\ddagger(l_C)$ respectively, where $\theta'_\ddagger(l_C)$ has been determined after expanding

$$\begin{aligned} \theta_\ddagger(l_C) &= \big((l_C + \tfrac{1}{2} - \varrho_C e_C)^2 - \tfrac{1}{4} + \varrho_C e_C\big) \cdot s_C e_C \\ &= \big(l_C^2 + 2l_C(\tfrac{1}{2} - \varrho_C e_C) + (\tfrac{1}{2} - \varrho_C e_C)^2 - \tfrac{1}{4} + \varrho_C e_C\big) \cdot s_C e_C \\ &= (l_C^2 + l_C - 2\varrho_C e_C l_C + (\tfrac{1}{2})^2 - \varrho_C e_C + \varrho_C^2 e_C^2 - \tfrac{1}{4} + \varrho_C e_C) \cdot s_C e_C \\ &= (l_C^2 + l_C - 2\varrho_C e_C l_C + \varrho_C^2 e_C^2) \cdot s_C e_C, \end{aligned}$$

the proof can be given as:

$$\begin{aligned} \overbrace{\theta'_\ddagger(\varrho_C e_C)}^{\theta'(l_C)\text{ for } l_C \leq \varrho_C e_C} &= \overbrace{\theta'_\ddagger(\varrho_C e_C)}^{\theta'(l_C)\text{ for } l_C > \varrho_C e_C} \\ s_C e_C &= (2\varrho_C e_C + 1 - 2\varrho_C e_C) \cdot s_C e_C \\ s_C e_C &= s_C e_C. \quad \square \end{aligned}$$

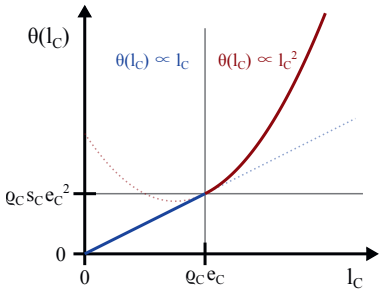


Figure 3.20: Tension in a connection between two participants, depending on the connection's length.

3.3.1.3 Protrusion

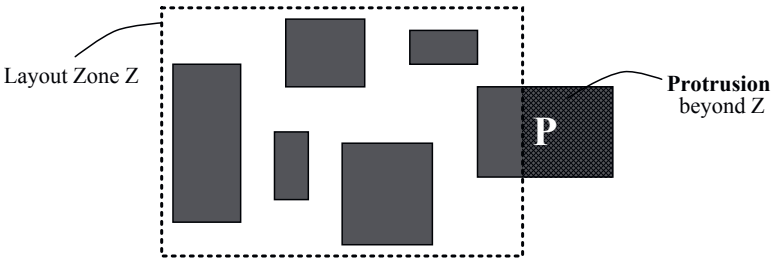


Figure 3.21: Illustration of protrusion for a participant P.

Protrusion is the case when a participant does not completely lie within the current outline of the layout zone, as depicted in Figure 3.21. In that case, the participant is forced to take an action. Geometrically, the protrusion Ψ of a participant P can be written as

$$\Psi_P = \square P \sqcap \overline{Z} \tag{3.50}$$

where $\overline{Z}$ (the complement of Z) denotes the territory beyond the layout zone.

Depending on the grade of protrusion, a participant is denoted as *lost* (entirely outside Z), *prone* (partially outside Z), or *safe* (entirely inside Z). Each of these three cases is illustrated in Figure 3.22 and bijectively maps to a distinct geometrical condition as follows:

$$P \text{ is } \begin{cases} \textit{lost} & \Leftrightarrow \quad \Psi_P = \square P \\ \textit{prone} & \Leftrightarrow \quad \Psi_P \sqsubset \square P \\ \textit{safe} & \Leftrightarrow \quad \Psi_P = \varnothing. \end{cases} \tag{3.51}$$

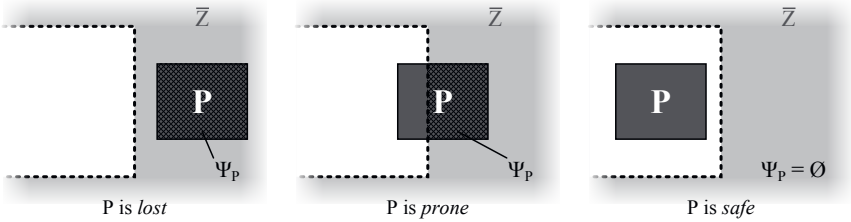


Figure 3.22: The three grades of protrusion: a participant can be either lost, prone, or safe.

Participant's Desire 3 (Protrusion):
As mentioned above, a participant that does not at all protrude the layout zone is said to be safe. *During the interaction, every participant strives to act in a way such that it becomes safe. So, the desire of a participant P regarding protrusion is to achieve that*

$$\Psi_P = \varnothing. \tag{3.52}$$

If a participant would become *prone* after performing one of its potential actions, it is good to know how far exactly the participant would protrude the layout zone horizontally and vertically. As will be discussed in the end of Section 3.3.3.1, this *protrusion extent* $(\psi_x,\ \psi_y)$ can be used to correct the potential action such that it leads the participant into a safe location. To determine the protrusion extent of a prone participant, eight different cases of protrusion are to be distinguished, as shown in Figure 3.23. In that regard, it is important to note that the layout zone may be rectilinear, not just rectangular.

Referring to the eight images in Figure 3.23, there are five cases in which it is not possible to determine an unambiguous protrusion extent, depending on the number and the form of those parts of a participant P that lie inside and outside the layout zone Z:

(a1) There are more than one disjunct parts of P that lie outside of Z.
(a2) There are more than one disjunct parts of P that lie inside of Z.
(b1) The part of P inside Z is rectangular, but the outside part is a concave polygon with more than one reflex interior angle (i.e., an interior angle greater than $180°$).
(b2) The part of P outside Z is rectangular, but the inside part is a concave polygon with more than one reflex interior angle.
(c1) The part of P outside Z and the part inside Z are both not rectangular.

In these five cases, no *action correction* is performed due to the lack of an unequivocal protrusion extent. Furthermore, this is justified by the conjecture that the likelihood of encountering one of these cases is rather low. If Z is rectangular, cases (a1) and (b1) are merely theoretical, while the occurrence of cases (a2), (b2) or (c1) is even impossible. In the remaining three cases, the calculation of the protrusion extent is well-defined and trivial:

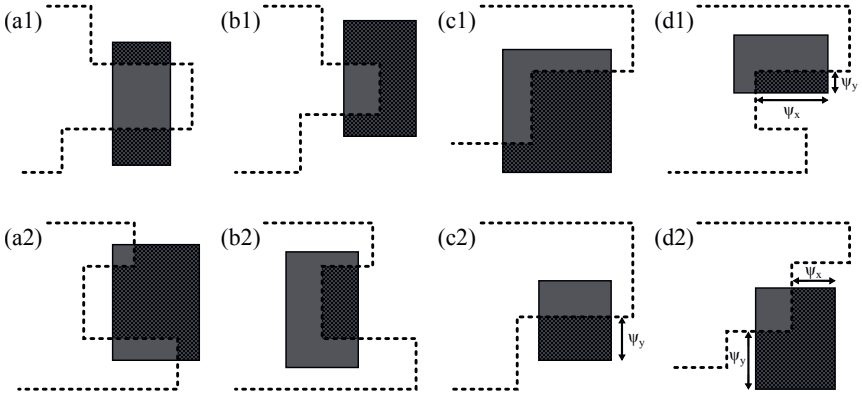


Figure 3.23: The different cases of protrusion for a prone participant and a rectilinear layout zone.

(c2) The part of P outside Z and the part inside Z are both rectangular. In this case, the protrusion extent is given by the dimensions of the part outside Z.

(d1) The part of P outside Z is rectangular and the inside part is a concave polygon with only one reflex interior angle. As in (c2), the protrusion extent is given by the dimensions of the part outside Z.

(d2) The part of P inside Z is rectangular and the outside part is a concave polygon with only one reflex interior angle. Then, the protrusion extent is given by the bounding box dimensions of P minus the dimensions of the part inside Z.

With the aim of performing an action correction, an important matter must be respected if the protrusion occurs at an edge (not at a vertex) of the layout zone: either the horizontal or the vertical protrusion extent must be zero since an *axis-oriented* action correction is desired in this case. More specifically, if P protrudes Z upwards or downwards –as in the example of (c2)– this requires that $\psi_x = 0$ (to obtain a purely vertical action correction). If P protrudes Z leftwards or rightwards, then $\psi_y = 0$ (horizontal action correction). This can be achieved by calculating the protrusion extent in the same way as described above regarding case (d2).

Whether a geometrical shape G is a rectangle, can be determined by evaluating if the condition $\boxdot G = G$ holds true. When it is known that every angle in G is a right angle, an alternative option is to check if the number of vertices is $|G| = 4$. Whether a rectilinear, not self-intersecting polygon G has exactly one reflex interior angle, can be determined by evaluating if $|G| = 6$. Thus, the calculation of the protrusion extent for the cases (c2), (d1), and (d2)

can be done according to the following formulas

$$\psi_x = \begin{cases} \dashv(\square P \sqcap \overline{Z}) - \vdash(\square P \sqcap \overline{Z}) & \Leftrightarrow \ |\square P \sqcap Z| = 6 \wedge |\square P \sqcap \overline{Z}| = 4 \\ \dashv(\square P) - \vdash(\square P) - \dashv(\square P \sqcap Z) + \vdash(\square P \sqcap Z) & \Leftrightarrow \ |\square P \sqcap Z| = 4 \wedge |\square P \sqcap \overline{Z}| \leq 6 \end{cases} \tag{3.53a}$$

$$\psi_y = \begin{cases} \top(\square P \sqcap \overline{Z}) - \bot(\square P \sqcap \overline{Z}) & \Leftrightarrow \ |\square P \sqcap Z| = 6 \wedge |\square P \sqcap \overline{Z}| = 4 \\ \top(\square P) - \bot(\square P) - \top(\square P \sqcap Z) + \bot(\square P \sqcap Z) & \Leftrightarrow \ |\square P \sqcap Z| = 4 \wedge |\square P \sqcap \overline{Z}| \leq 6 \end{cases} \tag{3.53b}$$

where $|\square P \sqcap Z| = 4 \wedge |\square P \sqcap \overline{Z}| = 4$ represents case (c2), $|\square P \sqcap Z| = 6 \wedge |\square P \sqcap \overline{Z}| = 4$ represents case (d1), and $|\square P \sqcap Z| = 4 \wedge |\square P \sqcap \overline{Z}| = 6$ represents case (d2).

3.3.1.4 Wounds

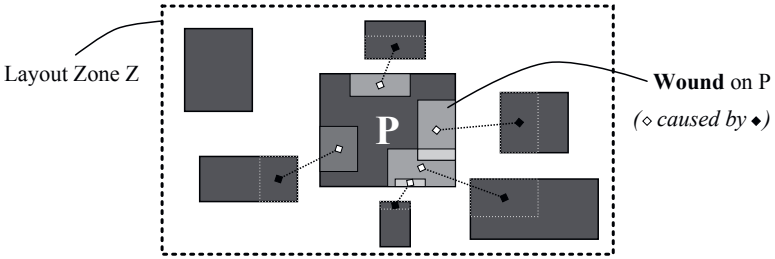


Figure 3.24: Illustration of wounds for a participant P.

When a participant P assesses its condition, every overlap ω between P and another participant inflicts wounds on P, as exemplified in Figure 3.24. Formally, a wound W is a pair (ρ, ς) which has a specific rectangular *region* ρ on P and a discrete *severity* ς represented as a natural number $\varsigma \in \mathbb{N}^0$.

An overlap between P and another participant P' inflicts a new wound W on P which has a severity $\varsigma_W = 1$ and covers the overlap region $\rho_W = \square P \sqcap \square P'$. If that region also overlies an existing wound W^* on P, then the wound is *aggravated* throughout the part of W^* overlain by ρ_W. That is to say, the severity of the wounded part $\rho_W \sqcap \rho_{W^*}$ is incremented by $\Delta\varsigma$, wherein the value for $\Delta\varsigma$ is either 2 (if W^* was inflicted by P') or 1 (if W^* was inflicted by a participant other than P'). Those parts of a wound, that do not become aggravated due to an overlap, can *ameliorate*. This is achieved by decrementing the severity of those parts by 1. If the severity drops below zero, then that part of the wound vanishes and is said to be *healed*.

One particular comment should be made about the implementation of the wounds concept. Aggravating the overlain part ρ^* of an existing wound can be done by geometrically separating that part from the wound and then incrementing its severity to the new value ς^*. However, this leads to the necessity of dealing with arbitrarily intricate polygonal contours of wounded parts. A convenient alternative is simply to inflict a new wound with the increased severity ς^* on top of the existing wound, covering the region ρ^*. With this practice, geometrical operations on wounds involve only rectangular regions, as Figure 3.25 illustrates in an example.

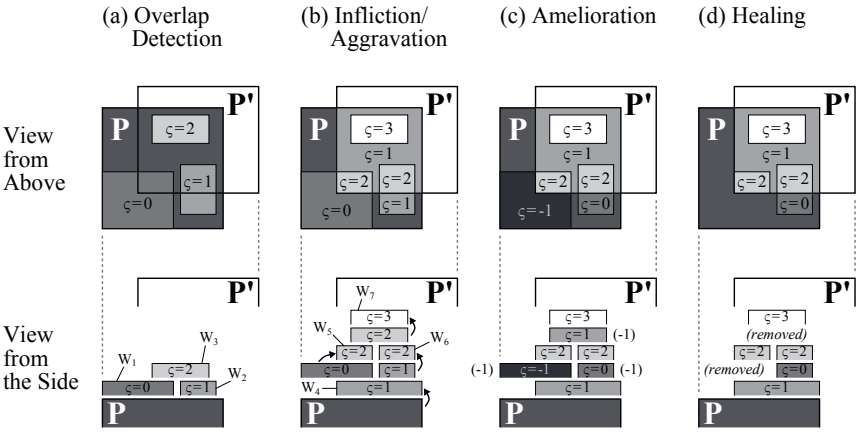


Figure 3.25: Exemplary depiction of how a participant P gets wounded by another participant P'.

For this example, Figure 3.25 shows a view from above onto the layout, and a cross section through the "stack" of wounds. Both depictions are given for the four different stadiums of the wound infliction which can be described as follows:

(a) *Overlap Detection:*
Given is a participant P with three wounds: a wound W_1 with a severity of $\varsigma = 0$, a wound W_2 with $\varsigma = 1$, and a wound W_3 with $\varsigma = 2$. As illustrated in the image, P is overlapped by another participant P' which was already responsible for having inflicted wound W_1 in a previous round of interaction.

(b) *Infliction/Aggravation:*
The overlap inflicts a new wound W_4 on P with a severity of $\varsigma = 1$, covering the overlap region $\square P \sqcap \square P'$. Since the overlap also intersects W_1 and W_2 in parts and encompasses W_3 in full, these wounds are aggravated. That means, the overlap region $\square P' \sqcap \rho_{W_1}$ is aggravated by causing another wound W_5 with $\varsigma = 2$ (i.e., $\Delta\varsigma = 2$, since W_1 was inflicted by P'), while the regions $\square P' \sqcap \rho_{W_2}$ and $\square P' \sqcap \rho_{W_3}$ are aggravated from $\varsigma = 1$ to $\varsigma = 2$ and

from $\varsigma = 2$ to $\varsigma = 3$, which is achieved via new wounds W_6 and W_7 respectively (i.e., $\Delta\varsigma = 1$, since W_2 and W_3 were not inflicted by P').

(c) *Amelioration:*
Now that the new wounds (W_4, W_5, W_6, and W_7) have been added to P, the old wounds (W_1, W_2, and W_3) are ameliorated. This is done via decrementing the respective severity by $\Delta\varsigma = -1$. For W_3, the severity thus becomes $\varsigma = 1$, and for W_2, the severity reaches $\varsigma = 0$. For W_1, the severity drops to the (actually excluded) value $\varsigma = -1$, which is thus subject to healing in the next stadium.

(d) *Healing:*
Since the severity of W_1 has gone below zero, the wound has fully healed and is therefore removed from the stack of wounds. Hence, it is again true that the severity of all remaining wounds is $\varsigma \in \mathbb{N}^0$. Since the new wound W_7 covers the old wound W_3 not only in parts but in full, W_3 is in fact inane and can therefore also be removed.

When the aggravation of a wound exceeds a *critical severity* ς_c, the wound is said to be critical and the participant is forced to perform an action. In that case, the participant begins a strategy of *recuperation* and chooses its subsequent actions such that the wound is not aggravated any further before being fully healed. By the same token, a participant is not allowed to aggravate another participant's wound if that wound is currently subject to recuperation.

Participant's Desire 4 (Wounds):
If a participant's wound is overlapped by another participant despite being subject to recuperation, or if a participant overlaps another participant's wound which is currently subject to recuperation, then the participant is said to be in an unhealthy *location, otherwise in a* healthy *location. During the interaction, every participant strives to act in a way such that it does not get into an unhealthy location. If $\mathcal{P}$ denotes the set of all participants and $\mathcal{W}_P^\dagger$ represents a participant P's set of wounds that are currently subject to recuperation, then the desire of a participant P regarding wounds is to achieve that*

$$\forall P' \in \mathcal{P} \setminus P\,,\ (\nexists W \in \mathcal{W}_P^\dagger : \square P' \sqcap \rho_W \neq \varnothing) \wedge (\nexists W' \in \mathcal{W}_{P'}^\dagger : \square P \sqcap \rho_{W'} \neq \varnothing). \tag{3.54}$$

Using

$$\mathcal{W}_{P\ltimes}^{\ddagger} = \{W \in \mathcal{W}_P^\dagger \mid \exists P' \in \mathcal{P} \setminus P : \square P' \sqcap \rho_W \neq \varnothing\} \tag{3.55}$$

to denote the set of P's recuperating wounds currently overlapped by other participants, and using

$$\mathcal{W}_{P\rtimes}^{\ddagger} = \{W' \mid \exists P' \in \mathcal{P} \setminus P : W' \in \mathcal{W}_{P'}^\dagger \wedge \square P \sqcap \rho_{W'} \neq \varnothing\} \tag{3.56}$$

to denote the set of other participants' recuperating wounds currently overlapped by P, then P is in a healthy location if the union $\mathcal{W}_{P\bowtie}^{\ddagger}$ of both sets is empty:

$$\mathcal{W}_{P\bowtie}^{\ddagger} = \mathcal{W}_{P\ltimes}^{\ddagger} \cup \mathcal{W}_{P\rtimes}^{\ddagger} = \emptyset. \tag{3.57}$$

Wounds are a pivotal element of SWARM, akin to the idea of aversion in Section 3.3.1.1 but with different motives in two regards. First, aversion has a long-term effect in that it hinders a perpetual interference of two participants, whereas a critical wound has an immediate impact. Second, as aversion correlates with the area of an overlap, wounds are more effective for preventing marginal interferences. A careful balance in the modeling of both aversions and wounds is one key to a fluent progress of self-organization in SWARM.

3.3.1.5 Noncompliance

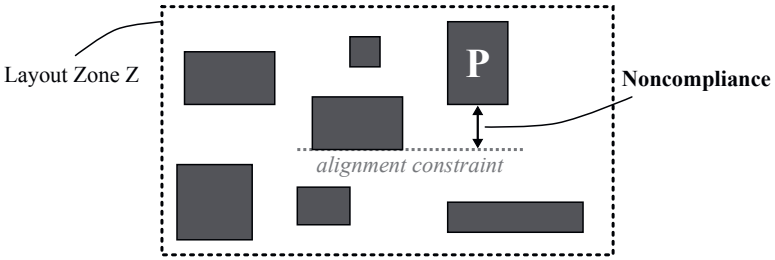


Figure 3.26: Illustration of noncompliance for a participant P.

Noncompliance indicates that a participant, in its current location, violates at least one explicitly formulated design constraint. An example is given in Figure 3.26, where participant P violates an Alignment constraint. Noncompliance specifically addresses only *hard constraints* (strict confinements), since *soft constraints* (optimization goals) are incorporated into SWARM via other mechanisms. Mentioning two examples for the latter, wirelength minimization is achieved with the consideration of turmoil (Section 3.3.1.2), while minimization of the total area is pursued through the successive tightening of the layout zone (Section 3.4).

Formally, a hard constraint imposes a certain restriction on a set $\mathcal{M}$ of design objects denoted as the *constraint members*. For a concrete, hard constraint H of constraint type t_H, there must be a type-dependent *verification function* $v_{t_H}(\mathcal{M})$ whose codomain is the Boolean domain $\mathbb{B} = \{0, 1\}$. A return value of 1 indicates that the constraint H is currently satisfied for its constraint members $\mathcal{M}_H$, whereas a return value of 0 indicates that the constraint is currently not satisfied.

For example, in the case of an Alignment constraint such as in Figure 3.26, where the constraint members $\mathcal{M}$ are supposed to be aligned by their bottom edge, the verification function $v_{\text{Alignment}}$ can be expressed as

$$v_{\text{Alignment}}(\mathcal{M}) \;=\; \begin{cases} 1 & \Leftrightarrow \;\; \exists y \in \mathbb{R} : \forall P \in \mathcal{M}, \bot(\square P) = y \\ 0 & \Leftrightarrow \;\; \text{otherwise.} \end{cases} \tag{3.58}$$

If $\mathcal{H}_P$ denotes the set of all hard constraints that have been imposed on a participant P, i.e.,

$$\mathcal{H}_P = \{H \mid H \in \mathcal{H} \wedge P \in \mathcal{M}_H\}, \tag{3.59}$$

with $\mathcal{H}$ representing the set of all hard constraints in the current design, then participant P is in a state of noncompliance if

$$\exists H \in \mathcal{H}_P : v_{t_H}(\mathcal{M}_H) = 0. \tag{3.60}$$

Participant's Desire 5 (Noncompliance):
If a participant does not violate any explicitly formulated hard design constraint, then the participant is referred to as compliant. *During the interaction, every participant strives to act in a way such that it becomes compliant. So, the desire of a participant P regarding noncompliance is to achieve that*

$$\forall H \in \mathcal{H}_P,\ v_{t_H}(\mathcal{M}_H) = 1. \tag{3.61}$$

Using $\mathcal{H}_P^{\ddagger} = \{H \in \mathcal{H}_P \mid v_{t_H}(\mathcal{M}_H) = 0\}$ to denote the set of hard constraints imposed on participant P that are currently not satisfied, P is compliant if that set is empty:

$$\mathcal{H}_P^{\ddagger} = \emptyset. \tag{3.62}$$

As has been stated in Chapter 2 of [1], it is the essential aim of this thesis to provide the means for a comprehensive consideration of design constraints. To that effect, the conception of *noncompliance* represents only one of several instruments. While it should again be remarked, that each participant is supposed to take care of its own particular design constraints implicitly, there are also multiple distinct flavors of explicit constraint consideration in SWARM:

- As already mentioned above, wirelength minimization is achieved by the conception of turmoil, while minimization of the total area is addressed through the successive tightening of the layout zone. Wirelength minimization and area minimization represent optimization goals, also referred to as soft constraints.
- Noncompliance, as an influencing factor, can be utilized to target hard design constraints. In that regard, noncompliance is feasible for constraints that are supposed to be satisfiable via SWARM's native catalog of actions (which will be described in Section 3.3.3).
- Certain design constraints cannot be adequately covered by SWARM's native catalog of actions. However, for such constraints a participant may provide and pursue its own dedicated kind of action. An example is the action called *Imitation*, by which a participant mimics another participant's transformations, as will be discussed in Section 3.3.3.2.
- Design constraints pertaining to the distance between two participants, can be included in the model of *tension* (see Section 3.3.1.2). Apart from the *emphasis* value (which acts as a "soft" quantifier), a hard restriction –such as a maximum distance– can be imposed by overriding a connection's *relaxation threshold* with the desired value.

- Some design constraints just imply, that a participant may not assume certain layout variants. That is to say, the participant is not allowed to exploit its entire *variability* during the interaction. Instead, the permitted variability is reduced to a subset of the participant's total variability. For example, a Current Mirror may be prevented from deforming itself into a single-row variant for reasons of matching.

In summary, if a participant is not affected by any of the above influencing factors (interference, turmoil, protrusion, wounds, noncompliance), then the participant is said to be *contented*. Otherwise, the participant strives for action because it is *discontented*. When contented, no action is required, but nonetheless the participant attempts to perform a movement in this case if possible: to center itself within its *free peripheral space* (which will now be covered in Section 3.3.2). If that is not possible, the participant lingers where it is. Although the centering does not improve the participant's condition, it is regarded as a betterment of its situation, imagining that the participant "feels best" when it can equalize the distances to its nearby neighbors.

3.3.2 Perception of the Free Peripheral Space

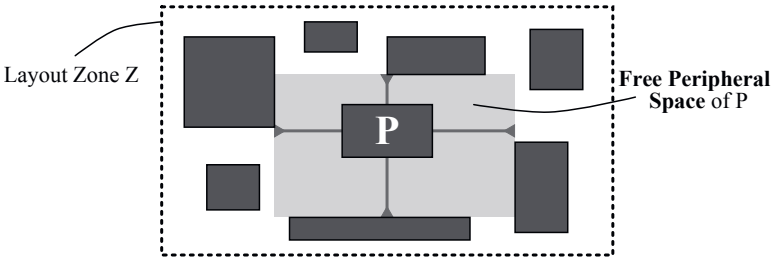


Figure 3.27: The free peripheral space of a participant P is the vacant area around it.

The basis for a participant's exploration of possible actions is the vacant area around it (and, for actions involving other participants, also the vacant areas around *them*). While there are various possible conceptions of how this *free peripheral space* could be specified, SWARM provides the following unambiguous definition, in accordance with the exemplary depiction of Figure 3.27.

Definition 3.3. *Given the layout zone Z as a rectilinear polygon, let P be a participant located (at least partially) inside Z. Let B be the rectangular* bounding box around the part *of P that is completely inside Z. For every* edge *E of B, let the* corridor *K_E be a rectangle beginning at E and sprawling away from P to infinity with a width equal to the length of E. Let $\mathcal{K}$ be a set containing the four* corridors *of P and let $\mathcal{U}$ be a* set of obstacles *containing the complement of Z as well as the bounding boxes of all participants except P. Then, the free peripheral*

space S_P of P is uniquely defined as the largest possible rectangle around P not containing any intersection $\mathcal{K} \sqcap \mathcal{U}$.

3.3.2.1 Geometrical Recipe for Perceiving the Free Peripheral Space

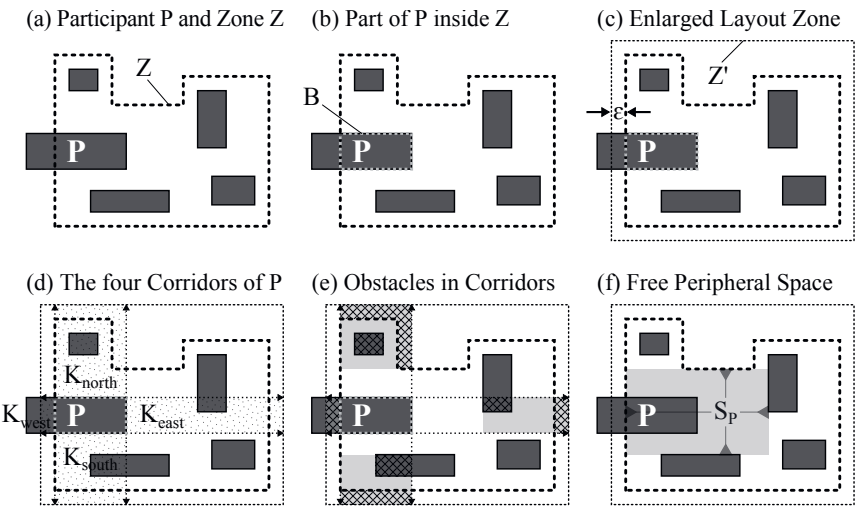


Figure 3.28: A participant's geometrical recipe for perceiving its free peripheral space.

Geometrically, a participant can perceive its free peripheral space as described below and illustrated in Figure 3.28. For a participant P and a layout zone Z –such as in Figure 3.28 (a)– the bounding box around the part of P inside Z –see (b)– is given by

$$B = \boxdot(Z \sqcap \boxdot P) \tag{3.63}$$

and the four edges of B can be determined as lines $\big((x_1, y_1), (x_2, y_2)\big)$ via

$$E_{\text{north}} = \big((\vdash B, \top B), (\dashv B, \top B)\big), \tag{3.64a}$$
$$E_{\text{east}} = \big((\dashv B, \bot B), (\dashv B, \top B)\big), \tag{3.64b}$$
$$E_{\text{south}} = \big((\vdash B, \bot B), (\dashv B, \bot B)\big), \tag{3.64c}$$
$$E_{\text{west}} = \big((\vdash B, \bot B), (\vdash B, \top B)\big). \tag{3.64d}$$

Notwithstanding the theoretical definition above, the corridors of these four edges cannot sprawl to infinity in practice. For calculating the free peripheral space, the corridors just need to go *beyond* the zone outline Z, no matter how far. For that purpose, as shown in image (c), it is

suitable to define an auxiliary rectangular zone outline

$$Z' = \circledcirc_{\varepsilon}(\square Z) \tag{3.65}$$

which is obtained by getting the bounding box of Z and enlarging it by an arbitrarily small positive number $\varepsilon > 0$. Then, the four corridors of $\mathcal{K}$ –depicted in (d)– can be defined as

$$K_{\text{north}} = \big((\vdash B, \top B), (\dashv B, \top Z')\big), \tag{3.66a}$$
$$K_{\text{east}} = \big((\dashv B, \bot B), (\dashv Z', \top B)\big), \tag{3.66b}$$
$$K_{\text{south}} = \big((\vdash B, \bot Z'), (\dashv B, \bot B)\big), \tag{3.66c}$$
$$K_{\text{west}} = \big((\vdash Z', \bot B), (\vdash B, \top B)\big). \tag{3.66d}$$

If $\mathcal{P}$ represents the set of all participants, then the set of obstacles $\mathcal{U}$ containing the complement of Z and the bounding boxes of all participants expect P is given by

$$\mathcal{U} \; = \; \overline{Z} \; \sqcup \; \square(\mathcal{P} - P) \tag{3.67}$$

and allows to determine the obstacles' intersections with each individual corridor via

$$\mathcal{U}_{\text{north}} = K_{\text{north}} \sqcap \mathcal{U}, \tag{3.68a}$$
$$\mathcal{U}_{\text{east}} = K_{\text{east}} \sqcap \mathcal{U}, \tag{3.68b}$$
$$\mathcal{U}_{\text{south}} = K_{\text{south}} \sqcap \mathcal{U}, \tag{3.68c}$$
$$\mathcal{U}_{\text{west}} = K_{\text{west}} \sqcap \mathcal{U}, \tag{3.68d}$$

as indicated by the criss-cross hatching in image (e). With the bounding boxes around these intersections –lightly grayed in (e)–, the four bounding box edges that face towards P demarcate the free peripheral space S_P, which can thus be perceived by evaluating the expression

$$S_P = \Big(\big(\dashv(\square\mathcal{U}_{west}), \top(\square\mathcal{U}_{south})\big), \big(\vdash(\square\mathcal{U}_{east}), \bot(\square\mathcal{U}_{north})\big)\Big) \tag{3.69}$$

and therefore yields the free peripheral space S_P illustrated in image (f) of Figure 3.28.

3.3.2.2 Pervasion (Obstacles in the Free Peripheral Space)

Although the given definition of free peripheral space is quite convenient, it may occur that the free peripheral space is not really "free" for the reason of being pervaded by an obstacle. This situation is denoted as *pervasion* and can be divided into two different cases:

- The first case is illustrated in Figure 3.29. Image (a) shows a participant P being overlapped by another participant P'. This means that P is in a state of *interference* (see Section 3.3.1.1), and therefore the overlapping part of P' intersects the free peripheral

space of P. The situation is similar if the obstacle is not another participant but the complement of the layout zone. As can be seen in image (b), participant P may be in a state of *protrusion* (see Section 3.3.1.3) where the part of P inside Z is not rectangular. In this situation, the bounding box B around that part –and thus, the free peripheral space as well– also protrudes beyond the given layout zone. So, in both Figure 3.29 (a) and Figure 3.29 (b) the free peripheral space is not truly vacant due to the impaired *condition* of the participant.

- The second case is depicted in Figure 3.30, which points out the fact that the free peripheral space can contain up to four so-called *blind spots*. These blind spots are comprised of $S_P \hookleftarrow \mathcal{K}$ (i.e., those parts of the free peripheral space outside the four corridors K_{north}, K_{east}, K_{south}, K_{west}). Obstacles in a blind spot, such as participant P' in image (a), are accidentally overlooked when P perceives its free peripheral space. As above, a similar situation arises when the obstacle is the complement of the layout zone. Unless the zone outline is strictly rectangular, the free peripheral space can inadvertently exceed the bounds of the layout zone as in the example of image (b). So, in both Figure 3.30 (a) and Figure 3.30 (b) the free peripheral space is not entirely vacant because of its blind spots.

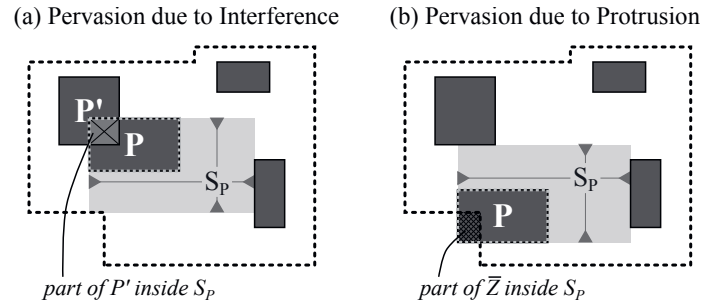


Figure 3.29: Pervasion of a participant's free peripheral space due to its bad condition.

Luckily, neither of these cases is problematic, since eventual situations of interference or protrusion will be detected before a participant performs an action. Furthermore, it should be noted that blind spots become less and less troublesome throughout the course of a SWARM run: due to the successive tightening of Z, the arrangement of participants becomes increasingly compact and thus the area of blind spots approaches zero. This observation is visualized in Figure 3.31. In the initial constellation (a), there is a high degree of pervasion: almost the entire layout zone is littered with blind spots, such that every single participant is overlooked by at least one of its neighbors. As shown by the intermediate constellation of (b), where only five out of the nine participants pervade another participant's free peripheral space, the degree of pervasion becomes smaller throughout the self-organization flow. In the final constellation (c),

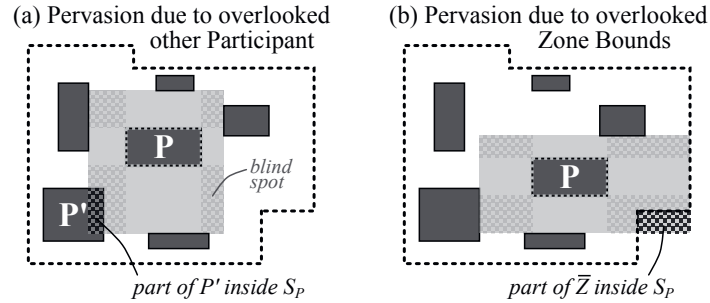


Figure 3.30: Pervasion of a participant's free peripheral space due to blind spots.

the arrangement is so compact that no pervasion at all is encountered in the relatively small areas of the remaining blind spots.

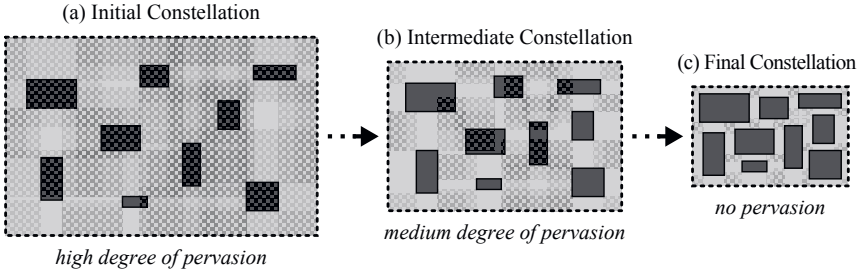


Figure 3.31: Successive decline of free peripheral space pervasion caused by blind spots.

In addition to the above two cases, one can identify a third case, in which the idea of free peripheral space becomes completely moot. That is, if a participant has become *lost* (again see Section 3.3.1.3), S_P entirely lies beyond the given layout zone. In this case, a dedicated *Re-entering* action, which does not rely on the participant's free peripheral space, is used to hurl the participant back into the layout zone as will now be described in Section 3.3.3.

3.3.3 Exploration and Evaluation of Possible Actions

Every action that a participant can perform is basically a set of transformations for all participants $\mathcal{P}_\iota$ *involved* in the action. As introduced in Section 3.2.3.3, each transformation T is a triple $T = (M, R, D)$ that may comprise a movement M, a rotation R, and a deformation D:

- The movement M is a two-dimensional vector $M = (\Delta x, \Delta y)$ that displaces the participant without altering its aspect ratio. To obtain another aspect ratio, a rotation or a deformation need to be included in the transformation.

- The rotation R either rotates the participant around its center point or preserves its orientation. If the participant can be treated as a "black box" during the self-organization phase of the SWARM run, then it suffices to consider rotations $R \in \{0°, 90°\}$ which either keep or invert the participant's aspect ratio.
- The deformation $D \in \{V_1, V_2, \ldots, V_d\}$ refers to one of the d layout variants that the participant can assume (as given by its cumulative variability $\widetilde{\mathcal{V}}$). The higher the ductility d is, the more variants are available for potential deformations. A deformation always changes the participant's aspect ratio to that of the assumed layout variant but does not dislocate its center point.

Depending on the prospective aspect ratio, the participant's next measure is to spot an assortment of possible locations it might eventually decide to head for. Serving this purpose, different kinds of actions are at hand, including SWARM's catalog of native actions (Section 3.3.3.1), constraint-specific custom actions (Section 3.3.3.2), and a special kind of functionality to account for full variability (Section 3.3.3.3).

Every action that the participant explores, also has to be evaluated. First and foremost, this is necessary in order to see whether an action is *valid* at all.

Definition 3.4. *An action is* valid *if it leads the participant into a location that is devoid of* protrusion*,* wounds *and* noncompliance*. Otherwise the action is* invalid*. If the participant currently is in an invalid location, then an invalid action that somehow improves the participant's condition (as done by the* ***Re-entering*** *action described below) is called* tolerable*. If an action is exacted by a constraint (such as the* ***Imitation*** *action in Section 3.3.3.2), the action is* mandatory*. An action is defined as* acceptable *if it is valid, tolerable, or mandatory.*

Actions that are not acceptable can be immediately discarded. Among the acceptable actions, the respective evaluation then allows to compare the actions' prospective *interference* and *turmoil*. This decides, which action will be finally performed, if an action is performed at all (Section 3.3.4).

3.3.3.1 Native Actions

Although particular design requirements may necessitate dedicated actions (such as the already mentioned *Imitation* that will be discussed in Section 3.3.3.2), SWARM implements a fundamental catalog of nine different actions considered to be adequate as a kind of natural, "instinctive" demeanor. Regarding a participant P and a layout zone Z, each of these actions is subsequently discussed in a paragraph of its own, with each paragraph providing

- a description of the respective action,
- the instructions to put the action into practice,
- additional comments about the action in general, and

- an illustration that depicts the action in a demonstrative example.

As already mentioned, some actions only involve participant P whereas other actions may also involve further participants. In either case, participant P (i.e., the participant who decides about the action to be performed) is denoted as the *leading participant*.

It should again be emphasized that each of the following paragraphs concentrates on *how* an action is explored, but not *when* that action is explored nor when (if at all) it gets indeed performed within SWARM's common action scheme. These open questions will be answered in Section 3.3.4, covering the preference of actions as well as the overall flow of action execution.

Re-entering

Description: *Re-entering* is performed when participant P is *lost*. This action has the sole aim of catapulting P back into the given layout zone Z at the nearest possible location. *Re-entering* thus remedies protrusion, while disregarding all other *influencing factors* covered in Section 3.3.1.

Instructions:

(1) If participant P is lost, then for each edge E of the rectilinear layout zone Z, determine whether E is a horizontal edge or a vertical edge.

(2) If E is a horizontal edge, determine and memorize a purely vertical move by which P aligns its southern (if P is below E) or northern (if P is above E) edge with E. If E is a vertical edge, determine and memorize a purely horizontal move by which P aligns its western (if P is to the left of E) or eastern (if P is to the right of E) edge with E.

(3) If a memorized move would lead P into a state of protrusion, correct the move such that it leads P to an allegedly *safe* location. For a vertical move, this implies an offset to the right (if P is to the left of E) or to the left (if P is to the right of E). For a horizontal move, this implies an offset upwards (if P is below E) or downwards (if P is above E). Formally, one can say:
$\forall E \in Z, M_E = (\Delta x, \Delta y)$ with

$$\Delta x = \begin{cases} \max(\vdash E - \vdash(\textcircled{P}), 0) + \min(\dashv E - \dashv(\textcircled{P}), 0) & \Leftrightarrow \ E \text{ is horizontal} \\ \max(x_E - \vdash(\textcircled{P}), 0) + \min(x_E - \dashv(\textcircled{P}), 0) & \Leftrightarrow \ E \text{ is vertical} \end{cases}$$

$$\Delta y = \begin{cases} \max(y_E - \perp(\textcircled{P}), 0) + \min(y_E - \top(\textcircled{P}), 0) & \Leftrightarrow \ E \text{ is horizontal} \\ \max(\perp E - \perp(\textcircled{P}), 0) + \min(\top E - \top(\textcircled{P}), 0) & \Leftrightarrow \ E \text{ is vertical.} \end{cases}$$

(4) Sort the memorized moves by the Euclidean distance of the movements and let P perform the move with the smallest distance.

Comments: As will be discussed in Section 3.4, the tightening of the layout zone is supposed to be realized in a way such that no participant gets lost through a tightening. However, this is not that easy to accomplish in case the layout zone is a nonrectangular polygon. Keeping the implementation simple, and conceding that a participant can thus get lost, *Re-entering* is a convenient action to remedy such situations.
Of course, *Re-entering* actions can also be explored for participants that are only *prone* but not entirely lost. In some cases, such a move may turn out to be a better alternative than an *Evasion* action (described next). For that reason, *Re-entering* actions can also be encountered in SWARM runs where the layout zone is rectangular.

Illustration:

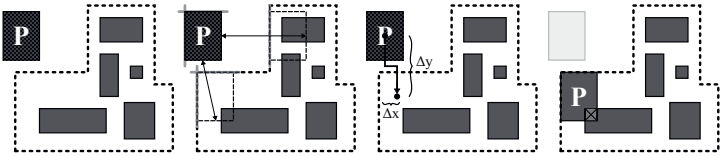


Figure 3.32: Exemplary visualization of a *Re-entering* move.

Evasion

Description: *Evasion* is only applicable when P is prone. With this action, P moves back into the layout area Z, evading other participants by going sideways.

Instructions:

(1) If P is prone, then perceive its free peripheral space S_P (based on the rectangular part B of P that lies inside Z, as described in Section 3.3.2).

(2) Get the width w_P and the height h_P of P. These quantities are relative to the coordinate system and can be formally calculated as follows:

$$w_P = \dashv(\textcircled{i}P) - \vdash(\textcircled{i}P),$$
$$h_P = \top(\textcircled{i}P) - \bot(\textcircled{i}P).$$

(3) Determine the three locations inside S_P by which P aligns its bounding box with the nearest edge or with one of the two nearest vertices of S_P. In case S_P lies to the right of P (as in the illustration below), the set of locations for potential moves is

$$\mathcal{L}_{\text{new}} = \Big\{ \big(\vdash S_P + \tfrac{1}{2}w_P, \top S_P - \tfrac{1}{2}h_P\big), \qquad \textit{// north-west of } S_P$$
$$\big(\vdash S_P + \tfrac{1}{2}w_P, \tfrac{1}{2}(\bot S_P + \top S_P)\big), \quad \textit{// center-west of } S_P$$
$$\big(\vdash S_P + \tfrac{1}{2}w_P, \bot S_P + \tfrac{1}{2}h_P\big) \Big\}. \qquad \textit{// south-west of } S_P$$

For the cases in which S_P lies above, below, or to the left of P, the set of locations is to be determined analogously.

(4) Let P move such that its center point lies at one of the new locations $\mathcal{L}_{\text{new}}$. As is the case with most of the actions that follow, it is up to the participant's decision-making which one of the explored actions will eventually be performed (see Section 3.3.4). Thus, the subsequent action descriptions are only meant to expound the principal idea behind each action, regardless of the question which action will finally be chosen.

Comments: If P protrudes Z at one of its convex vertices (not at one of its edges as shown in the example below), i.e., if $w_B < w_P \wedge h_B < h_P$, then S_P does neither definitely lie above, nor below, nor to the left, nor to the right of P. In that case, the participant should try to align its bounding box with the nearest vertex and with one of the two nearest edges of S_P.

If Z is not a rectangular layout zone, it can happen that P protrudes Z at a concave vertex. This represents a case of pervasion due to protrusion (as depicted in Figure 3.29). To address such situations, the *Evasion* action would have to be enhanced, but for convenience it is also possible to skip the *Evasion* attempt in favor of one of the other actions which may just as well help P to get back into the layout zone.

Illustration:

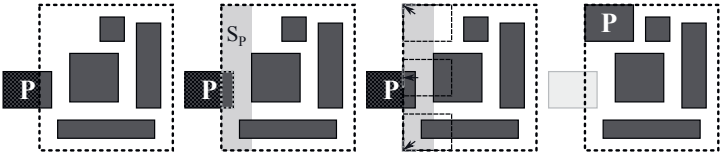


Figure 3.33: Exemplary visualization of a *Evasion* move.

Centering

Description: *Centering* is an elementary action where P aligns its center point with the center point of its free peripheral space. As such, *Centering* is not necessarily meant to solve conflict situations, but rather to balance the distances between participants when they are *contented* (also see the remark at the end of Section 3.3.1).

Instructions:
(1) Perceive P's free peripheral space S_P (as explained in Section 3.3.2).
(2) Determine the center point of S_P to go for it as the new location L_{new}:

$$L_{\text{new}} = \big(\tfrac{1}{2}(\vdash S_P + \dashv S_P), \tfrac{1}{2}(\perp S_P + \top S_P)\big).$$

(3) Let P move such that its center point lies at the new location L_{new}.

Comments: A mandatory concern here and in general is the definition of a *minimal movement distance* m to prevent infinitesimal actions. This is not only for reasons of performance, but to prevent a group of contented participants from *Centering* themselves ad infinitum. As the current implementation of SWARM shows, it is feasible to correlate m with the amount of free space in Z such that it changes dynamically during the self-organization.

Illustration:

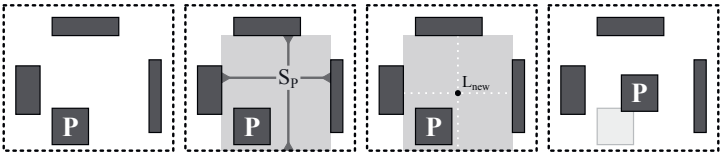


Figure 3.34: Exemplary visualization of a *Centering* move.

Lingering

Description: *Lingering* occurs if P is contented but cannot perform a *Centering* (e.g., due to prospective interference with an obstacle in a *blind spot*). In that case, P deliberately does nothing but to stay where it is, waiting for the next round of interaction.

Instructions:
(1) The situation is expected to be such that P is contented and attempts to perform a *Centering*.
(2) Determine if P would become discontented through the *Centering*.
(3) If P would become discontented, let P stay at its current location.

Comments: It should again be emphasized that *Lingering* is an action where the participant remains idle intentionally, as opposed to the case where a participant cannot perform an action although it would like to do so for the reason of being discontented.

Illustration:

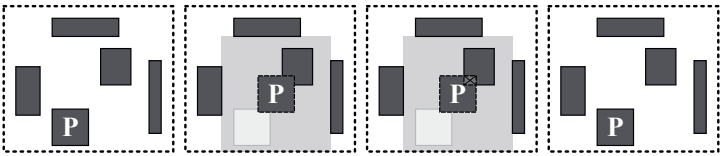


Figure 3.35: Exemplary visualization of a *Lingering* move.

Budging

Description: *Budging* makes P spot additional vacant room by perceiving the four free peripheral spaces from the viewpoints of its four corners. Then, P tries to slip into an appropriate location within that "secondary" free peripheral space.

Instructions:
(1) For each vertex of P's bounding box, perceive the secondary free peripheral space from the viewpoint of the vertex (with infinitely narrow *corridors* emanating from the vertex).
(2) Find a suitable location L_{new} inside S_P^{NE} (free peripheral space from the viewpoint of the north-eastern vertex), S_P^{SE} (of the south-eastern vertex), S_P^{SW} (of the south-western vertex), or S_P^{NW} (of the north-western vertex). One basal location is the center point of the free peripheral space (see comments below).
(3) Let P move such that its center point lies at the new location L_{new}.

Comments: Of course, vertices that lie inside another participant's bounding box, can be discarded right off the bat. For the other vertices, several locations inside the respective free peripheral space (in addition to its center point) might be probed as suitable targets. The specification, which of these targets are to be explored, is denoted as an *exploration plan* (also see the formal definition in Section 3.3.4.2).

For example, an exhaustive exploration plan would be letting P try to align its center point, its northern edge, north-eastern vertex, eastern edge, south-eastern vertex, southern edge, south-western vertex, western edge, and north-western vertex with the corresponding element of the free peripheral space. However, in the current implementation of SWARM, only the center point of the participant's free peripheral space is taken into consideration as the potential new location for P.

This is prompted by the desire to find an adequate trade-off for a participant's efforts on the local level. On the one hand, it is important to explore a rich fund of diverse actions that the participant may then choose from. On the other hand, a participant should refrain from investing too much labor into exploring a single kind of action, because the impairments of a participant's condition (e.g., interference or protrusion) are ultimately remediated by the totality of all participants' actions across several tightening-settlement cycles.

Balancing the three different core concepts (to bring the variability of the *responsive modules*, the possibilities of the *module interaction*, and the tightening conduct of the *interaction control* organ into a well-adjusted flow of self-organization) is a particular necessity but also a powerful setscrew of the SWARM methodology.

Illustration:

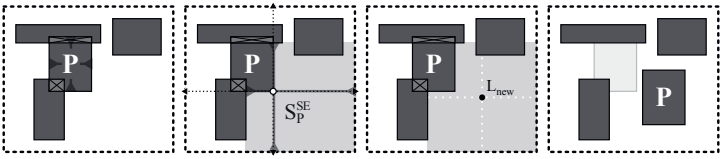


Figure 3.36: Exemplary visualization of a *Budging* move.

Swapping

Description: *Swapping* lets P trade places with another participant P'. That is, P jumps into the free peripheral space of P' and P' in turn jumps into the free peripheral space of P.

Instructions:

(1) Perceive the free peripheral space S_P of P and, considering another participant P', perceive the free peripheral space $S_{P'}$ of P'.

(2) Determine if a *Swapping* with P' should be explored. In that case, the *Swapping* is said to be *promising* (see comments below).

(3) If a *Swapping* appears to be promising, determine the center point of $S_{P'}$ as the new location L_{new} for P, and determine the center point of S_P as the new location L'_{new} for P':

$$L_{\text{new}} = \big(\tfrac{1}{2}(\vdash S_{P'} + \dashv S_{P'}), \tfrac{1}{2}(\bot S_{P'} + \top S_{P'})\big),$$
$$L'_{\text{new}} = \big(\tfrac{1}{2}(\vdash S_P + \dashv S_P), \tfrac{1}{2}(\bot S_P + \top S_P)\big).$$

(4) Let P move with its center point to the new location L_{new} and pull P' with its center point to the new location L'_{new}.

Comments: In general, it is worthwhile to explore *Swappings* with all other participants. But as indicated above, it might be feasible to elide a presumably futile *Swapping* to save computation time – according to a specific heuristic. As an example for such a heuristic, one may try to make a prediction by comparing the size of P and S_P with the size of P' and $S_{P'}$, and then only explore that *Swapping* if the deviation regarding size does not exceed a certain proportion.

An action worth being explored according to such a –or a similar– heuristic is qualified as promising. Here, it must be clearly understood, that the adjective *promising* approves the exploration of an action, not the execution of that action (which is authorized by the predicate *acceptable*, as specified in Definition 3.4). The postulated saving in computation time results from the fact, that comparing two sizes is computationally less expensive than having to assess the prospective conditions of the two participants (if the action is indeed explored and therefore also has to be evaluated – considering all influencing factors).

Unfortunately, it is not trivial to find an appropriate rule of thumb that reliably tells whether a *Swapping* would probably be useful or not. Experiments show that surprisingly often P opts for a *Swapping* that would otherwise have been rejected by a heuristic such as comparing sizes (or aspect ratios, or both, for that matter). On these grounds, the current implementation of SWARM by default explores all possible *Swappings* with all other participants. So, the issue of finding an adequate heuristic is worth being kept in mind and may be examined in future research, but is not broached any further within the scope of this thesis.

Another remark should be made about the fact that a *Swapping* need not necessarily imply that the two participants *center* themselves within the free peripheral space of their counterpart. Instead, the participants might pursue a more exhaustive exploration plan as also discussed in the comments of the *Budging* action above.

Illustration:

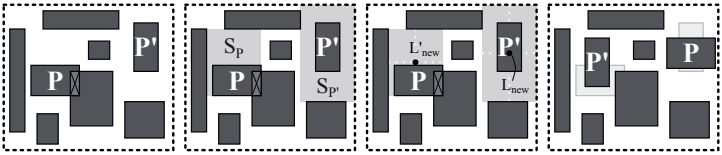


Figure 3.37: Exemplary visualization of a *Swapping* move.

Pairing

Description: *Pairing* means that P jumps next to another participant P', thereby pushing P' aside such that both participants then share the free peripheral space of P'.

Instructions:

(1) Considering another participant P', determine if a horizontal *Pairing*, a vertical *Pairing*, or both (or none) are promising and should therefore be explored.

(2) For a horizontal *Pairing*, get the width w_P of P and the width $w_{P'}$ of P'. For a vertical *Pairing*, get the height h_P of P and the height $h_{P'}$ of P'.

(3) For a horizontal *Pairing*, determine a horizontal movement where P' is pushed to the right, and a horizontal movement where P' is pushed to the left (such that P and P' will be centered around the vertical symmetry axis of P' after the *Pairing*). The respective Δx is

$$\Delta x = -\tfrac{1}{2}w_{P'} + \tfrac{1}{2}(w_P + w_{P'}) = \tfrac{1}{2}w_P$$
$$\text{and} \quad \Delta x = \tfrac{1}{2}w_{P'} - \tfrac{1}{2}(w_P + w_{P'}) = -\tfrac{1}{2}w_P$$

such that the potential movements $M_{P'}$ for P' can be shortly written as

$$M_{P'} = (\pm\tfrac{1}{2}w_P, 0).$$

With a vertical *Pairing*, the potential movements for an upwards-push and a downwards-push can be equivalently given as

$$M_{P'} = (0, \pm\tfrac{1}{2}h_P).$$

(4) Calculate the new location L_{new} for P accordingly, which can be done in a way that is analogous to the calculation of $M_{P'}$ above. For a horizontal *Pairing*, the respective locations accompanying a rightwards-push and a leftwards-push can thus be compactly expressed as

$$L_{\text{new}} = \Big(\tfrac{1}{2}\big(\vdash(\square P') + \dashv(\square P')\big) \mp \tfrac{1}{2}w_{P'}, \tfrac{1}{2}\big(\perp(\square P') + \top(\square P')\big)\Big)$$

while the respective locations for a vertical *Pairing* can be calculated via

$$L_{\text{new}} = \Big(\tfrac{1}{2}\big(\vdash(\square P') + \dashv(\square P')\big), \tfrac{1}{2}\big(\perp(\square P') + \top(\square P')\big) \mp \tfrac{1}{2}h_{P'}\Big).$$

(5) Push P' by $M_{P'}$ and let P move with its center point to the new location L_{new}.

Comments: In the basic form of *Pairing*, as described above and as currently implemented in SWARM, participant P aligns its vertical (horizontal) center with the vertical (horizontal) center of P' when performing a horizontal (vertical) *Pairing*. Additionally, it would also be possible to follow a more comprehensive exploration plan and let P explore further options of alignment, where P aligns its northern or southern edge with the corresponding edge of P' (for a horizontal *Pairing*), and likewise its western or eastern edge (for a vertical *Pairing*).
On the other hand, as indicated above, the participant might dismiss the exploration of a *Pairing* if it does not look promising at first sight (based on a heuristic such as a comparison of the participants' aspect ratios, for example). This idea has also been discussed in the comments of the *Swapping* action, with the mentioned caveats unfortunately also applying here to a similar extent. Therefore, the participants in the current implementation of SWARM by default decide that all possible *Pairings* are promising (and thus worthy of exploration). Still, the idea of not exploring certain actions based on some heuristic to save computation time might be investigated in the future.

Illustration:

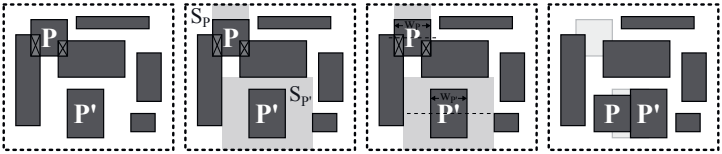


Figure 3.38: Exemplary visualization of a *Pairing* move.

Hustling

Description: *Hustling* is an action by which P remains where it is, but pushes away all other participants that currently overlap P. These other participants are pushed as far as necessary such that P gets rid of their interference and becomes *clear* again. So far, *Hustling* is the only action that may involve more than two participants.

Instructions:

(1) For every partipant P' that overlaps P, measure the width $w'_{\sqcap}$ and the height $h'_{\sqcap}$ of the overlap region $\boxplus P \sqcap \boxplus P'$.

(2) For each of these participants, check whether the horizontal extent of the overlap is larger than the vertical extent. In the first case, determine a vertical movement to get rid of P'. In the second case, determine a horizontal movement for that purpose. If $\mathcal{P}'$ denotes the set of participants that overlap P, one can formally say: $\forall P' \in \mathcal{P}'$ the respective movement $M_{P'}$ is

$$M_{P'} = \begin{cases} (0, h'_{\sqcap}) \text{ or } (0, -h'_{\sqcap}) & \Leftrightarrow \quad w'_{\sqcap} > h'_{\sqcap} \\ (w'_{\sqcap}, 0) \text{ or } (-w'_{\sqcap}, 0) & \Leftrightarrow \quad \text{otherwise} \end{cases}$$

where the polarity of the pushing depends on whether P' overlaps P from the northern ($\Rightarrow h'_{\sqcap}$), southern ($\Rightarrow -h'_{\sqcap}$), eastern ($\Rightarrow w'_{\sqcap}$), or western ($\Rightarrow -w'_{\sqcap}$) side of P.

(3) Push every overlapping participant P' away via the respective $M_{P'}$.

Comments: Similar to some of the other actions discussed so far (i.e., ***Budging***, ***Swapping***, ***Pairing***), a more exhaustive exploration plan might be pursued here. For example, P might try both the horizontal and the vertical movement above, regardless of the overlap's dimensions. In addition, P may even attempt to exert a movement $M_{P'} = (w'_{\sqcap}, h'_{\sqcap})$ such that P' is pushed away both horizontally and vertically. In its current implementation however, SWARM sticks to the instructions above, exploring either a vertical movement or a horizontal movement for every overlapping participant P', but not both. As with the other actions mentioned at the beginning of this comment, this is done to keep the participant's exploration effort rather low for this particular kind of action.

There are situations, where P does not become completely *clear* by performing a ***Hustling***, for example when P is *properly enclosed* by another participant P' (i.e., $P \sqsubset P'$). Generally speaking, this is the case if P pushes P' away in a direction where P' juts beyond P on *both* sides of P. However, such a situation is not that problematic since it does not jeopardize the entire ***Hustling*** action: despite the remaining overlap, the ***Hustling*** may still be better than all other actions explored by P.

Illustration:

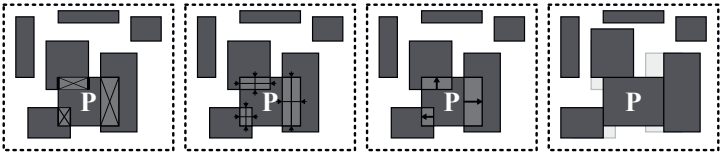


Figure 3.39: Exemplary visualization of a ***Hustling*** move.

Yielding

Description: *Yielding* is only done if P suffers from interference but cannot find an appropriate action. In this case, P determines the polygon Y_P (the so-called *yielding region*, which circumscribes the part of P inside layout zone Z excluding the overlap regions) and aligns its center with the geometric centroid (i.e., the barycenter) of Y_P.

Instructions:

(1) If P is in a state of interference, determine the polygon Y_P. If $\mathcal{P}'$ denotes the set of participants that overlap P, Y_P can be formally obtained with

$$Y_P = \complement P \hookleftarrow (\overline{Z} \sqcup \complement \mathcal{P}').$$

(2) Find the geometric centroid of the yielding region Y_P, which is to be used as the new location L_{new} for P. Calculating the signed area of Y_P via

$$[Y_P]^{\pm} = \frac{1}{2} \sum_{i=1}^{n} (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i)$$

where $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are the vertices along the contour of Y_P (with $x_{n+1} = x_1$ and $y_{n+1} = y_1$), the centroid of Y_P is given by the coordinates

$$x_c = \frac{1}{6\,[Y_P]^{\pm}} \sum_{i=1}^{n} (x_i + x_{i+1})\ (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i),$$
$$y_c = \frac{1}{6\,[Y_P]^{\pm}} \sum_{i=1}^{n} (y_i + y_{i+1})\ (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i).$$

(3) Let P move with its center point to the new location $L_{\text{new}} = (x_c, y_c)$.

Comments: The name of this action suggests that the participant abandons itself to some sort of makeshift maneuver. Although this can not be entirely disavowed (since indeed no helpful action could be found so far by the participant), *Yielding* has a rightful place among the catalog of native actions and should not be frivolously repudiated, because a *Yielding* often has a reviving effect on the self-organization flow when stuck in a period of stagnation.

One reason for this is that a yielding participant ignores all influencing factors apart from protrusion (like in a *Re-entering*). Another reason is that *Yielding* typically leads to interference with all neighboring participants that P is currently oppressed by (as in the example below). In many cases, this animates the neighbors to move away a bit instead of continuing to shove P around in endless repetitive circles.

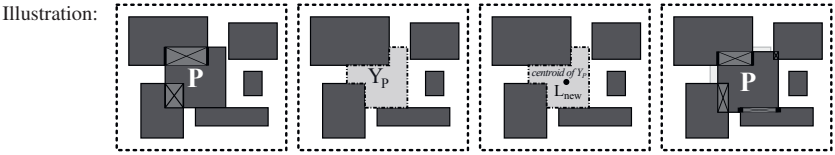


Figure 3.40: Exemplary visualization of a *Yielding* move.

Although each of the above illustrations depicts mere movement-based transformations, every kind of action can also involve a rotation and/or a deformation. During a participant's action exploration, this is basically taken into account by *first* determining the would-be aspect ratio and *then* spotting the new location for the participant. This proceeding is mandatory for those kinds of action, where the new location depends on the participant's aspect ratio, as in the *Evasion* example given by Figure 3.41 (a). On the other hand, there are also actions where the new location is independent from the aspect ratio, like with the *Centering* example shown in Figure 3.41 (b). For these actions, it is –of course– wise not to recalculate the new location again and again for all possible orientations and layout variants that the participant can assume.

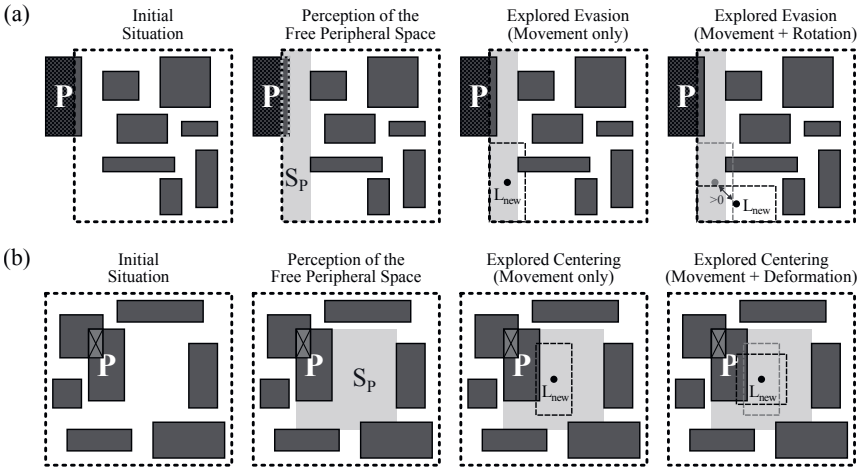


Figure 3.41: The new location of a participant can be (a) dependent or (b) independent of its aspect ratio.

As described above, certain actions (e.g., *Evasion*) make the participant align itself with the layout zone Z, such that the participant is definitely *safe* afterwards. But with most of the other

actions, the participant takes the risk of getting into a state of protrusion. While it is unlikely to get completely *lost* (which can only happen in the course of a *Pairing*), the participant must often face the chance of becoming *prone*. In such a case, the action need not be turned down, but can be corrected via the *protrusion extent* ψ introduced in Section 3.3.1.3. This is done by simply subtracting the horizontal and the vertical part of the protrusion extent from the respective element of the movement vector such that

$$M = (\Delta x - \psi_x, \Delta y - \psi_y). \tag{3.70}$$

However, one must pay attention here: if ψ_x and ψ_y are not signed (since equation 3.53 returns absolute values), the action correction above only works if the participant protrudes Z in upward or rightward direction. Otherwise, the respective protrusion extent has to be *added* to Δx and Δy, not subtracted. An example of an action correction is presented in Figure 3.42. Initially, P and P' both suffer from interference with other participants. After perceiving the free peripheral spaces S_P and $S_{P'}$, P explores a potential *Swapping* with P'. Originally, the action would lead both participants into a state of protrusion, but when adjusting the movements by $-\psi_y$ and ψ_x respectively, the prospective situation is *safe* for P and P'. Although the action correction again entails some overlaps with other participants, the *Swapping* might still be performed because the overall interference is reduced thereby.

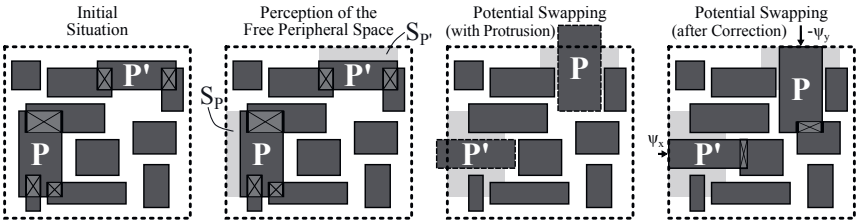


Figure 3.42: Action correction example: P's move is corrected downwards, the move of P' rightwards.

To sum it up, Table 3.7 provides an overview that lists SWARM's catalog of native actions. For each action, the table indicates how many participants would be moved (i.e., transformed) thereby, under which condition of the *leading participant* the action is explored, under which prospective condition of the *involved* participants the action is deemed to be *acceptable*, and whether this predicate stems from the fact that the action is *valid* or *tolerable*. The third possibility according to Definition 3.4 –the action being *mandatory*– so far only applies to custom actions, which will now be discussed in Section 3.3.3.2.

Table 3.7: Overview of SWARM's catalog of native actions.

Action	Number of moved Participants	Explored if leading Participant P is	Acceptable if all involved Participants will be	Action is
Re-entering	1	lost	safe	tolerable
Evasion	1	prone	safe & healthy & compliant	valid
Centering	1	not lost	safe & healthy & compliant	valid
Lingering	0	contented	contented	valid
Budging	1	safe	safe & healthy & compliant	valid
Swapping	2	safe	safe & healthy & compliant	valid
Pairing	2	safe	safe & healthy & compliant	valid
Hustling	$1 \ldots n$	safe & not clear	safe & healthy & compliant	valid
Yielding	1	not lost & not clear	safe	tolerable

3.3.3.2 Custom Actions

Depending on the layout problem at hand, it may occur that SWARM's catalog of native actions is not adequate when a participant needs to satisfy some particularly rigorous design constraints. In such cases, it can be advisable to equip the participant with its own particular behavior by implementing some custom action that the participant is to pursue. This idea will now be elucidated by means of the *Imitation* action that was already brought up in Section 3.3.1.5.

For matching reasons, it may be demanded that a participant behaves symmetrically to another participant, such that (1) the two participants have equal dimensions, that (2) they are aligned in one direction, and that –in the orthogonal direction– (3) they are kept at the same distance on either side of a certain axis. There is no need to point out that the odds of meeting all these demands via SWARM's native actions are quite bad. An alternative is to let the two participants be managed by a superordinate governing module such as the Symmetric Pair known from Figure 3.10. However, that module is meant to keep its two submodules closely together with a certain fixed space in between. This approach would be infeasible if there was another module between the two participants, the more so when that module changes its aspect ratio throughout the self-organization. This is where the *Imitation* action comes to the rescue.

Imitation

Description: *Imitation* is a custom action by which a participant P mimics another participant's transformations. The *Imitation* action can thus be employed to satisfy a Symmetry constraint (see Figure 3.3 in Section 3.1.1.2 in [1]). Since an *Imitation* action is irrevocable, it does not pay respect to the *influencing factors* described in Section 3.3.1.

Instructions:

(1) If P is to behave in horizontal or vertical symmetry to another participant P', get the location $L' = (x_{P'}, y_{P'})$, which is supposed to be the center point of P':

$$L' = \Big(\frac{1}{2}\big(\vdash(\text{▯}P') + \dashv(\text{▯}P')\big), \frac{1}{2}\big(\perp(\text{▯}P') + \top(\text{▯}P')\big)\Big).$$

(2) Calculate the new location L_{new} for participant P. If x_{sym} and y_{sym} denote the horizontal and vertical coordinate of the specified iso-oriented symmetry axis respectively, the new location can be determined via

$$L_{\text{new}} = \begin{cases} (2\,x_{\text{sym}} - x_{P'}, y_{P'}) & \Leftrightarrow \quad \text{horizontal symmetry} \\ (x_{P'}, 2\,y_{\text{sym}} - y_{P'}) & \Leftrightarrow \quad \text{vertical symmetry.} \end{cases}$$

(3) Get the dimensions $(w_{P'}, h_{P'})$ of P' with the width $w_{P'} = \dashv(\text{▯}P') - \vdash(\text{▯}P')$ and the height $h_{P'} = \top(\text{▯}P') - \perp(\text{▯}P')$.

(4) Make P assume another layout variant that conforms to the dimensions $(w_{P'}, h_{P'})$ and let P move with its center point to the new location L_{new}.

Comments: In what way P is able to assume another layout variant that comes up to the dimensions of P', depends on the *variability* of P (see Section 3.2.4). If P has only *discrete variability*, but is –as a layout module– constructed identically to P', then P can simply take over the parameters of P'. If P and P' are constructed differently, some additional savvy might be necessary to help P choose the layout variant that at least comes closest to that of P'.

The instructions above assume that P has *full variability* (also see the next Section 3.3.3.3) and can take on dimensions in a continuous range. In this case, the purpose of the *Imitation* action is to match P with P' not (or not only) for the sake of P or P', but to achieve overall symmetry in the layout constellation as a whole – in this manner enforced *explicitly*.

As indicated above (and illustrated below), the *Imitation* action allows to leave some mutable room for other participants scrimmaging between P and P' – a feature that could not be adequately effectuated by using a superordinate governing module instead (in particular with respect to the consideration that this mutable room dynamically adjusts itself to the in-between participants during the self-organization). Vice versa, the *Imitation* action does not facilitate interdigitation, which in turn is an easy job for a governing module (like the Quad shown in Figure 3.6, for example). Following a given interdigitation pattern, such a module can take care of corresponding placement constraints *implicitly*.

So, the ability to team suchlike modules (incorporating nonformalized expert knowledge) with custom actions like *Imitation* (embodying formalized expert knowledge) is a powerful competence of SWARM regarding the title of the original dissertation [1].

Illustration:

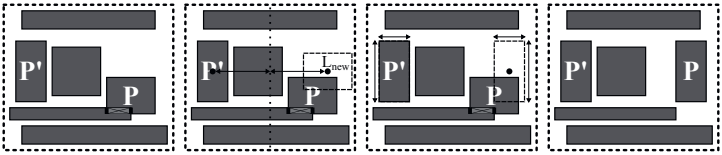


Figure 3.43: Exemplary visualization of a *Imitation* move.

So far, no coercive remarks have been made about the order in which the individual participants should act. But for custom actions, this order might be of exceptional relevance with regard to the efficiency of the self-organization. This is particularly evident in the *Imitation* action, where it is most reasonable to let participant P act as soon as P' has taken an action. Elsewise, all transformations performed by other participants in response to the action of P' might be more or less in vain, because they need to be remedied after P performs its irrevocable, ensuing *Imitation* move.

As the name indicates, custom actions are special-purpose actions and are thus not as generic as SWARM's catalog of native actions. Hence, it is not too much to ask that a participant provide its own custom actions as required. Still, a custom action –including the *Imitation* action above– may prove useful on such a regular basis that the action is deemed to be more universal than other custom actions. For future developments, it might therefore be worth a thought to include such actions into SWARM as extensions to the catalog of native actions. One action-related functionality, that has already been implemented in SWARM apart from the native actions covered in Section 3.3.3.1 addresses design components with full variability (as will now be described in Section 3.3.3.3).

3.3.3.3 Full Variability

If a participant covers multiple discrete layout variants, SWARM considers these via deformations during the action exploration, as has been discussed at the end of Section 3.3.3.1. The proceeding is different with participants that support full variability and are here denoted as *elastic*. An elastic participant P has a width w_P, a height h_P, and an area $\square P = w_P \cdot h_P$. It starts out with some initial dimensions w_P^{init} and h_P^{init}, but during the interaction P can change w_P and h_P into the new values w_P^{new} and h_P^{new} within the respective range

$$\check{w}_P \leq w_P^{\text{new}} \leq \underbrace{\frac{w_P\, h_P}{\check{h}_P}}_{=\hat{w}_P} \quad \text{and} \quad \check{h}_P \leq h_P^{\text{new}} \leq \underbrace{\frac{w_P\, h_P}{\check{w}_P}}_{=\hat{h}_P} \tag{3.71}$$

(where $\check{w}_P$ and $\check{h}_P$ allow to individually specify a minimal width and a minimal height for P, which also defines the maximum width $\hat{w}_P$ and the maximum height $\hat{h}_P$), such that the area of P remains unaltered:[9]

$$w_P^{\text{new}} \cdot h_P^{\text{new}} = w_P \cdot h_P. \tag{3.72}$$

Most of SWARM's native actions are geared towards some free peripheral space (e.g., for the ***Centering*** of a participant P, it is that of P; for the ***Swapping*** with another participant P', it is that of P'). If P is elastic, then P must be able to fit its aspect ratio to the dimensions (w_S, h_S) of such a free peripheral space S, thereby literally "pouring" into the vacant region. Thus, P's new layout variant is not one of a discrete set of layout variants $\{V_1, V_2, \ldots, V_d\}$, but rather a function of the free peripheral space that is currently available. With this function, five elastic deformation behaviors are explored per action, such that participant P either

(a) keeps its current aspect ratio,
(b) reassumes its initial aspect ratio,
(c) changes into a variant with square aspect ratio,
(d) fits its width to that of the free peripheral space (and adjusts its height),
(e) or fits its height to that of the free peripheral space (and adjusts its width).

These behaviors and the respective change of P's current width w_P and current height h_P into the new values w_P^{new} and h_P^{new} are listed in Table 3.8 and illustrated in Figure 3.44 by means of a ***Centering*** action for an exemplary, given situation.

Table 3.8: Listing of the five deformation behaviors explored by an elastic participant.

New Variant	New Width	New Height
(a) Current variant	$w_P^{\text{new}} = w_P$	$h_P^{\text{new}} = h_P$
(b) Initial variant	$w_P^{\text{new}} = w_P^{\text{init}}$	$h_P^{\text{new}} = h_P^{\text{init}}$
(c) Square variant	$w_P^{\text{new}} = \sqrt{w_P\, h_P}$	$h_P^{\text{new}} = \sqrt{w_P\, h_P}$
(d) Width-fitted	$w_P^{\text{new}} = \min(\max(w_S, \check{w}_P), w_P h_P / \check{h}_P)$	$h_P^{\text{new}} = w_P\, h_P \,/\, w_P^{\text{new}}$
(e) Height-fitted	$w_P^{\text{new}} = w_P\, h_P \,/\, h_P^{\text{new}}$	$h_P^{\text{new}} = \min(\max(h_S, \check{h}_P), w_P h_P / \check{w}_P)$

The five deformation behaviors do not constitute an action of its own, but are automatically fathomed during a participant's exploration of SWARM's native actions in case the participant is elastic. Thereupon, the "static" traversal of discrete layout variants (including different orientations) during the action exploration is simply substituted by probing the five alternatives of

[9] In reality, the area of such a layout component is usually not exactly constant for different aspect ratios, but only approximately – this is due to certain layout structures on the fringe. The smallest possible variant then is that of a square layout because the circumference-to-area quotient (and thus the portion of the fringe structures) reaches a minimum in that case.

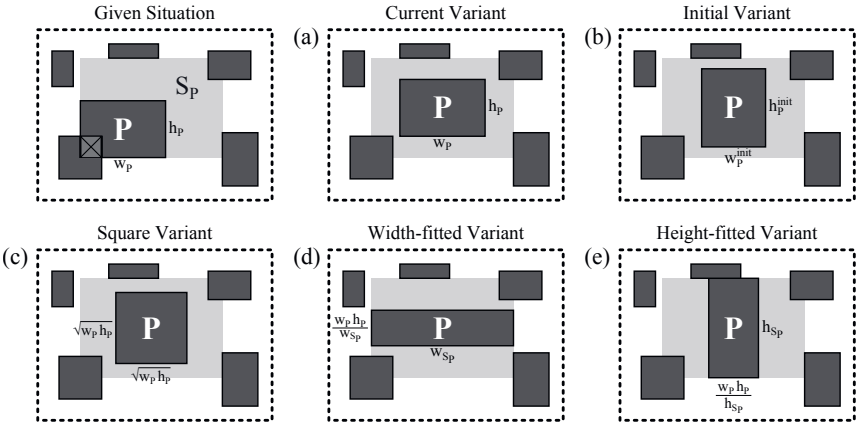


Figure 3.44: Illustration of the five deformation behaviors explored by an elastic participant.

"dynamic" deformation. As with SWARM's native actions, every elastic deformation has to be evaluated to check whether it represents a valid action or not. Otherwise, it could –for example– happen, that the participant P moves into an *unhealthy* location, overlapping a critical wound of another participant which penetrates a blind spot of P's free peripheral space.

3.3.4 Execution of the Preferred Action

After having explored and evaluated an assortment of potential actions, the fourth and final measure of SWARM's action scheme makes a participant choose and execute the most preferred one of these actions. To understand this decision-making, the following Section 3.3.4.1 illuminates by what judgment one action is preferred over another. In some cases, no such comparison has to be made because a satisfactory action can be performed right away, thus eliminating the need to investigate further alternatives. Such cases can best be comprehended from regarding the action execution in its entirety, which is therefore done in Section 3.3.4.2 and rounds out the current Section 3.3 in a synoptical fashion.

3.3.4.1 Action Preference

Comparing two actions to determine which one is preferable presupposes that both actions are *valid*. As already mentioned at the outset of Section 3.3.3, the preference then depends on two criteria: *interference* and *turmoil*. If no turmoil is involved (i.e., the connectivity is disregarded), then the comparison metric described in paragraph (1) suffices, otherwise the more elaborate comparison metric of paragraph (2) is employed. For the sake of convenience, an action is

preliminarily considered as a single transformation T of only *one* participant P – this can be assumed without loss of generality. Afterwards, paragraph (3) discusses how the metrics can be generalized to address actions that involve multiple participants.

(1) Comparison Metric for Interference Only

Some layout problems elude the necessity to take distances between participants into consideration during the module interaction. For instance, this is the case if the locations of the participants in relation to each other have been predefined in advance via a *placement template* (in SWARM enacted by constraints, as in the examples of Section 4.3). Then, no turmoil needs to be accounted for, which simplifies the action comparison significantly: if two actions are given as a transformation T_a and a transformation T_b, with $\Upsilon_P^{\text{new}}(T_a)$ and $\Upsilon_P^{\text{new}}(T_b)$ denoting the prospective interference that the acting participant P will sustain when performing the corresponding action, then action T_a is preferred over action T_b if

$$\Upsilon_P^{\text{new}}(T_a) < \Upsilon_P^{\text{new}}(T_b). \tag{3.73}$$

In the case that $\Upsilon_P^{\text{new}}(T_a) > \Upsilon_P^{\text{new}}(T_b)$, then action T_b is preferred over action T_a. If the prospective interference of the two actions are equal, then the participant chooses the action with the lesser Euclidean distance of the involved translational movement $M_a = (\Delta x_a, \Delta y_a)$ or $M_b = (\Delta x_b, \Delta y_b)$. Therefore, if

$$\sqrt{(\Delta x_a)^2 + (\Delta y_a)^2} < \sqrt{(\Delta x_b)^2 + (\Delta y_b)^2} \tag{3.74}$$

then action T_a is preferred over T_b. If the distances of the movements are equal, the action that has been explored first, will be preferred over the other action. In that regard, it is worthwhile to think about the order in which the actions –and in particular the layout variants that the participant can assume– are to be explored. This order should not be arbitrary, since it gives some valuable control over the decision-making such that –for example– squarish layout variants are preferred over layout variants with more extreme aspect ratios (or vice versa, depending on the problem at hand), e.g., to achieve a better matching. However, a detailed investigation of this topic is beyond the scope of this thesis.

If the participant finds an action T with $\Upsilon_P^{\text{new}}(T) = 0$ (such that there will be no interference at all), then this action can be immediately performed without exploring further alternatives. If $\Upsilon_P^{\text{new}} > 0$ for *all* explored actions, then the comparison condition 3.73 makes P execute the action with the least prospective interference – but only if

$$\Upsilon_P^{\text{new}}(T) - \Upsilon_P < 0 \tag{3.75}$$

which means that the prospective interference $\Upsilon_P^{\text{new}}(T)$ must be less than P's present interference Υ_P.[10] Such an action is called ***beneficial***. If $\Upsilon_P^{\text{new}}(T) = 0$, the action is denoted as ***adjuvant***. In all these cases, no constraint (e.g., template-induced) is violated since only valid actions are considered, as said above.

If no beneficial action can be found, but P is discontented, then P performs a ***Yielding***, even if it is not beneficial. As already mentioned in Section 3.3.3.1, ***Yielding*** leaves P in an unsatisfactory condition but has the effect that, if P is jammed in between multiple other participants $\mathcal{P}'$, then P will deliberately interfere with all of them. Thus every $P' \in \mathcal{P}'$ is provoked to move away such that P can again try to find a beneficial action on its next turn.

(2) Comparison Metric for Interference and Turmoil

To take distances between participants into account as well, condition 3.73 has to be enhanced such that it also includes a participant's turmoil. The following definition explains how this enhancement leads to condition 3.77 and condition 3.78. In that regard, it should be recapitulated that the distance between two participants is modeled as a straight connection C, and that the connection is said to be ***relaxed*** if the length of C is below its ***relaxation threshold*** ϱ_C (as described in Section 3.3.1.2).

Definition 3.5. *For a participant P, the value ζ_P is defined as the number of its* unrelaxed connections *(also see Section 3.3.1.2). An action T changes ζ_P by the amount $\Delta\zeta_P(T)$, which is denoted as the so-called* relaxation delta*. If $\Delta\zeta_P(T) < 0$, then T represents a* relaxing action*. Two actions T_a and T_b are said to be* equally relaxing *if both are relaxing or if they have the same relaxation delta. This definition can be written in form of the following logical expression:*

$$\Delta\zeta_P(T_a) < 0 \wedge \Delta\zeta_P(T_b) < 0 \vee \Delta\zeta_P(T_a) = \Delta\zeta_P(T_b). \tag{3.76}$$

The idea of relaxation delta is illustrated in the example of Figure 3.45. Participant P, having a total of five connections $(C_1, C_2, \ldots, C_5)$, is about to act. Initially (a), connections C_1, C_2 and C_3 are not relaxed (i.e., unrelaxed), while C_4 and C_5 are relaxed, which means that $\zeta_P = 3$. With the ***Centering*** shown in image (b), connections C_1 and C_2 become relaxed, while C_3 remains unrelaxed, C_4 remains relaxed, and C_5 becomes unrelaxed. Thus, the new number of unrelaxed connections is $\zeta_P = 2$, such that the relaxation delta is $\Delta\zeta_P = -1$. Hence, the action performed here represents a relaxing action.

With these conceptions in the sphere of relaxation, the comparison condition 3.73 is now enhanced, distinguishing two cases. If two actions are *not* equally relaxing, they are simply compared by their relaxation deltas. Otherwise, as will be detailed farther below, the prospective interference *and* the prospective turmoil are taken into account to rate an action. This is done by

[10] It is not necessary to take the absolute value $|\Upsilon|$ of interference here (nor in equation 3.73), since it is always the case that $\Upsilon \geq 0$, which means that interference can never be negative.

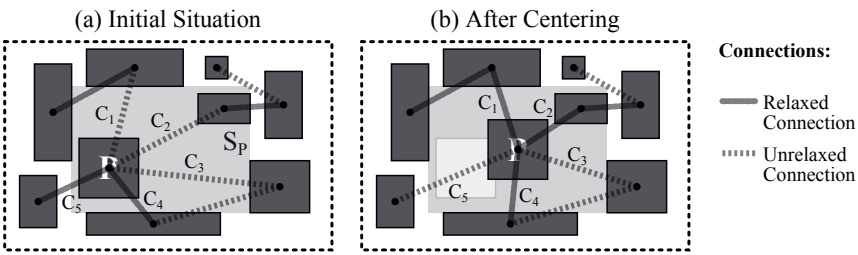


Figure 3.45: A relaxing action decreases a participant's number of unrelaxed connections.

adding up the *relative change* of interference and turmoil, written as $\delta\Upsilon_P$ and $\delta\Theta_P$ respectively. Thus, if two actions T_a and T_b are *not* equally relaxing, they are compared via condition

$$\Delta\zeta_P(T_a) < \Delta\zeta_P(T_b) \tag{3.77}$$

while otherwise –i.e., if T_a and T_b *are* equally relaxing– the following condition

$$\delta\Upsilon_P(T_a) + \delta\Theta_P(T_a) < \delta\Upsilon_P(T_b) + \delta\Theta_P(T_b) \tag{3.78}$$

is applied. Therein, the relative change of interference $\delta\Upsilon_P(T)$ achieved via action T is defined as

$$\delta\Upsilon_P(T) = \frac{\Upsilon_P^{\text{new}}(T) - \Upsilon_P}{\Upsilon_P^{\text{ref}}} \tag{3.79}$$

where Υ_P again is P's present interference while $\Upsilon_P^{\text{new}}(T)$ denotes the prospective interference that P achieves with T. The reference value Υ_P^{ref} is given as

$$\Upsilon_P^{\text{ref}} = \begin{cases} \Upsilon_P & \Leftrightarrow \quad \Upsilon_P > 0 \\ \left[\!\left[P\right]\!\right]^2 & \Leftrightarrow \quad \text{otherwise.} \end{cases} \tag{3.80}$$

Here, the value $\left[\!\left[P\right]\!\right]^2$ is used as a worst-case baseline for interference, reflecting the situation where P is completely overlapped by another participant. This *trouble* is approximated by putting $\left[\!\left[P\right]\!\right]$ (the area of P) to the square, such that it represents the intensity of the trouble on the one hand, and the trouble's tenacity on the other hand (also see equation 3.21 in Section 3.3.1.1). Since it is always the case that $\left[\!\left[P\right]\!\right] > 0$, the calculation of the reference value Υ_P^{ref} does never risk to run into a *division by zero* problem. Furthermore, $\left[\!\left[P\right]\!\right] > 0$ (or rather $\left[\!\left[P\right]\!\right]^2 > 0$) means that the relative change of interference $\delta\Upsilon_P(T)$ is beneficial if and only if

$$\delta\Upsilon_P(T) < 0. \tag{3.81}$$

Considering distances between the participants, $\delta\Theta_P(T)$ defines the relative change of turmoil for a participant P (equivalent to equation 3.79 above) as follows:

$$\delta\Theta_P(T) = \frac{\Theta_P^{\text{new}}(T) - \Theta_P}{\Theta_P^{\text{ref}}} \tag{3.82}$$

with Θ_P and $\Theta_P^{\text{new}}(T)$ denoting P's present and prospective turmoil. With $\hat{\Theta}_P$ being the greatest turmoil that participant P has encountered so far, the reference value in equation 3.82 is defined as

$$\Theta_P^{\text{ref}} = \begin{cases} \Theta_P & \Leftrightarrow \quad \Theta_P > 0 \\ \hat{\Theta}_P & \Leftrightarrow \quad \text{otherwise.} \end{cases} \tag{3.83}$$

Having equation 3.83 cover the circumstance that the present turmoil could be zero is analogous to the case distinction made in equation 3.80. However, during the module interaction the chance of running into a constellation where the turmoil is indeed $\Theta_P = 0$, can be assumed to be minuscule. As discussed in Section 3.3.1.2, a participant's turmoil is calculated as the sum of tensions in its connections. For a connection C with its length l_C, its emphasis e_C, its strength s_C, and its relaxation threshold ϱ_C, it is always true that $l_C \geq 0$, $e_C > 0$, $s_C > 0$ and $\varrho_C > 0$. From equation 3.48, this means that the tension θ in a connection can never be negative (as confirmed by Figure 3.20), such that also the turmoil of a participant is always $\Theta_P \geq 0$. For a participant P with connections $\mathcal{C}_P$, the turmoil Θ_P can only become zero if the tension in –and thus the length of– every connection in $\mathcal{C}_P$ is zero:

$$\Theta_P = 0 \iff \forall C \in \mathcal{C}_P, l_C = 0. \tag{3.84}$$

This incident can be neglected since it is extremely unlikely to occur. But in the beginning of a SWARM run, one would definitely encounter that $\Theta_P = 0$ (which inevitably means that $\hat{\Theta}_P = 0$ as well because $\hat{\Theta}_P = \Theta_P$ in that moment) if all participants were centered at the very same location. So, in order to avoid a ***division by zero*** problem here, the initial constellation of the participants $\mathcal{P}$ must be chosen such that in formal terms (using $C_{P,P'}$ to denote that C connects participant P with participant P') the following statement is true:

$$\forall P \in \mathcal{P}, \ \exists C_{P,P'} \in \mathcal{C}_P : L_P \neq L_{P'} \tag{3.85}$$

which means that for every participant P, at least one participant P' among its connected participants must be stationed at a location $L_{P'}$ that is different from the location L_P of P. This is easily achieved by choosing an initial constellation without any interference (i.e., $\forall P \in \mathcal{P}, \Upsilon_P = 0$). Since it can be expected that $\Theta_P^{\text{ref}} > 0$, the relative change of turmoil $\delta\Theta_P(T)$ is beneficial if and only if

$$\delta\Theta_P(T) < 0. \tag{3.86}$$

Hence, condition 3.81 and condition 3.86 justify why the sum of $\delta\Upsilon_P(T)$ and $\delta\Theta_P(T)$ is used for the action comparison in condition 3.78, which achieves that the action with the best compromise between prospective interference and prospective turmoil will finally be picked. Again, that action will only be executed if it is beneficial, which implies that

$$\delta\Upsilon_P(T) + \delta\Theta_P(T) < 0. \tag{3.87}$$

This condition indicates, that an acion is considered to be beneficial even if it concedes an increase in either (a) interference or (b) turmoil, as long as that increase is more than counterbalanced by a decrease in (a) turmoil or (b) interference, respectively.

(3) Actions Involving Multiple Participants

As to the *without loss of generality* remark at the beginning of this Section 3.3.4.1, the entire comparison metric described above can also be applied to actions that involve transformations of more than one participant. This is basically done by accumulating the respective quantities (i.e., the interference Υ, the turmoil Θ, and the relaxation delta $\Delta\zeta$).

To generalize the notation used so far, let $\Upsilon^{\text{new}}_{\mathcal{P}}(\mathcal{T})$ with $\mathcal{P} = \{P, P', P'', \dots\}$ denote the prospective *collective* interference of an action $\mathcal{T}$ (i.e., a set of transformations $\{T, T', T'', \dots\}$) that is initiated by the leading participant P, but involves further participants (i.e., P', P'', etc.). Then, the comparison condition 3.73 can also be generalized such that it reads

$$\Upsilon^{\text{new}}_{\mathcal{P}}(\mathcal{T}_a) < \Upsilon^{\text{new}}_{\mathcal{P}}(\mathcal{T}_b). \tag{3.88}$$

In the same manner, but now to compare the *collective* relaxation delta, condition 3.77 becomes

$$\Delta\zeta_{\mathcal{P}}(\mathcal{T}_a) < \Delta\zeta_{\mathcal{P}}(\mathcal{T}_b) \tag{3.89}$$

while condition 3.78 –in order to compare the *collective* relative change of Υ and Θ– turns into

$$\delta\Upsilon_{\mathcal{P}}(\mathcal{T}_a) + \delta\Theta_{\mathcal{P}}(\mathcal{T}_a) < \delta\Upsilon_{\mathcal{P}}(\mathcal{T}_b) + \delta\Theta_{\mathcal{P}}(\mathcal{T}_b). \tag{3.90}$$

One might fall prey to the misbelief that the collective interference $\Upsilon^{\text{new}}_{\mathcal{P}}(\mathcal{T})$ is calculated by adding up *all* prospective interferences of the individual participants that perform a transformation due to the action. This *can* be correct in some cases such as for example in Figure 3.46 (a): regarding the potential ***Pairing*** of P with P', the ***trouble*** of P and the trouble of P' are disjunct and must both be added together. However, the situation is different if two (or more) of the involved participants interfere with each other. That is because in such a case, the mutual trouble would be wrongly included twice, as exemplified in Figure 3.46 (b), where P explores a ***Pairing*** with P' which involves an action correction to make sure that P' is ***safe***: P overlaps P' and P' overlaps P, so they share the same trouble, which should therefore only be counted once.

Considering an action that involves two participants (i.e., the leading participant P and another participant P'), for example a ***Pairing*** (as in Figure 3.46) or a ***Swapping***, the prospective collective interference is correctly calculated via

$$\Upsilon^{\text{new}}_{\{P,P'\}}(\mathcal{T}) = \sum_{P^* \in \mathcal{P}^* \text{-} P} \tau^{\text{new}}_{P,P^*}(\mathcal{T}) + \sum_{P^* \in \mathcal{P}^* \setminus \{P',P\}} \tau^{\text{new}}_{P',P^*}(\mathcal{T}) \tag{3.91}$$

where $\mathcal{P}^*$ represents the set of all participants and the notation $\tau^{\text{new}}_{P,P^*}(\mathcal{T})$ is used to denote the prospective trouble between two participants P and P^*. With this formula, potential trouble

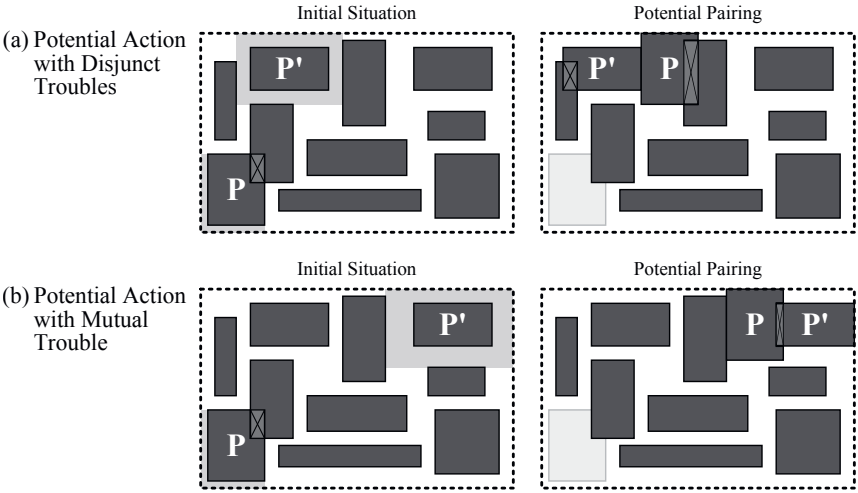


Figure 3.46: In contrast to disjunct troubles (a), mutual trouble (b) must be counted only once.

between the involved participants P and P' is included in the first addend but excluded from the second addend, and is thus only counted once, as desired. In this spirit, the formula can be extended to target the calculation of $\Upsilon_{\mathcal{P}}^{\text{new}}(\mathcal{T})$ in general, thus also addressing actions with $|\mathcal{T}| > 2$ (where more than two participants are involved) such that the prospective collective interference is correctly compared in condition 3.88.

Just as well, one must pay heed to this subtlety in the calculation of the present collective interference $\Upsilon_{\mathcal{P}}$, and –equivalently– in the calculation of the prospective collective turmoil $\Theta_{\mathcal{P}}^{\text{new}}(\mathcal{T})$ and the present collective turmoil $\Theta_{\mathcal{P}}$, such that a connection's tension is not counted more than once. This guarantees, that the collective relative change of Υ and Θ in condition 3.90 is correct. Finally, also the calculation of the collective relaxation delta $\Delta\zeta_{\mathcal{P}}(\mathcal{T})$ –used for the comparison in condition 3.89– must make sure not to doubly count a connection that becomes relaxed due to action $\mathcal{T}$.

3.3.4.2 Action Execution

The two tasks of exploring potential actions and choosing the most preferred action cannot be clearly separated from each other in terms of consecution. Instead, they proceed in a rather intermingled way before the chosen action is finally executed. This will now be explained by the subsequent description, which elucidates the liaison of the two tasks within the entirety of SWARM's action scheme. The prosaic description is literally "in line" with the more algorithmic representation provided by Algorithm 1.

Algorithm 1 SWARM's action scheme as acted upon by a participant P

Measure 1: Assessment of the Participant's Condition
$\Upsilon_P \leftarrow$ interference
$\zeta_P \leftarrow$ unrelaxed connections (turmoil)
$\Psi_P \leftarrow$ protrusion
$\mathcal{W}_{P\bowtie}^{\ddagger} \leftarrow$ unhealthy wounds
$\mathcal{H}_P^{\ddagger} \leftarrow$ dissatisfied constraints (noncompliance)
Measure 2: Perception of the Free Peripheral Space
$S_P \leftarrow$ free peripheral space
Measure 3: Exploration and Evaluation of Possible Actions
$\mathcal{T}_{\text{final}} \leftarrow \texttt{null}$
$\mathbb{T}_{\text{valid}} \leftarrow \{\,\}$
for all *kinds of custom actions* defined on P **do**
  if $\mathcal{T}_{\text{final}} = \texttt{null}$ **then**
    determine custom action $\mathcal{T}_{\text{custom}}$
    if $\mathcal{T}_{\text{custom}}$ is acceptable **then** ▷ this means the action is valid, tolerable, or mandatory
      $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{custom}}$; **exit for-loop**
    end if
  end if
end for
if $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Psi_P = \text{▯}P$ **then**
  determine ***Re-entering*** action $\mathcal{T}_{\text{Re-entering}}$ ▷ the action is tolerable since $\Psi_P^{\text{new}} = \varnothing$
  $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Re-entering}}$
end if
if $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Psi_P \neq \text{▯}P$ **then**
  for all $(O, V_P) \in (\mathbb{O} \times \widetilde{\mathcal{V}}_P)$ **do**
    determine ***Centering*** action $\mathcal{T}_{\text{Centering}}$ with respect to S_P
    if $\Psi_P^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_P^{\ddagger\ \text{new}} = \emptyset$ **then** ▷ this means the action is valid
      $\mathbb{T}_{\text{valid}} \leftarrow \mathbb{T}_{\text{valid}} \dotplus \mathcal{T}_{\text{Centering}}$
      if $\Upsilon_P^{\text{new}} = 0$ **AND** $\zeta_P^{\text{new}} = 0$ **then** ▷ this means the action is adjuvant
        $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Centering}}$; **exit for-loop**
      end if
    end if
  end for
end if
if $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Upsilon_P = 0$ **AND** $\zeta_P = 0$ **AND** $\Psi_P = \varnothing$ **AND** $\mathcal{H}_P^{\ddagger} = \emptyset$ **then** ▷ P is contented
  $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Lingering}}$
end if
if $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Psi_P = \varnothing$ **then**
  for all N of $\text{▯}P$ **do**
    perceive secondary free peripheral space S_P^N
    for all $(O, V_P, X) \in (\mathbb{O} \times \widetilde{\mathcal{V}}_P \times \mathcal{X})$ **do**
      determine ***Budging*** action $\mathcal{T}_{\text{Budging}}$ with respect to S_P^N
      if $\Psi_P^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_P^{\ddagger\ \text{new}} = \emptyset$ **then** ▷ this means the action is valid
        $\mathbb{T}_{\text{valid}} \leftarrow \mathbb{T}_{\text{valid}} \dotplus \mathcal{T}_{\text{Budging}}$
        if $\Upsilon_P^{\text{new}} = 0$ **AND** $\zeta_P^{\text{new}} = 0$ **then** ▷ this means the action is adjuvant
          $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Budging}}$; **exit for-loops**
        end if
      end if
    end for
  end for
end if

Algorithm 2 SWARM's action scheme as acted upon by a participant P (continued)

if $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Upsilon_P > 0$ **AND** $\Psi_P = \varnothing$ **then**
  determine ***Hustling*** action $\mathcal{T}_{\text{Hustling}}$ for participants $\mathcal{P}_\iota$
  if $\forall P_\iota \in \mathcal{P}_\iota, \Psi_{P_\iota}^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P_\iota \bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_{P_\iota}^{\ddagger\ \text{new}} = \emptyset$ **then** ▷ means the action is valid
    $\mathbb{T}_{\text{valid}} \leftarrow \mathbb{T}_{\text{valid}} \dot{+} \mathcal{T}_{\text{Hustling}}$
    if $\forall P_\iota \in \mathcal{P}_\iota, \Upsilon_{P_\iota}^{\text{new}} = 0$ **AND** $\zeta_{P_\iota}^{\text{new}} = 0$ **then** ▷ this means the action is adjuvant
      $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Hustling}}$; **exit for-loop**
    end if
  end if
end if
if $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Psi_P = \varnothing$ **then**
  for all $P' \in \mathcal{P} \setminus P$ **do**
    perceive free peripheral space $S_{P'}$ of P'
    if promising **then**
      for all $X \in \mathcal{X}$ **do**
        determine ***Swapping*** action $\mathcal{T}_{\text{Swapping}}$ for P and P' with respect to S_P and $S_{P'}$
        if $\Psi_P^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_P^{\ddagger\ \text{new}} = \emptyset$ **then**
          if $\Psi_{P'}^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P'\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_{P'}^{\ddagger\ \text{new}} = \emptyset$ **then**
            ▷ this means the action is valid
            $\mathbb{T}_{\text{valid}} \leftarrow \mathbb{T}_{\text{valid}} \dot{+} \mathcal{T}_{\text{Swapping}}$
            if $\Upsilon_P^{\text{new}} = 0$ **AND** $\zeta_P^{\text{new}} = 0$ **then**
              if $\Upsilon_{P'}^{\text{new}} = 0$ **AND** $\zeta_{P'}^{\text{new}} = 0$ **then** ▷ this means the action is adjuvant
                $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Swapping}}$; **exit for-loops**
              end if
            end if
          end if
        end if
      end for
    end if
  end for
end if
if $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Psi_P = \varnothing$ **then**
  for all $P' \in \mathcal{P} \setminus P$ **do**
    perceive free peripheral space $S_{P'}$ of P' ▷ can be re-used from the ***Swapping*** exploration
    if promising **then**
      for all $X \in \mathcal{X}$ **do**
        determine ***Pairing*** action $\mathcal{T}_{\text{Pairing}}$ for P and P' with respect to $S_{P'}$
        if $\Psi_P^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_P^{\ddagger\ \text{new}} = \emptyset$ **then**
          if $\Psi_{P'}^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P'\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_{P'}^{\ddagger\ \text{new}} = \emptyset$ **then**
            ▷ this means the action is valid
            $\mathbb{T}_{\text{valid}} \leftarrow \mathbb{T}_{\text{valid}} \dot{+} \mathcal{T}_{\text{Pairing}}$
            if $\Upsilon_P^{\text{new}} = 0$ **AND** $\zeta_P^{\text{new}} = 0$ **then**
              if $\Upsilon_{P'}^{\text{new}} = 0$ **AND** $\zeta_{P'}^{\text{new}} = 0$ **then** ▷ this means the action is adjuvant
                $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Pairing}}$; **exit for-loops**
              end if
            end if
          end if
        end if
      end for
    end if
  end for
end if

Algorithm 3 SWARM's action scheme as acted upon by a participant P (continued)

98: **if** $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Psi_P \sqsubset \text{▯}P$ **then**
99: **for all** $(O, V_P, X) \in (\mathbb{O} \times \widetilde{\mathcal{V}_P} \times \mathcal{X})$ **do**
100: determine *Evasion* action $\mathcal{T}_{\text{Evasion}}$ with respect to S_P
101: **if** $\Psi_P^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_P^{\ddagger\ \text{new}} = \emptyset$ **then** ▷ this means the action is valid
102: $\mathbb{T}_{\text{valid}} \leftarrow \mathbb{T}_{\text{valid}} \Plus \mathcal{T}_{\text{Evasion}}$
103: **if** $\Upsilon_P^{\text{new}} = 0$ **AND** $\zeta_P^{\text{new}} = 0$ **then** ▷ this means the action is adjuvant
104: $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Evasion}}$; **exit for-loop**
105: **end if**
106: **end if**
107: **end for**
108: **end if**
109: **if** $\mathcal{T}_{\text{final}} = \texttt{null}$ **then**
110: **if** $\mathbb{T}_{\text{valid}} \neq \{\,\}$ **then** ▷ this means that at least one valid action has been found
111: sort $\mathbb{T}_{\text{valid}}$ by benefit ▷ sorting is done according to the comparison metric
112: store the best action in $\mathcal{T}_{\text{best}}$
113: **if** $\mathcal{T}_{\text{best}}$ is beneficial **OR** $\Psi_P \neq \varnothing$ **OR** $\mathcal{W}_{P\bowtie}^{\ddagger} \neq \emptyset$ **OR** $\mathcal{H}_P^{\ddagger} \neq \emptyset$ **then**
114: $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{best}}$
115: **end if**
116: **end if**
117: **end if**
118: **if** $\mathcal{T}_{\text{final}} = \texttt{null}$ **AND** $\Upsilon_P > 0$ **AND** $\Psi_P \neq \text{▯}P$ **then**
119: identify the polygonal yielding region Y_P
120: determine *Yielding* action $\mathcal{T}_{\text{Yielding}}$ with respect to Y_P
121: **if** $\Psi_P^{\text{new}} = \varnothing$ **AND** $\mathcal{W}_{P\bowtie}^{\ddagger\ \text{new}} = \emptyset$ **AND** $\mathcal{H}_P^{\ddagger\ \text{new}} = \emptyset$ **then** ▷ this means the action is valid
122: $\mathcal{T}_{\text{final}} \leftarrow \mathcal{T}_{\text{Yielding}}$
123: **end if**
124: **end if**

Measure 4: Execution of the Preferred Action

125: **if** $\mathcal{T}_{\text{final}} \neq \texttt{null}$ **then**
126: **if** $\exists T \in \mathcal{T}_{\text{final}} : \sqrt{(\Delta x_T)^2 + (\Delta y_T)^2} \geq m \vee O_{P_T}^{\text{new}} \neq O_{P_T} \vee V_{P_T}^{\text{new}} \neq V_{P_T}$
OR $\Psi_P \neq \varnothing$ **OR** $\mathcal{W}_{P\bowtie}^{\ddagger} \neq \emptyset$ **OR** $\mathcal{H}_P^{\ddagger} \neq \emptyset$ **then** ▷ m is the minimal movement distance
127: **for all** $T \in \mathcal{T}_{\text{final}}$ **do**
128: apply transformation T to the respective participant P_T
129: **end for**
130: **end if**
131: **end if**

Let there be a participant P, whose turn it is to take an action. Following the action scheme, P's **first measure** (line 1ff.) –as covered in Section 3.3.1– is to assess its condition in order to determine if

- P is clear: $\Upsilon_P = 0$ (see Section 3.3.1.1 on interference),
- P is relaxed: $\zeta_P = 0$ (see Section 3.3.1.2 on turmoil),
- P is safe: $\Psi_P = \varnothing$ (see Section 3.3.1.3 on protrusion),
- P is healthy: $\mathcal{W}_{P\bowtie}^{\ddagger} = \emptyset$ (see Section 3.3.1.4 on wounds),
- P is compliant: $\mathcal{H}_P^{\ddagger} = \emptyset$ (see Section 3.3.1.5 on noncompliance).

Next, P's **second measure** (line 6) is to perceive its free peripheral space S_P according to the geometrical recipe given in Section 3.3.2.

Then, P's **third measure** is to explore and evaluate potential actions, whereby an *adjuvant* action can already be chosen for execution during the action exploration. Choosing an action is done by storing the action in a variable here denoted as $\mathcal{T}_{\text{final}}$, which is initialized with a null value in the beginning of the action exploration (line 7). As long as that variable is null, the action exploration continues, otherwise the participant proceeds with the execution of that action. Looking forward to the case that no action is chosen during the action exploration, every *valid* action is memorized in a set $\mathbb{T}_{\text{valid}}$ (initialized in line 8), from which the best action can then be chosen afterwards.

Custom actions: If custom actions such as the *Imitation* action from Section 3.3.3.2 are defined on P, these are explored first of all (see the for-loop in line 9ff.). When a custom action $\mathcal{T}_{\text{custom}}$ is deemed *acceptable*, the action is stored in $\mathcal{T}_{\text{final}}$ and the participant can immediately exit the for-loop.

Re-entering: Given that no custom action has been chosen, SWARM's native actions from Section 3.3.3.1 are explored. Starting off with the case that P finds itself being lost, then a *Re-entering* action $\mathcal{T}_{\text{Re-entering}}$ is determined and stored in $\mathcal{T}_{\text{final}}$ (line 17ff.). By definition, the action is *tolerable* because it eliminates P's protrusion.

Centering: Next, the participant –if it is not lost– explores potential *Centering* actions (line 21 ff.). For all orientations $O \in \mathbb{O}$ and for all layout variants $V_P \in \widetilde{\mathcal{V}}_P$ that P can assume, an action $\mathcal{T}_{\text{Centering}}$ is determined and evaluated. If the prospective location is safe, healthy and compliant, then the action is valid and therefore memorized in $\mathbb{T}_{\text{valid}}$. If P would even become clear and relaxed, then the action is adjuvant and is thus stored in $\mathcal{T}_{\text{final}}$, letting the participant exit the for-loop.

Lingering: In case no adjuvant *Centering* action could be found, the participant checks if it is currently clear, relaxed, safe and compliant (line 32). If that is so, P is *contented* (this can be said without checking for wounds, since a participant is definitely healthy if it is clear). Then, an "empty" *Lingering* action with $\Delta x = 0$ and $\Delta y = 0$ is stored in $\mathcal{T}_{\text{final}}$. Mathematically speaking, the *Lingering* action represents the *identity element* among SWARM's actions (i.e., a *neutral element* such as the ones used in algebraic structures).

Budging: Unless P lingers where it is, potential *Budging* actions are explored if the participant is safe (line 35ff.). For each vertex of P's bounding box $\square P$, represented as a *node* N, the secondary free peripheral space S_P^N is perceived for action exploration. How exhaustive that exploration is done, depends on the *exploration plan* which can be abstractly thought of as a set $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$. Therein, each X represents one possible *exploration scenario* with respect to the kind of action being explored. For a *Budging*, every exploration scenario specifies one possible edge or vertex of S_P^N that P can align itself with. So, for every possible orientation O, for any layout variant V_P of P, and for each explo-

ration scenario X, an action $\mathcal{T}_{\text{Budging}}$ is determined and evaluated. If the action is valid, it is memorized in $\mathbb{T}_{\text{valid}}$; if the action is even adjuvant, it is stored in $\mathcal{T}_{\text{final}}$ (as also done in the exploration of *Centering* actions).

Hustling: Provided that no adjuvant action has yet been found, a *Hustling* action $\mathcal{T}_{\text{Hustling}}$ is explored (line 49ff. in Algorithm 2), subject to the condition that the participant is safe but not clear (otherwise the exploration of a *Hustling* action would be futile). As above, the explored action is memorized in $\mathbb{T}_{\text{valid}}$ in case it is valid; and is further stored in $\mathcal{T}_{\text{final}}$ if it is adjuvant – however, this has to take not only P into account, but all participants $\mathcal{P}_\iota$ that are involved in the action.[11] Computationally, determining a *Hustling* action is cheap (even if a more exhaustive exploration plan is pursued), therefore it is opportunely done *prior* to the subsequent exploration of the other kinds of actions, which raise higher combinatorial expenses (especially *Swapping* and *Pairing*).

Swapping: If P is safe, the action exploration continues with *Swapping* actions (line 58ff.). For every participant P' among the set $\mathcal{P} - P$ (the set of all participants except P), the free peripheral space $S_{P'}$ is perceived. Then, as discussed under *Swapping* in Section 3.3.3.1, the participant might try to predict whether a *Swapping* of P and P' is promising or not. If yes, then an action $\mathcal{T}_{\text{Swapping}}$ is explored for every exploration scenario X of the exploration plan $\mathcal{X}$. If the prospective locations are safe, healthy and compliant for *both* participants P and P', then the action is valid and hence gets memorized in $\mathbb{T}_{\text{valid}}$. If both participants would even become clear and relaxed, then the action is adjuvant and is thus used as the final action $\mathcal{T}_{\text{final}}$.

Pairing: In the same manner, potential *Pairing* actions are explored (line 78ff.). For every other participant P', the free peripheral space $S_{P'}$ is to be determined. In view of this, it is feasible to save the free peripheral spaces during the *Swapping* exploration such that they can now be retrieved without the need to perceive them anew. If a *Pairing* of P with P' looks promising (depending on $S_{P'}$), then for every exploration scenario X in the exploration plan $\mathcal{X}$ an action $\mathcal{T}_{\text{Pairing}}$ is explored. If the action is valid (again regarding *both* participants), it is memorized in $\mathbb{T}_{\text{valid}}$. If the action is adjuvant (regarding both participants), then it is used as the final action $\mathcal{T}_{\text{final}}$.

Evasion: Be it that still no adjuvant action is available and that the participant is prone, potential *Evasion* actions are finally explored by the participant (line 98ff. in Algorithm 3). For each orientation O, for any layout variant V_P of P and for every exploration scenario X in the exploration plan $\mathcal{X}$, an action $\mathcal{T}_{\text{Evasion}}$ is determined and evaluated. As with the

[11] In the implementation, checking for $\Psi_P^{\text{new}} = \varnothing$ (prospective protrusion of the leading participant P) is in fact superfluous because P itself is already safe and does not move. However, checking for $\Upsilon_P^{\text{new}} = 0$ (prospective interference) *is* necessary since the *Hustling* action may be unsuccessful, either because P is *properly enclosed* by another participant (as explained in Section 3.3.3.1 under *Hustling*) or because an *action correction* is involved (see Section 3.3.1.3).

other kinds of actions, the explored action is memorized in $\mathbb{T}_{\text{valid}}$ if it is valid; in case the action is even adjuvant, it is stored in $\mathcal{T}_{\text{final}}$.

Best action: In the event that the participant has not been able to find an adjuvant action, the best non-adjuvant valid action is chosen (line 109ff.). This presupposes that at least one valid action has been found (i.e., $\mathbb{T}_{\text{valid}} \neq \{\,\}$). The memorized actions of $\mathbb{T}_{\text{valid}}$ are sorted according to the comparison metric given in Section 3.3.4.1, such that the best action (the action with the greatest benefit) can be chosen and stored in $\mathcal{T}_{\text{best}}$. However, that action is only used as the final action $\mathcal{T}_{\text{final}}$ if (a) it is *beneficial* –see condition 3.75 and condition 3.87– or (b) if the participant's current location is not safe ($\Psi_P \neq \varnothing$), not healthy ($\mathcal{W}_{P\bowtie}^{\ddagger} \neq \emptyset$), or not compliant ($\mathcal{H}_P^{\ddagger} \neq \emptyset$). Thus in case (b), the best action will be performed even if it is not beneficial, otherwise such a best action is rejected.

Yielding: Given that by now the action exploration has not put forth a valid action $\mathcal{T}_{\text{final}}$ to perform, the participant checks if it suffers from interference ($\Upsilon_P > 0$), because then a *Yielding* action can be explored. This is done if the participant is not lost (line 118ff.). For that purpose, the polygon Y_P (introduced as the *yielding region* in Section 3.3.3.1) is identified, which in turn allows P to determine the *Yielding* action $\mathcal{T}_{\text{Yielding}}$. If the action is valid, then it is used as the final action $\mathcal{T}_{\text{final}}$, regardless of whether this represents a beneficial action or not.

The action exploration is followed by the action execution, which marks P's **fourth measure** (line 125ff.). Executing an action first of all implies that a valid action has been determined (i.e., $\mathcal{T}_{\text{final}} \neq \texttt{null}$). In that case, the participant then affirms that the action is not *trivial*.

Definition 3.6. *An action $\mathcal{T}$ is not trivial if at least one of its transformations $T \in \mathcal{T}$ is not trivial. A transformation T is not trivial (a) if the length of its movement vector M_T is not smaller than the minimal movement distance m, or (b) if T would turn the respective participant P_T into a different orientation $O_{P_T}^{new}$ due to a rotation R_T, or (c) if T would change that participant into another layout variant $V_{P_T}^{new}$ due to a deformation D_T, or (d) if the leading participant P currently is in an invalid location (not safe, not healthy, or not compliant).*

Trivial actions are discarded because their potential benefit is insignificant, which tends to retard the overall self-organization flow rather than to enliven it. If the action $\mathcal{T}_{\text{final}}$ is not trivial, then it is executed, i.e., every transformation T of the action is applied to the respective participant P_T. This concludes the way in which the leading participant P acts upon SWARM's action scheme. Directing the overall *inter*-action towards the desired layout outcome is up to the interaction control organ – SWARM's third core concept, that will now be addressed in Section 3.4.

3.4 Interaction Control

As exposed in Chapter 2, immense power can be ascribed to self-organizing, decentralized structures. But to tap the full potential of such structures, regarding the design of bottom-up systems for practical utilization, [158] points out: "the bottom is not enough. You need a bit of top-down as well." This is why SWARM includes an *interaction control* organ which supervises the modules' actions of Section 3.3 from a top-down perspective and carefully steers the interaction in order to enforce the emergence of a constraint-compliant, compact constellation that fits within the user-defined layout zone Z.

As illustrated in the control-flow depiction of Figure 3.3, the steering of the module interaction proceeds in an indirect fashion which is achieved by repeatedly changing the size of the user-defined layout zone. This represents the main task of the interaction control organ and will be explicated in Section 3.4.1. Extending these considerations, the notion of *transient tightening policies* is then introduced in Section 3.4.2. A further idea within that orbit is to envelop each participating module in a personal *comfort padding* correlated with the tightening of the layout zone. As will be discussed in Section 3.4.3, this idea is meant to smooth the self-organization flow and to equilibrate the evocation of emergent behavior throughout a SWARM run.

3.4.1 Scaling the Layout Zone

In line with the depiction of Figure 3.3, the interaction control organ's task of scaling the user-defined layout zone incorporates two discernible jobs: (1) setting and enlarging the zone in the beginning of the self-organization phase, and then (2) tightening the layout zone after each settlement. The former job will be addressed in Section 3.4.1.1, while Section 3.4.1.2 covers the details of the latter job. For arbitrarily rectilinear[12] layout zones, some particular considerations must be made, as Section 3.4.1.3 will discuss.

3.4.1.1 Setting and Enlarging the Layout Zone

Demarcating the layout territory that is available for the design problem at hand, the layout zone Z is expected to be given by the user as a rectangular or rectilinear polygon such as in Figure 3.47 (a). Then, for an initial module constellation –i.e., a set of n participants $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$– as in Figure 3.47 (b), the interaction control organ determines the rectangular bounding box $\square\mathcal{P}$ around all participants, denoted as the *constellation frame* F and calculated via

$$F = \square\mathcal{P} = \left(\left(\min_{\forall P \in \mathcal{P}} \left(\vdash(\square P) \right), \min_{\forall P \in \mathcal{P}} \left(\bot(\square P) \right) \right), \left(\max_{\forall P \in \mathcal{P}} \left(\dashv(\square P) \right), \max_{\forall P \in \mathcal{P}} \left(\top(\square P) \right) \right) \right) \tag{3.92}$$

[12] In the remainder of this thesis, the term *rectilinear* especially points to the case where the layout zone is *not rectangular*.

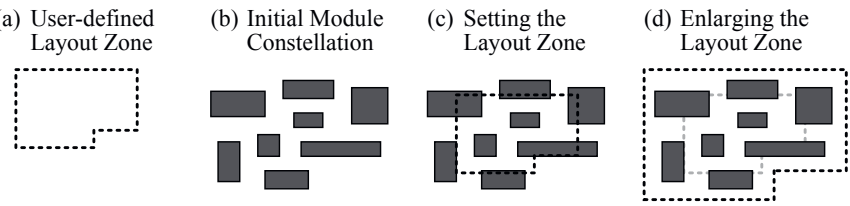


Figure 3.47: Setting and enlarging the user-defined layout zone in the beginning of the self-organization.

in rectangle notation. To put Z into its initial location as in the example of Figure 3.47 (c), the interaction control organ moves it by

$$\Big(\frac{1}{2}\big(\vdash F - \vdash Z + \dashv F - \dashv Z\big), \frac{1}{2}\big(\perp F - \perp Z + \top F - \top Z\big)\Big) \tag{3.93}$$

which is the vectorial difference of the constellation frame's center point and the center point of the layout zone's bounding box. Throughout the remaining SWARM run, the layout zone is never again moved or transformed in another way apart from being resized. Foremost, prior to the first round of interaction, Z is enlarged such that the area of the layout zone becomes considerably larger than the totaled areas of the individual participants, as can be seen in Figure 3.47 (d).

It is important to note, that the aspect ratio of Z is not modified thereby. This implies that the enlargement of Z is not achieved through an equal, absolute dilation of its edges (as done by the *grow operator* $\circledcirc_\varepsilon$ on page 227), but via a scaling operation that works in relation to the current size of Z. Considering a geometrical shape G in general, this scaling operation takes a relative, percental *contraction amount* $\xi \in \mathbb{R}$ in the right-open interval $(-\infty, 0.5)$ and applies it to the shape quadrilaterally, i.e., in all four directions of the coordinate system (in the two-dimensional layout plane).

Figure 3.48 exemplifies this contraction operation with a given rectangle (a) of width w and height h: a contraction amount of $\xi = -0.5$ stretches the rectangle by an absolute *edge displacement* $\epsilon = 0.5 \cdot w$ to the left and to the right, whereas the vertical edge displacement is $0.5 \cdot h$ both upwards and downwards, such that the rectangle becomes four times as large in this particular case (b). The increase in size with respect to ξ can be universally given as the quotient of the shape's new area $\lceil G_{\text{new}} \rceil$ and its old area $\lceil G_{\text{old}} \rceil$ via the following formula (which holds true for $\xi \leq 0.5$):

$$\frac{\lceil G_{\text{new}} \rceil}{\lceil G_{\text{old}} \rceil} = \frac{(w - 2\xi w) \cdot (h - 2\xi h)}{w \cdot h} = \frac{w(1 - 2\xi) \cdot h(1 - 2\xi)}{w \cdot h} = (1 - 2\xi)^2. \tag{3.94}$$

Thus, as the term contraction suggests, the shape is enlarged by a negative contraction amount ($\xi < 0$) and made smaller by a positive contraction amount ($\xi > 0$). This is deliberate, owing to the fact that the layout zone Z is much more often subject to a tightening than to an enlargement.

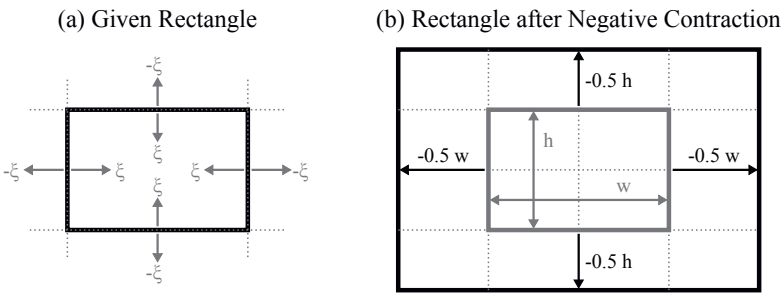


Figure 3.48: Relative enlargement of a given rectangle via negative contraction.

To generalize the contraction operation in regard of arbitrary polygons, the *contraction operator* $\circledast_\xi$ is introduced. Formally, describing the polygonal shape G as a closed polygonal chain, i.e., a sequence $G = (N_1, N_2, \dots)$, where each node N represents one vertex of G with coordinates (x_N, y_N), the contraction operator moves every node N to the new location N_{new} according to the directive

$$N_{\text{new}} = \Big((1-2\xi)\, x_N + \xi\, (\vdash G + \dashv G), (1-2\xi)\, y_N + \xi\, (\perp G + \top G) \Big). \tag{3.95}$$

This directive guarantees that the center point of the shape is not dislocated through the contraction. The proof can be given by inserting the center point's coordinates $x_N = \frac{1}{2}(\vdash G + \dashv G)$ and $y_N = \frac{1}{2}(\perp G + \top G)$ into equation 3.95, as this leads to the new location

$$N_{\text{new}} = \Big((1-2\xi)\, \tfrac{1}{2}(\vdash G + \dashv G) + \xi\, (\vdash G + \dashv G), (1-2\xi)\, \tfrac{1}{2}(\perp G + \top G) + \xi\, (\perp G + \top G) \Big) \tag{3.96}$$

$$= \Big((\tfrac{1}{2} - \tfrac{1}{2}2\xi + \xi)\, (\vdash G + \dashv G), (\tfrac{1}{2} - \tfrac{1}{2}2\xi + \xi)\, (\perp G + \top G) \Big) \tag{3.97}$$

$$= \Big(\tfrac{1}{2}\, (\vdash G + \dashv G), \tfrac{1}{2}\, (\perp G + \top G) \Big) \tag{3.98}$$

which is the same location as before. The preservation of the center point can also be observed in Figure 3.49, where a contraction operation with $\xi = -0.5$ is applied to a rectilinear octagon. Following the directive of equation 3.95, Table 3.9 lists how the coordinates of each node change from their old into their new values. Also shown is that –like in the example of Figure 3.48– the area of the polygon is quadruplicated in this case.

By applying such a negative contraction, the enlargement of the layout zone's initial user-defined size $\lceil Z_0 \rceil$ in the beginning of the self-organization –see Figure 3.47 (d)– can be correctly achieved. For that purpose, a *kickoff enlargement multiplier* $\varkappa \in \mathbb{R}$ with $\varkappa \geq 1$ allows to specify the size $\lceil Z_1 \rceil$ of the layout zone for the 1st tightening-settlement cycle in relation to the *intrinsic minimum* $\lceil \boxplus \mathcal{P} \rceil$ (the sum of the participants' individual areas, as introduced in Defini-

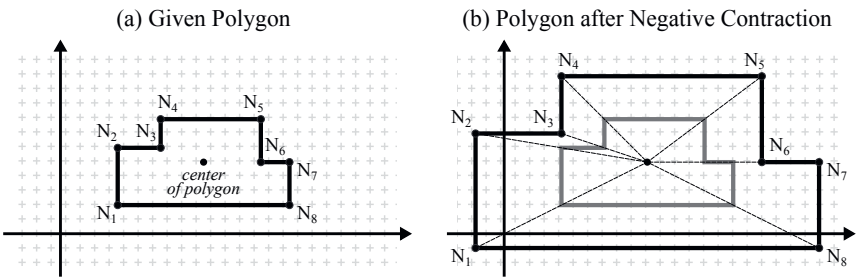


Figure 3.49: Relative enlargement of a given polygon via negative contraction.

Table 3.9: Coordinates of the octagon nodes from the negative contraction example of Figure 3.49.

Nodes	**Old**	**New**
N_1	$(4, 2)$	$(-2, -1)$
N_2	$(4, 6)$	$(-2, 7)$
N_3	$(7, 6)$	$(4, 7)$
N_4	$(7, 8)$	$(4, 11)$
N_5	$(14, 8)$	$(18, 11)$
N_6	$(14, 5)$	$(18, 5)$
N_7	$(16, 5)$	$(22, 5)$
N_8	$(16, 2)$	$(22, -1)$
center	$(10, 5)$	$(10, 5)$
area	60	240

tion 3.2). The kickoff enlargement multiplier is to be provided by the user as desired, considering the rule of thumb that Z should be made roughly thrice as large as the intrinsic minimum.

Internally, the respective contraction amount ξ must be calculated from the kickoff enlargement multiplier $\varkappa$. This relationship can be easily deduced from equation 3.94 as follows:

$$\frac{\lceil Z_1 \rceil}{\lceil Z_0 \rceil} = (1 - 2\xi)^2 \tag{3.99}$$

$$\frac{\varkappa \cdot \lceil \mathcal{P} \rceil}{\lceil Z_0 \rceil} = (1 - 2\xi)^2 \tag{3.100}$$

$$\sqrt{\frac{\varkappa \cdot \lceil \mathcal{P} \rceil}{\lceil Z_0 \rceil}} = 1 - 2\xi \tag{3.101}$$

$$\xi = \frac{1}{2} - \frac{1}{2}\sqrt{\frac{\varkappa \cdot \lceil \mathcal{P} \rceil}{\lceil Z_0 \rceil}}. \tag{3.102}$$

It should be mentioned, that no $\pm\sqrt{...}$ distinction must be made in the root extraction of equation 3.101 because the contraction amount is always $\xi < 0.5$ (which means that the term $1 - 2\xi$ above cannot be less than zero). An exemplary kickoff enlargement is given in Figure 3.50: the individual areas of the seven participants (a) add up to the intrinsic minimum $\lceil \mathfrak{P} \mathcal{P} \rceil = 18$, while the initial size of the layout zone (b) with width $w = 6$ and height $h = 4$ is $\lceil Z_0 \rceil = 24$. For a kickoff enlargement of $\varkappa = 3$, the contraction amount is calculated as

$$\xi = \frac{1}{2} - \frac{1}{2}\sqrt{\frac{3 \cdot 18}{24}} = \frac{1}{2} - \frac{1}{2}\sqrt{\frac{9}{4}} = \frac{1}{2} - \frac{1}{2}\frac{3}{2} = -\frac{1}{4}. \tag{3.103}$$

With this negative contraction, the new dimensions (c) of the layout zone are $w = 9$ and $h = 6$ such that its area becomes $\lceil Z_1 \rceil = 9 \cdot 6 = 54$ which is thrice the intrinsic minimum, as desired.

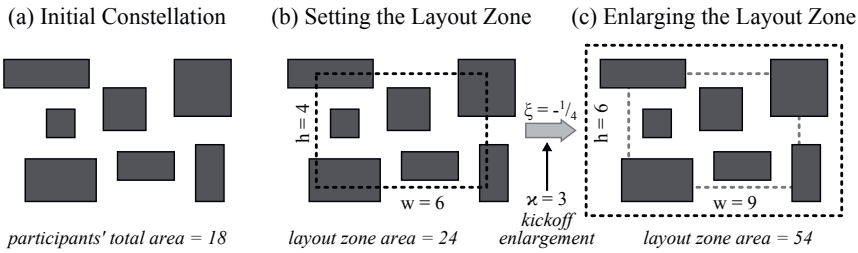


Figure 3.50: Exemplary illustration of a layout zone's kickoff enlargement.

In the initial module constellation, the participants are expected to be loosely scattered inside the kickoff zone Z_1 (as in the example of Figure 3.50). Just as well, the situation is fine if the participants are heaped up in the middle, overlapping each other. In that case, the participants will first disperse across Z_1 in the incipient tightening-settlement cycle before successively congregating in the remaining tightening-settlement cycles (as illustrated in Figure 3.4). However, one might also encounter quite the opposite case where the participants are initially located so far apart that some of them even lie beyond Z_1. In that situation, it may be preferable to overrule $\varkappa$ in order to enlarge the layout zone such that all participants are definitely *safe* before the interaction begins.

This can be achieved by determining the difference in height between the initial layout zone Z_0 and the constellation frame F (which defines the *vertically minimal* kickoff enlargement), as well as the respective difference in width (which defines the *horizontally minimal* kickoff enlargement). Including the contraction amount calculated from the user-specified kickoff enlargement multiplier $\varkappa$, the safe contraction amount ξ_{safe} can be obtained from the following

formula:[13]

$$\xi_{\text{safe}} = \min\left(\underbrace{\frac{1}{2} - \frac{1}{2}\sqrt{\frac{\varkappa \cdot [\mathfrak{P}\mathcal{P}]}{[Z_0]}}}_{\substack{\text{user-specified}\\ \text{kickoff enlargement}}}, \underbrace{\frac{\top Z_0 - \top F}{\top Z_0 - \bot Z_0}, \frac{\bot F - \bot Z_0}{\top Z_0 - \bot Z_0}}_{\substack{\text{vertically minimal}\\ \text{kickoff enlargement}}}, \underbrace{\frac{\dashv Z_0 - \dashv F}{\dashv Z_0 - \vdash Z_0}, \frac{\vdash F - \vdash Z_0}{\dashv Z_0 - \vdash Z_0}}_{\substack{\text{horizontally minimal}\\ \text{kickoff enlargement}}}\right). \tag{3.104}$$

Figure 3.51 gives an example. Presented in image (a) are eight participants with a total area of $[\mathfrak{P}\mathcal{P}] = 44$, with the initial layout zone Z_0 being a 6 by 8 rectangle and the constellation frame F having a width of 12 and a height of 14. For a user-specified kickoff enlargement multiplier $\varkappa = 3$, the contraction amount is $\xi = \frac{1}{2} - \frac{1}{2}\sqrt{(3 \cdot 44)/(6 \cdot 8)} = -0.33$, but one participant would be completely lost thereby while three participants would be prone, as can be seen in image (b). With the vertically minimal kickoff enlargement $\varkappa = 3.34$ –achieved via $\xi = \frac{-3}{8} = -0.375$ and illustrated in image (c)– two participants would still be prone. However, a contraction amount of $\xi = \frac{-3}{6} = -0.5$ leads to the horizontally minimal kickoff enlargement $\varkappa = 4.36$ which finally produces a zone size for Z_1 where all participants are safe, as shown in image (d), and thus represents the safe contraction amount in this case.

(a) Layout Zone and Constellation Frame (b) User-specified Kickoff Enlargement (c) Vertically Minimal Kickoff Enlargement (d) Horizontally Minimal Kickoff Enlargement

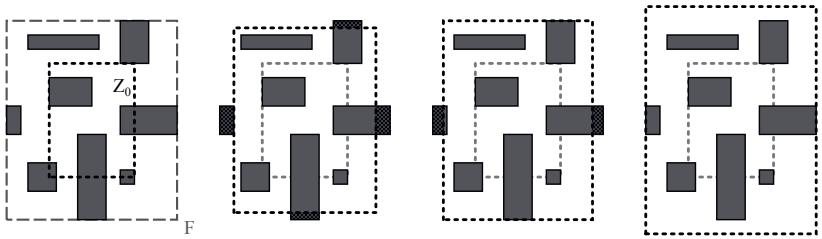


Figure 3.51: Safe contraction amount, corresponding to the horizontally minimal kickoff enlargement.

3.4.1.2 Tightening the Layout Zone

Now that a negative contraction provided the kickoff enlargement which turned Z_0 into Z_1, the module interaction begins. Therein, the interaction control organ applies positive contraction operations to the layout zone in order to achieve a tightening of Z_1, Z_2, Z_3 and so on after each settlement. Here, the respective contraction amount defines how rigorous the *tightening policy* is – considering the following two tenets, which represent two extremes in this regard:

[13] In fact, the two terms for the vertically minimal kickoff enlargement are redundant in equation 3.104 because the layout zone Z_0 is centered on the constellation frame F. The same is true regarding the two terms for the horizontally minimal kickoff enlargement.

- First, the tightening policy should sufficiently "put the screws" on the participants, which is to say that at least one participant should be *prone* after each tightening. Otherwise, i.e., if all participants are completely *safe* after each tightening, the tightening policy is too *lenient* and the layout zone does never reach its initial size but will instead approach a limit much greater than $\lceil Z_0 \rceil$. This would even be true in the hypothetical case that the minimal movement distance m is set to zero. The final layout zone size approached by Z would depend on the constellation itself, as exemplified in Figure 3.52.
- Second, the tightening policy should refrain from overpacing, which is to say that no participant should be *lost* after a tightening. Otherwise, the lost participants are forced to perform a ***Re-entering*** move instead of being able to choose the best action from a set of alternative options. Altough this is not a dealbreaking issue, such a tightening policy can in general be considered too *aggressive* as it tends to bring about unwanted chaotic behavior.

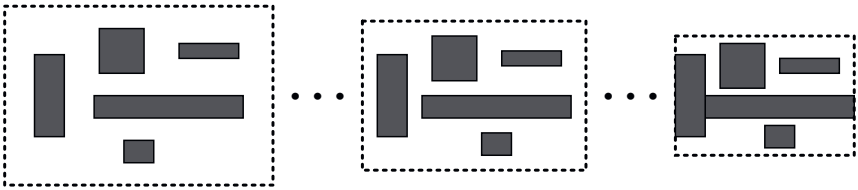


Figure 3.52: If the tightening policy is too lenient, the desired zone size cannot be reached.

Whether the tightening policy is too lenient or too aggressive, the ultimate implications are the same: the module interaction does not converge towards the desired goal. Hence, for every i^{th} tightening-settlement cycle (with $i \in \mathbb{N}^+$) the two tenets above define the task of finding a contraction amount ξ_i which tightens the layout zone from Z_i into Z_{i+1} after the settlement such that

$$\forall P \in \mathcal{P},\ (\exists P : \Psi_P \sqsubset \lceil P) \wedge (\nexists P : \Psi_P = \lceil P). \tag{3.105}$$

To facilitate a concise specification of the tightening policy's rigorousness, the *pressing rate* $\varpi \in \mathbb{R}$ in the open interval $(0, 1)$ is introduced. It allows to set ξ_i depending on the desired *protrusion extent* ψ of the participants (see Section 3.3.1.3), wherein each ψ is calculated relative to the participant's dimensions. This means, ξ_i is chosen such that the layout zone is tightened as much as possible without letting any participant P have a horizontal protrusion extent $\psi_{x,P}$ larger than ϖ-times the width w_P of P, nor a vertical protrusion extent $\psi_{y,P}$ larger than ϖ-times the height h_P of P.

In this manner, the pressing rate makes it possible to fine-tune the tightening policy in a continuous range between the two excluded endpoints $\varpi = 0$ (lenient) and $\varpi = 1$ (aggressive), as illustrated in Figure 3.53. In general, it is sensible to work with a pressing rate near $\varpi = 0.5$,

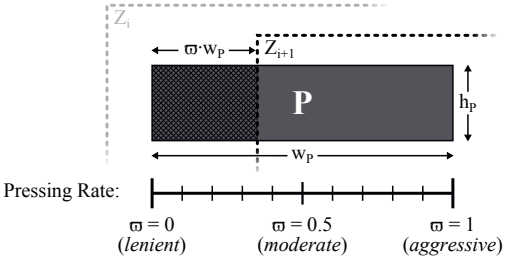


Figure 3.53: Visualization of a tightening policy's rigorousness, depending on the pressing rate.

which is referred to as a *moderate* tightening policy. For a given ϖ, the interaction control organ's consequential task per each tightening is to minimize the area of the layout zone while making sure that

$$(\forall P \in \mathcal{P},\ \psi_{x,P} \leq \varpi{\cdot}w_P \wedge \psi_{y,P} \leq \varpi{\cdot}h_P) \wedge (\exists P \in \mathcal{P} : \psi_{x,P} = \varpi{\cdot}w_P \vee \psi_{y,P} = \varpi{\cdot}h_P). \quad (3.106)$$

This is achieved by calculating the respective contraction amount ξ_i according to the following formula:

$$\begin{aligned}
\xi_i = \min \Bigg(& \min_{\forall P \in \mathcal{P}} \Bigg(\frac{\overbrace{\top Z_i - \top(\text{▯}P) + \varpi \top(\text{▯}P) - \varpi \bot(\text{▯}P)}^{\text{northern edge}}}{\top Z_i - \bot Z_i} \Bigg), \\
& \min_{\forall P \in \mathcal{P}} \Bigg(\frac{\overbrace{\bot(\text{▯}P) + \varpi \top(\text{▯}P) - \varpi \bot(\text{▯}P) - \bot Z_i}^{\text{southern edge}}}{\top Z_i - \bot Z_i} \Bigg), \\
& \min_{\forall P \in \mathcal{P}} \Bigg(\frac{\overbrace{\dashv Z_i - \dashv(\text{▯}P) + \varpi \dashv(\text{▯}P) - \varpi \vdash(\text{▯}P)}^{\text{eastern edge}}}{\dashv Z_i - \vdash Z_i} \Bigg), \\
& \min_{\forall P \in \mathcal{P}} \Bigg(\frac{\overbrace{\vdash(\text{▯}P) + \varpi \dashv(\text{▯}P) - \varpi \vdash(\text{▯}P) - \vdash Z_i}^{\text{western edge}}}{\dashv Z_i - \vdash Z_i} \Bigg) \Bigg).
\end{aligned} \quad (3.107)$$

This formula bears some resemblance to the formula for ξ_{safe} in Section 3.4.1.1, because the first two fractions here (which concern the northern edge and the southern edge) represent the equivalent to those of the vertically minimal kickoff enlargement in equation 3.104 while the other two fractions (eastern edge and western edge) correspond to those of the horizontally minimal kickoff enlargement therein. However, it is important to note that –in contrast to equation 3.104– the terms are not redundant here for two reasons: (1) the layout zone is not perfectly centered on the constellation since the minimal movement distance is larger than zero, and (2) the maximally permitted protrusion extent is relative to a participant's dimensions, as already mentioned above.

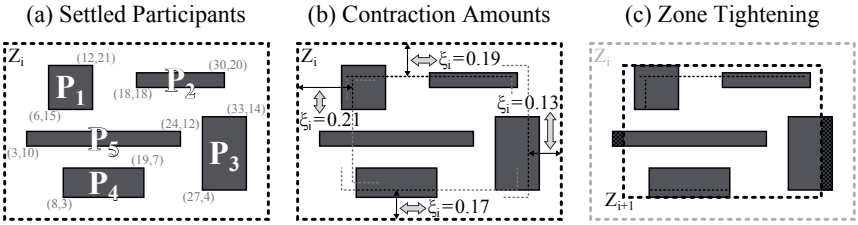


Figure 3.54: Exemplary illustration of a zone tightening, calculated from the pressing rate.

Table 3.10: Bounding box coordinates of the objects in the tightening example of Figure 3.54.

Object	Bounding Box
Z	$((0,0),(36,24))$
P_1	$((6,15),(12,21))$
P_2	$((18,18),(30,20))$
P_3	$((27,4),(33,14))$
P_4	$((8,3),(19,7))$
P_5	$((3,10),(24,12))$

To illustrate the calculation of the contraction amount via equation 3.107, Figure 3.54 shows an exemplary zone tightening for a constellation of five settled participants inside a layout zone Z_i, as given in image (a). The coordinates of the layout zone rectangle and those of the particicpants' bounding boxes are given in Table 3.10. With a –rather lenient– pressing rate of $\varpi = 0.25$ the contraction amount according to equation 3.107 becomes

$$\begin{aligned}\xi_i = \min\big(\min(\overbrace{0.19}^{P_1},\overbrace{0.19}^{P_2}, 0.52, 0.75, 0.52), \min(0.69, 0.77, 0.27, \overbrace{0.17}^{P_4}, 0.44),\\ \min(0.71, 0.25, \underbrace{0.13}_{P_3}, 0.55, 0.48), \min(\underbrace{0.21}_{P_1}, 0.58, 0.79, 0.3, 0.23)\big)\end{aligned} \tag{3.108}$$

where the five values in each of the four ancillary $\min$-terms correspond to the five participants (P_1, P_2, P_3, P_4, P_5) respectively. In the first term (concerning the northern edge), the smallest contraction amount is given by P_1 and P_2, in the remaining terms it is given by P_4 (southern edge), P_3 (eastern edge), and again P_1 (western edge), as indicated in image (b). Among these four values, the smallest one is

$$\xi_i = \min(0.19, 0.17, 0.13, 0.21) = 0.13 \tag{3.109}$$

which is the contraction amount construed from the eastern edge. Using this value, image (c) shows how the layout zone is tightened into Z_{i+1}. As can be seen, P_3 now has a horizontal

protrusion extent of $\psi_{x,P_3} = \varpi \cdot w_{P_3}$ while all other participants have lesser protrusion extents, which means that condition 3.106 is fulfilled.

3.4.1.3 Considering Rectilinear Layout Zones

The tightening formulas described so far work well for rectangular layout zones. But for rectilinear (i.e., nonrectangular) layout zones one can inadvertently ensnarl in situations where condition 3.106 is violated. This is not overly problematic because the surplus protrusion extent usually is rather small; and even if a participant is completely lost, a ***Re-entering*** move helps to bring it back into the layout zone. On this account, the tightening formulas above can be adequately used for rectangular as well as for rectilinear layout zones. Implementing a safe tightening procedure (where all participants definitely adhere to the maximally permitted protrusion extent) is more elaborate in the rectilinear case. To explicate this topic, the following ruminations sketch out the subtleties that must be paid attention to.[14]

Figure 3.55 (a) displays a settlement of seven participants inside a hexagonal layout zone Z_i. Due to the notch, there are two edges (named E_1 and E_2) which deviate from the bounding box $\square Z_i$. Applying the formula of equation 3.107 for a pressing rate of $\varpi = 0.5$ would lead to the tightened layout zone Z^*_{i+1} by which P_2 violates its permitted protrusion extent ψ_{y,P_2} in vertical direction. This violation –referred to as the *protrusion excess*– is colored deep black in image (b). To obtain the correct contraction amount, equation 3.107 must be enhanced such that it considers all edges of the layout zone. However, one further issue must be taken into account: when determining a potential contraction amount with respect to a particular edge E and a particular participant P, it may be that E does not even touch P because the length of E is smaller than the layout zone's respective dimension (i.e., the width of Z_i if E is a horizontal edge, the height of Z_i if E is a vertical edge). In that case, the potential contraction amount is to be discarded. Image (b) exemplifies such a situation: calculatorily, P_1 and E_2 determine the maximal contraction amount, but it is discarded since E_2 remains completely aloof from P_1 (not only for that contraction amount, but even regardless of ξ – the trajectory in the image visualizes how the first node of E_2 wanders distant to P_1 when scaling the layout zone). Next in size is the contraction amount determined by P_2 and E_2. In contrast to the (P_1, E_2) pair, E_2 is in touch with P_2 for that contraction amount and is therefore called ***afferent*** to P_2. In fact, this value is the smallest among all determinable contraction amounts and thus represents the desired ξ_i. The resultant zone tightening is illustrated in image (c): as can be seen, the desired protrusion extent is now met by P_2 and undercut by all other participants as should be.

For rectilinear layout zones, another circumstance has to be contemplated regarding the fact that a participant can protrude the layout zone at one of its concave vertices due to a tightening.

[14] Without going into a detailed elaboration, the considerations made in this Section 3.4.1.3 about the tightening of a rectilinear layout zone can likewise be applied to its kickoff enlargement.

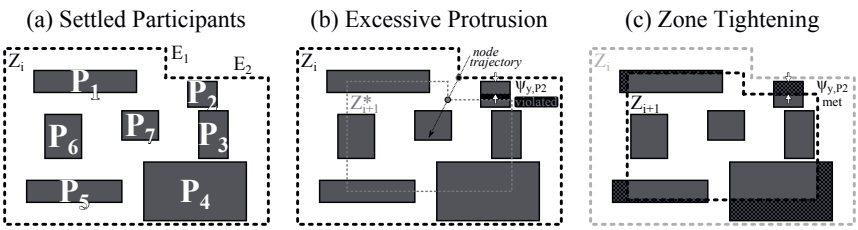


Figure 3.55: Exemplary illustration of a zone tightening for a rectilinear layout zone.

But first, row [a] in Figure 3.56 discriminates the possible cases of protrusion at a convex vertex as known from rectangular layout zones. Each of these cases represents a subsidiary offshoot of case (d2) in Figure 3.23. For a supposed pressing rate of $\varpi = 0.5$, neither ψ_x nor ψ_y violate the permitted protrusion extent in image [a1] of Figure 3.56. Hence, both are said to be *admissible*, which makes the overall protrusion Ψ admissible as well. In [a2], both ψ_x and ψ_y violate the permitted protrusion extent and are therefore called *excessive* (in line with the blackened protrusion excess outlined in the image). In that case, the overall protrusion is also excessive, i.e., the inherent contraction amount is too large and the tightening would be more rigorous than desired. In [a3], ψ_x is excessive while ψ_y is admissible; in [a4], the situation is vice versa. It goes without saying, that in those two cases again the overall protrusion is excessive.

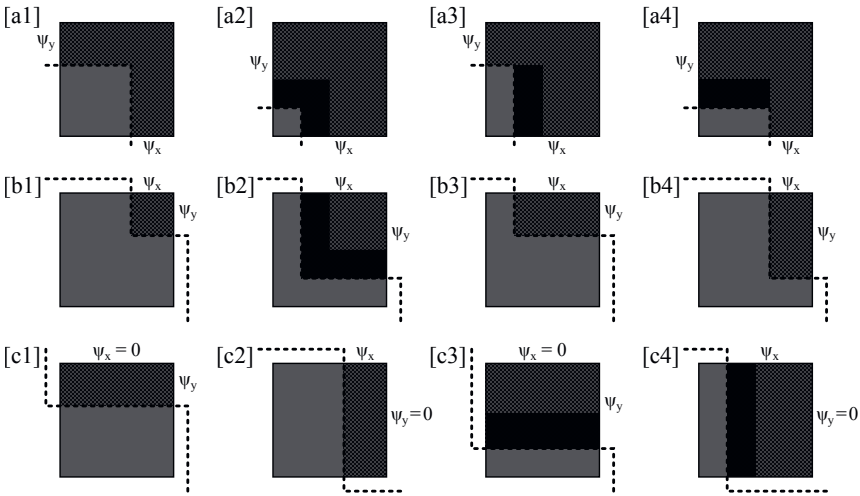


Figure 3.56: Cases of protrusion after a tightening: [a] convex vertex, [b] concave vertex, [c] edge.

If the protrusion occurs at a concave vertex (see row [b] in Figure 3.56, covering the different offshoots of case (d1) in Figure 3.23), the affair is the same as above if both ψ_x and ψ_y are admissible [b1] or if both are excessive [b2]. However, things are different if the protrusion extent is only violated in one direction. In [b3], ψ_x is excessive but the overall protrusion is admissible because ψ_y is admissible. In [b4], the overall protrusion is admissible since ψ_x is admissible although ψ_y is excessive. These two cases are justified by looking at row [c], which displays the offshoots of case (c2) from Figure 3.23, where the protrusion does not occur at a vertex but at an edge of the layout zone. In [c1], the protrusion is admissible because ψ_y is admissible. Although the participant protrudes the layout zone with its entire width, the horizontal protrusion extent is $\psi_x = 0$ (according to equation 3.53). The anomaly with this reckoning is, that in case [b3] –where ψ_y is the same and ψ_x is much greater– the protrusion area is less than in [c1]. Yet, this conclusion logically justifies why the protrusion is admissible in [b3]. The preceding considerations about [b3] and [c1] can be applied to [b4] and [c2] in the orthogonal direction to reason that the protrusion in [b4] is admissible as well. For the sake of completeness, the remaining two images in Figure 3.56 present cases of excessive edge protrusion through ψ_y in [c3] and through ψ_x in [c4].

Table 3.11 lists the admissibility/excessiveness for the three superordinate cases of protrusion in a matrix-like fashion, depending on the dimensionality of the protrusion excess (i.e., whether there is excess neither through ψ_x nor through ψ_y, excess both by ψ_x and ψ_y, or excess in either ψ_x or ψ_y exclusively). Following [159], protrusion at a convex vertex is denoted as *ear protrusion* while protrusion at a concave vertex is referred to as *mouth protrusion* in the table.

Table 3.11: Cases of protrusion after a tightening. The signifiers in square brackets refer to the images in Figure 3.56.

	Protrusion Excess		
	none	ψ_x **and** ψ_y	ψ_x **or** ψ_y
Ear Protrusion (convex vertex)	admissible [a1]	excessive [a2]	excessive [a3],[a4]
Mouth Protrusion (concave vertex)	admissible [b1]	excessive [b2]	admissible [b3],[b4]
Edge Protrusion	admissible [c1],[c2]	*not applicable*	excessive [c3],[c4]

The findings of Table 3.11 deliver an important insight: in the case of mouth protrusion, the greater one of the two potential contraction amounts is to be favored over the lesser one. This is in contrast to equation 3.107 (where the smallest value for ξ_i was sought) and further distinguishes it from the case of ear protrusion, where the lesser contraction amount is to be taken

instead of the greater one. To illuminate this statement, Figure 3.57 (a1) shows a participant P next to an ear (convex vertex) of the layout zone Z_i. The two edges E_1 and E_2 of the ear offer two potential contraction amounts. For a $\varpi = 0.5$ pressing rate, image (a2) illustrates the potential tightening into Z^*_{i+1} based on the ear's horizontal edge E_1. As can be seen, the overall protrusion for this contraction amount ξ_i^{P,E_1} is admissible. But based on the ear's vertical edge E_2, the other potential tightening entails an overall excessive protrusion (a3). Hence, the respective contraction amount ξ_i^{P,E_2} is to be rejected in favor of the smaller ξ_i^{P,E_1}. In (b1) however, the participant lies next to a mouth (concave vertex) of Z_i. As Table 3.11 tells right away, both potential tightenings (b2) and (b3) definitely lead to an overall admissible protrusion – the protrusion *cannot* be excessive in these cases because either ψ_x (b2) or ψ_y (b3) is congenitally admissible. For that reason, the larger contraction amount (ξ_i^{P,E_2} in this example) may overtrump the smaller one (ξ_i^{P,E_1} here) in the case of mouth protrusion.

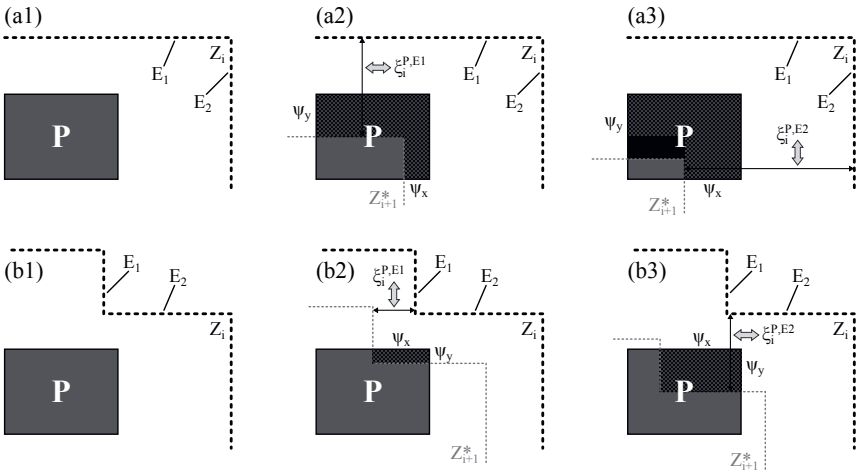


Figure 3.57: Depending on the case of protrusion, the smaller or the greater contraction amount matters.

In a nutshell, four issues have been revealed in this Section 3.4.1.3: for the tightening of an arbitrarily rectilinear layout zone (1) all possible participant/edge pairs have to be regarded, (2) it is necessary to check whether the respective edge is afferent to the participant, (3) it must be determined if the prospective protrusion of the participant is admissible at all, and (4) in the case of mouth protrusion the larger contraction amount is to be favored over the smaller one. Each of these issues can be found in Algorithm 4, which provides a pseudocode description of the contraction amount calculation in the rectilinear case. A couple of even further details need to be paid attention to, as the following explanation of Algorithm 4 points out.

Explanation of Algorithm 4

First of all, a set Ξ_{all} is initialized in line 1 that will later be used to gather all potential contraction amounts. Then, for each participant P among the set $\mathcal{P}$ of all participants, every node N of the layout zone Z_i is traversed. Every node N joins two edges: a horizontal edge denoted as ${E_{\bar{h}}}^N$ and a vertical edge referred to as E_v^N. If ${E_{\bar{h}}}^N$ is a northern edge[15] of Z_i whose vertical coordinate $y_{{E_{\bar{h}}}^N}$ lies above the center of Z_i, then the potential contraction amount $\xi_i^{P,{E_{\bar{h}}}^N}$ is calculated as shown in line 6. If ${E_{\bar{h}}}^N$ is a southern edge of Z_i and lies below the center of Z_i, then the potential contraction amount is calculated as in line 8. If neither is true, then $\xi_i^{P,{E_{\bar{h}}}^N}$ is set to a null value (line 10) – no contraction amount is calculated in this case because it would falsely lead to an enlargement of the layout zone instead of a tightening, as illustrated in Figure 3.58 (a).[16] The formulas in line 6 and line 8 correspond to the terms found in equation 3.107 for a rectangular layout zone's northern edge and southern edge. However, there is one pivotal difference: in the rectilinear case, the contraction amount must be projected from the considered edge onto the respective side of the layout zone's bounding box. For that purpose, the *projection divisor* χ is introduced. Figure 3.59 amplifies this on a rectilinear layout zone Z_i with height h_{Z_i}. For any contraction amount ξ_i, the absolute *edge displacement* of the layout zone's northernmost and southernmost edge is $\epsilon = \xi_i \cdot h_{Z_i}$. Since edge E is nearer to the center point of Z_i, its edge displacement ϵ_E is smaller by the projection divisor χ:

$$\epsilon_E = \frac{\epsilon}{\chi}. \tag{3.110}$$

Looking at the two plotted node trajectories and the horizontal line through the center point of Z_i, all three of which are intercepted by the pair of auxiliary parallels drawn into Figure 3.59, the projection divisor can be deduced from the relations between the four line segments a, b, c, d

[15] To determine if a horizontal edge $E_{\bar{h}}$ of a rectilinear polygon is a northern edge of that polygon, one can simply draw a vertical line through $E_{\bar{h}}$. If the line crosses an *even* number of other horizontal edges *above* $E_{\bar{h}}$ (and thus an *odd* number of other horizontal edges *below* $E_{\bar{h}}$), then $E_{\bar{h}}$ is a northern edge. Otherwise, the line would cross an *odd* number of other horizontal edges *above* $E_{\bar{h}}$ (and an *even* number of other horizontal edges *below* $E_{\bar{h}}$), indicating that $E_{\bar{h}}$ is a southern edge.

[16] As desired, the potential contraction amount calculated from the (P_1, E_1) pair in Figure 3.58 (a) would result in a tightening because E_1 is a northern edge of Z_i above its center point. In contrast, the (P_2, E_2) pair would lead to an enlargement because E_2 is a northern edge below the center point of Z_i. For that reason, E_2 is to be ignored in the calculation of the contraction amount. One might say that E_2 should not be completely ignored because the (P_3, E_2) pair would then be elided – yet, this *is* legitimate because the (P_3, E_3) pair would define the correct contraction amount in that case.
Analogous to these considerations about northern edges below the center of the layout zone, southern edges above the center of the layout zone also need to be ignored in the calculation of the contraction amount for the same reason.

via Thales' theorem (intercept theorem) as follows:

$$\frac{a}{b} = \frac{c}{d} \tag{3.111}$$

$$\frac{a}{c} = \frac{b}{d} \tag{3.112}$$

$$1 - \frac{a}{c} = 1 - \frac{b}{d} \tag{3.113}$$

$$\frac{c-a}{c} = \frac{d-b}{d} \tag{3.114}$$

$$\frac{c-a}{d-b} = \frac{c}{d} \tag{3.115}$$

$$\frac{c-a}{d-b} + 1 = \frac{c}{d} + 1 \tag{3.116}$$

$$\frac{c-a+d-b}{d-b} = \frac{c+d}{d} \tag{3.117}$$

wherein the four terms of the fractions can be substituted to produce the following equation

$$\frac{\epsilon}{\epsilon_E} = \frac{\frac{1}{2}\, h_{Z_i}}{y_E - \frac{1}{2}(\bot Z_i + \top Z_i)} \tag{3.118}$$

which represents the projection divisor χ. To calculate the projected contraction amount from a given edge displacement ϵ_E, equation $\epsilon = \xi_i \cdot h_{Z_i}$ can now be rearranged into

$$\xi_i = \frac{\epsilon}{h_{Z_i}} = \frac{\epsilon_E \cdot \chi}{h_{Z_i}} = \frac{\epsilon_E}{h_{Z_i}} \cdot \frac{\frac{1}{2}\, h_{Z_i}}{y_E - \frac{1}{2}(\bot Z_i + \top Z_i)} = \frac{\frac{1}{2}\, \epsilon_E}{y_E - \frac{1}{2}(\bot Z_i + \top Z_i)} \tag{3.119}$$

$$= \frac{\epsilon_E}{2 \cdot y_E - \bot Z_i - \top Z_i}. \tag{3.120}$$

For a northern edge of Z_i and a participant P, the absolute edge displacement is

$$\epsilon_E = y_{E_{\bar{h}}} - \top(\text{▯}P) + \varpi \cdot \top(\text{▯}P) - \varpi \cdot \bot(\text{▯}P) \tag{3.121}$$

which explicates line 6 in Algorithm 4. Since $y_{E_{\bar{h}}}$ corresponds to $\top Z_i$ in the dividend of the northern-edge term in equation 3.107, the formula in Algorithm 4 differs from that term by the factor

$$\frac{\top Z_i - \bot Z_i}{2 \cdot y_E - \bot Z_i - \top Z_i} = \frac{h_{Z_i}}{2 \cdot y_E - \bot Z_i - \top Z_i} = \frac{\frac{1}{2}\, h_{Z_i}}{y_E - \frac{1}{2}(\bot Z_i + \top Z_i)} \tag{3.122}$$

which –as expected– is precisely the projection divisor χ deduced above (see equation 3.118). For a southern edge of Z_i, the projection divisor becomes

$$\chi = \frac{\frac{1}{2}\, h_{Z_i}}{\frac{1}{2}(\bot Z_i + \top Z_i) - y_E} \tag{3.123}$$

which is reflected in line 8 of Algorithm 4. Considering the vertical edge E_v^N of the traversed node N, the formulas in line 14, line 16, and line 18 –which are analogous to those from the

horizontal edge $E_{\bar{h}}{}^N$ (line 6, line 8, and line 10 respectively)– provide the potential contraction amount ξ_i^{P,E_v^N} for the (P, E_v^N) pair. Equivalently, if E_v^N is an eastern edge to the left of Z_i's center or if E_v^N is a western edge to the right of Z_i's center, then it is discarded because it would lead to an enlargement of the layout zone (as visualized in Figure 3.58 (b)). It should be noted that the calculations in line 6, line 8, line 14, and line 16 cannot run into a *division by zero* because the respective if-conditions guarantee that $y_{E_{\bar{h}}{}^N} \neq \frac{1}{2}(\perp Z_i + \top Z_i)$ and $x_{E_v^N} \neq \frac{1}{2}(\vdash Z_i + \dashv Z_i)$.

Next (line 21 in Algorithm 4), those contraction amounts among $\{\xi_i^{P,E_{\bar{h}}{}^N}, \xi_i^{P,E_v^N}\}$ that have indeed been calculated (i.e., are not `null`) will be stored in a set $\Xi_{P,N}$ if the respective edge E^N is *afferent* to P and if the prospective protrusion Ψ_P^{new} is *admissible*. If $\Xi_{P,N}$ is not empty, the appropriate contraction amount must be chosen from that set. In the case that node N is a mouth, then the larger value found in $\Xi_{P,N}$ is added to the set Ξ_{all} in line 24 (concordant with the ruminations made in Figure 3.57); otherwise, N is an ear and the smaller value in $\Xi_{P,N}$[17] is taken (line 26). After all nodes have been traversed for each participant, the smallest value from the set Ξ_{all} is chosen in line 31 as the final contraction amount ξ_i for the subsequent tightening of the rectilinear layout zone.

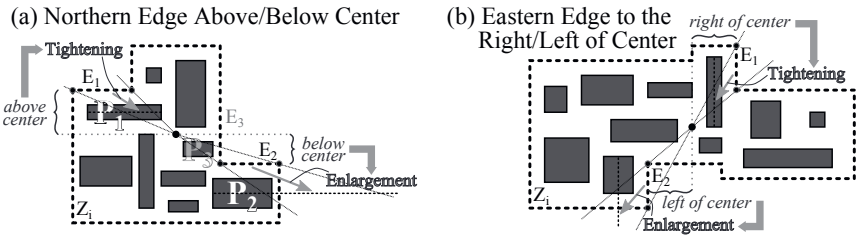


Figure 3.58: Certain edges of a rectilinear layout zone –such as (a) northern edges below the zone center and (b) eastern edges to the left of the zone center– need to be discarded when calculating the contraction amount.

3.4.2 Transient Tightening Policies

Although the successive tightening based on a single pressing rate ϖ is a quite dynamic ingredient of a SWARM run, it can still be regarded as being a bit too coarse because the layout zone gets tightened only *after* each individual settlement. This manner of *abrasive tightening*

[17] If node N is an ear, then $\Xi_{P,N}$ is in fact a *singleton* here. To understand the reason for this, two cases are to be distinguished: if the node would lie aloof from participant P after the prospective tightening, one of the two potential contraction amounts has been discarded in line 21 because the respective edge is not *afferent* to P; otherwise (i.e., if the node would lie on P after the tightening) one of the two potential contraction amounts has been discarded in line 21 because the prospective protrusion of P would not be *admissible*.

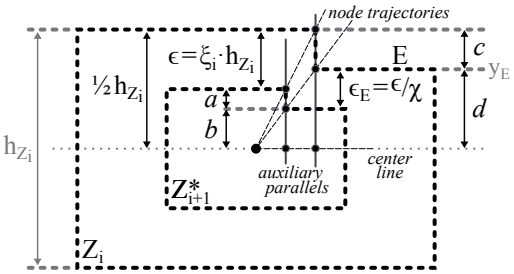


Figure 3.59: Exemplification of edge projection, as necessary in the case of a rectilinear layout zone.

can be visualized in form of a staircase-shaped *tightening profile*, as exemplarily done in Figure 3.60: every tightening (henceforth also denoted as a *major tightening*) can be recognized as an abrupt change in the size of the layout zone. To obtain a more fine-grained tightening characteristic, the idea of *transient tightening policies* is introduced. A transient tightening policy is a tightening policy where changes in the size of the layout zone may occur *during* a tightening-settlement cycle. Basically, two kinds of transient tightening policies can be conceived: a *progressive tightening policy* and a *regressive tightening policy*, both of which will be discussed in Section 3.4.2.1 and Section 3.4.2.2 respectively.

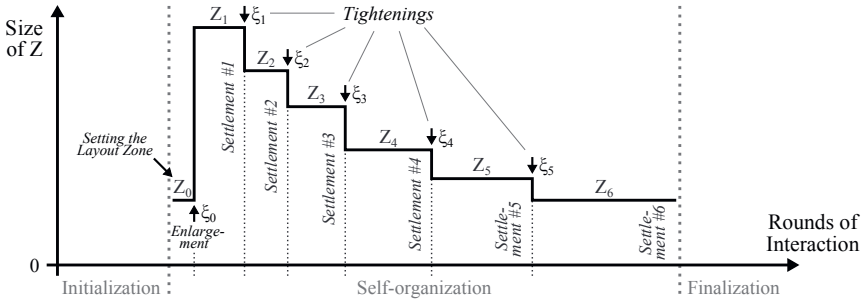


Figure 3.60: Exemplary tightening profile of an abrasive tightening policy.

3.4.2.1 Progressive Tightening

A *progressive tightening policy* is a transient tightening policy where each major tightening is in fact chopped up into several *minor tightenings*. Thus, the tightening policy is not specified solely through the pressing rate ϖ, but may for example be defined via (ϖ, u) wherein $u \in \mathbb{N}^+$ denotes the number of minor tightenings. With this tightening policy, every contraction

Algorithm 4 Contraction amount for the tightening of a rectilinear layout zone

$\Xi_{\text{all}} \leftarrow \{\,\}$
for each participant $P \in \mathcal{P}$ **do**
  for each node N of Z_i **do**
    ▷ **Horizontal edge of the node ($E_{\bar{h}}{}^N$):**
    if $E_{\bar{h}}{}^N$ is a northern edge of Z_i **AND** $y_{E_{\bar{h}}{}^N} > \frac{1}{2}(\perp Z_i + \top Z_i)$ **then**
      $\xi_i^{P,E_{\bar{h}}{}^N} \leftarrow (y_{E_{\bar{h}}{}^N} - \top(\mathbb{o}P) + \varpi\top(\mathbb{o}P) - \varpi\perp(\mathbb{o}P))/(2 \cdot y_{E_{\bar{h}}{}^N} - \perp Z_i - \top Z_i)$
    else if $E_{\bar{h}}{}^N$ is a southern edge of Z_i **AND** $y_{E_{\bar{h}}{}^N} < \frac{1}{2}(\perp Z_i + \top Z_i)$ **then**
      $\xi_i^{P,E_{\bar{h}}{}^N} \leftarrow (\perp(\mathbb{o}P) + \varpi\top(\mathbb{o}P) - \varpi\perp(\mathbb{o}P) - y_{E_{\bar{h}}{}^N})/(\perp Z_i + \top Z_i - 2 \cdot y_{E_{\bar{h}}{}^N})$
    else
      $\xi_i^{P,E_{\bar{h}}{}^N} \leftarrow \texttt{null}$
    end if
    ▷ **Vertical edge of the node (E_{υ}^N):**
    if E_{υ}^N is an eastern edge of Z_i **AND** $x_{E_{\upsilon}^N} > \frac{1}{2}(\vdash Z_i + \dashv Z_i)$ **then**
      $\xi_i^{P,E_{\upsilon}^N} \leftarrow (x_{E_{\upsilon}^N} - \dashv(\mathbb{o}P) + \varpi\dashv(\mathbb{o}P) - \varpi\vdash(\mathbb{o}P))/(2 \cdot x_{E_{\upsilon}^N} - \vdash Z_i - \dashv Z_i)$
    else if E_{υ}^N is a western edge of Z_i **AND** $x_{E_{\upsilon}^N} < \frac{1}{2}(\vdash Z_i + \dashv Z_i)$ **then**
      $\xi_i^{P,E_{\upsilon}^N} \leftarrow (\vdash(\mathbb{o}P) + \varpi\dashv(\mathbb{o}P) - \varpi\vdash(\mathbb{o}P) - x_{E_{\upsilon}^N})/(\vdash Z_i + \dashv Z_i - 2 \cdot x_{E_{\upsilon}^N})$
    else
      $\xi_i^{P,E_{\upsilon}^N} \leftarrow \texttt{null}$
    end if
    ▷ **Memorizing the appropriate contraction amount**
    $\Xi_{P,N} \leftarrow \left\{\xi \in \{\xi_i^{P,E_{\bar{h}}{}^N}, \xi_i^{P,E_{\upsilon}^N}\} \mid \xi \neq \texttt{null} \wedge E^N \text{ is afferent to } P \wedge \Psi_P^{\text{new}} \text{ is admissible}\right\}$
    if $|\Xi_{P,N}| > 0$ **then**
      if N is a mouth **then**
        $\Xi_{\text{all}} \leftarrow \Xi_{\text{all}} \dot{+} \max(\Xi_{P,N})$
      else
        $\Xi_{\text{all}} \leftarrow \Xi_{\text{all}} \dot{+} \min(\Xi_{P,N})$
      end if
    end if
  end for
end for
$\xi_i \leftarrow \min(\Xi_{\text{all}})$

amount ξ_i that was calculated from the desired pressing rate ϖ is simply divided into u minor contraction amounts $\xi_{i,1}, \xi_{i,2}, \ldots, \xi_{i,u}$ all of which reduce the area of the layout zone by the same magnitude. These minor contraction amounts are then successively applied to the layout zone –one after each round of interaction– until u minor tightenings have been performed. In the remaining rounds of interaction, the layout zone is not tightened any further before all participants have settled. The respective tightening profile of this *linear progressive tightening policy* looks like the one displayed in Figure 3.61 for an exemplary value of $u = 4$ as the number of minor tightenings.

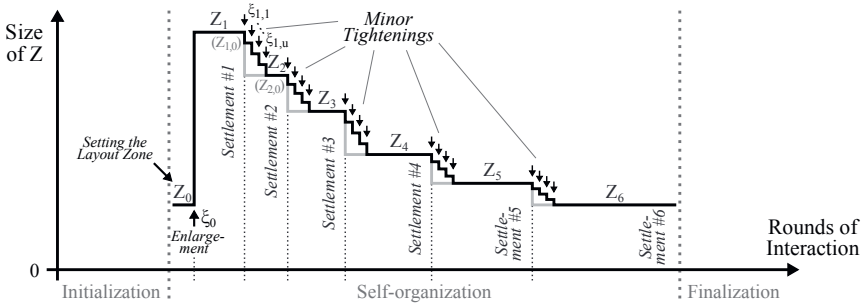


Figure 3.61: Exemplary tightening profile of a linear progressive tightening policy.

The first minor contraction amount $\xi_{i,1}$ can be calculated by setting the areic difference $\lceil Z_{i+1} \rceil - \lceil Z_i \rceil$ based on $\xi_{i,1}$ equal to the desired fraction of the areic difference based on ξ_i:

$$w(1-2\xi_{i,1}) \cdot h(1-2\xi_{i,1}) - w \cdot h = \frac{w(1-2\xi_i) \cdot h(1-2\xi_i) - w \cdot h}{u} \tag{3.124}$$

$$wh \cdot \left((1-2\xi_{i,1})^2 - 1\right) = wh \cdot \left((1-2\xi_i)^2 - 1\right) \cdot \frac{1}{u} \tag{3.125}$$

$$1 - 4\xi_{i,1} + 4\xi_{i,1}^2 - 1 = (1 - 4\xi_i + 4\xi_i^2 - 1) \cdot \frac{1}{u} \tag{3.126}$$

$$-\xi_{i,1} + \xi_{i,1}^2 = (-\xi_i + \xi_i^2) \cdot \frac{1}{u} \tag{3.127}$$

$$\xi_{i,1}^2 - \xi_{i,1} + \frac{1}{u}(\xi_i - \xi_i^2) = 0. \tag{3.128}$$

This quadratic equation can be easily solved with the well-known *quadratic formula*, which produces

$$\xi_{i,1} = \frac{1 \pm \sqrt{1 - \frac{4}{u}(\xi_i - \xi_i^2)}}{2}. \tag{3.129}$$

It should be noted that $\xi_i - \xi_i^2 \leq 0.25$ for any ξ_i (and that $u \geq 1$ as already stated), which means that the discriminant of the polynomial in equation 3.129 never becomes negative and the equation is therefore always solvable (without consulting complex numbers). But, concerning the case where the square root is *added* in the dividend of equation 3.129, the minor contraction amount is $\xi_{i,1} \geq 0.5$ for any u and for any ξ_i, and thus lies out of the valid range (defined in Section 3.4.1.1). Hence, the minor contraction amount is unequivocally given only by the case where the square root is *subtracted* in the dividend:

$$\xi_{i,1} = \frac{1 - \sqrt{1 - \frac{4}{u}(\xi_i - \xi_i^2)}}{2}. \tag{3.130}$$

To calculate the second minor contraction amount $\xi_{i,2}$ one may contemplate that the total ***edge displacement*** of the western edge[18] is destined to be $\epsilon_i = \xi_i \cdot w_{i,0}$ where $w_{i,0}$ denotes the width w of the layout zone's bounding box before the first minor tightening. After the first minor tightening, the edge displacement obtained so far is $\epsilon_{i,1} = \xi_{i,1} \cdot w_{i,0}$, which means that the remaining edge displacement is $\epsilon_i - \epsilon_{i,1} = (\xi_i - \xi_{i,1}) \cdot w_{i,0}$. If this remaining edge displacement was to be achieved with one single tightening, the ***residual contraction amount*** $\xi_{i,[2,u]}$ could be determined via the following equation, where $w_{i,1}$ with $w_{i,1} = w_{i,0} \cdot (1-2\xi_{i,1})$ denotes the width of the layout zone's bounding box after the first minor tightening:

$$\xi_{i,[2,u]} \cdot w_{i,1} = (\xi_i - \xi_{i,1}) \cdot w_{i,0} \tag{3.131}$$

$$\xi_{i,[2,u]} = (\xi_i - \xi_{i,1}) \cdot \frac{w_{i,0}}{w_{i,1}} \tag{3.132}$$

$$\xi_{i,[2,u]} = (\xi_i - \xi_{i,1}) \cdot \frac{w_{i,0}}{w_{i,0} \cdot (1 - 2\xi_{i,1})} \tag{3.133}$$

$$\xi_{i,[2,u]} = \frac{\xi_i - \xi_{i,1}}{1 - 2\xi_{i,1}}. \tag{3.134}$$

With this value, equation 3.130 can be used to calculate the second minor contraction amount $\xi_{i,2}$, wherein the number of minor tightenings now must be $u - 1$ since one minor tightening has already been performed. On that account, equation 3.130 becomes

$$\xi_{i,2} = \frac{1 - \sqrt{1 - \frac{4}{u-1}(\xi_{i,[2,u]} - \xi_{i,[2,u]}^2)}}{2}. \tag{3.135}$$

Based on this proceeding, equation 3.130 can be written for $j = 1 \ldots u$ in general form as

$$\xi_{i,j} = \frac{1 - \sqrt{1 - \frac{4}{u+1-j}(\xi_{i,[j,u]} - \xi_{i,[j,u]}^2)}}{2} \tag{3.136}$$

to calculate the minor contraction amounts from $\xi_{i,1}$ to $\xi_{i,u}$. To provide the residual contraction amounts $\xi_{i,[j,u]}$ that appear on the right-hand side of equation 3.136, equation 3.134 can be generalized into

$$\xi_{i,[j,u]} = \frac{\xi_i}{\prod_{k=1}^{j-1}(1 - 2\xi_{i,k})} - \sum_{k=1}^{j-1} \frac{\xi_{i,k}}{\prod_{n=k}^{j-1}(1 - 2\xi_{i,n})}. \tag{3.137}$$

[18] Taking the western edge here is arbitrary. Any other edge (eastern, southern, northern) would ultimately lead to the same result in equation 3.134.

As a plausibility check, evaluating equation 3.136 for $j = u$ unfurls into

$$\xi_{i,u} = \frac{1 - \sqrt{1 - \frac{4}{u+1-u}(\xi_{i,[u,u]} - \xi_{i,[u,u]}^2)}}{2} \tag{3.138}$$

$$= \frac{1 - \sqrt{1 - 4\xi_{i,[u,u]} + 4\xi_{i,[u,u]}^2}}{2} \tag{3.139}$$

$$= \frac{1 - \sqrt{(1 - 2\xi_{i,[u,u]})^2}}{2} \tag{3.140}$$

$$= \frac{1 - 1 + 2\xi_{i,[u,u]}}{2} \tag{3.141}$$

$$= \xi_{i,[u,u]} \tag{3.142}$$

which means, that the last minor contraction amount $\xi_{i,u}$ is equal to the last residual contraction amount $\xi_{i,[u,u]}$ (as expected). Furthermore, a spot sample can be evaluated from equation 3.137 for $j = 1$: comprehending that the *empty product* is by convention equal to the multiplicative identity 1 just as the *empty sum* is by convention equal to the additive identy 0, the spot sample folds up into

$$\xi_{i,[1,u]} = \frac{\xi_i}{\prod_{k=1}^{1-1}(1 - 2\xi_{i,k})} - \sum_{k=1}^{1-1} \frac{\xi_{i,k}}{\prod_{n=k}^{1-1}(1 - 2\xi_{i,n})} \tag{3.143}$$

$$= \frac{\xi_i}{1} - 0 = \xi_i \tag{3.144}$$

which explains the appearance of ξ_i instead of $\xi_{i,[j,u]}$ in equation 3.130.

An even sleeker tightening characteristic than the one of the linear progressive tightening policy above can be obtained with an *exponential progressive tightening policy*. Such a tightening policy is defined through (ϖ, q) where q in the open interval $(0, 1)$ represents the so-called *contraction quotient*. With this tightening policy, every contraction amount ξ_i that is calculated from the pressing rate ϖ, is spread over multiple minor contraction amounts all of which reduce the area of the layout zone by the same factor. One by one, these minor contraction amounts are then applied to the layout zone after each round of interaction until all participants have settled. For a contraction quotient of $q = 0.5$ the tightening profile of this tightening policy is exemplarily depicted in Figure 3.62.

The first minor contraction amount $\xi_{i,1}$ can be calculated from equation 3.130 by setting $u = \frac{1}{q}$, which leads to

$$\xi_{i,1} = \frac{1 - \sqrt{1 - 4q \cdot (\xi_i - \xi_i^2)}}{2}. \tag{3.145}$$

After the first minor tightening, the second minor contraction amount $\xi_{i,2}$ likewise is

$$\xi_{i,2} = \frac{1 - \sqrt{1 - 4q \cdot (\xi_{i,[2,u]} - \xi_{i,[2,u]}^2)}}{2}. \tag{3.146}$$

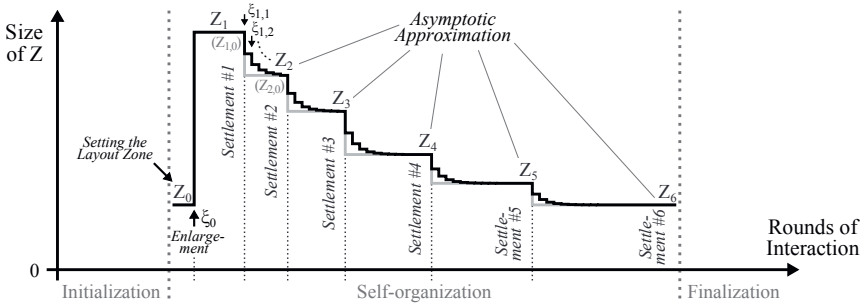


Figure 3.62: Exemplary tightening profile of an exponential progressive tightening policy.

So, it becomes obvious that the generalized formula for calculating all minor contraction amounts is

$$\xi_{i,j} = \frac{1 - \sqrt{1 - 4q \cdot (\xi_{i,[j,u]} - \xi_{i,[j,u]}^2)}}{2} \tag{3.147}$$

for which the residual contraction amounts $\xi_{i,[j,u]}$ can again be determined via equation 3.137 as before. Alternatively, equation 3.137 can also be turned into a recursive formula as follows:

$$\xi_{i,[j,u]} = \frac{\xi_i}{\prod_{k=1}^{j-1}(1-2\xi_{i,k})} - \sum_{k=1}^{j-1} \frac{\xi_{i,k}}{\prod_{n=k}^{j-1}(1-2\xi_{i,n})} \tag{3.148}$$

$$= \frac{\xi_i}{(1-2\xi_{i,j-1}) \cdot \prod_{k=1}^{j-2}(1-2\xi_{i,k})} - \sum_{k=1}^{j-2} \frac{\xi_{i,k}}{(1-2\xi_{i,j-1}) \cdot \prod_{n=k}^{j-2}(1-2\xi_{i,n})} - \frac{\xi_{i,j-1}}{1-2\xi_{i,j-1}} \tag{3.149}$$

$$= \frac{1}{1-2\xi_{i,j-1}} \cdot \left(\underbrace{\frac{\xi_i}{\prod_{k=1}^{j-2}(1-2\xi_{i,k})} - \sum_{k=1}^{j-2} \frac{\xi_{i,k}}{\prod_{n=k}^{j-2}(1-2\xi_{i,n})}}_{= \xi_{i,[j-1,u]}} - \xi_{i,j-1} \right) \tag{3.150}$$

$$= \frac{\xi_{i,[j-1,u]} - \xi_{i,j-1}}{1-2\xi_{i,j-1}} \tag{3.151}$$

which is applicable for $j > 1$. With this case distinction, the formula for all $j = 1 \ldots u$ becomes

$$\xi_{i,[j,u]} = \begin{cases} \xi_i & \Leftrightarrow \quad j = 1 \\ \frac{\xi_{i,[j-1,u]} - \xi_{i,j-1}}{1-2\xi_{i,j-1}} & \Leftrightarrow \quad j > 1. \end{cases} \tag{3.152}$$

With an exponential progressive tightening policy, the layout zone gets arbitrarily close to its designated size during each tightening-settlement cycle. This can be proven as follows: for the j^{th} minor tightening, the change in the size of the layout zone is

$$\lceil Z_{i,j} \rceil - \lceil Z_{i,j-1} \rceil = q \cdot (\lceil Z_{i+1} \rceil - \lceil Z_{i,j-1} \rceil). \tag{3.153}$$

For $j-1$ it can thus equally be said that

$$\lceil Z_{i,j-1}\rceil - \lceil Z_{i,j-2}\rceil = q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-2}\rceil) \tag{3.154}$$

$$\lceil Z_{i,j-1}\rceil = q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-2}\rceil) + \lceil Z_{i,j-2}\rceil \tag{3.155}$$

which may be used to substitute $\lceil Z_{i,j-1}\rceil$ on the right-hand side of equation 3.153 to produce

$$\begin{aligned}
\lceil Z_{i,j}\rceil - \lceil Z_{i,j-1}\rceil &= q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-1}\rceil) && (3.156)\\
&= q\cdot(\lceil Z_{i+1}\rceil - (q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-2}\rceil) + \lceil Z_{i,j-2}\rceil)) && (3.157)\\
&= q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-2}\rceil - q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-2}\rceil)) && (3.158)\\
&= q\cdot(1-q)\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-2}\rceil) && (3.159)\\
&= q\cdot(1-q)^2\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,j-3}\rceil) && (3.160)\\
&\quad\vdots && (3.161)\\
&= q\cdot(1-q)^{j-1}\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,0}\rceil). && (3.162)
\end{aligned}$$

Now, the layout zone's total change in size after j minor tightenings can be expressed as the sum

$$\lceil Z_{i,j}\rceil - \lceil Z_{i,0}\rceil = \lceil Z_{i,j}\rceil \underbrace{-\lceil Z_{i,j-1}\rceil + \lceil Z_{i,j-1}\rceil}_{=0} \underbrace{-\lceil Z_{i,j-2}\rceil + \lceil Z_{i,j-2}\rceil}_{=0} - \ldots \underbrace{-\lceil Z_{i,1}\rceil + \lceil Z_{i,1}\rceil}_{=0} - \lceil Z_{i,0}\rceil \tag{3.163}$$

$$= \sum_{k=1}^{j} \lceil Z_{i,k}\rceil - \lceil Z_{i,k-1}\rceil \tag{3.164}$$

$$= \sum_{k=1}^{j} q\cdot(1-q)^{k-1}\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,0}\rceil) \tag{3.165}$$

$$= q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,0}\rceil)\cdot\sum_{k=0}^{j-1}(1-q)^k \tag{3.166}$$

and the value $\lceil Z_{i,\infty}\rceil$ that the zone size tends to, can be calculated through the mathematical limit

$$\lceil Z_{i,\infty}\rceil = \lceil Z_{i,0}\rceil + q\cdot(\lceil Z_{i+1}\rceil - \lceil Z_{i,0}\rceil)\cdot\lim_{j\to\infty}\sum_{k=0}^{j-1}(1-q)^k. \tag{3.167}$$

In mathematics, the term q^k for $k\geq 0$ is denoted as a *geometric sequence*, while the sum $\sum \mathrm{q}^k$ represents its *geometric series*. The mathematical limit of such a geometric series is known to be

$$\sum_{k=0}^{\infty} = \frac{1}{1-\mathrm{q}} \tag{3.168}$$

for $|\mathfrak{q}| < 1$. Since the contraction quotient q must be $0 < q < 1$ by definition (as already stated), it is sure that $|1 - q| < 1$ which allows to resolve equation 3.167 into

$$\lceil Z_{i,\infty} \rceil = \lceil Z_{i,0} \rceil + q \cdot (\lceil Z_{i+1} \rceil - \lceil Z_{i,0} \rceil) \cdot \lim_{j \to \infty} \sum_{k=0}^{j-1} (1-q)^k \tag{3.169}$$

$$= \lceil Z_{i,0} \rceil + q \cdot (\lceil Z_{i+1} \rceil - \lceil Z_{i,0} \rceil) \cdot \Big(\underbrace{\lim_{j \to \infty} \sum_{k=0}^{j} (1-q)^k}_{= \frac{1}{1-(1-q)} = \frac{1}{q}} - \underbrace{\lim_{j \to \infty} (1-q)^j}_{=0} \Big) \tag{3.170}$$

$$= \lceil Z_{i,0} \rceil + q \cdot (\lceil Z_{i+1} \rceil - \lceil Z_{i,0} \rceil) \cdot \frac{1}{q} \tag{3.171}$$

$$= \lceil Z_{i+1} \rceil. \tag{3.172}$$

Thus (in contrast to a geometric series, where the limit depends on the base of the exponentiation), the q and $\frac{1}{q}$ cancel each other out in equation 3.171 such that the layout zone asymptotically approaches its designated size $\lceil Z_{i+1} \rceil$ independent of the contraction quotient. So, compared to a linear progressive tightening policy, an exponential progressive tightening policy has the advantage that it is more rigorous in the beginning of a settlement and then lets the layout zone snuggle down towards the desired dimensions quite smoothly, not abruptly.

With an exponential progressive tightening policy, one might come to believe that the participants will never be able to conclude a settlement because the layout zone gets tightened unceasingly. However, this is prevented by the *minimal movement distance* (see Section 3.3.3.1): eventually, there is a minor tightening in each tightening-settlement cycle whose contraction amount is so small that no participant performs an action because the length of the action's movement vector is smaller than the minimal movement distance.

3.4.2.2 Regressive Tightening

A *regressive tightening policy* is a transient tightening policy where a major tightening is indeed performed after every settlement, but is then followed by several *minor enlargements*. In this way, every tightening is at first as rigorous as with an abrasive tightening policy – however, the reins are loosened afterwards, which can be helpful if the participants have trouble finding a viable constellation. Although various kinds of regressive tightening policies can be conceived in theory (such as a linear or an exponential one, in the style of those discussed in Section 3.4.2.1), three particular traits should be envisaged here per tightening-settlement cycle: (1) immediately after the major tightening, the minor enlargements should be rather slight so as not to depart from the designated zone size too quickly, (2) then, the minor enlargements should become more and more lavish to oblige the rivaling participants, and (3) towards the end of the tightening-settlement cycle, the layout zone should be prevented from ever attaining the

size that it had before the major tightening – otherwise, the flow of self-organization would not make any headway towards the final layout zone size.

A transient tightening policy that exhibits these three traits can be deduced from a *logistic function* and is therefore denoted as a *logistic regressive tightening policy*. A logistic function has a characteristic sigmoid curve, as illustrated in Figure 3.63 (a). The tightening policy is specified by (ϖ, z, z^*), where z and z^* are *zone size quotients*: z defines the maximal minor enlargement that is to occur during a tightening-settlement cycle, and z^* specifies the first minor enlargement after each major tightening. Both z and z^* relate the desired zone size change to the total difference in zone size that results from the major tightening. In mathematics, a logistic function is in general defined by the formula

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{3.173}$$

and can now be carried over to SWARM in order to describe the desired layout zone size $\lceil Z_{i,j} \rceil$ for all $j > 1$ (i.e., the zone size after every minor enlargement). As will now be explained in detail, carrying over the formula involves three steps, displayed in Figure 3.63 (b)–(d).

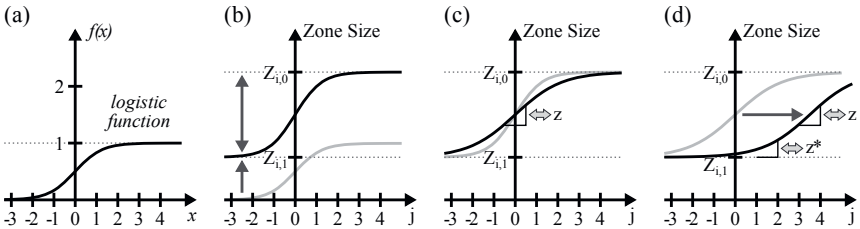


Figure 3.63: Utilizing a logistic function's sigmoid curve for a logistic regressive tightening policy.

First, the sigmoid curve has to be stretched and shifted in vertical direction as shown in Figure 3.63 (b), such that the layout zone size regresses from $\lceil Z_{i,1} \rceil$ (zone size after the major tightening) towards $\lceil Z_{i,0} \rceil$ (zone size before the major tightening). Thus, the formula becomes

$$\lceil Z_{i,j} \rceil = \frac{\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil}{1 + \mathrm{e}^{-j}} + \lceil Z_{i,1} \rceil \tag{3.174}$$

where the dividend $\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil$ stretches the curve and the addend $\lceil Z_{i,1} \rceil$ shifts the curve upwards.

Second, referring to Figure 3.63 (c), the sigmoid curve must be stretched horizontally to meet the given zone size quotient z. For that purpose, the formula is rewritten with a different base β as

$$\lceil Z_{i,j} \rceil = \frac{\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil}{1 + \beta^{-j}} + \lceil Z_{i,1} \rceil \tag{3.175}$$

where β can now be deduced from z to obtain the desired steepness. The derivate of the logistic function $f(x)$ above is largest at $x = 0$ (the abscissa of the function's inflection point). Hence, the greatest vertical difference Δy between two points on $f(x)$ that are horizontally apart by $\Delta x = 1$ is found between $x = -\frac{1}{2}$ and $x = \frac{1}{2}$ and amounts to

$$\Delta y = f\left(\tfrac{1}{2}\right) - f\left(-\tfrac{1}{2}\right). \tag{3.176}$$

This equation can be expressed in a different way because $f(x)$ possesses point symmetry around the point $(0, f(0))$. The proof is given by checking if $f(x) - f(0) = -f(-x) + f(0)$:

$$\frac{1}{1+\mathrm{e}^{-x}} - \frac{1}{1+\mathrm{e}^{0}} = -\frac{1}{1+\mathrm{e}^{x}} + \frac{1}{1+\mathrm{e}^{0}} \tag{3.177}$$

$$\frac{1}{1+\frac{1}{\mathrm{e}^{x}}} - \frac{1}{2} = -\frac{1}{1+\mathrm{e}^{x}} + \frac{1}{2} \tag{3.178}$$

$$\frac{\mathrm{e}^{x}}{\mathrm{e}^{x}+1} - \frac{1}{2} = -\frac{1}{1+\mathrm{e}^{x}} + \frac{1}{2} \tag{3.179}$$

$$\frac{\mathrm{e}^{x}}{\mathrm{e}^{x}+1} + \frac{1}{1+\mathrm{e}^{x}} = \frac{1}{2} + \frac{1}{2} \tag{3.180}$$

$$\frac{\mathrm{e}^{x}+1}{\mathrm{e}^{x}+1} = 1. \quad \square \tag{3.181}$$

Now, due to this point symmetry, the function slope expressed in equation 3.176 can be rewritten as

$$\Delta y = 2 \cdot \left(f\left(\tfrac{1}{2}\right) - f(0)\right) \tag{3.182}$$

which represents the maximal minor enlargement, and here also constitutes the zone size quotient z because the height of the $f(x)$ curve is 1. Substituting base e with a variable β allows to attain a specific z, for which β can be deduced as follows:

$$\Delta y = 2 \cdot \left(f\left(\tfrac{1}{2}\right) - f(0)\right) \tag{3.183}$$

$$z = 2 \cdot \left(\frac{1}{1+\beta^{-\frac{1}{2}}} - \frac{1}{2}\right) \tag{3.184}$$

$$z + 1 = \frac{2}{1+\beta^{-\frac{1}{2}}} \tag{3.185}$$

$$1 + \beta^{-\frac{1}{2}} = \frac{2}{z+1} \tag{3.186}$$

$$\beta = \left(\frac{2}{z+1} - 1\right)^{-2} \tag{3.187}$$

$$\beta = \left(\frac{1+z}{1-z}\right)^{2}. \tag{3.188}$$

Inserting this term in equation 3.175 to replace the base variable β leads to the new formula

$$\lceil Z_{i,j} \rfloor = \frac{\lceil Z_{i,0} \rfloor - \lceil Z_{i,1} \rfloor}{1 + \left(\frac{1+z}{1-z}\right)^{-2j}} + \lceil Z_{i,1} \rfloor. \tag{3.189}$$

According to Figure 3.63 (d), the third step is to shift the sigmoid curve in horizontal direction such that the first minor enlargement after the major tightening is achieved as was specified through z^*. A look at Figure 3.63 (c) suggests that the curve must be shifted rightwards by the distance between the vertical axis and the abscissa at which $f(x)$ veers away from $\lceil Z_{i,1} \rceil$ by the specified zone size difference. This can be formally expressed as follows:

$$\frac{\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil}{1 + \left(\frac{1+z}{1-z}\right)^{-2j}} + \lceil Z_{i,1} \rceil - \lceil Z_{i,1} \rceil = z^* \cdot (\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil) \tag{3.190}$$

$$\frac{\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil}{z^* \cdot (\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil)} = 1 + \left(\frac{1+z}{1-z}\right)^{-2j} \tag{3.191}$$

$$\frac{1}{z^*} - 1 = \left(\frac{1+z}{1-z}\right)^{-2j} \tag{3.192}$$

$$-2j = \log_{\frac{1+z}{1-z}} \left(\frac{1}{z^*} - 1\right). \tag{3.193}$$

As is the case with the contraction quotient q in an exponential progressive tightening policy, both z and z^* lie in the open interval $(0, 1)$. This means, that computing the logarithm is indeed allowed here because $\frac{1+z}{1-z} > 1$ and $\frac{1}{z^*} - 1 > 0$ for all values that z and z^* can assume. So, equation 3.193 can now be solved with the aid of the logarithmic laws, leading to

$$-2j = \frac{\ln\left(\frac{1}{z^*} - 1\right)}{\ln\left(\frac{1+z}{1-z}\right)} \tag{3.194}$$

$$j = \frac{\ln\left(\frac{1}{z^*} - 1\right)}{-2 \cdot \ln\left(\frac{1+z}{1-z}\right)}. \tag{3.195}$$

This value for j delivers the desired horizontal shift of the sigmoid curve.[19] But it should be noted, that in effect the curve must be shifted even further since the specified zone size difference is to be effectuated by the first minor enlargement, which in turn occurs after the initial major tightening. To be consistent with the indexing of the kickoff enlargement (where the index i is 0) and the major contraction amounts (with $i \geq 1$), the additional horizontal shift must be 2. Hence, equation 3.189 finally becomes

$$\lceil Z_{i,j} \rceil = \frac{\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil}{1 + \left(\frac{1+z}{1-z}\right)^{-2\left(j + \frac{\ln\left(\frac{1}{z^*}-1\right)}{-2 \cdot \ln\left(\frac{1+z}{1-z}\right)} - 2\right)}} + \lceil Z_{i,1} \rceil. \tag{3.196}$$

which –by expanding the exponent in the divisor– can be simplified into

$$\lceil Z_{i,j} \rceil = \frac{\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil}{1 + \left(\frac{1+z}{1-z}\right)^{-2j + \ln\left(\frac{1}{z^*}-1\right) / \ln\left(\frac{1+z}{1-z}\right) + 4}} + \lceil Z_{i,1} \rceil. \tag{3.197}$$

[19] A plausibility check is to set $z^* = 0.5$ which correctly results in $j = 0$ as expected.

For example, let there be a logistic regressive tightening policy with $z = \frac{1}{4}$ and $z^* = \frac{1}{40}$, wherein a major tightening occurs from $\lceil Z_{i,0} \rceil = 23$ to $\lceil Z_{i,1} \rceil = 19$. Table 3.12 shows the zone sizes that are (based on equation 3.197) calculated for $1 < j \leq 10$ and the corresponding curve is depicted in Figure 3.64. As can be seen, the first minor enlargement increases the size of the layout zone by 0.1 (which is precisely $\frac{1}{40}$ of $\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil = 4$). The greatest minor enlargement in this example occurs between $j = 5$ and $j = 6$ and amounts to 0.9982 (which is near to $\frac{1}{4}$ of $\lceil Z_{i,0} \rceil - \lceil Z_{i,1} \rceil$). The reason why this minor enlargement does not exactly match the given zone size quotient z here, is that the greatest minor enlargement may –depending on $\lceil Z_{i,0} \rceil$, $\lceil Z_{i,1} \rceil$, and z^*– also occur between non-integral values for j (in this example between $j = 5.0859$ and $j = 6.0859$, where the zone size changes from 20.5 to 21.5 understandably).

Table 3.12: Calculation values of a tightening example with a logistic regressive tightening policy.

Round j	**Zone Size** $\lceil Z_{i,j} \rceil$	**Absolute Change** $\lceil Z_{i,j} \rceil - \lceil Z_{i,j-1} \rceil$	**Contraction Amount** $\xi_{i,j}$
0	23.0000	—	0.0456
1	19.0000	-4.0000	-0.0013
2	19.1000	0.1000	-0.0022
3	19.2660	0.1660	-0.0051
4	19.6607	0.3947	-0.0095
5	20.4186	0.7580	-0.0121
6	21.4168	0.9982	-0.0095
7	22.2367	0.8199	-0.0050
8	22.6870	0.4503	-0.0021
9	22.8814	0.1944	-0.0008
10	22.9565	0.0751	-0.0003

Now that the desired zone sizes can be expressed via equation 3.197, the respective contraction amounts are to be calculated. In line with the determination in Section 3.4.1.2 that a major contraction amount ξ_i tightens the layout zone from Z_i into Z_{i+1}, a minor contraction amount $\xi_{i,j}$ here changes the layout zone from $Z_{i,j}$ into $Z_{i,j+1}$. Following the previous ruminations, such a contraction amount $\xi_{i,j}$ scales both the width and the height of the layout zone by a factor of $(1 - 2\xi_{i,j})$ respectively. The combined change of width and height must of course be equal to the relative change in zone size. Formally articulated, this allows to calculate the

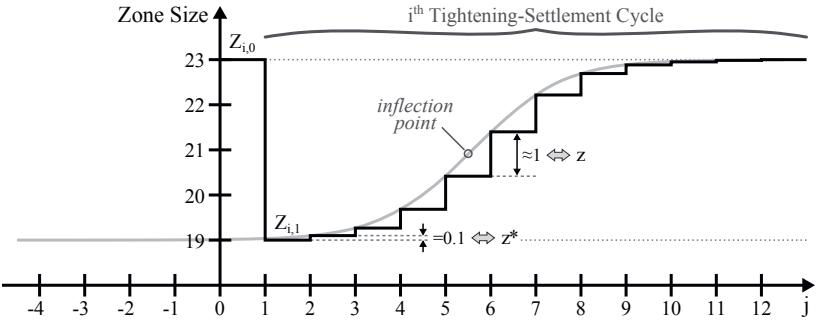


Figure 3.64: Zone size curve of the logistic regressive tightening example from Table 3.12.

sought contraction amount as follows:

$$(1 - 2\xi_{i,j})^2 = \frac{\lceil Z_{i,j+1} \rceil}{\lceil Z_{i,j} \rceil} \tag{3.198}$$

$$1 - 2\xi_{i,j} = \sqrt{\frac{\lceil Z_{i,j+1} \rceil}{\lceil Z_{i,j} \rceil}} \tag{3.199}$$

$$2\xi_{i,j} = 1 - \sqrt{\frac{\lceil Z_{i,j+1} \rceil}{\lceil Z_{i,j} \rceil}} \tag{3.200}$$

$$\xi_{i,j} = \frac{1}{2} - \frac{1}{2}\sqrt{\frac{\lceil Z_{i,j+1} \rceil}{\lceil Z_{i,j} \rceil}}. \tag{3.201}$$

Regarding the given example, the contraction amounts calculated with this formula are also listed in Table 3.12. Naturally, $\xi_{i,0}$ is positive because the contraction is indeed a tightening, whereas the other contraction amounts are negative since they correspond to the minor enlargements of the layout zone.

As it is the case in Figure 3.64, z^* is supposed to be much smaller than z. This statement calls for attention to one further subtlety: if z^* is too large, then the tightening profile can look awkward because the change from $Z_{i,1}$ to $Z_{i,2}$ is greater than the change from $Z_{i,2}$ to $Z_{i,3}$. Such a situation, referred to as *hyperregression*, is depicted in Figure 3.65 (a). On the other hand, it may be possible to find a certain value z^*_{iso} for which there is a situation of *isoregression*, where the zone size differences in these two minor enlargements are equal, as in image (b). If z^* is smaller than that value, then the first minor enlargement is lesser than the second one, which is denoted as *hyporegression* (c).

In principle, Figure 3.65 makes it obvious that finding z^*_{iso} is possible by taking equation 3.173 and determining the abscissa x at which the difference between $f(x+1)$ and $f(x)$ is

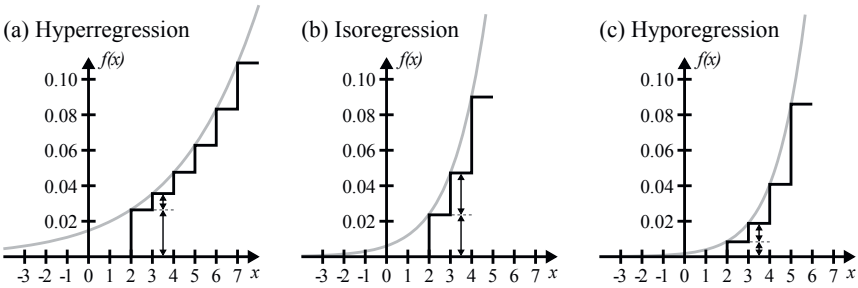


Figure 3.65: Depending on the first minor enlargement, three types of regression can be discerned.

equal to $f(x)$. Again, the base may be a variable β, which formally produces:

$$\frac{1}{1+\beta^{-(x+1)}} - \frac{1}{1+\beta^{-x}} = \frac{1}{1+\beta^{-x}} \tag{3.202}$$

$$\frac{1}{1+\beta^{-x-1}} - \frac{2}{1+\beta^{-x}} = 0 \tag{3.203}$$

$$1+\beta^{-x} - 2 \cdot \left(1+\beta^{-x-1}\right) = 0 \tag{3.204}$$

$$1+\beta^{-x} - 2 - 2 \cdot \beta^{-x} \cdot \beta^{-1} = 0 \tag{3.205}$$

$$\beta^{-x} \cdot \left(1 - \frac{2}{\beta}\right) = 1 \tag{3.206}$$

$$\beta^{-x} = \frac{1}{\frac{\beta-2}{\beta}} \tag{3.207}$$

$$x = -\log_\beta \left(\frac{\beta}{\beta-2}\right). \tag{3.208}$$

Having β as the base is fine for the logarithm because $\beta = \left(\frac{1+z}{1-z}\right)^2$ and $\frac{1+z}{1-z} > 1$ (as already mentioned before), which means that $\beta > 1$. However, the argument $\frac{\beta}{\beta-2}$ is only positive for $\beta > 2$ (and for $\beta < 0$ but that is not relevant here since $\beta > 1$), as visualized by the graph in Figure 3.66. This means, that a value for z^*_{iso} can only be found if $\beta > 2$, which in turn implies

that z must be[20]

$$\left(\frac{1+z}{1-z}\right)^2 > 2 \tag{3.209}$$

$$\frac{1+z}{1-z} > \sqrt{2} \tag{3.210}$$

$$1+z > \sqrt{2} - \sqrt{2}z \tag{3.211}$$

$$z \cdot (1+\sqrt{2}) > \sqrt{2} - 1 \tag{3.212}$$

$$z > \frac{\sqrt{2}-1}{\sqrt{2}+1} \tag{3.213}$$

$$z > 0.1716 \tag{3.214}$$

which can also be seen in Figure 3.66. For convenience, equation 3.208 can be simplified into

$$x = -\log_\beta\left(\frac{\beta}{\beta-2}\right) \tag{3.215}$$

$$= -\frac{\ln\left(\frac{\beta}{\beta-2}\right)}{\ln(\beta)} \tag{3.216}$$

$$= -\frac{\ln(\beta) - \ln(\beta-2)}{\ln(\beta)} \tag{3.217}$$

$$= \frac{\ln(\beta-2)}{\ln(\beta)} - 1. \tag{3.218}$$

To illustrate the matter that isoregression requires $\beta > 2$, subtracting the left-hand side of equation 3.202 from its right-hand side produces the function $\Delta_{\text{regression}}(x)$:

$$\Delta_{\text{regression}}(x) = \frac{2}{1+\beta^{-x}} - \frac{1}{1+\beta^{-x-1}}. \tag{3.219}$$

Now, the curve family in Figure 3.67 is obtained by plotting $\Delta_{\text{regression}}(x)$ for various β values. As the image illustrates, the curves for $\beta \leq 2$ do not cross the x-axis, whereas each of the curves for $\beta > 2$ has exactly one zero point, depicted as white circles on the x-axis (for $\beta = 2.78$, $\beta = 4$, and $\beta = 9$). The abscissae of these zero points represent the x-values of isoregression (for the given curves, according to equation 3.218, these are: $x = -1.24$, $x = -0.5$, and $x = -0.11$). To the right of such a zero point, the figure shows that $\Delta_{\text{regression}} > 0$, which signifies hyperregression. To the left of such a zero point, there is hyporegression since the respective curve runs below the x-axis.

The zero points from Figure 3.67 are confirmed by Figure 3.68, where equation 3.218 is plotted in the lower-right quadrant as a function $x(\beta)$. The upper-left quadrant displays the function $x^{-1}(\beta) = \beta(x)$ which represents the inverse function of $x(\beta)$. The white circles from

[20]The multiplication with $1 - z$ in condition 3.210 does not involve an inversion of the inequation because $1 - z > 0$.

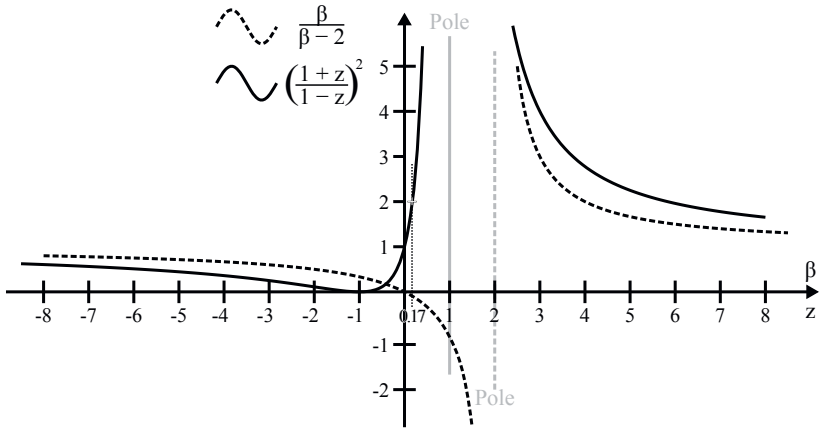


Figure 3.66: Relevant graphs for the calculation of isoregression, showing that β must be greater than 2 (to achieve that $\frac{\beta}{\beta-2} > 0$) and that z must be greater than 0.17 (such that $\beta = (\frac{1+z}{1-z})^2 > 2$).

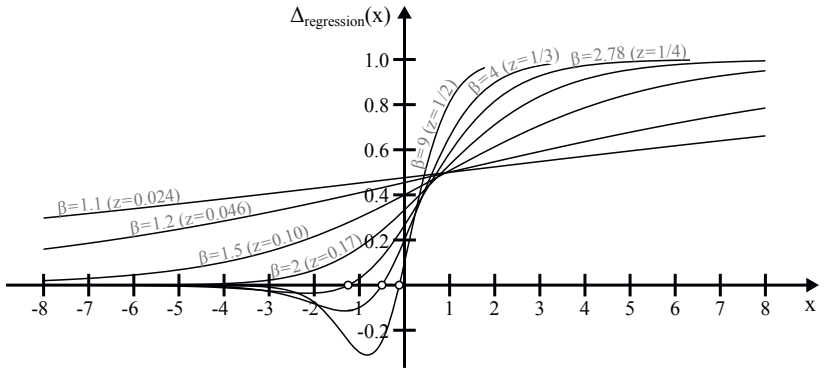


Figure 3.67: Curve family to illustrate that isoregression can only be obtained if β is greater than 2.

Figure 3.67 can also be found in Figure 3.68 to visualize that the vertical lines through these circles cross the $\beta(x)$ function precisely at $\beta = 2.78$, $\beta = 4$, and $\beta = 9$ (as expected).

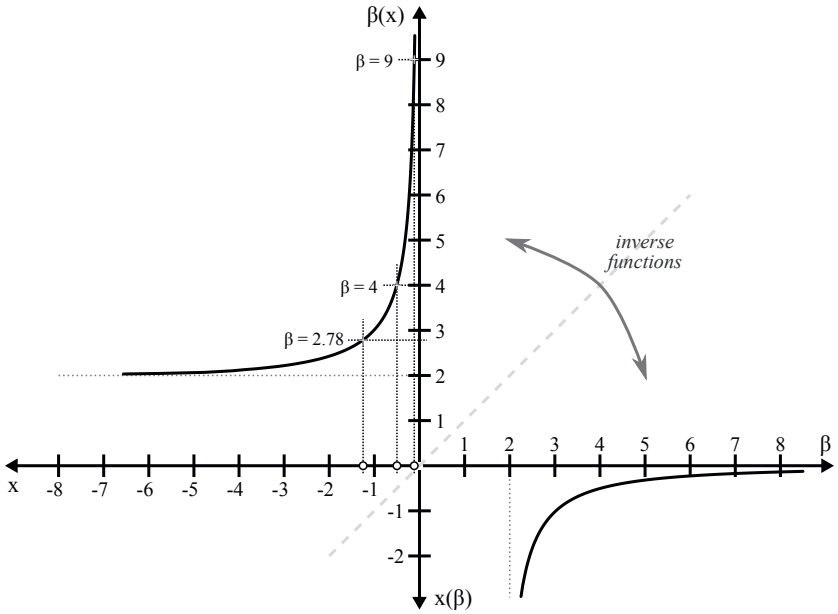


Figure 3.68: Plot of the isoregression equation 3.218 as a function $x(\beta)$ and its inverse function $\beta(x)$.

Based on these considerations, the calculation of z^*_{iso} can now be achieved by inserting the term from equation 3.208 into the logistic function of equation 3.173 (again, with e substituted by β). This delivers the abscissa's ordinate, which simultaneously represents the sought value for z^*_{iso}:

$$z^*_{\text{iso}} = \frac{1}{1+\beta^{-\left(-\log_\beta\left(\frac{\beta}{\beta-2}\right)\right)}} \tag{3.220}$$

$$= \frac{1}{1+\frac{\beta}{\beta-2}} \tag{3.221}$$

$$= \frac{\beta-2}{2\beta-2}. \tag{3.222}$$

Figure 3.69 illustrates z^*_{iso} as a function of β. Since $\beta > 1$ by definition, legal values for z^*_{iso} (with $0 < z^*_{\text{iso}} < 1$) are only obtained for $\beta > 2$ (as already discussed before). With respect to

z, the isoregression value z^*_{iso} is given by the formula

$$z^*_{\text{iso}} = \frac{\left(\frac{1+z}{1-z}\right)^2 - 2}{2\left(\frac{1+z}{1-z}\right)^2 - 2} \tag{3.223}$$

$$= \frac{\left(\frac{1+2z+z^2}{1-2z+z^2}\right) - 2}{2\left(\frac{1+2z+z^2}{1-2z+z^2}\right) - 2} \tag{3.224}$$

$$= \frac{\frac{1+2z+z^2-2(1-2z+z^2)}{1-2z+z^2}}{\frac{2(1+2z+z^2)-2(1-2z+z^2)}{1-2z+z^2}} \tag{3.225}$$

$$= \frac{-z^2+6z-1}{8z} \tag{3.226}$$

$$= -\frac{z}{8} + \frac{3}{4} - \frac{1}{8z} \tag{3.227}$$

which is also depicted in Figure 3.69. Here, the legal values for z^*_{iso} are found between the first zero crossing of the curve and $z = 1$. The zero crossings occur at

$$-z^2 + 6z - 1 = 0 \tag{3.228}$$

$$z = \frac{-6 \pm \sqrt{6^2-4}}{-2} \tag{3.229}$$

$$z = 3 \mp \sqrt{8} \tag{3.230}$$

and the first of these lies at $3 - \sqrt{8} = 0.1716$ (which equals the value from equation 3.214 of course). The fact, that the $z^*_{\text{iso}}(z)$ curve runs below the first median (i.e., the bisecting line of the Cartesian system's upper-right quadrant), indicates that z is always greater than its z^*_{iso} value.

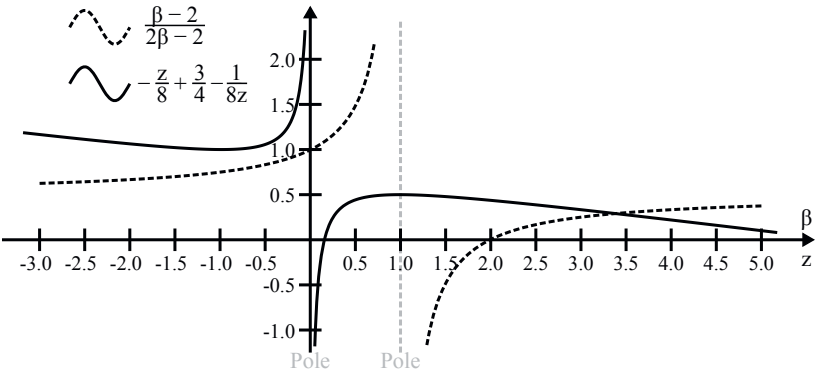


Figure 3.69: The isoregression zone size quotient z^*_{iso} as a function of β and as a function of z.

As an example for calculating z^*_{iso}, let there be a logistic regressive tightening policy with $z = \frac{1}{5}$. According to equation 3.227, isoregression is then effecuated for $z^* = -\frac{1}{5 \cdot 8} + \frac{3}{4} - \frac{5}{8} =$

$\frac{-1+30-25}{40} = \frac{1}{10}$. Imagining a major tightening from $\lceil Z_{i,0} \rceil = 20$ to $\lceil Z_{i,1} \rceil = 15$, equation 3.197 produces $\lceil Z_{i,2} \rceil = 15.5$ and $\lceil Z_{i,3} \rceil = 16.0$ (so, since $\lceil Z_{i,2} \rceil - \lceil Z_{i,1} \rceil = \lceil Z_{i,3} \rceil - \lceil Z_{i,2} \rceil$, isoregression has been achieved here).

With the formulas above, it is possible to give evidence of a statement that was made earlier about Figure 3.65 without any proof: if z^* is smaller than z^*_{iso}, then the obtained regression is hyporegression (otherwise hyperregression). This can be attested by looking at equation 3.220: a value for z^* that is smaller (greater) than z^*_{iso} implies that $-\log_\beta\left(\frac{\beta}{\beta-2}\right)$ must become smaller (greater).[21] As known from equation 3.208, this term corresponds to the abscissae in Figure 3.67, for which it is obvious that a smaller (greater) x value –to the left (right) of a zero point– leads to hyporegression (hyperregression), as already pointed out before.

Altogether, the tightening profile of a logistic regressive tightening policy is illustrated in Figure 3.70. This exemplary depiction reflects three characteristic *episodes* of a SWARM run's self-organization phase. In the beginning, when there is still a lot of free space available, the module interaction proceeds comparably quickly because the rivalry is low in this first episode. Later, the ongoing tightening of the layout zone effectuates increasingly competitive situations, such that more and more rounds of interaction are required to achieve a settlement wherein all participants are contented. Typically it is this second episode where the main hurdles of the problem need to be overcome and which decides whether the self-organization will be ultimately successful or not. If the participants manage to get through, then the remaining tightening-settlement cycles involve less and less disruptive actions which affect the layout arrangement only slightly. In this third episode, especially close to the end, it can usually be observed that only *Centering* actions are performed, such that the relative locations of the participants are maintained while their overall constellation is successively driven towards its final size.

The characteristics of that third episode can be important for transient tightening policies. In the case of a logistic regressive tightening policy, Figure 3.70 indicates that the last tightening must be abrasive, not regressive; otherwise, the tightening policy might be unable to reach the final layout size. Thus, since the interaction control organ does not loosen the layout zone during the last tightening-settlement cycle, it is vital that only marginal actions are performed therein.

[21] For an in-depth analysis, a case distinction can be made:

- If $-\log_\beta\left(\frac{\beta}{\beta-2}\right)$, subsequently referred to as x (from equation 3.208), is negative and becomes smaller (greater), then its absolute value $|x|$ becomes greater (smaller), β^{-x} is $\beta^{|x|}$ and becomes greater (smaller) since $\beta > 1$, and so the whole term of equation 3.220 becomes smaller.
- If $-\log_\beta\left(\frac{\beta}{\beta-2}\right)$, again referred to as x, is positive and becomes smaller (greater), then β^{-x} is $\frac{1}{\beta^{|x|}}$ and becomes greater (smaller) since $\beta^{|x|}$ becomes smaller (greater), and so the whole term of equation 3.220 becomes smaller.

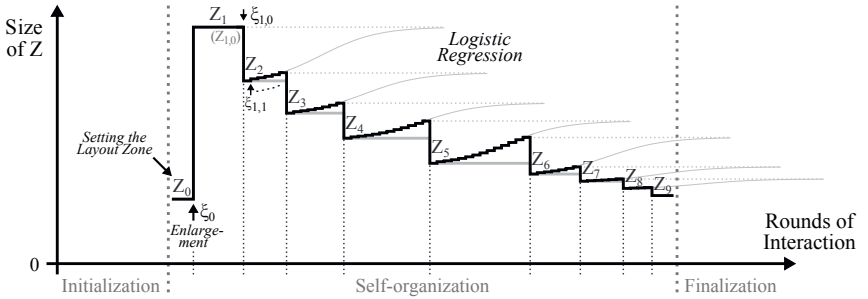


Figure 3.70: Exemplary tightening profile of a logistic regressive tightening policy.

3.4.3 Comfort Padding

Another topic which can be discussed within the interaction control organ's scope of activities, is the idea of *comfort padding*. Simply put, comfort padding allows to preserve layout space around a participant during the self-organization. SWARM distinguishes two forms of comfort padding that will be described in Section 3.4.3.1 and Section 3.4.3.2: *solid comfort padding* and *volatile comfort padding*.

3.4.3.1 Solid Comfort Padding

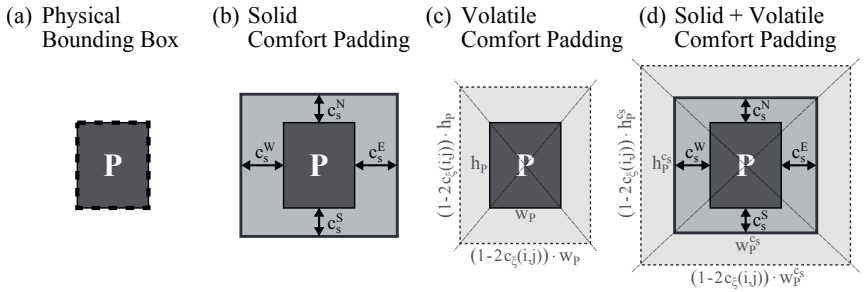


Figure 3.71: Depiction of the different forms of comfort padding around a participant.

Solid comfort padding preserves a fix amount of layout space around a participant, which means that the amount of layout space that is to be preserved does not change throughout a SWARM run. Figuratively speaking, the consideration of comfort padding is achieved by treating the participant as if it were bigger than it actually is. In a formal way, the expression of equation 3.23 (which takes the participant's physically relevant shapes into account) can be

said to define only the *physical bounding box* $\boxed{\,}_{\mathrm{p}}P$ of a participant P, as emphasized in Figure 3.71 (a). Then, the consideration of solid comfort padding can be achieved by specifying the bounding box operator $\boxed{\,}P$ (for usage throughout all the formulas in Chapter 3) to be

$$\boxed{\,}P \leftarrow \underset{\mathrm{p+s}}{\boxed{\,}}P. \tag{3.231}$$

If the amount of solid comfort padding is identical on each side of the participant, this *solid bounding box* operator $\underset{\mathrm{p+s}}{\boxed{\,}}P$ can be defined as

$$\underset{\mathrm{p+s}}{\boxed{\,}}P = \circledcirc_{c_{\mathrm{s}}}\big(\boxed{\,}_{\mathrm{p}}P\big) \tag{3.232}$$

whereby the physical bounding box of P is simply enlarged in all directions by the same *solid comfort padding amount* c_{s}. If different amounts of solid comfort padding are desired on each side of P (referred to as $c_{\mathrm{s}}^{\mathrm{N}}$, $c_{\mathrm{s}}^{\mathrm{E}}$, $c_{\mathrm{s}}^{\mathrm{S}}$, and $c_{\mathrm{s}}^{\mathrm{W}}$ for the northern, eastern, southern, and western side respectively), the operator can be formally defined in rectangle notation via

$$\underset{\mathrm{p+s}}{\boxed{\,}}P = \Big(\big(\vdash\!(\boxed{\,}_{\mathrm{p}}P) - c_{\mathrm{s}}^{\mathrm{W}}, \perp\!(\boxed{\,}_{\mathrm{p}}P) - c_{\mathrm{s}}^{\mathrm{S}}\big), \big(\dashv\!(\boxed{\,}_{\mathrm{p}}P) + c_{\mathrm{s}}^{\mathrm{E}}, \top\!(\boxed{\,}_{\mathrm{p}}P) + c_{\mathrm{s}}^{\mathrm{N}}\big)\Big) \tag{3.233}$$

which is illustrated in Figure 3.71 (b). The idea of solid comfort padding is useful when it is desired to preserve a certain amount of layout space between the interacting participants for a subsequent routing step that will then be performed in the finalization phase. This is also about to be demonstrated by the examples of Section 4.3.

3.4.3.2 Volatile Comfort Padding

In contrast to solid comfort padding, the volatile comfort padding of a participant changes throughout the self-organization phase of a SWARM run. The change depends on the size of the layout zone, so this is how the idea of comfort padding is correlated with the activities of the interaction control organ. Like solid comfort padding, volatile comfort padding preserves layout space around a participant during the module interaction – however, this is not done for the sake of the layout space, but with the intention to streamline the flow of self-organization. In fact, the amount of layout space preserved by the volatile comfort padding approaches zero towards the end of the self-organization phase. As above, volatile comfort padding can likewise be incorporated by specifying

$$\boxed{\,}P \leftarrow \underset{\mathrm{p+v}}{\boxed{\,}}P \tag{3.234}$$

for the formulas of Chapter 3. Similar to equation 3.232, the already known contraction operator $\circledast$ can be used to define the *volatile bounding box* operator $\underset{\mathrm{p+v}}{\boxed{\,}}P$ as

$$\underset{\mathrm{p+v}}{\boxed{\,}}P = \circledast_{c_\xi(i,j)}\big(\boxed{\,}_{\mathrm{p}}P\big) \tag{3.235}$$

but in contrast to c_{s} in equation 3.232, the *volatile comfort padding factor* $c_{\xi}(i,j)$ here is not a constant value but changes with each major zone tightening i (and every minor zone tightening j, if applicable). The $\circledast$ operator satisfies the notion that volatile comfort padding is supposed to keep the aspect ratio of the participant, as it is shown in Figure 3.71 (c). It may also be the case that both solid and volatile comfort padding are requested. For that purpose, the bounding box operator can be specified as

$$\boxdot P \leftarrow \underset{\text{p+s+v}}{\boxdot P} \tag{3.236}$$

with

$$\underset{\text{p+s+v}}{\boxdot P} = \circledast_{c_{\xi}(i,j)} \left(\underset{\text{p+s}}{\boxdot P} \right) \tag{3.237}$$

which signifies that the volatile comfort padding is clasped (as a kind of "outer" padding) around the participant's solid comfort padding (serving as its "inner" padding). This can also be seen in Figure 3.71 (d), where the combination of solid and volatile comfort padding is depicted. The following ruminations also concentrate on this case of combined (solid plus volatile) comfort padding since equation 3.237 can be considered a generalized form of equation 3.235.[22]

To determine the padding factor $c_{\xi}(i,j)$ for equation 3.237, the *volatile comfort padding share* c_{v} with $c_{\mathrm{v}} \geq 0$ is introduced. It allows to specify how much of the vacant space inside the layout zone $Z_{i,j}$ should be occupied by the volatile comfort padding, totaled over all participants $\mathcal{P}$. In formal terms, this can be expressed as

$$\sum_{P \in \mathcal{P}} \left\lceil \underset{\text{p+s+v}}{\boxdot P} \right\rceil - \sum_{P \in \mathcal{P}} \left\lceil \underset{\text{p+s}}{\boxdot P} \right\rceil = c_{\mathrm{v}} \cdot \left(\lceil Z_{i,j} \rceil - \sum_{P \in \mathcal{P}} \left\lceil \underset{\text{p+s}}{\boxdot P} \right\rceil \right). \tag{3.238}$$

According to equation 3.94, the contraction operator $\circledast$ in equation 3.237 increases the size of a participant by a factor of

$$\frac{\left\lceil \underset{\text{p+s+v}}{\boxdot P} \right\rceil}{\left\lceil \underset{\text{p+s}}{\boxdot P} \right\rceil} = \left(1 - 2 \cdot c_{\xi}(i,j)\right)^2 \tag{3.239}$$

and it goes without saying, that this can likewise be said about the sum of the participants' sizes:

$$\frac{\sum_{P \in \mathcal{P}} \left\lceil \underset{\text{p+s+v}}{\boxdot P} \right\rceil}{\sum_{P \in \mathcal{P}} \left\lceil \underset{\text{p+s}}{\boxdot P} \right\rceil} = \left(1 - 2 \cdot c_{\xi}(i,j)\right)^2. \tag{3.240}$$

[22] If only volatile comfort padding (without solid comfort padding) is concerned, the subsequent formulas still hold true, with $\underset{\text{p+s+v}}{\boxdot P}$ effectively being $\underset{\text{p+v}}{\boxdot P}$ and $\underset{\text{p+s}}{\boxdot P}$ being $\underset{\text{p}}{\boxdot P}$.

Now, inserting equation 3.240 into equation 3.238 and replacing $\sum_{P \in \mathcal{P}} \lceil P \rceil_{\text{p+s}}$ with the more compact notation $\lceil \mathcal{P} \rceil_{\text{p+s}}$ produces

$$\left(1 - 2 \cdot c_\xi(i,j)\right)^2 \cdot \lceil \mathcal{P} \rceil_{\text{p+s}} - \lceil \mathcal{P} \rceil_{\text{p+s}} = c_\text{v} \cdot \left(\lceil Z_{i,j} \rceil - \lceil \mathcal{P} \rceil_{\text{p+s}} \right) \tag{3.241}$$

$$\left(\left(1 - 2 \cdot c_\xi(i,j)\right)^2 - 1 \right) \cdot \lceil \mathcal{P} \rceil_{\text{p+s}} = c_\text{v} \cdot \left(\lceil Z_{i,j} \rceil - \lceil \mathcal{P} \rceil_{\text{p+s}} \right) \tag{3.242}$$

$$\left(-4 \cdot c_\xi(i,j) + 4 \cdot c_\xi(i,j)^2 \right) \cdot \lceil \mathcal{P} \rceil_{\text{p+s}} = c_\text{v} \cdot \left(\lceil Z_{i,j} \rceil - \lceil \mathcal{P} \rceil_{\text{p+s}} \right) \tag{3.243}$$

$$-4 \cdot c_\xi(i,j) + 4 \cdot c_\xi(i,j)^2 = \frac{c_\text{v} \cdot \left(\lceil Z_{i,j} \rceil - \lceil \mathcal{P} \rceil_{\text{p+s}} \right)}{\lceil \mathcal{P} \rceil_{\text{p+s}}} \tag{3.244}$$

$$4 \cdot c_\xi(i,j)^2 - 4 \cdot c_\xi(i,j) + c_\text{v} \cdot \left(1 - \frac{\lceil Z_{i,j} \rceil}{\lceil \mathcal{P} \rceil_{\text{p+s}}} \right) = 0 \tag{3.245}$$

and this quadratic equation can be solved with the *quadratic formula* to calculate the padding factor as

$$c_\xi(i,j) = \frac{4 - \sqrt{16 - 16 \cdot c_\text{v} \cdot \left(1 - \frac{\lceil Z_{i,j} \rceil}{\lceil \mathcal{P} \rceil_{\text{p+s}}} \right)}}{8} \tag{3.246}$$

which can be simplified to

$$c_\xi(i,j) = \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{4} \cdot c_\text{v} \cdot \left(1 - \frac{\lceil Z_{i,j} \rceil}{\lceil \mathcal{P} \rceil_{\text{p+s}}} \right)} \tag{3.247}$$

$$= \frac{1}{2} - \frac{1}{2} \cdot \sqrt{1 - c_\text{v} \cdot \left(1 - \frac{\lceil Z_{i,j} \rceil}{\lceil \mathcal{P} \rceil_{\text{p+s}}} \right)}. \tag{3.248}$$

As can be seen in equation 3.246, the other solution for $c_\xi(i,j)$ –where the root is added in the dividend, not subtracted– is discarded here, because the padding factor must be $c_\xi(i,j) < 0$ to obtain an enlargement (instead of a contraction). A closer look at equation 3.248 confirms, that –since the area of the layout zone is always greater than the sum of the participants' areas (without volatile comfort padding)– the padding factor is definitely smaller than zero (unless $c_\text{v} = 0$, in which case the padding factor also is zero):

$$c_\xi(i,j) = \underbrace{\frac{1}{2} - \underbrace{\frac{1}{2} \cdot \underbrace{\underbrace{\sqrt{1 - c_\text{v} \cdot \underbrace{\left(1 - \frac{\lceil Z_{i,j} \rceil}{\lceil \mathcal{P} \rceil_{\text{p+s}}} \right)}_{>1}}}_{\leq 0}}_{\geq 1}}_{\leq 0}. \tag{3.249}$$

Since $c_\xi(i,j)$ depends on $[\![Z_{i,j}]\!]$, the sum of the participants' areas with their volatile comfort padding changes by every zone tightening. Without volatile comfort padding, the sum of the participants' areas would remain constant throughout a SWARM run, irrespective of the layout zone which closes in towards its final size as illustrated in Figure 3.72 (a). Remembering the considerations at the end of Section 3.4.2, a characteristic drawback of such a tightening approach is that the fierceness in the competition of the interacting participants usually peaks out in the middle (i.e., in the second episode) of the self-organization phase.

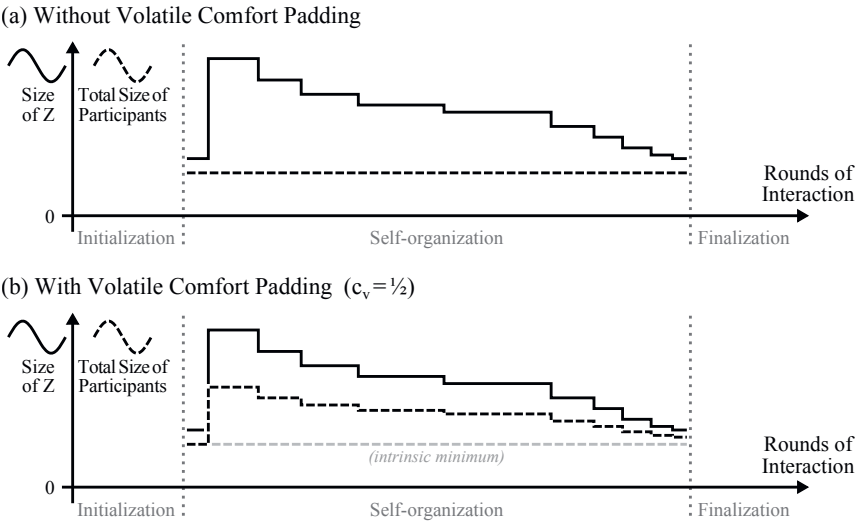


Figure 3.72: Total size of the participants (a) without and (b) with volatile comfort padding.

This is where the idea of volatile comfort padding can help to equilibrate the evocation of emergent behavior. Figure 3.72 (b) shows an exemplary graph for a volatile comfort padding share of $c_\mathrm{v} = \frac{1}{2}$. In mathematical terms, one could say that *without* volatile comfort padding, the derivative of the participants' totaled areas with respect to the progression (regression) of the major and minor tightenings (enlargements) –indexed with i and j– is zero, whereas *with* volatile comfort padding, the derivative is

$$\frac{\mathrm{d}}{\mathrm{d}(i,j)}[\![\mathcal{P}]\!]_{\text{p+s+v}} \;=\; c_\mathrm{v} \cdot \frac{\mathrm{d}}{\mathrm{d}(i,j)}[\![Z_{i,j}]\!]. \tag{3.250}$$

The positive effect of such a correlation is, that –on the one hand– the occurrence of the more competitive interaction situations is preponed and thus tackled sooner, while –on the other hand– the pressure on the interacting participants not only increases with every zone tightening but simultaneously decreases due to the diminution of the volatile comfort padding.

This statement is exemplified in Figure 3.73. For comparison, one may first consider a situation *without* volatile comfort padding as in (a1). After the tightening of the layout zone (a2), the ten participants pass a round of interaction. The primary intention of the participants is to get inside the layout zone, which leads to a constellation teeming with interference (a3). In particular, interference on a corner of a participant is impedimental since it rules out a possible *Budging* action. Thus, the participants in this example have immense difficulty to unravel the abundance of conflicts, even though the current constellation is quite close to a potential solution (a4).

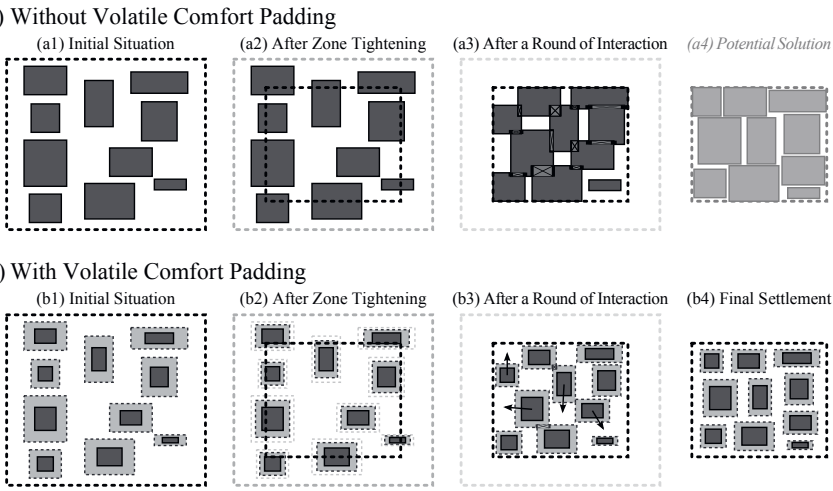


Figure 3.73: Two equivalent interaction situations showing the effect of volatile comfort padding.

Now, Figure 3.73 (b) devises an analogous example *with* volatile comfort padding. For that purpose, the ten participants are assumed to be identical to those of example (a), while the layout zone is initially larger. To make a point, one may consider a situation such as the one in (b1) which is –due to the volatile comfort padding of the participants– equivalent to (a1) from SWARM's point of view, albeit on a larger areal scale.[23] In temporal regard, this means that situation (b1) is encountered sooner than situation (a1) within the SWARM run. The zone tightening, shown in image (b2), is accompanied by a diminishment of the participants' volatile comfort padding, and for that reason the subsequent struggle is not as competitive as in example (a). Thus, after completing one round of interaction, the participants attain the constellation of

[23] Figuratively speaking, image (b1) in Figure 3.73 portrays the initial situation of example (b) from a farther distance than image (a1) portrays the initial situation of example (a).

(b3) which exhibits only two occurrences of interference. With just four *Budging* and *Centering* actions (see arrows), easily determined by the respective participants, the way is paved to reach the final settlement (b4). This constellation is equivalent to the potential solution of (a4), so the rest of the self-organization phase is trivial and demands only a couple of further *Centering* actions.

3.5 Final Remarks About the Conception of SWARM

Now that the layout automation methodology of SWARM has been described in detail, the following Section 3.5.1 throws a glance at a couple of methodological aspects in comparison to similar conceptions found in classical optimization algorithms (as covered in Section 3.1.1 of [1]). Then, Section 3.5.2 is about to elucidate where SWARM shares common ground with existing models of decentralized systems (as discussed in Section 2.5) and where it takes a different route.

3.5.1 Comparison with Optimization Algorithms

So far, this chapter may have left the impression that the SWARM methodology covertly borrows a couple of already-known ideas from the existing crop of optimization-based automation approaches. Although this is not entirely unfounded, a distinct line of demarcation can still be drawn between SWARM and those classical optimization algorithms, as the following explanations –covering several aspects– are about to show.

Nested Loops

In its basic form, as depicted in Figure 3.1 of [1], an optimization algorithm iteratively explores the solution space to determine possible candidate solutions and evaluates these candidates to assess their quality. Depending on the respective algorithm, this iterative strategy can involve multiple loops whereat such a loop may also be nested inside another one. For example, the partitioning algorithms referred to as *Kernighan-Lin* (KL) [160] and *Fiduccia-Mattheyses* (FM) [161] are both built upon a pair of nested loops. This set-up bears a striking resemblance to SWARM, whose self-organization phase recursively iterates over a loop that is nested inside another loop, as shown in Figure 3.3. But a detailed look on the respective mechanisms reveals the distinction between these partitioning algorithms and SWARM.

The KL algorithm works by repeatedly swapping two cells from different partitions while FM consecutively moves cells from one partition to the other. This is fictitiously done in an inner loop until all cells have been swapped/moved. After that, only some of these swaps/moves are effectively performed. This represents a so-called *pass*, which is followed by further passes due to the outer loop. In SWARM, the inner loop encompasses n rounds of interaction wherein each

participant may act n times (whereas in KL or FM each cell is only swapped/moved once –at most– within one pass). Another distinction is that there are no fictitious actions in SWARM – an action is either performed or not. SWARM's outer loop comprises the tightening-settlement cycles. As the name implies, the layout zone gets tightened between these cycles, i.e., the situation is changed prior to the outer loop's next iteration. In contrast, the input solution to a pass in KL and FM is always equal to the output solution of the previous pass.

Condition of a Participant

One elementary trademark of an optimization algorithm can be found in the formal metric used to rate the quality of a candidate solution [162]. As already mentioned, this is typically done via a cost function which incorporates several optimization goals and adds up to a numerical value enabling a relational comparison (as covered in [163], for example). This is reminiscent of a participant's *condition* in SWARM (see Section 3.3.1), a construct used to encapsulate several *influencing factors*. Two of these, *interference* and *turmoil*, are –as with a cost function– even used for a numerical comparison (Section 3.3.4.1), although other ones, such as *wounds*, affect the module interaction in a more elaborate way.

A decisive difference between SWARM and optimization algorithms in this regard is that in SWARM, following the idea of decentralization, a participant's condition (or, more precisely, the benefit of its prospective condition in relation to its present condition) merely pertains to that specific participant and affects nothing but that participant's next action. So, at no point during a SWARM run is the fitness of the overall layout solution quantified as a whole, since each participant only needs to care for its own personal well-being (namely described by its condition).

Handling Overlaps

The matter behind SWARM's influencing factor *interference* (Section 3.3.1.1) is also a common element of optimization-based placement algorithms: an overlap of design components. Yet again, this apparent similarity comes into play in a different way here. In a placement algorithm, the occurrence of overlaps depends on the representation used to describe the positions of the components. In a *topological representation*, where the component positions are described relative to each other (as in the *slicing model* introduced by [164]), the components cannot overlap. In contrast, an *absolute representation* (introduced in [165]) permits illegal overlaps during the optimization, taken into consideration by a (typically quadratic) penalty cost term that is to be minimized. In that case, the overlaps may even subsist in the final solution and need to be eliminated through a dedicated post-processing step [166].

SWARM differs from both of these practices as overlaps are –in contrast to placement algorithms based on topological representations– indeed allowed during the module interaction, but are –in contrast to placement algorithms based on absolute representations– definitely re-

duced until obliteration during each tightening-settlement cycle. Furthermore, the overlaps are not incorporated in a simple linear or quadratic fashion, but involve further expedients (such as *aversion*) shown to have a positive impact on the overall flow of self-organization. And above all, the overlaps in SWARM are not just an undesired nuisance, but also an essential driving force behind its module interaction.

Weight Versus Emphasis

The influencing factor *turmoil* (Section 3.3.1.2) includes a quantity named *emphasis* to specify the importance of a connection between two participants in a SWARM run. This may bring a similar notion to mind, generally known from optimization algorithms under the term *weight* or *weighting factor* [167]. Most prominently, each individual summand of an optimization algorithm's cost function is usually multiplied with a certain weighting factor. This way of controlling how the evaluated solution quality depends on a certain criterion allows to prioritize the various optimization goals that the algorithm is supposed to pursue. Since a cost function usually embraces the solution as a whole, the targets of these optimization goals can be quite diverse in nature. For example, the three penalty addends of the cost function in *TimberWolf* [168] –(1) total wirelength, (2) cell overlaps, and (3) inequality of row lenghts– represent different quantities with different dimensions.

In SWARM however, the emphasis is –unlike a weight– not applied to diverse optimization goals in general, but only with respect to a participant's connections in order to determine the participant's overall turmoil. Furthermore, a look at equation 3.34 shows that the emphasis does not appear therein as a numerator, but as a denominator (to calculate the so-called *relaxation threshold*) and can therefore not be considered a weighting factor in that sense. In equation 3.48, the emphasis does indeed appear as a numerator to compute the *tension* in a connection, but not to play different kinds of opposing optimization goals off against each other in order to make a compromise on the global scale. Instead, suchlike reciprocities are globally resolved by the locally incentivized actions of SWARM's participants.

Tractive Forces

As discussed in Section 3.3.1.2, the concept of *tension* influences a participant's actions like the tractive force of a rubber strap. The same suggestion also lies behind *Force-Directed Placement* [169], an optimization-based approach where –viewed from the physics of classical mechanics– the placement problem is modeled as a spring-mass system. For each pair of connected components, this approach lets the two components attract each other as if joined by an extension spring, which means that the tractive force is linearly proportional to the spring's deflection (given by the distance between the components).[24] Thus, the placement problem is solved by

[24] In fact, the analogy with springs is not entirely correct because a spring's ends already have a certain distance in the spring's resting state.

finding the *mechanical equilibrium*, i.e., by calculating a location for each component such that the *net force* on it becomes zero (or at least minimal).

Despite the striking similarity, SWARM's employment of a tractive force is quite different. Above all, the concept of tension in SWARM only belongs to one of several influencing factors. Apart from this, the magnitude of the force does not obey *Hooke's law* [170], but follows a combined linear and quadratic characteristic (Figure 3.20), also featuring a quantity called *strength* to consider a participant's total number of connections (thus streamlining the self-organization). Furthermore, the net force in SWARM is not calculated vectorially but represents a scalar value and can therefore never be zero (unless all participants lie centered on top of each other). Instead, SWARM defines the *relaxation threshold* which is correlated with the size of the layout zone –to account for its repetitive tightening– and facilitates overlap-free arrangements without the need to withdraw a component from an already occupied location (as would be necessary in Force-Directed Placement).

Free Peripheral Space and Prime Rectangles

Presumably, SWARM's conception of the so-called *free peripheral space* –as covered in Section 3.3.2– looks kind of crude for the reason that it may not be really "free" due to *pervasion* (for example because of its *blind spots*). Stating that this observation is not that much of a problem (see Section 3.3.2.2) may also have a stale aftertaste to it, sounding as if this deficiency has to be conceded due to the lack of a better conception. In contrast, other works such as [171] resort to the seemingly neater notion of *prime rectangles* since such a prime rectangle has the valuable predicate that is is always surely vacant – by definition [172].

Yet, the introduction of free peripheral space in SWARM is justified for two major motives. First, the geometrical recipe for perceiving the free peripheral space (Section 3.3.2.1) is relatively simple and can even be formulated as a closed functional expression. Hence, it does not necessitate any programming constructs (such as iteration loops and conditional statements) and gets by with quite simple graphical operations (e.g., determining the intersection of two rectangles). Second, there is –by definition– always exactly one free peripheral space per participant at any given point during a SWARM run (whereas there could be a multitude of prime rectangles). In that regard, a participant's doing is well-determined and straightforward. Altogether, both motives help to keep a participant's efforts low on the local scale, which is quite in line with SWARM's philosophy of decentralization and its aim of provoking emergent behavior on the global level. With this mindset, thinking of SWARM as a *complex system*, interference due to blind spots might even be crucial to achieve self-organization because –as already stated before– component overlaps represent a fundamental impetus for the overall flow of module interaction.

Actions Versus Perturbations

Some of SWARM's native actions –described in Section 3.3.3.1– might be evocative of similar implementations found in classical optimization algorithms. In particular, a counterpart of the *Swapping* action (whereby a participant trades places with another participant in SWARM) is commonly used by placement approaches based on *Simulated Annealing* to generate a new placement state with every iteration. For example, the TimberWolf tool referenced above can (apart from attempting a single-cell displacement) employ such a modification –in the context of Simulated Annealing typically called a *perturbation* [173]– to perform a pairwise interchange of cells [174]. That interchange may also include a change in orientation (as done by SWARM's *rotation* morphism). Thus, the idea of a cell interchange is nearly identical to SWARM's *Swapping* action – however, it differs in the way that it is utilized.

Simulated Annealing works stochastically, i.e., the decision whether to perform a single-cell displacement or a pairwise interchange (or, in general, any other perturbation) is random [175]. Likewise, the cell to displace or –accordingly– the pair of cells that are to be interchanged is selected randomly. In contrast, the action exploration (including the choice of involved participants) in SWARM proceeds pursuant to a completely deterministic agenda. Just as well, the action to perform in SWARM is picked according to a well-defined and unambiguous comparison metric (see Section 3.3.4.1). Concerning the single-cell displacement, it must be further recognized that SWARM does not feature only one suchlike perturbation but a couple of actions where the number of acting participants amounts to 1 (e.g., *Centering*, *Budging*, and *Yielding*). Choosing an action, every participant in SWARM follows clear preferences in its decision-making because every participant is meant to embody a rational agent (see Section 2.5.3). In Simulated Annealing however, the acceptance or rejection of a (worse) solution is again a question of probability and therefore not "rational" nor deterministic at all.

Greediness

The willingness of temporarily accepting an interim solution that is worse than the current candidate, represents a particular feature of Simulated Annealing and a couple of other optimization algorithms (such as the partitioning algorithms KL and FM mentioned above). In contrast, many layout automation algorithms are *greedy*: they only permit an improvement of the current layout solution by always taking a better neighboring candidate with every iteration (and coming to a stop if no immediate improvement can be found) [176]. Due to this attitude, such algorithms are not able to overcome local optima and therefore they often fail to find the globally optimal solution [177]. Since the participants in a SWARM run always strive for *beneficial* actions (to improve their condition), SWARM also seems to be greedy in this algorithmic sense. However, this can not be definitely said for several reasons.

First, it should be understood that the idea of *aversion* (see Section 3.3.1.1) and the influencing factor *wounds* (Section 3.3.1.4) have a kind of memory effect on their participants. Because

of this, an action that might have been rejected some rounds of interaction before may now become beneficial. So, although that action is now –from the participant's subjective point of view– better than before, it is –objectively perceived– still the same as it was back then. Thus, whether to call the participant greedy or not, becomes a rather moot question. Second, while a participant always chooses the best action explored, it can be the case that no beneficial action has been found at all. Then, as mentioned in Section 3.3.4.1, the participant deliberately executes a *Yielding* even if that is not beneficial. So, the participant acts non-greedily regarding its worsened prospective condition, but –on the other hand– yields on purpose (to revive the flow of self-organization). Third, it must be emphasized that all these considerations always apply to the decisions of an individual participant, investigating its local options within its current situation. For that reason alone, the term *greedy* (or non-greedy), which generally refers to the solution as a whole, is inappropriate to describe a decentralized methodology such as SWARM.

Placement Templates

Section 3.3.4.1 states that the locations of the participants in a SWARM run, relative to each other, can be predefined in advance via a placement template, thus enforcing a desired overall arrangement. This idea has already been exploited by a couple of optimization-based layout automation works in the past. For example, the "design by example" approach of [178] requires a sample layout –the template– to automatically produce other layout variants by availing the expert knowledge embedded in that template (such as the device placement, for example). The authors of [179] praise the good quality of the layouts obtained with this technique, but also criticize that a new sample layout must be created manually for each type of circuit. This is different in SWARM, where the relative placement of the participants is specified via constraints (as will be discussed in Section 4.3.2). Apart from this, there are –despite the common idea of using a template– a couple of further distinctions to be revealed on closer examination.

In [178], the placement topology of the sample layout is turned into a *slicing floorplan* description. To also include those device arrangements that would be ruled out by confining the multitude of potential solutions to a single slicing structure, the tool in fact considers an entire set of differing slicing structures. Since there exists an exponential number of slicing structure alternatives for a given placement topology, this in turn led to the introduction of so-called *option slices* (representing alternative binary slicing compositions) to cope with the significant increase in memory size and processing time. In contrast, the constraint-based template description of SWARM not only bypasses the need for suchlike tricks but also covers an even more exhaustive set of possible constellations because it allows for slicing and nonslicing arrangements alike. The tool of [180] converts an existing layout into a symbolic template described via *corner-stitching* [181], providing a separate representation for each mask layer to describe that plane in terms of rectangular *tiles*. While this representation also involves constraints, these are not used to specify the relative positions of design components as in SWARM, but to en-

force technological design rules (such as a minimum spacing between two tiles on the same mask layer).

In general, SWARM sets itself apart from other template-based approaches due to the fact that the placement template in SWARM does not apply to primitive devices (nor basic shapes) but compound modules. The relative locations of these modules are explicitly enforced through constraints, while the positioning of the individual devices –and thereby caring for detailed layout issues such as the spacing between these devices– is implicitly managed by the respective module along with the module-internal wiring. This is the power of delegating such low-level layout tasks down to the governing modules in a decentralized fashion. Speaking of decentralization, the following Section 3.5.2 is about to compare SWARM with other decentralized systems.

3.5.2 Comparison with Decentralized Systems

Comparing SWARM with cellular automata (Section 2.5.1), game theory (Section 2.5.2), multi-agent systems (Section 2.5.3), and agent-based models of collective motion (Section 2.5.4), the major similarities and the major differences can be summarized as in Table 3.13.

Table 3.13: Comparison of SWARM with existing models of decentralized systems.

	Cellular Automata	**SWARM Methodology**
Similarities:	• Realizes an interaction of cells on a usually planar two-dimensional grid. • Complex geometrical patterns dynamically emerge from the interaction of the cells. • A cell's state can change (e.g., from "dead" to "alive"), which influences its subsequent transitions.	• Realizes an interaction of modules in the two-dimensional layout plane. • The overall layout solution is expected to emerge from the interaction of the modules. • A module's condition can change (e.g., from "wounded" to "healed"), which influences its further actions.

to be continued on next page

Table 3.13: Comparison of SWARM with existing models of decentralized systems (continued).

	• A cell's transition to the next state involves the cell's local neighborhood (its adjacent cells).	• A module's decision-making involves the module's local surrounding (its free peripheral space).
Differences:	○ All cells are arrayed in a strictly tiled, space-discrete manner.	○ The layout plane is continuous (down to the marginally small design grid).
	○ The location, form, and size of a cell is fix and does not change.	○ The modules can move through the layout plane and deform themselves.
	○ The evolvement of patterns can be highly dependent on the initial population (e.g., in Conway's Game of Life).	○ The emergence of the final layout solution is desired to be largely independent of the initial constellation.
	Game Theory	**SWARM Methodology**
Similarities:	• Used to study the strategic interaction among independent players.	• Can be regarded as an infinitely repeated game played by the modules.
	• Distinguishes between different criteria by which various types of games can be classified.	• Each round of interaction represents a noncooperative, discrete, asymmetric, non-constant-sum, sequential, perfect-information stage game.
	• Is particularly targeted at situations where the desires of the different players conflict.	• The interaction of the participating modules is built on competition, not on cooperation.

to be continued on next page

Table 3.13: Comparison of SWARM with existing models of decentralized systems (continued).

	• Each player is self-interested and wants to maximize his or her own payoff.	• Each module is self-interested and wants to improve its own situation as much as possible.
Differences:	○ A player's desires are expressed via a utility function, mapping the player's preferences to a real number.	○ A module's desires are modeled through its condition, which in turn depends on five influencing factors.
	○ A player's decision-making may consider the potential actions of the other players (referred to as a *complete-information game*).	○ A module's decision-making ignores the potential subsequent actions of the other modules (denoted as an *incomplete-information game*).
	○ Noncooperation often leads to an outcome that is inferior for all players (e.g., see the Nash equilibrium in the Prisoner's Dilemma).	○ The noncooperative, competitive setting is supposed to result in an outcome that is optimal for all modules.
	Multi-Agent Systems	**SWARM Methodology**
Similarities:	• An agent is able to perform actions upon its environment.	• A module can see its layout context and react to environmental changes (such as a zone tightening).
	• An agent is autonomous in its decisions about how to act.	• Each module chooses its own actions by itself (possibly restricted by design constraints).
	• An agent has a particular degree of intelligence (and often mobility).	• The modules are implemented as computational entities and can move through the layout plane.

to be continued on next page

Table 3.13: Comparison of SWARM with existing models of decentralized systems (continued).

	• An agent has a limited viewpoint with only incomplete information about its environment.	• A module's free peripheral space is locally confined by the module's nearest obstacles.
	• An agent is rational in that it adheres to a goal-directed behavior following clear preferences.	• Every module acts in a clear and deterministic way, pursuing the goal of becoming (or staying) contented.
Differences:	○ The agents typically co-exist beside the problem that they solve (e.g., an object that they control).	○ The modules do not only help in solving the problem but also represent a part of its solution.
	○ Often involve control architectures for coordinating the agents and reaching consensus.	○ The interacting modules do not exchange information and do not rely on mutual consent.
	○ Usually each agent is implemented as a single and enclosed entity.	○ Multiple modules can be organized as a hierarchical module association.
	Agent-Based Models	**SWARM Methodology**
Similarities:	• Used to animate a spatially explicit, collective motion of an ensemble through a geographical sphere.	• Used to evoke a spatially explicit, collective motion of layout modules through the layout plane.
	• Inspired by the movement of large animal groups such as a flock of birds or a school of fish.	• Inspired by the movement of livestock during roundup (for example, a herd of sheep).

to be continued on next page

Table 3.13: Comparison of SWARM with existing models of decentralized systems (continued).

	• The ensemble members are mainly driven by biological emotions and sensations (e.g., fear and hunger).	• The modules are influenced by natural feelings and impacts (including tension, aversion and wounds).
	• A member's behavior can change over time due to the inclusion of memory into its behavioral model.	• Aversions and wounds abate only slowly and thus have a memory effect on a module's decision-making.
	• Interested in simulating how the ensemble manages to avoid collision with other moving environmental obstacles.	• The modules constantly need to adapt themselves to their environment which changes with every zone tightening.
Differences:	○ All ensemble members are created equal.	○ The modules may have individual looks, tasks, goals and abilities.
	○ The ensemble members are typically modeled as dot-like incorporeal entities.	○ Each module has a "body" with a nonzero width and height (and thus an area).
	○ The overall motion of the ensemble is a polarized coherent movement with smooth shifts in direction.	○ The overall motion of the modules is a centripetal aggregation with turbulences such as jumps and swaps.

As Table 3.13 shows, SWARM embeds a lot of ideas and concepts found in existing models of decentralized systems. Apart from this, every principle of self-organization covered in Section 2.4 can also be encountered in the SWARM methodology (see Figure 3.74):

The Basic Constituents of Self-organization: Above all, the three core concepts of SWARM embody the basic constituents of self-organization introduced in Section 2.4.1: the *responsive modules* correspond to the *workers* (and indeed perform the actual layout work), the *module interaction* follows well-defined behavioral *rules* of action (in accordance with each module's individual desires and abilities), and the *interaction control* organ

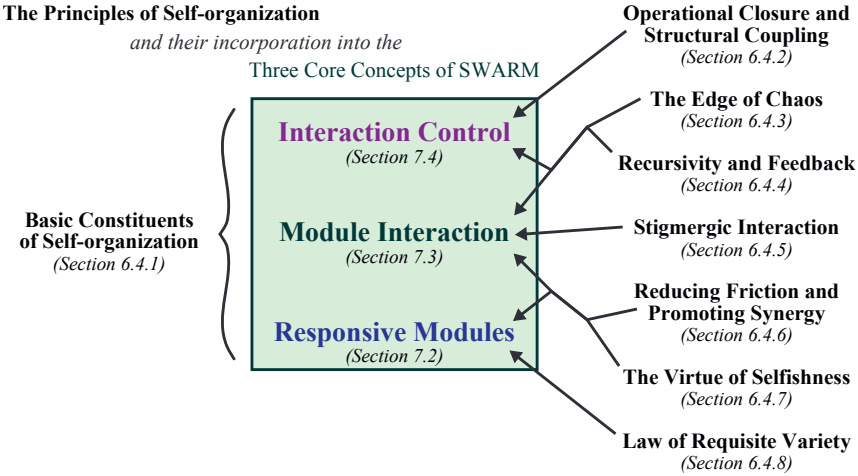


Figure 3.74: Incorporation of the principles of self-organization into the three core concepts of SWARM (the section numbers of the references in the image are those from the original dissertation [1]).

steers the overall flow towards the desired outcome by exerting *pressure* (and imposing the *goals* as constraints).

Operational Closure and Structural Coupling: The notions of *operational closure* and *structural coupling* (Section 2.4.2) can be identified in the limited authority of the interaction control organ which is not able to manipulate the module interaction directly but can only spur the modules by changing their environment (i.e., decreasing the size of the layout zone). Thereby, the interaction control organ leaves it up to the *cognitively open* network of interacting modules to adapt itself to the new situation by changing its internal structure (i.e., the layout arrangement) on its own behalf, using innate operations (i.e., the modules' catalog of actions).

The Edge of Chaos: Operating at the *edge of chaos* (Section 2.4.3) is a principle of self-organization which affects both the interaction control organ and the module interaction. Concerning the interaction control organ, the edge of chaos corresponds to a moderate tightening policy that is neither too lenient (which would risk to let the modules drift towards stagnation) nor too aggressive (which could lead the interaction into an erratic mêlée). Regarding the module interaction, the edge of chaos is found in the fine-tuned taring of aversions and wounds, whereby the self-organization proceeds on the thin red line between overly static (i.e., firm but unfertile) behavior and overly dynamic (i.e., fruitful but frantic) behavior.

Recursivity and Feedback: *Recursivity* (Section 2.4.4) can be ascribed to the way in which the interaction control organ drives the self-organization through successive tightenings of the layout zone. Therein, the output of each tightening (i.e., the settlement) serves as the input for the next tightening. One might claim that this is no different from any iterative algorithm, but whenever a layout constellation in SWARM is again encountered a second time, the subsequent actions will probably deviate from those of the first time due to the memory effect of aversions and wounds (internal quantities that make the system behavior history-dependent). The module interaction's *feedback* to the control organ is the difficulty (i.e., the number of necessary interaction rounds) to obtain a settlement. Feedback can also be observed in the interaction itself, both positive and negative. A module moving into a position of interference may be repulsed and driven back again (thus withdrawing the move = negative feedback) or may cause the other modules to relent and back off (thus endorsing the move = positive feedback).

Stigmergic Interaction: In contrast to request-and-response schemes commonly employed in multi-agent systems, the module interaction in SWARM is not based on direct communication but on *stigmergy* (Section 2.4.5). Just like the movement of an ant in an ant colony inevitably leads to a deposition of pheromones which in turn attracts other ants (see Figure 2.4), a module's actions inevitably modify the environment of the others (e.g., the free peripheral space of the module's neighbors). Furthermore, the modules also coordinate themselves via the influencing factors that they want to alleviate (such as overlaps and tension), equivalent to the stigmergic way in which the traffic lights of Figure 2.10 co-control each other via the traffic that they regulate.

Reducing Friction and Promoting Synergy: The idea to reduce *friction* and promote *synergy* (Section 2.4.6) has been worked into SWARM's comparison metric for evaluating explored actions (and thus sits on the fence between a responsive module's decision-making and its inherent effect on the overall module interaction). Actions that involve multiple participants are rated by determining the collective interference, turmoil, or relaxation delta of all involved participants (not just the leading participant). Hence, synergistic actions with mutual benefit for multiple participants are automatically favored over frictional actions by which the leading participant only profits from the sacrifices of others.

The Virtue of Selfishness: The virtue of *selfishness* (Section 2.4.7) is another self-organization principle that can be detected in the decision-making of the individual responsive modules and thereby has an impact on the overall module interaction. The selfishness resides in the modules' egoistic pursuit of their personal contentment: an action is only performed by the respective module if it improves that module's situation. This is definitely true for actions that consists of a single transformation and for *Hustling* actions. Only in rare cases where the module performes a *Swapping* or a *Pairing* action it may happen that the module concedes certain losses for the greater benefit of its counterpart.

Law of Requisite Variety: Last but not least, Ashby's *law of requisite variety* (Section 2.4.8) is reflected in the layout variability of SWARM's responsive modules. The greater the variety of a system is, the better are its chances to cope with changes in its environmental conditions. In analog layout design, these conditions are represented by the constraints that define each specific design problem. So, following the law of requisite variety, encouraging a module to cover as many layout variants as possible, and exploiting that variability during a module's action exploration enlarges the system's room for maneuver and increases the probability of effectuating a layout arrangement wherein all constraints are really satisfied.

As can be seen, paying respect to the principles of self-organization is an ambition that has been embraced throughout all three core concepts of the developed SWARM methodology. From the perspective of complex adaptive systems, these principles are meant to equip SWARM with the ability to "survive" situations marked by unforeseen disturbances, i.e., to produce layout solutions in full-custom quality for various design problems with settings of design constraints that have not been completely anticipated in advance.

Beyond this aim, the conception of a layout methodology such as SWARM –implementing a system of self-organizing layout modules– can be considered *perfect* if the success of the self-organization is largely independent of the initial module constellation. In terms of chaos theory, this means that the system is engineered in a way such that the desired solutions represent *attractors*, i.e., stable states towards which the system tends to evolve for a wide range of starting conditions. The next part of this thesis is about to show inhowfar the current implementation of SWARM can already be seen to hit that mark.

Chapter 4

Implementation and Results

Any sufficiently advanced technology
is indistinguishable from magic.
Arthur C. Clarke (British writer)

The SWARM methodology, as described in Chapter 3, has been implemented in the programming language SKILL [182] to be used in the Cadence *Virtuoso* IC design environment. Although SWARM is still in its infancy, its current implementation is already advanced enough to achieve various remarkable results – as this chapter is about to show. First, Section 4.1 provides a couple of examples to illustrate the evocation of *emergent behavior* in SWARM. Next, Section 4.2 demonstrates how SWARM can be successfully employed for the practical purpose of *floorplanning*. Finally, Section 4.3 targets the main application area of SWARM, displaying how it can be utilized to tackle analog *place-and-route* problems. As will be seen, SWARM manages to coalesce the chief advantages of both *top-down automation* and *bottom-up automation*: the versatility of an automatism and the quality of its resulting layouts.

4.1 Examples of Emergence in SWARM

As has been explained in Chapter 2, the intention of the SWARM methodology is to let an optimal layout outcome *emerge* from the interaction of responsive layout modules, similar to the way captivating patterns can evolve in a *cellular automaton*. To substantiate this idea, the following examples illustrate inhowfar the phenomenon of emergence can really be observed in SWARM, distinguishing between the emergence of a collective motion (Section 4.1.1), the emergence of an optimal layout outcome (Section 4.1.2), and also the –unfortunate– emergence of interaction cycles that never terminate (Section 4.1.3).

4.1.1 Example of an Emerging Collective Motion

Figure 4.1 presents a simple SWARM example –also given in [183]– that shows the step-by-step interaction of two participants P_1 and P_2 inside a rectangular layout zone. For the sake of exemplification, these participants are only symbolic modules, which is to say that they are not imposed on real layout devices. Additionally, neither rotations nor deformations are taken into consideration during the action exploration in this example, i.e., the participants can only perform translational moves.

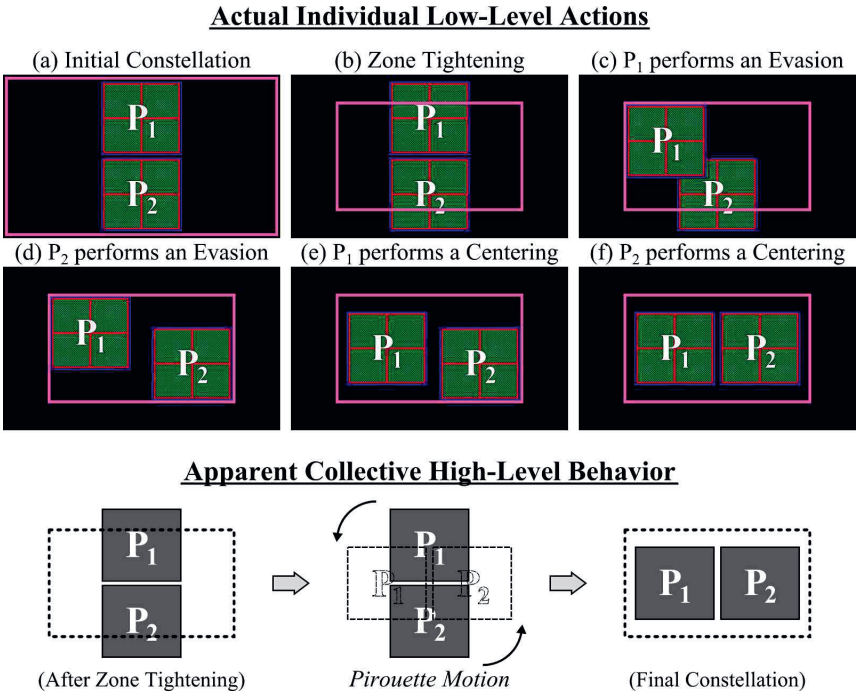


Figure 4.1: Example of how high-level behavior can emerge from SWARM's low-level interactions.

The individual actions of the two participants can be traced throughout images (a) to (f) in the upper portion of Figure 4.1. First, image (a) displays the initial constellation with the two participants lying completely inside the layout zone. Next, the layout zone is tightened as in (b) whereby both participants become *prone* and strive to take an action. Participant P_1, with the desire to become *safe* again, executes an *Evasion* move by going downwards and to the left, thus aligning itself with the north-western corner of the layout zone (c). In that location, there

is only little overlap with P_2 compared to the *interference* that would be sustained if P_1 moved straightforwardly down. For the same reason, P_2 then also performs and *Evasion*, wandering upwards and right into the layout zone's south-eastern corner (d). Since both participants have become *safe* and *clear* (and thus *contented* in this case), first P_1 begins *Centering* itself within its *free peripheral space* (e) just like P_2 also performs a *Centering* until both of them reach a stable settlement in their final locations (f).

Comparing the initial constellation (a) with the final constellation (f) makes it clear that P_1 and P_2 have been forced to re-organize themselves by turning from an above-below arrangement into a side-by-side arrangement. As the six discussed images in Figure 4.1 put on record, the two participants actually achieved this through a successive series of individual actions (utilizing two native types of actions: *Evasion* and *Centering*). But, looking at this low-level interaction from a higher-level perspective (as done in the lower part of Figure 4.1), the participants' movements between the incipient situation (after the zone tightening) and the resulting arrangement (the final constellation) give the impression as if the modules had virtually performed a *pirouette motion* around their common center point, keeping their orientations like the gondolas of a Ferris wheel.

This overall motion represents a fine example of how an apparently solidary, collective behavior can be seen to emerge from the basic actions of the self-interested participants. Just like the cells in Conway's Game of Life (Section 2.5.1) do not have any knowledge of the patterns that evolve from their aggregate state transitions, the emergence of the seemingly harmonic pirouette motion in Figure 4.1 is beyond the grasp of the involved participants and can only be perceived by an outside observer – as is typical for emergent phenomena.

4.1.2 Examples of an Emerging Optimal Layout Outcome

A complete placement exercise is depicted in Figure 4.2, with ten selected snapshots giving an account of the entire SWARM run that is witnessed to solve the problem. As in the example of Figure 4.1, the given layout zone is rectangular and again the seven participants are implemented as symbolic modules that may solely perform translational movements. Each participant has *volatile comfort padding* –but no *solid comfort padding*– (see Section 3.4.3), as visualized by the white rectangles that are drawn around the modules.

The initial constellation is not *viable* because no participant is clear and two of them (P_1 and P_3) are even prone. With several rounds of interaction, the participants manage to struggle themselves free from each other and occupy locations where all of them are contented – this viable constellation represents the 1st settlement (b). After the subsequent tightening of the layout zone, the overall arrangement changes only slightly, with participant P_4 moving from its location below P_1 to the right of P_1 (c). The relative placement in this 2nd settlement is maintained until the 5th settlement inclusively (d). Following another zone tightening, the sub-

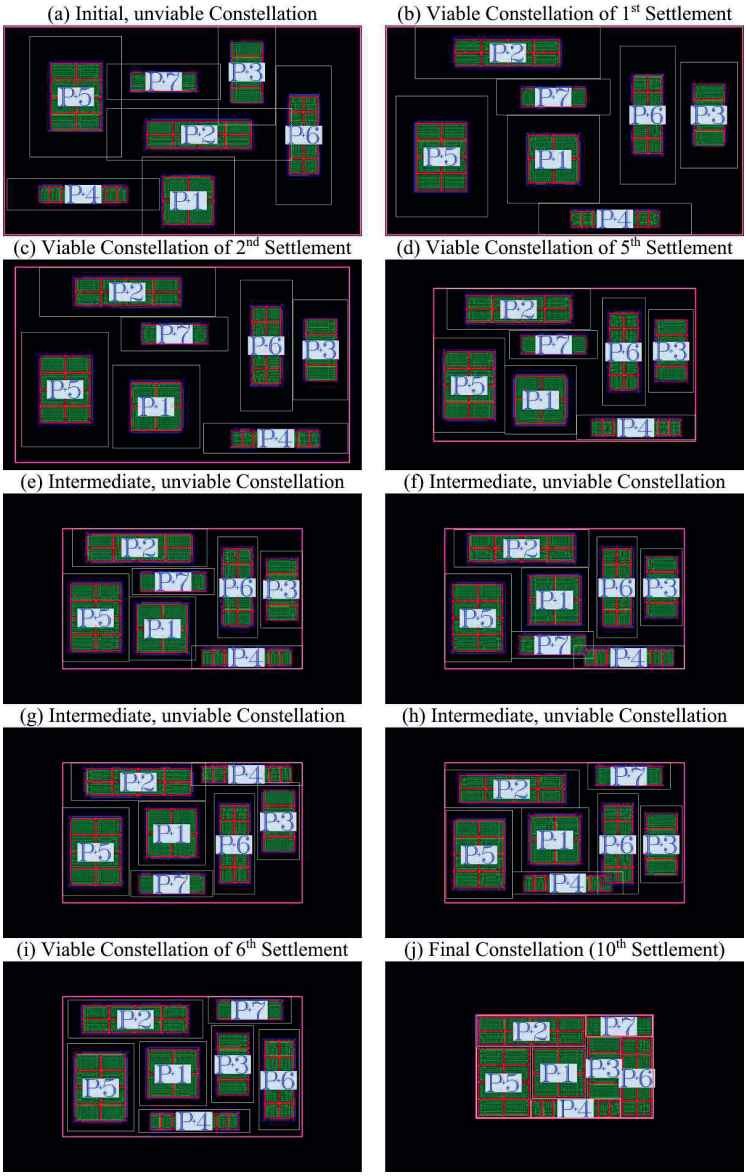


Figure 4.2: Example of a placement problem with the emergence of the globally optimal outcome.

sequent round of interaction leads to the intermediate, unviable constellation of (e), where P_4 now overlaps P_1. This arrangement persists until the interference causes the jammed participant P_1 to perform a ***Swapping*** with P_7 (f). After several more rounds of interaction, P_4 has built up much aversion against P_7 (in addition to the aversion against P_1), making it jump next to –and partially on top of– P_2 (g). A westwards ***Budging*** of P_2 can diminish but not eliminate the interference, which prompts P_4 to trade places with P_7 after some time (h). This results in an overlap between P_4 and P_6, which induces a ***Swapping*** of P_6 with P_3, at last leading to a viable constellation that marks the 6^{th} settlement (i). Thus, watching the interaction throughout (e) to (i), it seems as if the problematic interference gets passed around different pairs of participants until it is eventually resolved. The remaining interaction comprises only ***Centering*** moves, so the overall arrangement does not change apart from becoming more and more compact until the final constellation is reached with the 10^{th} settlement (j).

As should be visible in the depiction, this outcome of the SWARM run represents the *optimal* solution to the placement problem (from which the example was originally constructed). It should also be noted, that this arrangement is completely *nonslicing*: it is not even possible to cut out any rectangular part of the constellation unless at least one participant is torn in half. Therefore, the placement problem in this example truly has just *one* globally optimal solution without any alternative permutations (except for a horizontal and/or vertical flipping of the entire layout, to be sure).

In essence, this example demonstrates the basic ability of SWARM's interacting participants to obtain the *best* possible arrangement by themselves, even if there is only one single optimum available. Since every participant is selfishly focused on its own personal well-being, none of them is aware of the remarkable feat being accomplished at large. Again, only an outside observer recognizes that the participants' collective contentment in the end reflects their global achievement of solving the overall design problem. Hence, the optimal outcome truly *emerges* from the aggregate interaction – quasi as a "byproduct" of the participants' egoistic pursuit of happiness.

To analyze the presented SWARM run more deeply, Table 4.1 provides a so-called *interaction record*, i.e., a matrix listing the number of settlements, the number of interaction rounds per settlement, the number of actions per round of interaction, and the total number of actions per settlement. Peering at the facts and figures in this interaction record, four particular observations can be made:

- It may be that during the first round of interaction in a tightening-settlement cycle less participants take an action than in the second round. This is because participants that do not lie near an edge of the layout zone do not become prone from the tightening and therefore do not need to act (until coming in conflict with other participants when these move in to become safe again). From then on, the number of acting participants in general abates until no participant at all performs an action.

Table 4.1: Number of settlements, rounds, and actions in the SWARM run of Figure 4.2.

Settle-ment	Num. of Rounds	Actions per Round of Interaction													Num. of Actions
		1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th	
1st	5	7	5	1	1	0	-	-	-	-	-	-	-	-	14
2nd	4	4	7	1	0	-	-	-	-	-	-	-	-	-	12
3rd	3	6	4	0	-	-	-	-	-	-	-	-	-	-	10
4th	3	5	3	0	-	-	-	-	-	-	-	-	-	-	8
5th	4	6	7	1	0	-	-	-	-	-	-	-	-	-	14
6th	13	6	7	7	7	7	7	7	7	7	7	7	4	0	80
7th	3	5	1	0	-	-	-	-	-	-	-	-	-	-	6
8th	3	5	1	0	-	-	-	-	-	-	-	-	-	-	6
9th	4	5	7	1	0	-	-	-	-	-	-	-	-	-	13
10th	4	5	7	1	0	-	-	-	-	-	-	-	-	-	13

- The interaction record exhibits the three characteristic *episodes* that can often be discerned in the *self-organization phase* of a SWARM run (as mentioned in the context of Figure 3.70 on page 146). Here, one can roughly say that the first episode encompasses the first four settlements (with a maximum of 14 actions per settlement), while the second episode begins with the fifth settlement (where the declining number of actions starts to rise again) and peaks out in the sixth settlement (with a total of 80 actions), so the third episode spans the remaining four settlements (seeing the number of actions drop again).
- Although the three characteristic episodes of a SWARM run's self-organization phase can be identified in this example, the flow of self-organization is relatively fluent. This is reflected in the number of interaction rounds per tightening-settlement cycle since the maximum value (13 rounds for the sixth settlement) is less than thrice as large as the arithmetic average (which amounts to 4.6 rounds per settlement). This is a remarkably good achievement among the test cases put into effect so far, as a comparison with the results of the subsequent example is about to show (see Table 4.3).
- The interaction adds up to a total of only 176 performed actions. Of course, the number of explored actions is much greater (with a total of 3368 actions) but still smaller –by several orders of magnitude– than the number of iterations in a conventional optimization algorithm (which usually amounts to several millions). This reinforces the idea of decentralization and local decision-making in SWARM, where the fund of available actions is considerably richer and each individual choice of action is far more intelligent than the savage perturbations and probabilistic search techniques found in existing top-down approaches. The computational effort per action may be larger in SWARM, but nonetheless only 28 seconds of runtime were required to obtain the optimal solution in this case.

To underline SWARM's emergent problem-solving abilities, another placement task is given in Figure 4.3, similar to the previous example but with ten instead of seven layout modules this time. As before, the initial constellation is again unviable (a), yet the participants manage to eliminate all conflicts once again and attain a viable constellation with the 1st settlement (b). The 2nd settlement involves only a minor revision of the overall arrangement (c), which –from then on– remains basically the same until reaching the 12th settlement (d). With the subsequent tightening, the situation gets difficult and asks for relatively many rounds of interaction to settle. Traversing numerous intermediate, unviable constellations –a couple of which being displayed throughout images (e) to (h)– the participants succeed in resolving all overlaps, thus concluding the 13th settlement with the viable constellation depicted in (i). The last tightening incites a further compaction of the arrangement and produces the final constellation, which ends the SWARM run with the 14th settlement (j).

In this example, the participants' interaction has again lead to the optimal outcome, which is once more a nonslicing arrangement as in the previous example. However, there are multiple global optima in this case since some modules can be permuted without increasing the overall area occupation (such as the two participants in the north-eastern corner of the layout zone). A look at the interaction record for this SWARM run –given in Table 4.2– substantiates that the self-organization did not proceed as smoothly as in the previous example: the highest number of rounds amounts to 29 (for the 13th settlement) and is more than five times as large as the arithmetic average (5.5 rounds per settlement).

Table 4.2: Number of settlements, rounds, and actions in the SWARM run of Figure 4.3.

Settle-	Num. of	Actions per Round of Interaction													Num. of
ment	Rounds	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	...	27th	28th	29th	Actions
1st	3	10	10	0	-	-	-	-	-	-	...	-	-	-	20
2nd	6	5	10	10	3	1	0	-	-	-	...	-	-	-	29
3rd	2	3	0	-	-	-	-	-	-	-	...	-	-	-	3
4th	3	6	2	0	-	-	-	-	-	-	...	-	-	-	8
5th	3	6	3	0	-	-	-	-	-	-	...	-	-	-	9
6th	5	7	10	4	1	0	-	-	-	-	...	-	-	-	22
7th	3	7	3	0	-	-	-	-	-	-	...	-	-	-	10
8th	3	6	10	0	-	-	-	-	-	-	...	-	-	-	16
9th	3	6	10	0	-	-	-	-	-	-	...	-	-	-	16
10th	3	6	10	0	-	-	-	-	-	-	...	-	-	-	16
11th	3	7	10	0	-	-	-	-	-	-	...	-	-	-	17
12th	3	8	10	0	-	-	-	-	-	-	...	-	-	-	18
13th	29	9	10	10	10	10	10	10	10	10	...	10	10	0	279
14th	8	10	9	10	7	8	4	3	0	-	...	-	-	-	51

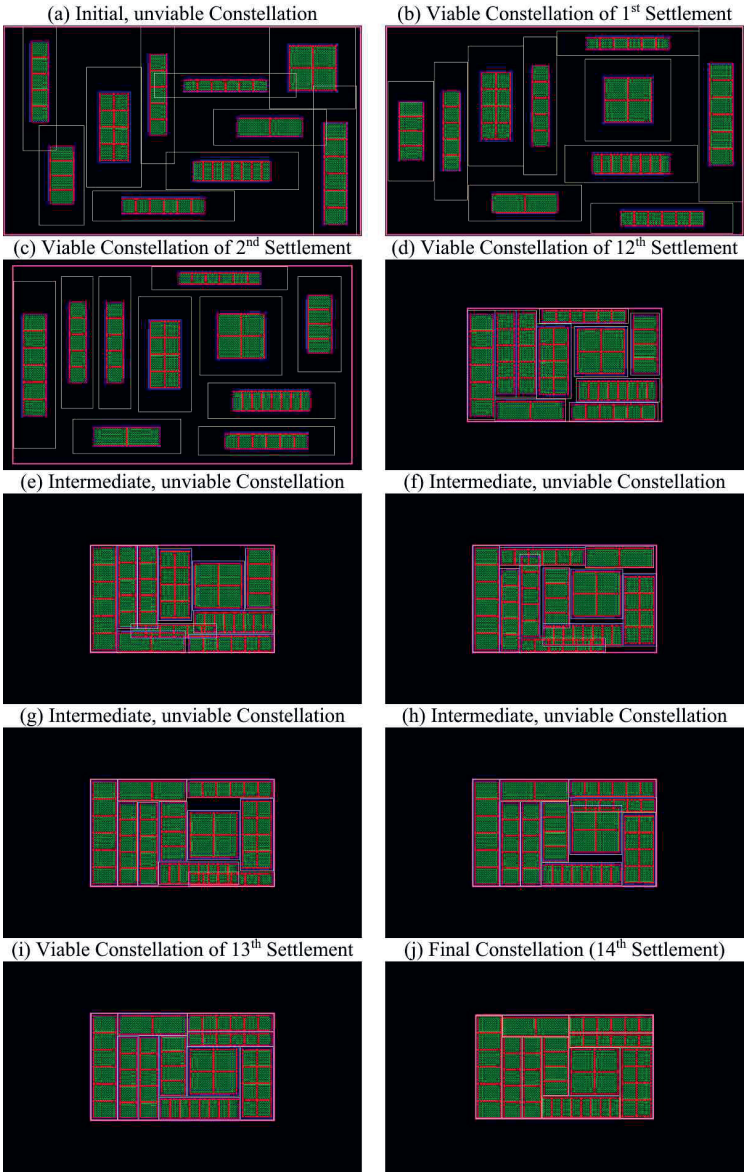
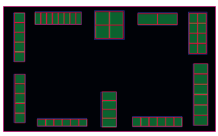
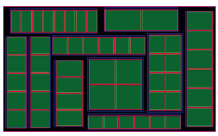
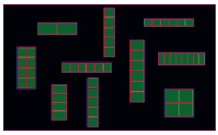
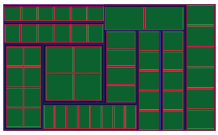


Figure 4.3: Second example of a placement problem with the emergence of an optimal outcome.

Another insight from the interaction record here is that the first episode of this SWARM run lasts extremely long, such that the second episode occurs quite late (in the penultimate settlement) and more or less merges with the third episode (i.e., the last settlement). Paradoxically, this shift is rooted in the participants' achievement of obtaining a nearly optimal arrangement already with the 2nd settlement. For that reason, the remainder of the problem-solving is entirely postponed to the very end. Yet, although this imbalance is detrimental for the fluency of the self-organization, this example demonstrates that the participants are still able to abandon such a local optimum –as good as it may be– in favor of an even better (here: the best) solution. The total number of performed actions adds up to 514 (out of 15476 explored actions), demanding a runtime of 145 seconds for the entire SWARM run.

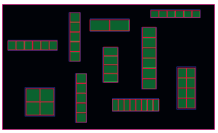
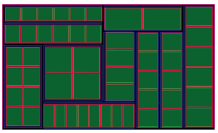
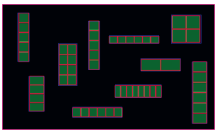
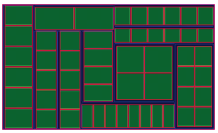
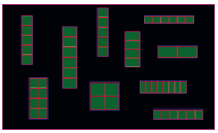
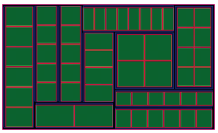
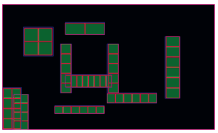
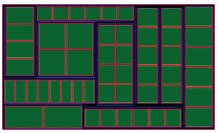
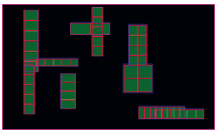
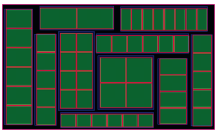
To fortify that the successful outcome in this example is not just a lucky strike, SWARM has been put to the test with ten different initial constellations (six of which were handcrafted while four were randomly created). For each SWARM run, the initial constellation and the final constellation, as well as certain statistics are listed in Table 4.3, sorted by the totally required runtime (with the SWARM run of Figure 4.3 appearing as #4). In six cases (covering runs with handcrafted and randomly created initial constellations), the participants manage to achieve the optimal solution while in the remaining four runs only a nearly optimal placement emerges from the interaction.

Table 4.3: Comparison of different SWARM runs for the same placement problem.

Run	Initial Constellation	Final Constellation	SWARM Run Statistics
#1	 (handcrafted)	 (nearly optimal)	Rounds per Settlement: ($\varnothing = \frac{54}{15} = 3.6$) 4, 4, 4, 2, 3, 3, 2, 3, 3, 4, 3, 5, 5, 3, 6 Number of Actions: 2089 (e) = 1549 (r) + 254 (p) + 286 (d) Total Runtime: 49 seconds
#2	 (handcrafted)	 (optimal)	Rounds per Settlement: ($\varnothing = \frac{77}{14} = 5.5$) 4, 3, 3, 3, 3, 3, 3, 3, 5, 27, 3, 3, 4, 10 Number of Actions: 10302 (e) = 9532 (r) + 531 (p) + 239 (d) Total Runtime: 106 seconds

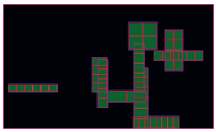
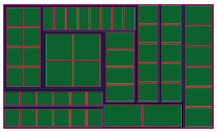
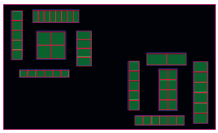
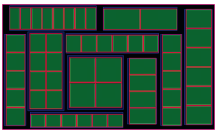
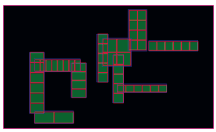
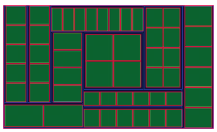
to be continued on next page

Table 4.3: Comparison of different SWARM runs for the same placement problem (continued).

#3			Rounds per Settlement: ($\varnothing = \frac{79}{13} = 6.1$) 37, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 9 Number of Actions: 15894 (e) = 15104 (r) + 598 (p) + 192 (d) Total Runtime: 137 seconds
	(handcrafted)	(optimal)	
#4			Rounds per Settlement: ($\varnothing = \frac{77}{14} = 5.5$) 3, 6, 2, 3, 3, 5, 3, 3, 3, 3, 3, 3, 29, 8 Number of Actions: 15476 (e) = 14706 (r) + 514 (p) + 256 (d) Total Runtime: 145 seconds
	(handcrafted)	(optimal)	
#5			Rounds per Settlement: ($\varnothing = \frac{95}{13} = 7.3$) 40, 3, 3, 3, 3, 3, 3, 4, 4, 10, 3, 4, 12 Number of Actions: 19529 (e) = 18579 (r) + 740 (p) + 210 (d) Total Runtime: 172 seconds
	(handcrafted)	(optimal)	
#6			Rounds per Settlement: ($\varnothing = \frac{104}{13} = 8.0$) 4, 3, 3, 5, 3, 8, 4, 5, 7, 32, 3, 12, 15 Number of Actions: 21702 (e) = 20662 (r) + 779 (p) + 261 (d) Total Runtime: 198 seconds
	(randomized)	(nearly optimal)	
#7			Rounds per Settlement: ($\varnothing = \frac{100}{14} = 7.1$) 4, 3, 3, 3, 3, 3, 9, 9, 5, 24, 4, 3, 21, 6 Number of Actions: 23098 (e) = 22098 (r) + 719 (p) + 281 (d) Total Runtime: 208 seconds
	(randomized)	(nearly optimal)	

to be continued on next page

Table 4.3: Comparison of different SWARM runs for the same placement problem (continued).

#8			Rounds per Settlement: ($\varnothing = \frac{112}{13} = 8.6$) 11, 3, 4, 4, 4, 9, 3, 52, 4, 3, 3, 4, 8 Number of Actions: 26597 (e) = 25477 (r) + 879 (p) + 241 (d) Total Runtime: 226 seconds
	(randomized)	(optimal)	
#9			Rounds per Settlement: ($\varnothing = \frac{145}{13} = 11.2$) 4, 5, 2, 4, 4, 4, 5, 5, 17, 81, 4, 4, 6 Number of Actions: 34528 (e) = 33078 (r) + 1177 (p) + 273 (d) Total Runtime: 296 seconds
	(handcrafted)	(nearly optimal)	
#10			Rounds per Settlement: ($\varnothing = \frac{267}{16} = 16.7$) 5, 3, 3, 4, 4, 28, 3, 177, 3, 3, 3, 4, 15, 3, 3, 6 Number of Actions: 83948 (e) = 81278 (r) + 2408 (p) + 262 (d) Total Runtime: 648 seconds
	(randomized)	(optimal)	

As can be seen, the required runtime is closely correlated with the number of actions involved. This primarily refers to the number of *explored actions* –marked (e)– although Table 4.3 also indicates inhowfar these explored actions divide into

- *rejected actions* (r):
  actions that are either deemed invalid or inferior to other actions,
- *performed actions* (p):
  the actions which are indeed executed by the participant(s), and
- *dismissed actions* (d):
  chosen actions that are not executed since they are *trivial* (primarily because they fall below the minimal movement distance).

A correlation between the runtime and the fluency of the self-organization can also be recognized in Table 4.3. The smoothest flow of self-organization is therefore found in example #1, where the number of rounds per settlement only varies between 2 and 6 (conceding that the final constellation is not the global optimum). A counterexample is given by the SWARM run of #10, climaxing in a total of 177 interaction rounds for the eigth settlement (not in vain since the best possible solution is finally obtained).

Table 4.4: Detailed number of actions for selected SWARM runs of Table 4.3. The abbreviated kinds of actions are: Re-entering (R), Centering (C), Lingering (L), Budging (B), Hustling (H), Swapping (S), Pairing (P), Evasion (E), and Yielding (Y).

		Number of Actions (Distinguished by Kind of Action)									
Run	**Qualifier**	R	C	L	B	H	S	P	E	Y	**Total**
#2	Explored:	106	770	1	3020	47	3224	3060	74	0	10302
	Rejected:	99	102	0	2967	41	3212	3051	60	0	9532
	Performed:	7	429	1	53	6	12	9	14	0	531
	Dismissed:	0	239	0	0	0	0	0	0	0	239
#4	Explored:	106	770	0	4621	69	5435	4400	75	0	15476
	Rejected:	106	137	0	4562	55	5413	4378	55	0	14706
	Performed:	0	377	0	59	14	22	22	20	0	514
	Dismissed:	0	256	0	0	0	0	0	0	0	256
#6	Explored:	100	1040	0	6637	89	7406	6322	107	1	21702
	Rejected:	94	171	0	6545	77	7386	6303	86	0	20662
	Performed:	6	608	0	92	12	20	19	21	1	779
	Dismissed:	0	261	0	0	0	0	0	0	0	261
#8	Explored:	118	1120	5	8430	110	8901	7810	102	1	26597
	Rejected:	118	208	0	8328	97	8869	7778	79	0	25477
	Performed:	0	671	5	102	13	32	32	23	1	879
	Dismissed:	0	241	0	0	0	0	0	0	0	241
#10	Explored:	148	2670	1	26757	364	29714	24150	143	1	83948
	Rejected:	141	642	0	26492	331	29524	24042	106	0	81278
	Performed:	7	1766	1	265	33	190	108	37	1	2408
	Dismissed:	0	262	0	0	0	0	0	0	0	262

To survey the number of actions in greater detail, Table 4.4 presents a so-called *action palette* covering selected SWARM runs of Table 4.3. For each of these runs, the number of explored, rejected, performed, and dismissed actions are listed – distinguishing between the different kinds of native actions.

An attentive reader will probably notice that in the listed SWARM runs, no explored *Lingering* or *Yielding* action is ever rejected. Considering the former kind of action, this is generally always the case (because the participant is definitely contented when the *Lingering* is being explored). In contrast, a *Yielding* move can be rejected for the reason of being invalid. Furthermore, it can be seen that all dismissed actions in the table are *Centering* actions. Although this

is usually the case indeed, it does not necessarily have to be like this because actions of other kinds can also be trivial.

More importantly, the following three basic insights can be gained from the given action palette:

- In each SWARM run, actions of almost every kind have been performed (in run #10 even all kinds of actions). This finding fleshes out that the general spectrum and the concrete implementation of SWARM's catalog of native actions is quite feasible.
- By far, the largest exploration effort (and highest rejection rate) naturally pertains to *Budging*, *Swapping*, and *Pairing* actions (to some extent rooted in the exhaustiveness of the respective *exploration plan*). This awareness presumably provides a major starting point for potential performance improvements in further developments of the SWARM methodology.
- The majority of performed actions usually represent *Centering* actions. To reduce runtime, it might be worthwhile to investigate the impact of eliding a *Centering* move in case the participant is contented anyway (especially in the early tightening-settlement cycles of a SWARM run, when there is still much space available).

Notwithstanding these performance aspects, one has to acknowledge that –to a certain degree– the required runtime is deliberately condoned in favor of visualizing a SWARM run since it allows the user to track the module interaction in real-time. This represents a valuable asset of SWARM that would be impossible and impractical to realize for a stochastic optimization algorithm which executes millions of perturbations to find a solution. Another concession is that SKILL is just a scripting language, whereas a potential implementation in a compilation-based programming language might reduce the runtime significantly. Nevertheless, this of course does not belittle the merit of the overall methodology.

4.1.3 Examples of Nonterminating Interaction Cycles

While the previous examples illustrated positive effects of emergent behavior in SWARM, there can unfortunately also be futile occurrences of emergence. Most intriguing, under certain circumstances it can happen that the interaction runs into a sequence of periodically recurring situations, which –unbeknownst to the interacting participants– represents a fatal infinity loop that will never terminate without intervention from outside.

An example for such a *nonterminating interaction cycle* is given in Figure 4.4, beginning with an initial constellation of nine participants as depicted in image (a). All participants are centered within their free peripheral space – except P_1, which perceives its free peripheral space S_{P_1} (limited by P_3, P_4, P_5 and P_8) and determines the center of S_{P_1} as its new location. After performing its *Centering* move (b), P_1 is still clear even though P_2 had been in a *blind spot*

of P_1's free peripheral space. But due to that move, the free peripheral spaces of P_2, P_3 and P_4 change (whereas the other participants remain unaffected). Therefore, P_2, P_3 and P_4 also perform *Centering* moves, leading to the new situation shown in (c). By now, the free peripheral space of P_1 has changed –in particular because its western corridor is not limited by P_8 anymore but by P_2– and P_1 once again centers itself (d), which in turn entails small *Centering* movements of P_2, P_3, and P_4 (e). Because of P_1's move, the obstacle limiting its free peripheral space to the north is now P_6 instead of P_5, causing P_1 to make a *Centering* leap downwards and slightly to the right (f). While this does not alter the free peripheral space of P_4, it provokes *Centering* actions of P_2 and P_3 (g). By then, the westwards obstacle for P_1's free peripheral space is P_8 once again, making P_1 walk to the left by performing yet another *Centering* (h). This is followed by *Centering* moves of P_3 and P_4 (i), which results in a situation that is identical to the initial constellation (a). Since moving from a clear location to another clear location –as was the case with all *Centering* actions above– does not consider aversions and wounds (thus eluding a memory effect in examples like this), the participants will definitely repeat the described succession of movements once again, and again and again without ever coming to an end.

Such an outcome of perpetual interaction (which is somehow reminiscent of a draw in chess due to *repetition of position*) cannot be detected from the limited viewpoint of a single participant but only by an outside observer who surveys the entire problem as a whole. The possibility to get lost in an infinity loop like this represents a tragic but natural weakness of decentralized systems that can also be encountered in reality. An enthralling example is given by a so-called *ant mill* as first reported in [184]. Although the pheromone-guided trailing habit of social insects helps an ant colony in accessing food resources (see Section 2.3.2), a self-intersection of the pheromone trail can delude the leading ants into tracing the tail of the foraging party. This leads to the formation of a rotating circular column (illustrated in Figure 4.5 [185]) which may go on forever, i.e., until the ants die of exhaustion.

Equivalent to an ant mill, the periodic interaction sequence of Figure 4.4 can be regarded as a cycle of rotatory *circulation* (with participant P_1 truly going round and round in endless circles). Another kind of nonterminating interaction cycle that can also be observed to emerge in SWARM is an *oscillation*. The simplest form is a two-period oscillation, where the movements in one round of interaction are entirely reversed in the subsequent round.

To visualize such an oscillation sequence, Figure 4.6 provides a plain example with seven participants. In the initial constellation (a), all participants except P_1 are centered within their free peripheral space. Naturally, P_1 also performs a *Centering* move leading to the situation shown in image (b). Thereby, the free peripheral space of P_2 gets trimmed, making the participant wander to the right via another *Centering* (c). In this fashion, the free peripheral space of P_3 is also cut, provoking a *Centering* move downwards (d). The *domino effect* continues to propagate through P_4 (e) and P_5 (f) while participants P_6 and P_7 are not bothered. So, the situ-

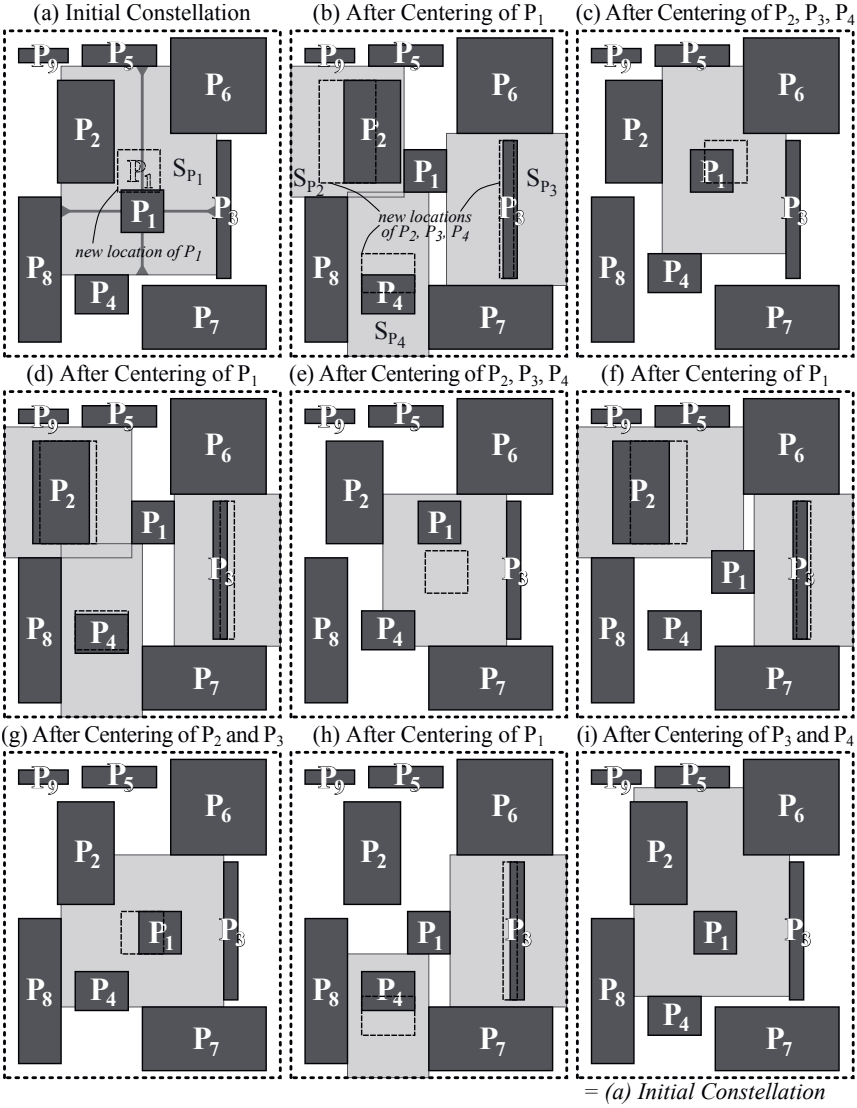


Figure 4.4: Example of a nonterminating interaction cycle: a rotatory circulation sequence.

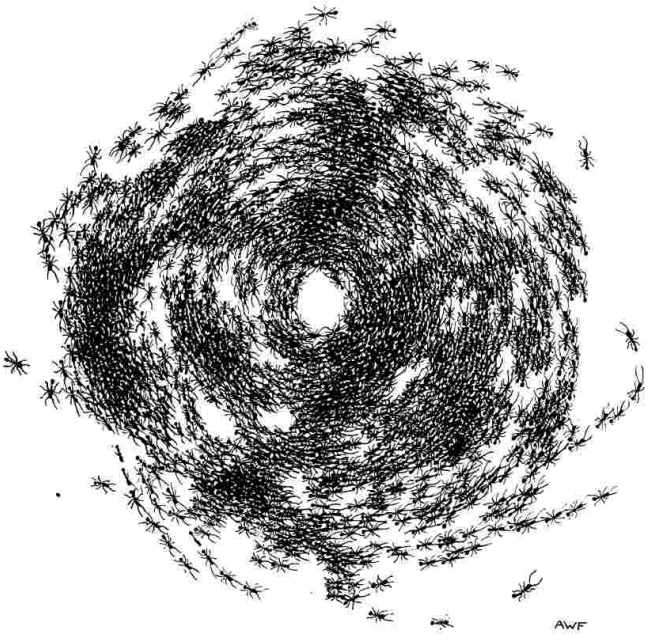


Figure 4.5: Circular column of an ant mill, as drawn according to a photograph (source: [185]).

ation shown in (f) represents the initial constellation for the next round of interaction (g). Since P_5 moved out of P_1's southern corridor with its previous move, the free peripheral space of P_1 now stretches down to the zone outline and lets it step back into its initial location (h). Thereby, the free peripheral space of P_2 analogously gets reverted to its original size, sucking P_2 back into its previous location (i). The same thing happens to P_3 (j) and this chain reaction proceeds through P_4 (k) as well as P_5 (l). Like before, P_6 and P_7 stay where they are so in the end, the participants' final arrangement –as can be seen in (l)– is equal to the initial constellation (a).

From then on, similar to the circulation example of Figure 4.4, the participants in Figure 4.6 will repeat the illustrated sequence of actions without ever settling in a definite location. But in contrast to the circular rotation in the former example, the interaction here rather represents an oscillation wave, consisting of a round with "forward" movements followed by a round of countermanding "backward" movements. Figuratively, the participants swing forth and back like electrical energy oscillates between the inductor and the capacitor in an LC resonant circuit.

Another parallel can be drawn between SWARM and cellular automata once again, since Conway's Game of Life also knows oscillators, i.e., periodic patterns which repeat themselves on the spot ad infinitum like the oscillation cycle in Figure 4.6 can be seen to do. Furthermore,

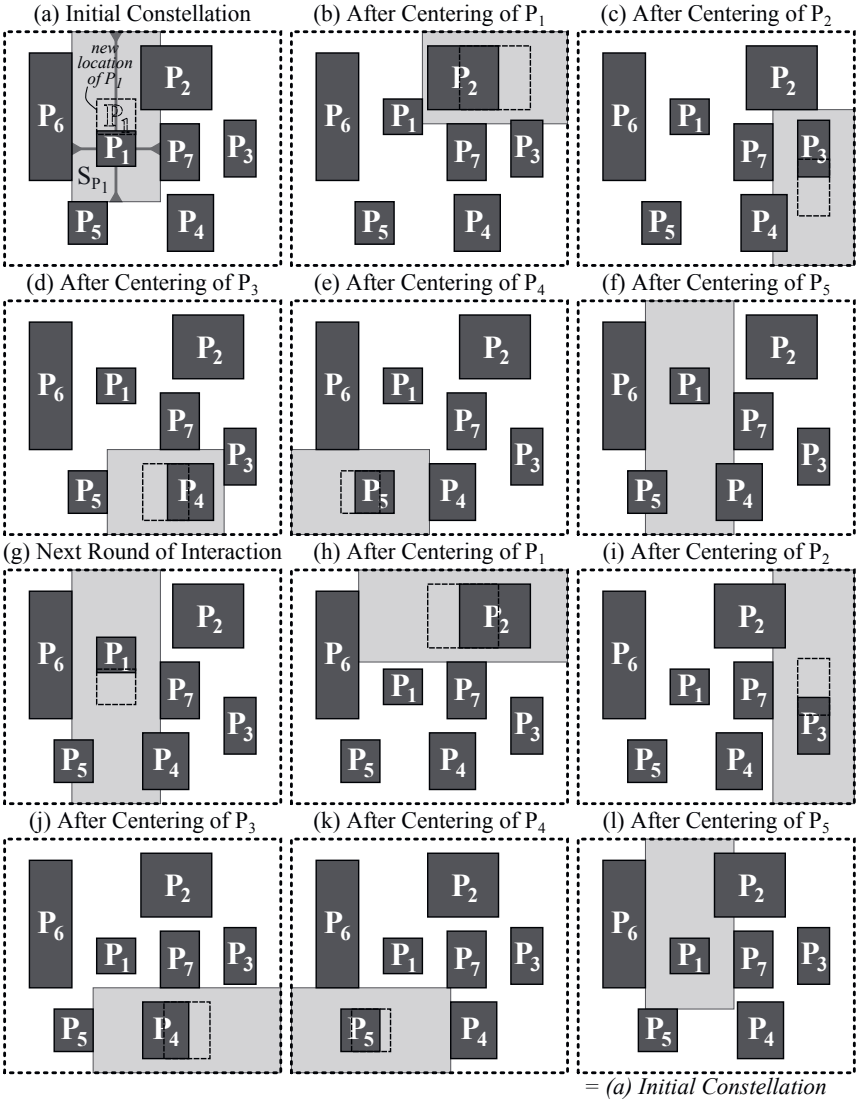


Figure 4.6: Example of a nonterminating interaction cycle: a two-period oscillation sequence.

moving patterns such as gliders and spaceships can be understood as the automaton's counterpart to the perpetual interaction in Figure 4.4 (although the pattern's trajectory is a straight line instead of a circular path). Completing the picture, a settlement in SWARM corresponds to a still life in the cellular automaton. This whole analogy fits snugly, but it also points to an important difference as follows.

A settlement in SWARM is stable in a static sense. A nonterminating interaction cycle is obviously not stable in that sense, no more than it is chaotic by any means. Yet, it would also be wrong to call such a cycle *metastable* in the sense of the *edge of chaos* (Section 2.4.3) since it is definitely not short-lived – the opposite is the case. From that perspective, one can say that a steady circulation or oscillation as exemplified above is also stable, not in a static sense but in a dynamic fashion. Of course, certain interaction dynamics are required in SWARM to reach a state of static stability after each tightening: a settlement which is tighter (i.e., of higher value) than the previous one. These dynamics ask for metastability – neither lazing in lower-level order, nor drifting into chaos and instability, nor (never-)ending in a periodic sequence of dynamic stability.

Now, to explicate the mentioned difference in the established analogy, one can contend that in Conway's Game of Life, the greatest fascination –at least concerning the visual spectacle– probably emanates from dynamically stable structures (such as gliders and spaceships) as well as the so-called methuselahs (i.e., highly chaotic patterns which require a large amount of transitions to reach a stable state or dissolve into nothing). In contrast, the success of SWARM depends on producing still lifes (above all: the ultimate settlement) whereas dynamically stable infinity loops and completely instable devolutions into utter chaos need to be avoided.

4.2 Practical Floorplanning Examples

Even though SWARM has not been primarily developed for targeting floorplanning problems, it can be readily employed for that purpose. This is demonstrated by the following two examples, the first one (Section 4.2.1) again featuring a rectangular outline like the examples before, the second one (Section 4.2.2) showcasing SWARM's ability to consider nonrectangular –i.e., rectilinear– outlines. An assessment of SWARM with respect to floorplanning tasks will then be given in Section 4.2.3.

4.2.1 Floorplanning Example with Rectangular Outline

Figure 4.7 depicts a rectangular floorplanning problem with 15 circuit blocks of various sizes. As is customary in floorplanning (see Table 2.2 on page 21 of [1]), these blocks are treated as black boxes (here displayed as plain rectangles) whose aspect ratios have yet to be determined. In line with the explication of Section 3.3.1.2, the distance between two blocks is modeled as a

straight *connection* between the centerpoints of the blocks. In this example, all connections are equally emphasized (i.e., each connection has an *emphasis* of $e = 1$). In the initial constellation, only 2 out of the 21 connections are *relaxed* (drawn as solid lines) while 19 connections are *unrelaxed* (drawn as patterned lines), as illustrated in image (a).

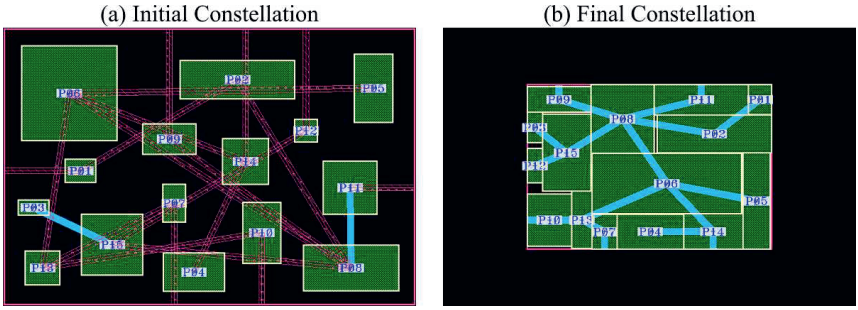


Figure 4.7: Example of applying SWARM to a floorplanning problem with a rectangular outline.

The distance between a block and an external component (e.g., a bondpad) is represented as a line which ends on the outline of the layout zone. It is handled like a block-to-block connection except that this endpoint automatically moves along the zone outline during the interaction, such that the distance is always kept as short as possible. The blocks themselves are implemented as *elastic* participants, exploiting the entire pool of elastic deformation behaviors covered in Section 3.3.3.3, to consider *full variability* during their action exploration.

The final constellation (b) reveals which aspect ratios the 15 floorplan blocks have assumed at the end of their self-organization. The resulting outcome is highly compact as the amount of dead space only makes up 2.9% of the layout zone area. Furthermore, all connections are relaxed at last: as can be seen in the final settlement, each pair of connected blocks is located in direct vicinity of each other and every block that has a connection to an external component lies next to the zone boundary. The figurative icing on the cake in this example is that the self-organization has led to the emergence of an arrangement wherein no connection lines cross one another.

To obtain this solution, the participants required a total of 71 interaction rounds, spread across 9 tightening-settlement cycles (interaction rounds per cycle: 5, 3, 6, 9, 14, 11, 3, 4, 16). An *abrasive tightening* policy and *volatile comfort padding* (with a volatile comfort padding share of $c_v = 0.4$) were used in this example. This can also be recognized in the tightening profile of the SWARM run, given in Figure 4.8, which shows how the size of the layout zone as well as the total area of the padded participants approach the *intrinsic minimum* (detailed numbers being listed in Table 4.5). Furthermore, the graph exhibits the three characteristic

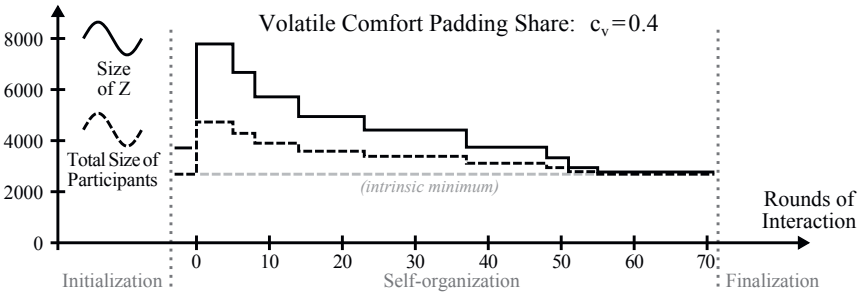


Figure 4.8: Tightening profile for the SWARM run of the floorplanning example from Figure 4.7.

episodes of a typical SWARM run, except that the last settlement peaks out with a large number of interaction rounds (which is not unusual in SWARM runs where the layout zone becomes extremely tight in the end, as it is the case in this example).

Table 4.5: Area decline in the SWARM run of the floorplanning example from Figure 4.7.

	Area per Tightening-Settlement Cycle								
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
Layout Zone:	7786.4	6674.3	5710.0	4939.6	4420.3	3749.8	3329.8	2945.3	2770.3
Participants:	4728.1	4284.0	3897.8	3589.5	3383.6	3114.0	2946.5	2791.8	2689.4
Dead Space:	39.3%	35.8%	31.7%	27.3%	23.5%	17.0%	11.5%	5.2%	2.9%

4.2.2 Floorplanning Example with Nonrectangular Outline

In Figure 4.9, SWARM is applied to a floorplanning problem where the layout zone has a nonrectangular outline. Such a contour can be encountered in cases where the desired floorplan only represents a part of the entire chip, using the notch to preserve layout space for other circuit components or IC sections. The 16 blocks feature 30 connections, and –in contrast to the previous example– the connections have been assigned different emphasis values (between $e = 0.4$ and $e = 1.0$), in the images indicated via the widths of the respective connection lines. As can be seen in image (a), 9 connections in the initial constellation are relaxed and the remaining 21 connections are unrelaxed.

With a total of 95 interaction rounds throughout 8 tightening-settlement cycles, the participants are able to obtain the final constellation shown in (b) where the floorplan blocks can again be seen to have changed their aspect ratios. In this outcome, the dead space amounts to 5.7% of

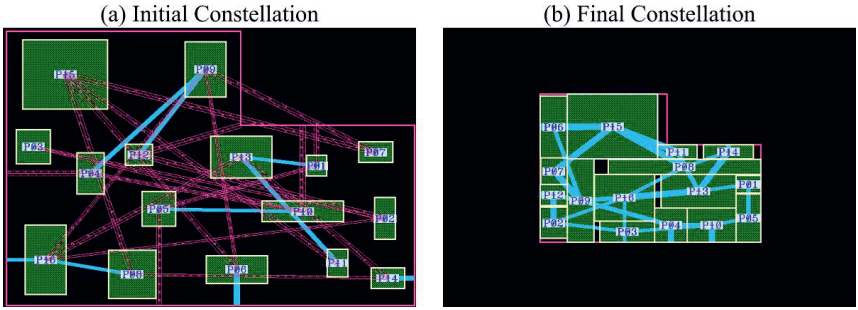


Figure 4.9: Example of applying SWARM to a floorplanning problem with a nonrectangular outline.

the layout zone and –as in the example of Figure 4.7– all connections are finally relaxed. And again, also reminiscent of the example before, tracking the number of interaction rounds per tightening-settlement cycle (5, 3, 10, 5, 9, 19, 12, 32) bares two maxima: an intermediate peak for the 6$^{\text{th}}$ settlement and a final peak for the 8$^{\text{th}}$ settlement. With an abrasive tightening policy and a volatile comfort padding share of $c_{\text{v}} = 0.4$, Table 4.6 displays the area decline observed during the SWARM run.

Table 4.6: Area decline in the SWARM run of the floorplanning example from Figure 4.9.

	Area per Tightening-Settlement Cycle							
	1$^{\text{st}}$	2$^{\text{nd}}$	3$^{\text{rd}}$	4$^{\text{th}}$	5$^{\text{th}}$	6$^{\text{th}}$	7$^{\text{th}}$	8$^{\text{th}}$
Layout Zone:	8183.1	6509.5	5130.4	3892.4	3272.3	2823.6	2497.6	2387.7
Participants:	4624.0	3955.7	3403.4	2908.2	2659.5	2479.7	2351.0	2252.2
Dead Space:	43.5%	39.2%	33.7%	25.3%	18.7%	12.2%	5.9%	5.7%

4.2.3 Assessment

Although SWARM has not been conceived with the task of floorplanning in mind initially, the examples of Section 4.2.1 and Section 4.2.2 demonstrate that the self-organization approach is quite adequate for addressing this application. Moreover, a look at Table 3.1 on page 39 of [1] reveals that the SWARM methodology even surpasses existing floorplanners in terms of five decisive aspects, as SWARM apparently is the first floorplanning approach that

(1) takes both area and block distances into account,
(2) pays respect to a rectilinear layout outline,

(3) considers fully variable block dimensions,
(4) supports nonslicing floorplan structures,
(5) and works completely deterministic.

The first three items are visibly evidenced by the examples above while the fifth issue is a logical consequence of SWARM's credo not to involve any kind of randomization. Regarding aspect (4), it is interesting to see that both the outcome in Figure 4.7 (b) as well as the final constellation of Figure 4.9 (b) represent slicing floorplans. Yet, it is important to realize that this predicate has not been explicitly imposed as a design-methodical restriction, but that it simply emerged from the interaction as an "energetically favorable state", so to speak. In other words, such slicing structures seem to be natural *attractors* of the system – but as the two examples published in [186] underline, it is generally possible that nonslicing solutions can result from a SWARM run.

Some particular floorplanning issues are not yet handled by SWARM. For instance, SWARM does not set pin positions on the floorplan blocks and is not intent on optimizing the power supply or the current flow. Inhowfar it is possible to incorporate such aspects into SWARM, e.g., via dedicated design constraints, may be subject to future works. Another common demand is to separate certain blocks from each other such that sensitive circuitry is not disturbed by noisy IC sections. In further developments of SWARM, this could be achieved by modeling compressive stress (instead of *tension*) into a connection, thereby reversing its effect on the interaction.

4.3 Practical Place-and-Route Examples

To demonstrate SWARM being employed for practical place-and-route purposes, the following Section 4.3.1 first elaborates on how the methodology has been integrated into the SDL design flow from the perspective of usage. Then, SWARM is shown creating layouts of various aspect ratios for a Symmetric P-Input OTA (Section 4.3.2) and a Folded Cascode P-Input OTA (Section 4.3.3). Finally, Section 4.3.4 assesses the SWARM methodology with regard to such place-and-route problems.

4.3.1 Usage of SWARM in the Design Flow

To facilitate the usage of SWARM in the design flow, it has been worked into the so-called *Constraint-Administered PCell-Applying Blocklevel Layout Engine* (CAPABLE) presented in [187]. CAPABLE is a designer-oriented framework that can be used to combine various automatisms into an executable layout automation script. While this concept is not restricted to including either algorithmic or procedural automatisms, CAPABLE is predominantly designated to realize a generator approach. In particular, the engine aims at applying *context aware* PCells to the layout in a hierarchical fashion (as it is intended for the *governing modules* in

SWARM). Hence, a CAPABLE script represents a command sequence that operates similar to a procedural generator, but on a higher level of abstraction.

CAPABLE is constraint-administered insofar as it provides a special *constraint interpreter* by which certain parts of the layout automation script can be automatically created from formally expressed design constraints that have been assigned to the input schematic by the designer. For example, a custom CurrMirr constraint –assigned to a group of transistors that are meant to constitute a Current Mirror circuit– is turned into a series of script commands that consecutively impose a Positioning module, a Wiring module, and an Info module on the devices (thus establishing a Current Mirror module association as depicted in Figure 3.8 on page 41). In this manner, high-level design requirements can be *explicitly* formulated as constraints, while the respective low-level implications are *implicitly* taken into account by the automatisms that implement these constraints.

Taking the point of critique to heart that analog layout automatisms often find only little acceptance due to a lack of insight into their inner workings, CAPABLE comes with a convenient GUI that allows to execute the developed layout scripts in a transparently traceable way. A graphical *execution cockpit* provides the user with different pacing modes – from running the entire script all at once to performing each command call in a fine-grained fashion, even separating different steps that the command may be comprised of. For a governing module, each individual operation of the *adoption process* described in Section 3.2.2.1 can be separately carried out as a single step. Utilizing this facility, the events occurring in the self-organization of a SWARM run can be recorded as a script and later be replayed step by step via CAPABLE.

The interface between a CAPABLE script and the design environment Virtuoso is made up by a small set of API commands referred to as *Constraint-Derived Procedural Commands* (CDPC). It is written in SKILL and can be considered as a domain-specific language for layout automation. Since CDPC is primarily focused on the application of PCells, the central CDPC command is `PCell` and serves the instantiation of a context aware PCell according to the adoption process mentioned above. Another CDPC command is named `Group` and allows to store a set of layout components in a single internal group (which may be helpful if multiple PCells are to be imposed on the same set of components, as in a Current Mirror module association). To give an example for a non-PCell automatism, the `Modgen` command can be used to invoke the interdigitation feature of Virtuoso's own ModGen tool [188]. Currently, the following commands are supported:

`Cellview`
: Opens the specified target layout (or creates an empty one if none yet exists).

`Revert`
: Discards any modifications that have been made to the layout since its last save.

`Clear`
: Removes all objects (instances, shapes, labels, markers, etc.) from the layout.

`GenFromSource`
: Invokes the design environment's native *Generate from Source* functionality.

`SetVar`
: Stores a given value in a specified local variable.

`GetVar`
: Retrieves the value of a specified local variable.

`Group`
: Assigns a set of layout components to an internal group.

`PCell`
: Instantiates a context aware PCell following the adoption process of Section 3.2.2.1.

`Modgen`
: Executes the interdigitation function of Virtuoso's proprietary ModGen tool.

`Rotate`
: Rotates a layout component (plus its adopted children and associated modules).

`Move`
: Moves a layout component (plus its adopted children and associated modules).

`PlacePin`
: Puts a generated layout pin to a given location and optionally sets its layer.

`Flatten`
: Flattens a layout component (as well as its adopted children) hierarchically.

`Geo`
: Uses the PCell Designer's *geometry query language* to determine layout geometries.

`Replay`
: Performs an event that was recorded during the self-organization of a SWARM run.

Calling the `PCell` command, a dedicated Wiring PCell can be instantiated in the layout. Such a PCell instance draws a single wire, comprised of different wire segments, between two component terminals. Together with the `Geo` command for retrieving the location of a component terminal, this enables the implementation of procedural routing scripts. Referring to the explanation in [189], a PCell parameter allows to specify the wire as a sequence of points with the possibility to set the metal layer and wire width for each individual segment. Although the points may be given in absolute coordinates, they are rather meant to be defined in relation to the bounding boxes of existing components in the layout so the wire can smoothly run along the edges of obstructive modules.

Such a routing approach can be quite appropriate if the overall module arrangement in the layout is already known in advance. This will be the case in the subsequent two examples and thus motivates the preconception of a procedural routing scheme for connecting the governing modules with each other (while the connectivity inside each module is still taken care of by the respective PCells). Therein, the need to preserve sufficient layout space between the modules is

where the idea of *solid comfort padding* comes into play. On that basis, going through the three phases of a SWARM run proceeds as follows:

(1) The *initialization phase* is carried out by a sequence of CDPC commands in the execution cockpit of CAPABLE. The corpus of this command sequence can be automatically derived from the constraints assigned to the input schematic via Virtuoso's constraint management system [190]. Custom edits and extensions to the command sequence are possible through manual scripting in a plain text editor.
(2) Switching to the SWARM tab of CAPABLE's user-interface, the user gains control over the *self-organization phase*. Similar to the different pacing modes in CAPABLE, the events may be driven by (a) stimulating a single participant, (b) initiating an entire round of interaction, (c) conducting a whole tightening-settlement cycle, or (d) making SWARM perform the complete self-organization until reaching the desired layout zone size.
(3) For the *finalization phase*, the user's focus returns to the execution cockpit of CAPABLE. Here, the preconceived routing scheme can be applied to the module arrangement of the self-organization's final settlement as a CDPC script. This complements the already existing *intra*-module wiring with the still missing *inter*-module wiring and finishes the layout.

4.3.2 Symmetric P-Input Operational Transconductance Amplifier

Figure 4.10 shows the schematic diagram of a Symmetric P-Input OTA circuit consisting of six functional units: a Differential Pair (1), three Current Mirrors (2 to 4), a Symmetric Current Mirror Pair (5) which in turn consists of two Current Mirrors (5a and 5b), as well as a Blocking Cap (6) to stabilize the supply voltage. By explicitly imposing a custom "Module" constraint (i.e., a DiffPair constraint, a CurrMirr constraint, etc.) on each of these functional units, SWARM is told to manage the corresponding layout devices with a governing module, module association, or hierarchical module association. Thereat, a user-definable interdigitation pattern can be provided for each Current Mirror (to be put into effect by the respective modules) for the benefit of a good device matching.

Based on the interdigitation pattern, each module implicitly takes care of its inherent matching requirements on device level. To achieve overall symmetry, the modules then are to be arranged according to a predefined *placement template* as visualized in Figure 4.11 (a). For the module interaction in SWARM, this sought-after arrangement is explicitly articulated via Relative Placement constraints (b). A constraint of this type –also appearing in [191]– forces a component to lie north, east, south, or west of another component. By implementing a *verification function* covering these four cases, the Relative Placement constraint is checked during the self-organization phase to avoid *noncompliance* whenever one of the involved modules explores its potential actions.

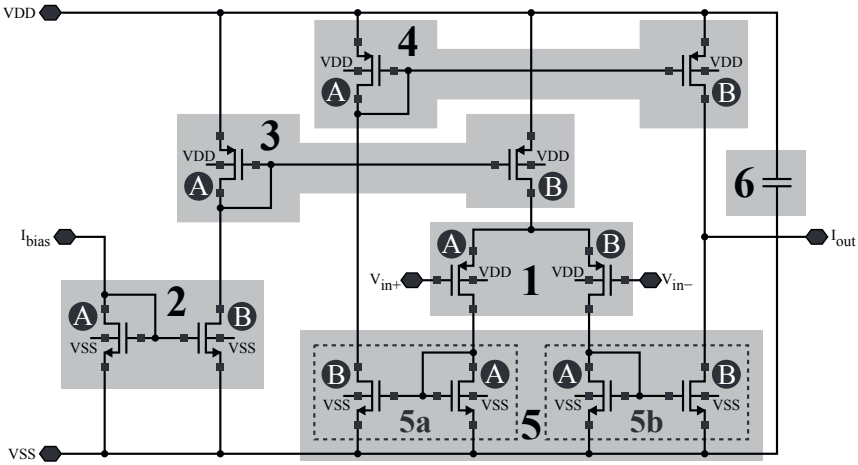


Figure 4.10: Schematic diagram of the Symmetric P-Input OTA circuit addressed with SWARM.

As an example for the circuit at hand, the input Current Mirror (2) is supposed to be located west of the Differential Pair (1). In this case, the verification function $v(P, P')$ –with the Current Mirror and the Differential Pair being assigned to P and P' respectively– looks like this:

$$v(P, P') \;=\; \begin{cases} 1 & \Leftrightarrow \;\; \frac{1}{2}\big(\vdash(P) + \dashv(P)\big) < \frac{1}{2}\big(\vdash(P') + \dashv(P')\big) \\ 0 & \Leftrightarrow \;\; \text{otherwise.} \end{cases} \tag{4.1}$$

The condition in this formula is less stringent than the one in [191] as it does not forbid an overlap of P and P'. This is deliberate: on the one hand, the condition is as lax as possible to temporarily permit situations of interference (which can be vital after a zone tightening to let prone modules become safe again); on the other hand, the condition is as strict as neccessary to effectively influence the modules in their attempt to become overlap-free at the end of each settlement. In other words, it is the combined power of two innate *desires* (formulated in Section 3.3.1.1 and Section 3.3.1.5 respectively) making sure that the modules are finally clear and constraint-compliant.

Table 4.7 gives an overview of the circuit's devices according to the module by which they are managed. Also listed are the respective interdigitation patterns and the *ductility* (number of possible deformation variants, as introduced in Section 3.2.4.4) of the module. As can be seen, the Blocking Cap (6) is handled with full variability (managed by a *variator module* as mentioned in Section 3.2.4.3). This is done to achieve that the Differential Pair is neatly placed in the middle. Paying heed to an Imitation constraint to enforce the action of the same name

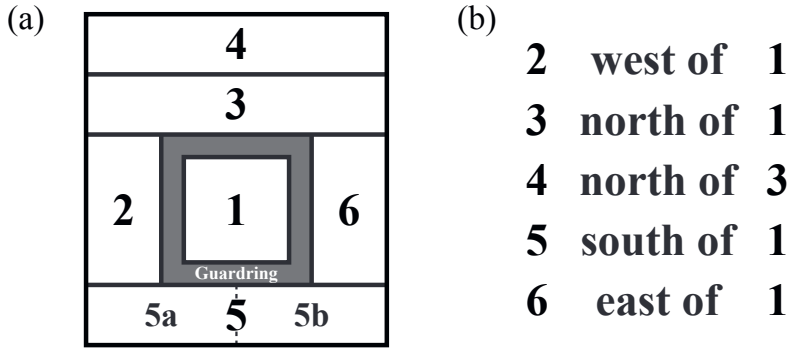


Figure 4.11: (a) Placement template and (b) respective constraints for the Symmetric P-Input OTA.

(see Section 3.3.3.2), the capacitor imitates the transformations of the input Current Mirror (2) on the other side of the Differential Pair (1), thereby permanently fitting its aspect ratio to that of the Current Mirror.

Table 4.7: Overview of the functional units constituting the Symmetric P-Input OTA circuit.

Index	Governing Module / Module Association	Device Type	Num. of Devices	Interdigitation Pattern	Ductility (Variants)
1	Differential Pair	PMOS	4	A B / B A (fix)	40
2	Current Mirror	NMOS	4	A B B A	56
3	Current Mirror	PMOS	6	B A B B A B	56
4	Current Mirror	PMOS	12	A B A B A B B A B A B A	56
5	Symm. Current Mirror Pair	-	-	-	112
└**5a**	Current Mirror	NMOS	8	A B A B B A B A	-
└**5b**	Current Mirror	NMOS	8	A B A B B A B A	-
6	Variator Module	Capacitor	1	-	full variability

To protect the Differential Pair from signal disturbances, it is to be enwrapped by a guardring (also indicated in Figure 4.11). This is accomplished with a Guardring constraint, telling SWARM to place a dedicated Guardring PCell aroung the Differential Pair module. Whenever the Differential Pair deforms itself during the module interaction, the Guardring automatically re-*absorbs* the Differential Pair and *adapts* itself to the new dimensions. Remember that *absorption* and *adaptation* represent the first two operations of SWARM's *adoption process* (Section 3.2.2.1).

Now, the initialization phase starts with a *Generate from Source* (called via the `GenFrom Source` command), in this case leading to the picture of Figure 4.12 (a) by producing a total of 43 primitive devices. Next, (based on the `PCell` command) the governing modules –implemented with Cadence PCell Designer [192]– are imposed on the respective groups of devices and then (using the `Modgen` command) put into the initial constellation shown in image (b). This arrangement, which already follows the template of Figure 4.11, ends the initialization phase (in line with the depiction of Figure 3.3 on page 33) and represents the starting point for the self-organization – irrespective of the layout zone's aspect ratio of which various ones are covered in the subsequent examples of this Section 4.3.2.

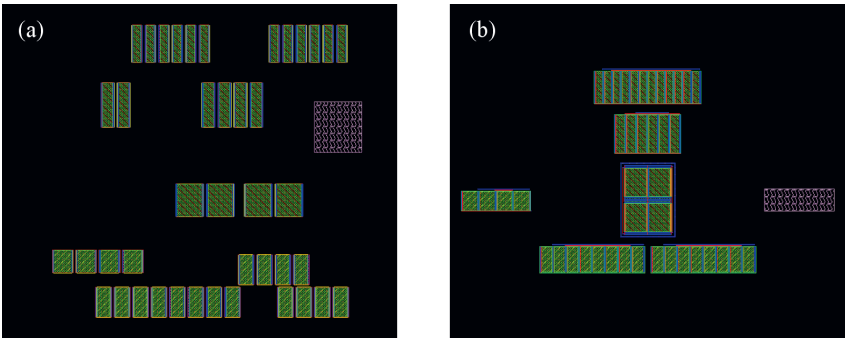


Figure 4.12: OTA layout (a) after device generation and (b) at the end of the initialization phase.

For a width-to-height aspect ratio of 2:3, the layout obtained by SWARM after going through the self-organization and the finalization phase is displayed in Figure 4.13. As can be seen, every module except the Differential Pair has deformed itself into a layout variant different from the one in the initial constellation of Figure 4.12 (b). A very fortunate advantage of a template-based arrangement as pursued here is that the locations of the modules in relation to each other do not change throughout the entire interaction. This allows to omit the exploration of *Swapping* and *Pairing* actions, which reduces the runtime tremendously (since these kinds of actions are computationally the most expensive ones, as mentioned in the context of Table 4.4).

For that reason, the SWARM run in this case just required a total of 22 interaction rounds throughout 5 tightening-settlement cycles. All in all, only 77 out of 1389 explored actions had to be performed to obtain the illustrated outcome, taking a runtime of 60 seconds for the mere self-organization phase. The observation that the quotient between this runtime and the number of performed or explored actions is larger than in the SWARM runs of Table 4.3 is simply rooted in the fact that –in contrast to the symbolic modules in those examples– a module transformation here is more elaborate because it has to include the corresponding co-transformations of all adopted children and associated modules (as explained in Section 3.2.2.2 and Section 3.2.3.3).

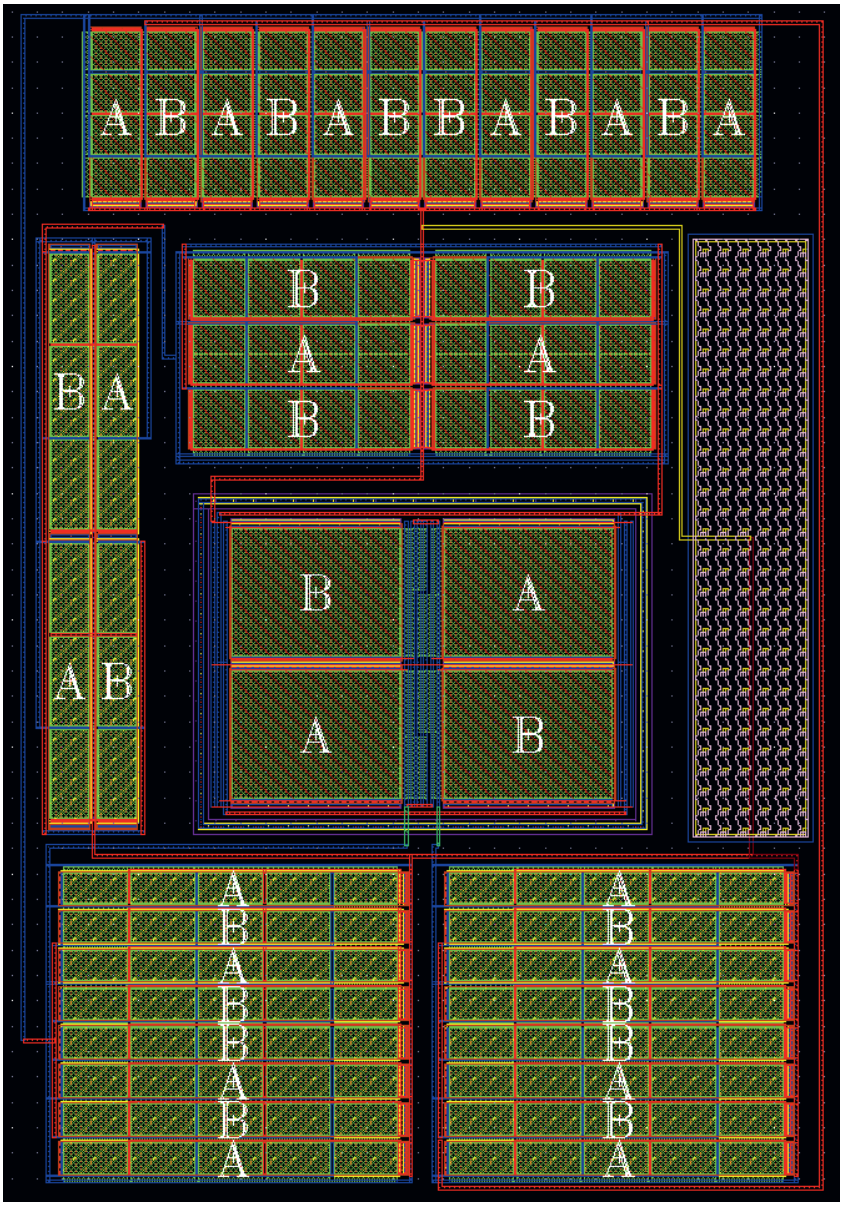


Figure 4.13: Finalized layout of the Symmetric OTA, obtained by SWARM for a 2:3 aspect ratio.

To demonstrate the immense layout flexibility achievable with SWARM, it been applied to the Symmetric P-Input OTA circuit of Figure 4.10 for a total of seven different aspect ratios. Table 4.8 displays the cardinal statistics for the respective SWARM runs, including the rounds of interaction per settlement, the number of explored, rejected, performed, and dismissed actions, as well as the number of performed actions distinguished by the kind of action, and the runtime required for the entire run (covering all three phases, i.e., initialization, self-organization, and finalization).

Table 4.8: Statistics of the different SWARM runs applied to the Symmetric OTA. The abbreviated kinds of actions are: Re-entering (R), Centering (C), Lingering (L), Budging (B), Hustling (H), Evasion (E), Yielding (Y), and Imitation (I).

Asp.	Rounds per	Total	Number of Actions				Perf. Actions (By Kind)								Total
Ratio	Settlement	Rounds	expl.	reje.	perf.	dism.	R	C	L	B	H	E	Y	I	Runtime
1:1	4, 5, 3, 8	20	1392	1272	71	49	1	47	0	7	1	1	1	13	108 sec.
1:2	4, 4, 3, 4, 10	25	364	214	97	53	0	77	1	0	0	1	0	18	113 sec.
1:3	5, 4, 4, 4, 5, 7	29	791	617	109	65	0	83	1	2	0	1	0	22	93 sec.
2:1	4, 16, 4, 10	34	1803	1599	109	95	2	62	0	6	10	1	1	27	233 sec.
2:3	4, 3, 4, 5, 6	22	1389	1257	77	55	1	55	0	5	0	0	0	16	113 sec.
3:2	4, 4, 6	14	935	851	46	38	1	32	0	3	0	2	0	8	95 sec.
5:2	4, 4, 5, 20	33	2158	1960	140	58	1	82	0	17	1	1	11	27	177 sec.

Taking a look at the rounds of interaction per settlement in Table 4.8 reveals that the self-organization in these runs proceeded quite smoothly on the whole. Even in the least fluent case (the SWARM run with a 5:2 aspect ratio, peaking out with 20 interaction rounds for the final settlement), only 140 actions out of 2158 explored ones had to be performed to obtain a compact and constraint-compliant layout solution in less than three minutes. Apart from this, it is again interesting to examine the number of performed actions per kind of action (understandably, *Swapping* and *Pairing* have been omitted here while *Imitation* was added to the action palette). As in the examples of Table 4.4, the most popular kind of action is *Centering*. But more importantly, it can be seen that every kind of action has been performed in at least two of the seven listed SWARM runs. This finding once more affirms the feasibility of the action catalog implemented by SWARM.

The finalized layouts for the 1:1 aspect ratio and the 5:2 aspect ratio are shown in Figure 4.14 and Figure 4.15 respectively (while the outcomes for the remaining aspect ratios have been moved to the appendix of [1]). As becomes evident from these images, the basic arrangement is the same in every case, exploiting the *cumulative variability* of each module to make the overall arrangement fit inside the outline of the layout zone.

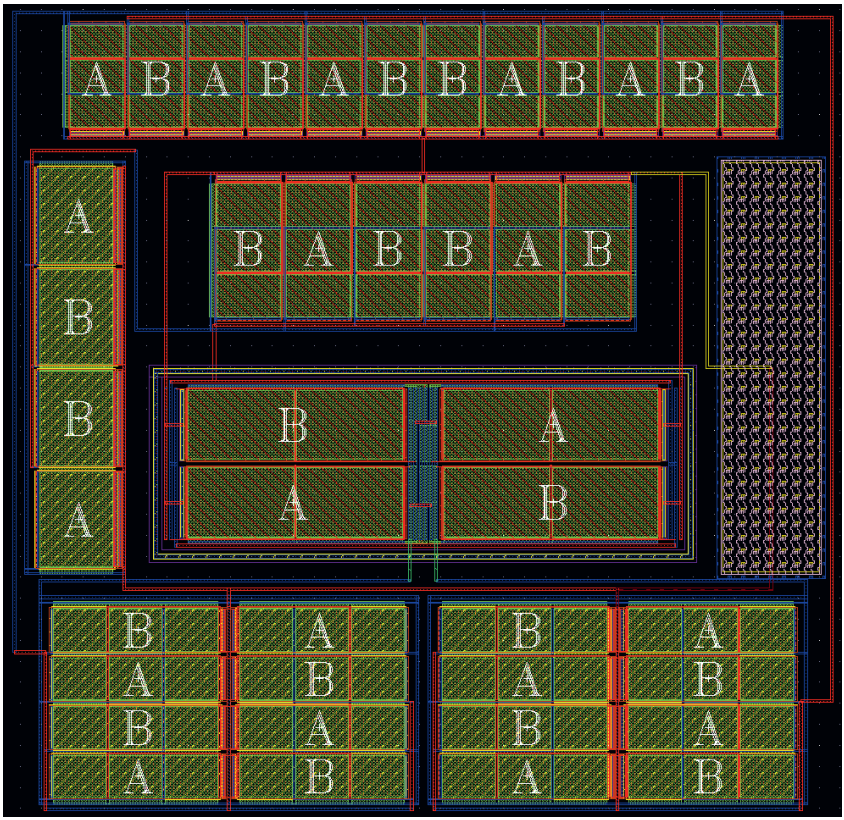


Figure 4.14: Finalized layout of the Symmetric OTA, obtained by SWARM for a 1:1 aspect ratio.

Considering this template-based approach, one might argue that the problem at hand could have been easier solved by modeling it as a *slicing tree* and tackling it with a so-called "floorplan sizing" algorithm (such as [193]) based on *shape functions*.[1] However, this would not just have simplified but trivialized the problem. Instead, translating the template (Figure 4.11) into a set of Relative Placement constraints –as explained above– leaves more degrees of freedom: the final constellation may look like Figure 4.16 (a) –i.e., precisely as in the depiction of Figure 4.11– or could alternatively be structured as in any of the other arrangements from (b) to (j).[2]

[1] A shape function describes the dependency between the width and the height of a layout component.

[2] The amount of potential layout arrangements here is given by the number of modules encountered when traversing the template in vertical direction through its center. Then, the amount of potential arrangements is

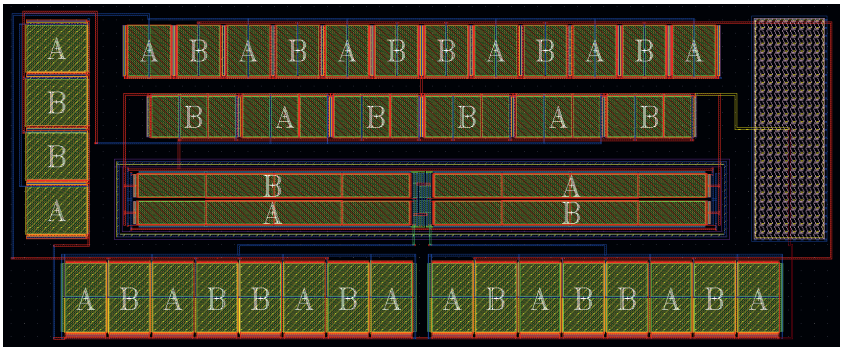


Figure 4.15: Finalized layout of the Symmetric OTA, obtained by SWARM for a 5:2 aspect ratio.

On the one hand, this is fine since it does not impair the desired overall symmetry. Moreover, on the other hand it is even serviceable because it keeps up a greater solution space. And rightly so, as certain solutions that would have been unattainable otherwise do indeed emerge in SWARM: the layouts of Figure 4.13 and Figure 4.14 correspond to that of Figure 4.16 (b) while the outcome of Figure 4.15 reflects the placement of Figure 4.16 (c).

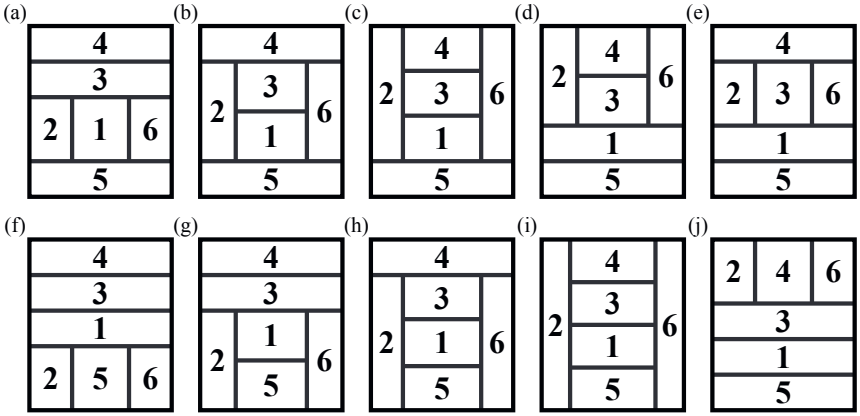


Figure 4.16: Potential layout arrangement alternatives for the Symmetric P-Input OTA circuit.

$\frac{1}{2} \cdot n \cdot (n+1)$: the Gaussian sum formula, which defines the sequence of triangular numbers. In this case, there are $\frac{1}{2} \cdot 4 \cdot (4+1) = 10$ arrangement alternatives.

So, although the overall module arrangement is comparably strictly predefined by the given placement template, SWARM concedes quite some flexibility here. Apart from this, the solution space is above all opened up by the combined variability of the governing modules. Concentrating only on these deformations, the number of mathematically possible layout solutions based on the multiplied module ductiliy *alone* amounts to a total of

$$40 \cdot 56 \cdot 56 \cdot 56 \cdot 112 = 786\,759\,680$$

variants in this example, thus drawing near to a billion potential combinations. Yet (in contrast to a conventional optimization algorithm, for which a larger solution space metaphorically corresponds to a bigger haystack making it more difficult to look for the proverbial needle), the many degrees of freedom are downright helpful for a decentralized approach as pursued here. A valuable conclusion for the topic of this thesis is that formalized design constraints are not necessitated by SWARM to diminish the solution space, but are specifically taken into consideration for their actual purpose: to guarantee the functionality of the layout.

4.3.3 Folded Cascode P-Input Operational Transconductance Amplifier

Applying SWARM to another place-and-route problem, Figure 4.17 presents the schematic diagram of a Folded Cascode P-Input OTA. The six functional units constituting this circuit are: a Differential Pair (1), two simple Current Mirrors (2 and 3, with two and three stages respectively), a Wide-swing Current Mirror (4) consisting of a Bank (4a) and a Cascode (4b), then a three-stage Cascode Current Mirror (5) with a Bank (5a) and a Cascode (5b), and finally also a Blocking Cap (6). As in the example of Section 4.3.2, customizable patterns can by given by the user to control the device interdigitation in the respective governing modules.

Once again, a placement template –see Figure 4.18 (a)– is turned into a set of Relative Placement constraints (b) to achieve overall symmetry, and (also like in the previous example) an Imitation constraint forces the Blocking Cap (6) to mimic the input Current Mirror (2). Just as well, a Guardring constraint is assigned to the Differential Pair (1) here to make SWARM employ a corresponding Guardring PCell. An overview of the circuit's devices and modules is provided in Table 4.9.

The initialization phase is depicted in Figure 4.19: image (a) displays the 61 primitive devices instantiated by calling *Generate from Source* while image (b) shows the situation after imposing appropriate governing modules on these devices and using the `Modgen` command to obtain an arrangement which reflects the given placement template. As in the example of Figure 4.12, this initial constellation serves as the starting point for the self-organization in seven different SWARM runs covering seven different layout zone aspect ratios.

SWARM's layout for an aspect ratio of 2:3 can be seen in Figure 4.20. Altogether, a total of 26 interaction rounds spanning 6 tightening-settlement cycles were required for the emergence

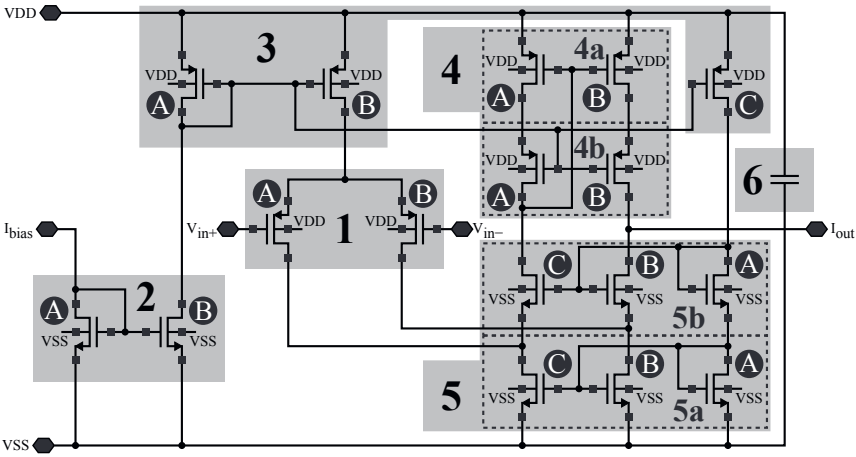


Figure 4.17: Schematic diagram of the Folded Cascode P-Input OTA circuit addressed with SWARM.

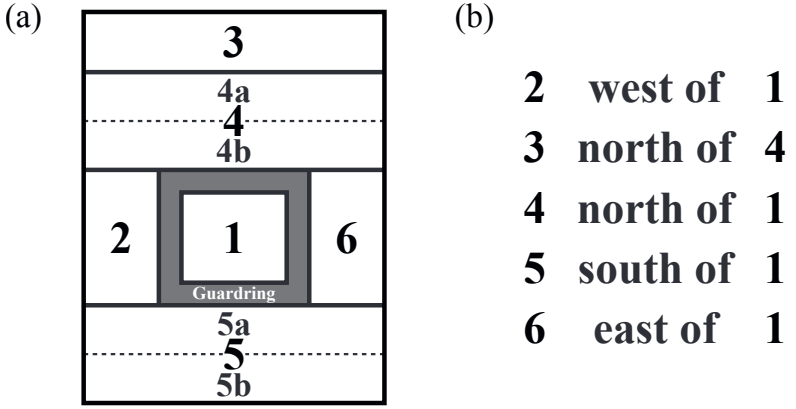


Figure 4.18: (a) Placement template and (b) respective constraints for the Folded Cascode P-Input OTA.

Table 4.9: Overview of the functional units constituting the Folded Cascode P-Input OTA circuit.

Index	Governing Module / Module Association	Device Type	Num. of Devices	Interdigitation Pattern	Ductility (Variants)
1	Differential Pair	PMOS	4	A B / B A (fix)	40
2	Current Mirror	NMOS	6	B A B B A B	64
3	Current Mirror	PMOS	10	B C A C B B C A C B	64
4	Wide-swing Current Mirror	-	-	-	32
└**4a**	Bank	PMOS	8	A B B A A B B A	-
└**4b**	Cascode	PMOS	8	A B B A A B B A	-
5	Cascode Current Mirror	-	-	-	32
└**5a**	Bank	NMOS	12	B A C B A C C A B C A B	-
└**5b**	Cascode	NMOS	12	B A C B A C C A B C A B	-
6	Variator Module	Capacitor	1	-	full variability

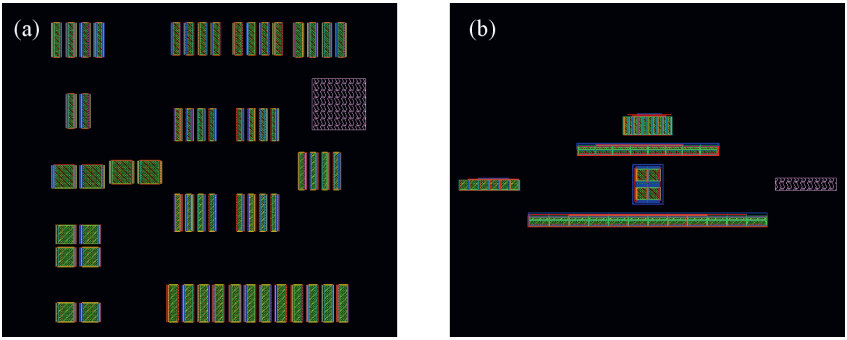


Figure 4.19: OTA layout (a) after device generation and (b) at the end of the initialization phase.

of this outcome. Although the number of rounds per settlement (4, 3, 3, 3, 4, 9) could have been distributed even smoother, the module interaction was exceptionally efficient with respect to the observation that only 388 actions had to be explored for the 105 actions that were eventually performed (the best rate among all SWARM runs carried out for either the Symmetric OTA or the Folded Cascode OTA). Thus, no more than 28 seconds of runtime were measured for the self-organization phase in this example, which is less than the time needed for the initialization plus the finalization (adding up to 41 seconds, which accounts for 69 seconds in total).

The cardinal statistics of this SWARM run, as well as those obtained for the other six runs, are listed in Table 4.10. At large, the self-organization was again quite fluent as the rounds per settlement indicate. And like in the examples of Table 4.8, every kind of action has played

Figure 4.20: Finalized layout of the Folded Cascode OTA, obtained by SWARM for a 2:3 aspect ratio.

its part here in one run or another. Regarding performance, the total runtimes of the listed SWARM runs (including the initialization, self-organization, and finalization phase) are actually shorter than those of Section 4.3.2: even in the worst case (encountered for an aspect ratio of 3:2), SWARM does not take much more than two minutes to produce a compact, constraint-compliant, completely placed and routed layout solution.

Table 4.10: Statistics of the different SWARM runs applied to the Folded Cascode OTA. The abbreviated kinds of actions are: Re-entering (R), Centering (C), Lingering (L), Budging (B), Hustling (H), Evasion (E), Yielding (Y), and Imitation (I).

Asp. Rat.	Rounds per Settlement	Total Rounds	Number of Actions expl.	reje.	perf.	dism.	Perf. Actions (By Kind) R	C	L	B	H	E	Y	I	Total Runtime
1:1	3, 3, 4, 4, 6	20	1283	1163	75	45	1	51	0	7	0	2	0	14	79 sec.
1:2	4, 3, 4, 3, 2, 5, 4, 10	35	1307	1097	135	75	2	102	0	4	0	2	0	25	91 sec.
1:3	6, 3, 3, 3, 3, 3, 3, 4, 4, 8	40	2742	2502	133	107	0	92	0	6	4	5	0	26	117 sec.
2:1	3, 3, 3, 4, 4	17	534	432	49	53	2	30	0	3	0	3	0	11	74 sec.
2:3	4, 3, 3, 3, 4, 9	26	388	232	105	51	1	81	1	1	0	2	0	19	69 sec.
3:2	3, 2, 2, 3, 3, 10, 4	27	2246	2084	97	65	3	58	0	12	1	3	2	18	129 sec.
5:2	3, 3, 3, 4	13	982	904	40	38	1	22	0	7	0	2	0	8	81 sec.

As illustrated in Figure 4.20, SWARM's outcome for a 2:3 aspect ratio precisely realizes the overall module arrangement specified by the placement template (Figure 4.18). The same is true for the 1:1 layout depicted in Figure 4.21, again relying on the cumulative variability of the modules to satisfy the contour of the square layout zone. In the result of Figure 4.22, achieved by SWARM for a 5:2 aspect ratio, the final module constellation represents a slightly different alternate arrangement (corresponding to that of Figure 4.16 (g) with modules 3 and 4 permuted) which is alright because it still attains the overall symmetry enforced by the set of Relative Placement constraints. This is also the case with the SWARM solutions for the other four aspect ratios (as confirmed by the respective screenshots that can be found in the appendix of [1]).

4.3.4 Assessment

As discussed in Chapter 4 of [1], assessing the practical worth of the SWARM methodology for real place-and-route problems should distinguish between its impact on *design productivity* (Section 4.2.1 in [1]) and the degree of achievable *layout quality* (Section 4.2.2 in [1]). The latter is investigated in the following Section 4.3.4.1 while the former will be subject to Section 4.3.4.2.

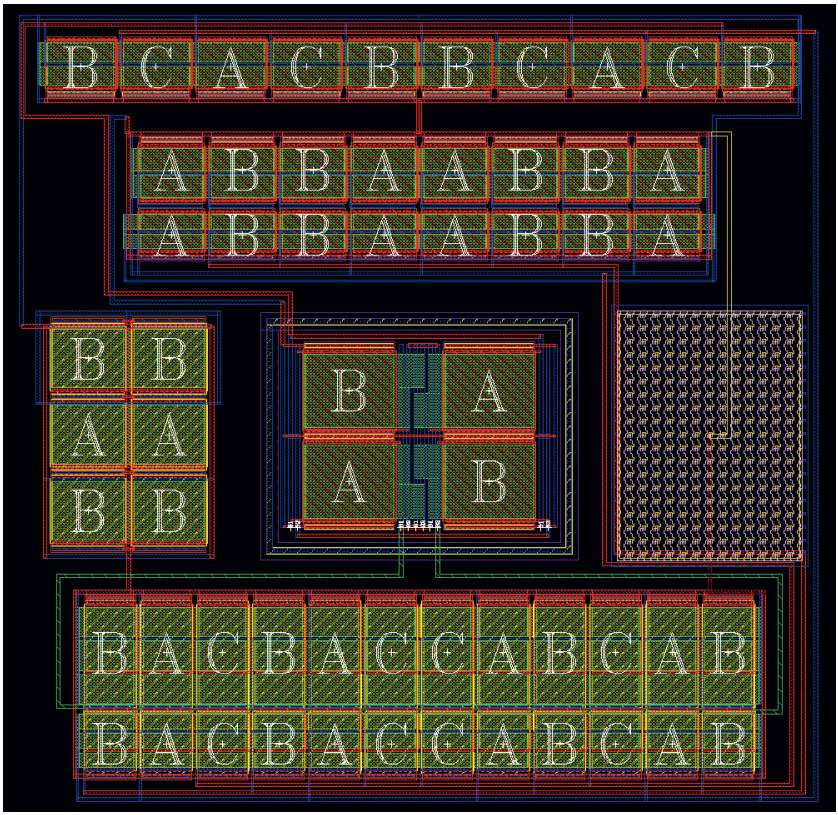


Figure 4.21: Finalized layout of the Folded Cascode OTA, obtained by SWARM for a 1:1 aspect ratio.

4.3.4.1 Assessment Regarding Layout Quality

Above all, a fundamental asset of SWARM is its ability to produce layout results which are completely placed and routed – a merit that already sets the methodology apart from placement-only approaches (Table 3.2 on page 41 of [1]) and mere routing algorithms (Table 3.3 on page 42 of [1]). Thereto, SWARM closely follows the common procedure of manual layout design, tackling the positioning of a module's devices and its intra-module wiring simultaneously to create the inter-module connectivity after the modules have been placed in the desired arrangement. Based on this modular approach, the layouts achieved by SWARM are qualitatively reminiscent of handcrafted solutions, both in terms of *functionality* and *consistency*.

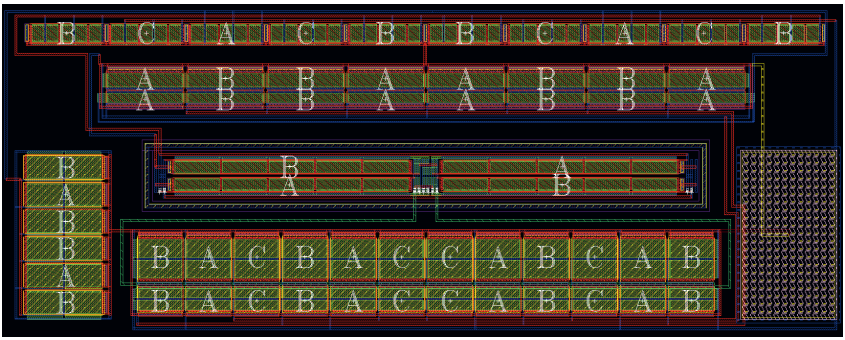


Figure 4.22: Finalized layout of the Folded Cascode OTA, obtained by SWARM for a 5:2 aspect ratio.

Concerning functionality (see Section 4.2.2.1 in [1]), the decisive strength of SWARM is to satisfy all relevant design constraints that would have been taken into account if the layout was created by a human expert. This involves an explicit *and* implicit constraint consideration, thus coming up to the technical aim of this thesis to include both formalized and nonformalized expert knowledge. Regarding the two exemplified OTA circuits, the following constraints have been taken into consideration by SWARM explicitly:[3]

- "Module" constraints imposed on each functional unit (e.g., a DiffPair constraint or a CurrMirr constraint) tell SWARM to manage the respective devices with the designated governing modules and module associations.
- If a governing module provides a dedicated parameter for specifying a custom device interdigitation pattern, then the desired pattern can by formally entered by the user as a string to be explicitly put into effect by the module.
- A Guardring constraint assigned to the Differential Pair effectuates the instantiation of a Guardring PCell around the four transistors, able to incessantly re-adapt itself during the module interaction.
- Using Relative Placement constraints derived from the predefined placement template, the interacting modules are confined in their action exploration via *noncompliance* – one of SWARM's five influencing factors.
- An Imitation constraint forces the Blocking Cap to mimic the transformations of the input Current Mirror, thus supporting the Relative Placement constraints in their aim to achieve overall symmetry.

[3]Constraints marked with an asterisk (*) belong to those addressed in literature (see Section 3.1.1.2 in [1]).

- With the influencing factor *protrusion*, all modules are obliged to get into (and stay within) the contour of the user-defined layout zone. This represents the consideration of a Fixed Outline* constraint.
- Based on SWARM's general idea of a successively tightened layout zone, the modules are explicitly driven towards an increasingly compact arrangement (which effectively implements the optimization goal of area minimization).

If the modules were not bound to a template-based arrangement, some additional constraints can be added to the picture:

- The influencing factor *turmoil* may be used to pursue the optimization goal of wirelength minimization, and –by replacing a connection's *tension* with compressive stress– to separate certain modules from each other.
- Prompting a module to locate itself near the zone border can be achieved with a connection between the module and a (hypothetical) external component, as described in Section 4.2.1). This corresponds to a Boundary* constraint.
- To realize a Distance* constraint, a connection's *relaxation threshold* may be overridden with a maximum or minimum value acting as a hard restriction on the proximity or separation between two modules.

Turning to the internal layout of the governing modules, it goes without saying that their introversive behavior in general can, should, and in practice usually does indeed profit from emulating the best practice design procedures cultivated throughout a design team's history (and informally available as their legacy know-how). In the presented examples, constraints implicitly taken into consideration by the governing modules are:[3]

- If a module does not allow for custom device interdigitation, then the module is inherently bound to a specific interdigitation pattern. For example, the Differential Pair module always sticks to a crosswise AB/BA layout (thus implicitly satisfying a Common Centroid* constraint).
- The devices of a module are cohesively positioned according to a single-row or dual-row Alignment*, also taking care of consistent Orientation* (across or upright). This fulfills the demands of a Symmetry Island* constraint.
- Whenever a module deforms itself into a different layout variant by changing the *number of fingers* in its devices, this is uniformly done for all of these devices to secure that they still have Equal Parameters*.
- In each module, the respective devices are positioned in a highly compact fashion without lacking sufficient space for the wiring. The modules automatically see to an Abutment* that makes use of diffusion sharing.
- By means of the PCell Designer's geometry query language, the modules are able to geometrically determine whether and where their adopted devices feature bulk contacts.

These are then automatically connected by the modules (which may even foster the application of bulk contacts and thus reduce substrate debiasing).

- No wires are routed on top of the devices but exclusively between them. This fundamental maxim reflects an implicit consideration of the constraint type introduced as Blockage*.
- Answering the Layer Limitation* constraint, only metal1 and metal2 wires are employed for every intra-module wiring. Apart from a few exceptions which additionally ask for metal3, this is also true for the inter-module wiring.
- In regard to current-carrying capacity, dedicated module parameters allow to make each *trunk* (lateral wire to connect a row of devices) wider than the perpendicular wires coming from the devices (also known as *pin-to-trunk* routes). This possibility pertains to the Wire Width* constraint.
- Without exemption, only double-cut vias (in remembrance of the Double-Cut Vias* constraint) are used throughout the entire layouts, not just for the connections inside each governing module but also by the Wiring PCells that create the connections between the modules.
- The connections between the modules are not tied to any layer preferences. For that reason, unnecessary layer changes between horizontal and vertical wire segments are avoided, which in turn minimizes the number of required vias.
- To avoid the antenna effect, all Current Mirror modules draw a metal1-only wire to connect their transistor gates with the drain of the reference transistor. On the same account, the Differential Pair module connects its transistor gates as depicted in Figure 3.14 (b) of [1], also using merely metal1.
- The modules generate their internal connections in a well-balanced fashion, evenly distributing the wires around (and –if appropriate– in between) the respective devices. This attains a high degree of symmetry, as well as homogeneity in the wire density.

Comparing this itemization with the state of the art (as surveyed in Chapter 3 of [1]) underlines that the overall amount of constraints taken into consideration in SWARM by far exceeds the constraint coverage achieved with purely optimization-based or purely generator-based automation approaches. Living up to the technical aim of the thesis (as articulated in Chapter 1), this accomplishment –of considering constraints both explicitly and implicitly– plays to the facet of circuit functionality. Beyond that, the SWARM methodology also goes strong in terms of consistency (see Section 4.2.2.2 in [1]):

- Since SWARM relies on well-established module PCells to automate the analog basic circuits that constitute the design as functional units, it effectuates an intuitive approach of functional modularization that the design team is supposed to be familiar with.
- The modularization eases a visual inspection of the layout as it allows to pick a single PCell (e.g., a certain Wiring module) and examine it in isolation. Furthermore, signing off

the layout is disburdened by inspiring trust on the designer's side, rooted in the confidence and experience that many design constraints are implicitly satisfied by the PCells.

- SWARM works completely deterministic, which means that any of its layout solutions can be exactly reproduced if necessary. In conjunction with the observation that a SWARM run typically comprises only a modest number of actions (rather than millions of incomprehensible perturbations), this offers the great opportunity that the run itself can be surveyed by the user and thus visually inspected in real-time.
- The final layouts are neither entirely flat nor utterly secluded. Instead, the procedural powers of the modules are still available such that parametrical adjustments can be made at module level. Even if these adjustments affect the inter-module connections, the Wiring modules can be deleted in order to repeat the finalization phase of the SWARM run.
- A pivotal advantage about context awareness is that even though the given schematic may be flat (i.e., built from primitive devices), modular facilities can be utilized in the layout via governing modules – without sacrificing SDL-conformity, because a distinct one-to-one device correspondence is maintained.[4]
- SWARM has been seamlessly integrated into the design environment (using native Virtuoso features such as *Generate from Source*, the ModGen tool, and –of course– PCells). This implementation merely enhances the design flow instead of replacing it, so it does not introduce any artifacts incompatible with the existing tool chain. If the layout produced by a SWARM run is indistinguishable from a genuine manual solution, further processibility is definitely guaranteed.
- Design iterations due to a re-sizing of the circuit or a PDK update (that involves a revision of the governing modules' master PCells) should be no problem as long as the circuit topology and the design constraints don't change. A new SWARM run, based on the same arguments as before, is supposed to spawn an adequate layout once again. To that end, the arguments of a SWARM run can be stored in a separate file and reloaded the next time.
- As already stated in Section 3.1.2.2 of [1], employing PCells to automate simple modules facilitates re-use of expert knowledge in a way much smarter than through rough-and-ready copy-pasting. Naturally, the same inventory of governing modules can be utilized in different SWARM runs of the same project and in other projects of the same semiconductor technology.
- To address projects in different technologies, the governing modules have been implemented in a way by which technology-specific data (e.g., layer names) is separated from the actual PCell code and instead passed to the PCell via parameters. In the best case,

[4] Difficulties can arise concerning net correspondence because symbolic LVS checkers may be unable to descend into a Wiring PCell and discern its internal connectivity. However, modern SDL flows are beginning to offer suchlike hierarchical support to some degree.

migrating a PCell to another technology is achieved solely by adjusting the technology-specific parameter values without having to touch the PCell's command sequence.

4.3.4.2 Assessment Regarding Design Productivity

Due to the large amount of assessment criteria (see Figure 4.1 on page 69 of [1]), calculations on design productivity are no exact science, but certain statements can be made when distinguishing between different orders of magnitude. For giving estimations on the involved design effort, Table 4.11 defines a vocabulary of temporal quantities from 1' (one minute) to 1Y (one year). For each of these quantities, the table lists the respective time span in minutes since the subsequent calculations also utilize minutes as the basic unit of time. Every quantity represents the labor costs ascribed to one human expert (i.e., a month –1M– in fact denotes a person-month, for example). Where it is appropriate, intermediate values such as 5' (five minutes), 2D (two days), or 2W (two weeks) can be found.

Table 4.11: Temporal quantities used to estimate the design effort. Please note that a day is understood as a working day, a week is understood as a working week, and so on.

Quantity	Meaning	Calculated As	In Minutes
1'	1 minute	-	1
1h	1 hour	60 minutes	60
1D	1 day	8 hours	480
1W	1 week	5 days	2400
1M	1 month	4 weeks	9600
1Q	1 quarter	3 months	28800
1H	1 half-year	2 quarters	57600
1Y	1 year	2 half-years	115200

The subsequent assessment provides a general comparison of the four fundamental layout strategies discussed in this thesis: manual layout design (without any kind of automation), generator-based automation, optimization-based automation, and SWARM. The intention of this comparison is to gauge every strategy's suitability with respect to the *component magnitude* (reflecting the functional complexity of the component being layouted), let it be a *primitive device*, a *simple module*, an *advanced module*, or a high-level *block* (as introduced in Table 2.1 on page 20 of [1]). For each of these component magnitudes, the respective number of occurrences n_C per chip is reckoned in decades from 1000 instances down to 1 instance.

For every layout strategy and component magnitude, regarding one type of component (which may cover an entire circuit class if it does not represent a primitive device), Table 4.12 makes a best-case and a worst-case educated guess on the respective layout design effort. In

both cases, the estimation covers the fixed preliminary effort $\mathit{Eff}_{\mathrm{P}}$ (which incurs prior to the actual design work) and the effort $\mathit{Eff}_{\mathrm{O}}$ per occurrence (i.e., the labor required to layout each individual instance of the component in the design project). Based on these estimations, the total average effort $\mathit{Eff}_{\mathrm{C}}$ per chip in each case is calculated via

$$\mathit{Eff}_{\mathrm{C}} = \frac{\mathit{Eff}_{\mathrm{P}} + \mathit{Eff}_{\mathrm{O}} \cdot n_{\mathrm{C}} \cdot n_{\mathrm{T}}}{n_{\mathrm{T}}} = \frac{\mathit{Eff}_{\mathrm{P}}}{n_{\mathrm{T}}} + \mathit{Eff}_{\mathrm{O}} \cdot n_{\mathrm{C}} \tag{4.2}$$

where n_{T} represents the total number of chips under consideration for a particular semiconductor technology. Concerning the estimation at hand, a value of $n_{\mathrm{T}} = 1$ is assumed first. To give further insight into the figures of Table 4.12, a couple of remarks should be made:

- Financial expenses whatsoever are omitted from this assessment – only effort in the sense of labor costs is taken into account.
- For simple modules, advanced modules and blocks, it is assumed that the primitive devices are readily given as procedural generators.
- As discussed in the context of Figure 3.16 (page 55 of [1]), automating advanced modules –or even blocks– as genuinely procedural generators is virtually impossible since their variability covers a continuous and infinitely huge parameter space. For the sake of comparison, Table 4.12 supposes that it *is* possible – but only if a correspondingly large amount of preliminary effort is spent on their development.
- By the same token, Table 4.12 hypothesizes the availability of optimization algorithms that are able to perform both placement and routing, that can explicitly consider all kinds of placement and routing constraints, and that indeed find constraint-compliant solutions in expert quality. The difficulty with (or nonexistence of) such algorithms is reflected by the exorbitant amount of constraining effort necessary to express all design constraints in a formal fashion.
- It goes without saying that optimization algorithms are deemed to be *not applicable* (n/a) for automating primitive devices, and the same is of course also true for the SWARM methodology. Just as well, SWARM is not applicable to produce layouts of simple modules because such modules already represent the interacting entities in a SWARM run.
- When employing the SWARM methodology to target an advanced module, the preliminary effort for implementing the simple modules that are to be impelled into the interaction may seem quite low. However, much substance can be obtained from the module exemplifications of this thesis (and –much more– from generators that might already be in use within the design team). Furthermore, a lot of effort can be saved if the module implementations are cleverly derived from generic modules for the positioning and the wiring – as done in this thesis (see Table 3.5 on page 54).
- Regardless of whether SWARM is used for an advanced module or an entire block, the preliminary effort is expected to be roughly the same because the cadre of simple modules

is basically always the same. In the latter case, although not realized in the scope of this thesis, the –discretely variable– simple modules are meant to interact with each other and thus form advanced modules (as in the examples of Section 4.3.2 and Section 4.3.3) while these –fully variable– advanced modules interact with each other on the next-higher hierarchy level (like the floorplan blocks in the examples of Section 4.2). To that end, only the utilization effort during design is expected to become greater (as indicated in Table 4.12).

Table 4.12: Estimated layout design effort for different layout strategies and component magnitudes.

	Primitive Device		**Simple Module**		**Advanced Module**		**Block**	
Occurrences per Chip	1000		100		10		1	
Best Case or Worst Case?	Best Case:	Worst Case:	Best Case:	Worst Case:	Best Case:	Worst Case:	Best Case:	Worst Case:
Effort with Manual Layout Design								
Preliminary Effort	0’	0’	0’	0’	0’	0’	0’	0’
Effort per Occurrence	10’	1h	1h	1D	1D	1W	1W	1M
Total Effort per Chip	10000’	60000’	6000’	48000’	4800’	24000’	2400’	9600’
Effort with Generator-based Automation								
Preliminary Effort	1h	1W	1W	1M	2M	1H	1Q	1Y
Effort per Occurrence	1’	5’	5’	20’	1h	1D	1D	1W
Total Effort per Chip	1060’	7400’	2900’	11600’	19800’	62400’	29280’	117600’
Effort with Optimization-based Automation								
Preliminary Effort	n/a	n/a	1h	1D	1h	1D	1h	1D
Effort per Occurrence	n/a	n/a	1D	1W	2D	1M	2W	1Q
Total Effort per Chip	n/a	n/a	48060’	240480’	9660’	96480’	4860’	29280’
Effort with SWARM Methodology								
Preliminary Effort	n/a	n/a	n/a	n/a	2W	2M	2W	2M
Effort per Occurrence	n/a	n/a	n/a	n/a	1h	1D	1h	1W
Total Effort per Chip	n/a	n/a	n/a	n/a	5400’	24000’	4860’	21600’

Grouped by the respective component magnitude, the total layout design effort per chip (covering both the best-case and the worst-case estimation) for every layout strategy is displayed in the chart of Figure 4.23 on a logarithmic scale. From this depiction, the following conclusions can be drawn:

- Considering a primitive device, it is generally advisable to implement it as a procedural generator. In regard to the estimation made here, even the worst-case effort on the generator’s side is lower than the best-case effort in manual design. This reckoning underlines

why it is common practice to utilize PCells at device level. Only in cases where a primitive device is assumed to be used quite rarely, it may be favorable to put up with manual polygon pushing.

- For a simple module, generator-based automation can basically be expected to outrival manual layout design. However, the situation is not as palpable as with primitive devices. Whether automating a simple module as a procedural generator will presumably pay off in the end, needs to be predetermined on a case-by-case basis within the respective design team.
- Optimization-based automation is the least appealing layout strategy on the level of simple modules where satisfying the relevant design constraints is a duty that expert layout engineers have off pat. This suggests to pass over the huge constraining effort for an optimization algorithm in favor of manual layout design or –maybe even better– a procedural generator.
- When the component magnitude reaches that of an advanced module, the effort for generator-based optimization already escalates into uneconomical regions. In contrast, optimization algorithms become more rewarding, but still they can beat manual layout design only in singular cases. That is why advanced modules are mostly done in a manual fashion – so far.
- Advanced modules are where the SWARM methodology may achieve substantial benefit for design productivity. If only one circuit class is targeted, the effort is –according to the experiences from this thesis– at least comparable to manual layout design. The true benefit unfolds when more than one circuit class is automated via SWARM because the set of governing modules –and thus the preliminary effort– is basically the same. Figure 4.23 illustrates the best-case estimation supposing that five circuit classes are covered: the preliminary effort of two weeks (ten working days) is virtually cut down to two days per circuit class, resulting in a total effort of $\mathit{Eff}_{\mathrm{P}} + \mathit{Eff}_{\mathrm{O}} \cdot n_{\mathrm{C}} = 2 \cdot 480' + 60' \cdot 10$ which amounts to 1560 minutes.
- Regarding a high-level layout block, optimization-based automation is thought to overtrump procedural generators as these become increasingly inappropriate when ascending towards larger component magnitudes. However, manual layout design is still unrivaled here (again except certain isolated cases, if at all). Inhowfar SWARM can compete for design productivity here, is to be investigated in future works. Following the projected numbers of Table 4.12, Figure 4.23 shows that SWARM is probably on a par with optimization-based automation if only one circuit class is concerned. As in the case of advanced modules, SWARM's benefit rises with the number of circuit classes being covered and may thereby noticeably outplay all of the other layout strategies. The illustrated best-case estimation for five circuit classes is $2 \cdot 480' + 60' \cdot 1$ (= 1020 minutes).

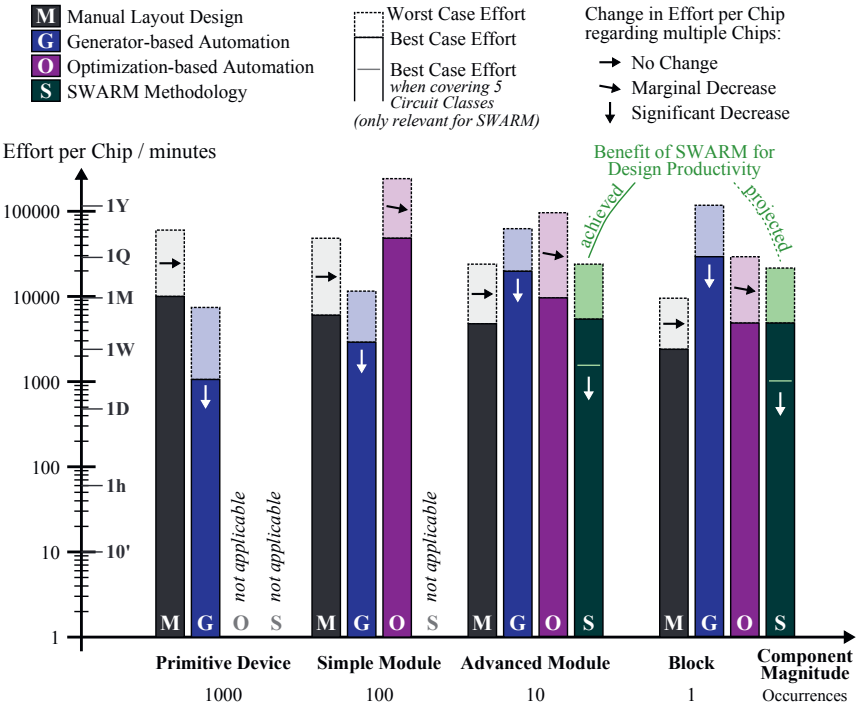


Figure 4.23: Depiction of the various layout design efforts estimated in Table 4.12.

While the estimations of Table 4.12 contemplated only one single chip, it is also interesting to analyze how the layout design effort does change when multiple chips in the same semiconductor technology can profit from a certain layout strategy. For values of $n_T > 1$, a look at equation 4.2 easily reveals what happens to the total average effort per chip: the greater the preliminary effort Eff_P is in relation to the effort Eff_O per occurrence, the larger the benefit is for design productivity when more than one chip is taken into account. This fundamental law is reflected by the differently oriented arrows in Figure 4.23, indicating the respective change in effort per chip.

With respect to manual layout design, there is no change in effort per chip for $n_T > 1$ since there is no preliminary effort involved. Concerning generator-based automation, Eff_P is much greater than Eff_O, which redounds to a significant decrease in design effort when the procedural generators are utilized throughout multiple chips. By contrast, the decrease in design effort is only marginal in the case of optimization-based automation due to the large constraining effort during design, which towers over the preliminary installation effort. With SWARM, the labor

again incurs much more on the predevelopment side rather than during an IC project, so the design effort per chip significantly decreases with every chip that is being targeted.

Regarding the degree of achievable layout quality (discussed in Section 4.3.4.1) as well as the good performance in terms of design productivity (examined in this Section 4.3.4.2), it is fair to say that the accomplished work on SWARM so far pressed the right buttons to successfully master the practical ambition of this thesis (as formulated in Chapter 1). To illuminate the scientific achievement of the presented work, the subsequent Chapter 5 puts the SWARM methodology in the greater context of a "higher-level" design flow addressing both schematic design and layout design.

Chapter 5

Towards a Holistic Design Flow on Module Level

From time to time you should step back from yourself like a painter from his painting.
Christian Morgenstern (German poet)

This chapter takes a step back from the developed SWARM methodology and frames its role inside the bigger picture of an improved design flow which seamlessly covers both of the two major design steps in IC design: schematic design and layout design. For this purpose, Section 5.1 touches on a couple of cognate topics accompanying SWARM in the pursuit of that holistic long-term vision. Then, Section 5.2 throws a glance at the scientific value of SWARM, considering the sketched-out EDA roadmap towards a novel *bottom-up meets top-down* automation paradigm.

5.1 Cognate Topics Across the Three Different Design Domains

Referring to the well-known *Y diagram* [194], the IC design flow can be said to traverse three different *design domains* denoted as the *behavioral domain* (which deals with the general function of the circuit), the *structural domain* (that the schematic representation of the circuit pertains to), and the *physical domain* (wherein the detailed layout geometries of the circuit are to be described). As illustrated in Figure 5.1, the step of schematic design is thus set between the behavioral domain and the structural domain while the step of layout design accounts for the transition from the structural domain to the physical domain.

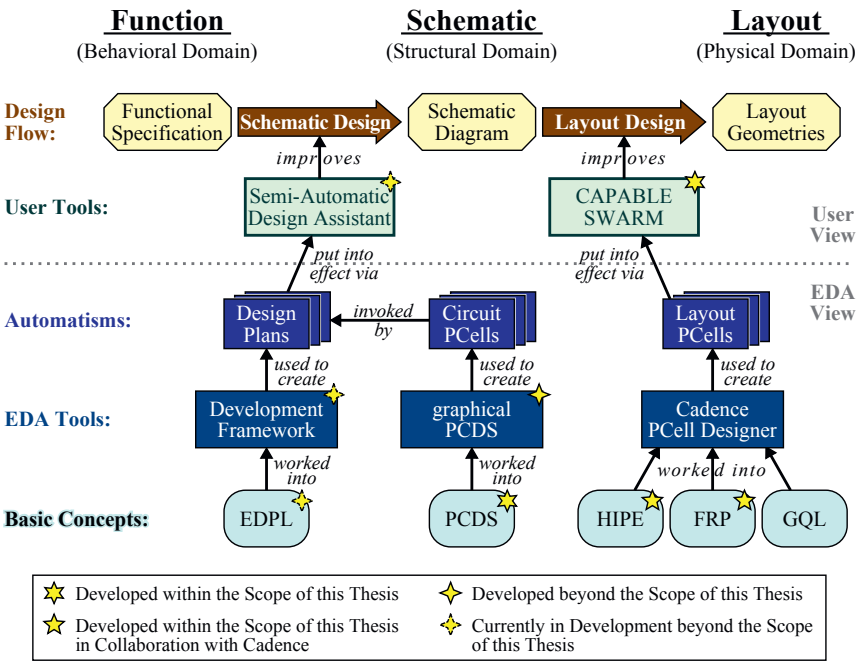


Figure 5.1: SWARM and its partner subjects in pursuit of a holistic design flow on module level.

In former times, when procedural generators did not reach beyond the level of primitive devices, the design flow exhibited a hierarchical break between the behavioral and the structural domain because creating flat schematics and flat layouts does not match the designers' way of thinking in functional units (i.e., modules) when the behavior of a circuit part is concerned. This inconsistency involves a loss of information about the purpose of a module and its respective design requirements. To prevent such discontinuities, this thesis has been conceived as part of an envisioned design flow on module level that holistically proceeds throughout all of the three design domains. Tackling both of the two intermediary design steps, Figure 5.1 displays related works that have been (or are being) developed within (or beyond) the scope of this thesis.

For historical reasons, these works are now described in backwards order (i.e., converse to the design flow), from the physical domain (Section 5.1.1) through the structural domain (Section 5.1.2) to the behavioral domain (Section 5.1.3).

5.1.1 Works Concerning the Physical Domain

The SWARM methodology (and its host tool CAPABLE) were initiated upon the success story of layout PCells by which generator-based automation began to pioneer in the physical domain. Especially with the advent of the Cadence PCell Designer tool, the potential of procedural generators –and likewise the need for further concepts to keep up with the new possibilities and challenges– became increasingly obvious. One fundamental pillar that the PCell Designer has been built upon from the beginning is the already-mentioned *geometry query language* (GQL).

Along with a couple of other enhancements, the idea of *FR PCells* (FRP) –which served as the conceptual trailblazer for SWARM (see Section 3.2.1)– was invented in collaboration with Cadence during the work on this thesis. Another basic concept added to PCell Designer is *Hierarchical Instance Parameter Editing* (HIPE) [195], which enables a parametrical customization of module PCells throughout arbitrary levels of subhierarchy (as demonstrated in [196]). The scientific earning of HIPE is that the PCell parameters need not be entirely predefined by the master PCell, but are dynamically determined depending on the contents of the specific PCell instance.

5.1.2 Works Concerning the Structural Domain

To accommodate the ascent of layout PCells in the physical domain, attention has come to the development of corresponding circuit PCells in the structural domain. As already explained in Section 3.1.2.5 of [1], a circuit PCell in fact consists of two PCells: a schematic PCell and a symbol PCell. With the intention of facilitating the implementation of such PCells, a domain-specific language named *Parameterized Circuit Description Scheme* (PCDS) has been worked out in the scope of this thesis. Apart from providing high-level *shape commands* and a *relative placement* notation, PCDS realizes a *dual interpretation* technique by which both the schematic PCell and the symbol PCell can be compiled from one single PCDS description. As outlined in [197], PCDS reduces the amount of necessary code lines by a factor of 50 on average.

Despite all these merits, PCDS still relies on a textual implementation of circuit PCells which is far from the mentality of circuit design experts accustomed to visually drawing schematic diagrams. Hence, the winning pitch here is to offer a graphical PCell programming approach in the fashion of PCell Designer. This has been prototypically achieved through *graphical PCDS* (gPCDS) [198], a circuit PCell development tool realized in a contemporaneous research project. Meanwhile, that approach has aroused so much interest that the basic concepts of PCDS are being integrated into PCell Designer to deliver a professionalized version of gPCDS. A first glimpse on these new features has been caught in [199].

5.1.3 Works Concerning the Behavioral Domain

To complete the route towards a holistic design flow on module level, the next logical measure is to transfer the idea of procedural automation over to the behavioral domain and thus address the step of schematic design. This step is about taking a functional circuit specification (mainly expressed via performance parameters) and producing a sized schematic (that meets the functional specification). For this purpose, procedural automation can be employed to handle the inherent tasks (finding an appropriate circuit topology, choosing the device types, and setting the device dimensions) through a so-called "expert design plan", which is a sequence of instructions mimicking a human expert's recipe-like design proceeding. In [200], the feasibility of doing so has been exemplarily demonstrated on the OTA circuit class.

As in the physical domain and the structural domain, the grand bargain for EDA is not to develop such design plans but rather to devise adequate tools for their development. In this spirit, an *Expert Design Plan Language* (EDPL) is currently being elaborated [201] as a companion concept of PCDS, so to speak. Based on EDPL, it is furthermore scheduled to realize a sophisticated development framework (serving as the logical counterpart to gPCDS and PCell Designer) for the creation of expert design plans. This approach becomes well-rounded if an expert design plan is enabled to invoke a corresponding circuit PCell (covering all topologies of the respective circuit class) that generates the schematics needed for simulation. On the user's side, a GUI-based tool is meant to bring an expert design plan to life (similar to a cassette deck playing a tape). Since such a tool relies on close interaction with the user, it is presently labeled as a *Semi-Automatic Design Assistant* (SADA).

5.2 The Scientific Value of SWARM: Meeting Bottom-up With Top-down

Beside the practical merits of SWARM, strategically more valuable is the academic achievement of illustrating how the two converse automation paradigms denoted as *bottom-up* and *top-down* can be made to work together. Interestingly, SWARM's journey towards this vision of a novel *bottom-up meets top-down* philosophy followed the five successive steps articulated in Section 3.2.3 of [1]:

(1a) The research project behind SWARM was encouraged by the development (and positive reception) of the PCell Designer tool, for which a lot of technical input –stemming from the work on SWARM– has been conveyed to Cadence.

(1b) The PCell Designer facilitated a swift implementation of the procedural generators for the simple modules (analog basic circuits) as layout PCells, utilizing parameters to cover the module-specific degrees of freedom.

(1c) Building on the idea of context awareness, the CAPABLE framework has been realized to incorprate the implemented layout PCells into the design flow as governing modules that are applied to primitive devices in the layout.

(2) The algorithmic way of optimizing a layout has been worked into the conception of a control organ which exerts pressure by repeatedly tightening the available layout space to enforce increasingly compact layouts indirectly.

(3) Devising a decentralized system, where constraint-compliant layout outcomes emerge from the self-organization of interacting modules (allowing the modules to consider their innate design requirements bottom-up and simultaneously satisfying the top-down demands of the control organ), represent the glue for uniting the two automation paradigms and let bottom-up meet top-down.

As already indicated, SWARM's ability to take both formalized and nonformalized expert knowledge into account –via explicit and implicit constraint consideration– is reflected in joining the cardinal strengths of optimization-based automation and generator-based automation: the versatility of the approach (i.e., not having to preconceive the entire solution space in advance) and the resulting layout quality (i.e., not having to specify every single detail through formal constraints). To provide a final overview, Table 5.1 summarizes the major characteristics of SWARM, as opposed to optimization algorithms and procedural generators.

From today's perspective, SWARM pioneers an entirely new field for EDA expected to blossom with many more innovative approaches that will further contribute to *bottom-up meets top-down* automation. Many questions –like whether upcoming works will also facilitate an interplay of bottom-up procedures with genuine top-down algorithms in their traditional form– are subject to future research and shall therefore be delegated to upcoming generations of EDA enthusiasts. Nevertheless, some embryos of big ideas –such as an interaction of context-aware, "smart" Wire PCells that perform a self-intelligent algorithmic pathfinding– have already come to mind.

Table 5.1: Major characteristics of optimization algorithms, procedural generators, and SWARM.

Automation Strategy	**Optimization Algorithm**	**Procedural Generator**	**SWARM Methodology**
Circum-scription	Problem-Solving Routine	Layout-Creation Script	Multi-Agent System
Working Principle	Loop of Exploration and Evaluation	Straight Sequence of Design Commands	Self-Organization of Interacting Modules
Expert Knowledge	Formalized	Nonformalized	Formalized and Nonformalized
Constraint Consideration	Explicit	Implicit	Explicit and Implicit
Designated Output	Layout Solution	Layout Result	Layout Outcome
Solution Policy	Solution Found by Optimization Loop	Solution Preconceived by Human Expert	Solution Emerges from the Interaction
Behavioral Determinacy	Unpredictable, often Randomized	Deterministic and even Predictable	Deterministic but not Predictable
Cardinal Strength	Versatility of the Automatism	Quality of the Resulting Layout	Autom. Versatility and Layout Quality
Automation Paradigm	Top-Down	Bottom-Up	Bottom-Up meets Top-Down

Chapter 6

Summary and Outlook

"Begin at the beginning," the King said very gravely,
"and go on till you come to the end: then stop."
Lewis Carroll: Alice in Wonderland

This thesis presents a new methodology for layout automation in the design of analog integrated circuits, referred to as *Self-organized Wiring and Arrangement of Responsive Modules* (SWARM).

While digital IC design has already been following highly automated flows for layout synthesis since the 1980s, many analog layout automation approaches do not find evident industrial acceptance so far, even though a couple of commercial tools –primarily focused on *placement* and *routing*– have arisen from the past three decades of EDA. The basic reason for this circumstance is that –in contrast to the *quantitative complexity* (More Moore) in the digital domain– the functional diversification of analog circuits and the continuous nature of its signals makes the analog design problem a matter of *qualitative complexity* (More than Moore). This involves many diverse and concurrent functional *constraints* (design restrictions and optimization goals) that need to be considered simultaneously.

Until today, thorough constraint consideration can only be satisfyingly achieved through a laborious and largely manual design flow, drawing heavily upon the *expert knowledge* of human layout engineers. For that reason, the presented methodology particularly concentrates on a comprehensive incorporation of expert knowledge into the automation. In real design flows, expert knowledge is communicated in *formalized* and *nonformalized* ways, which corresponds to an *explicit* and *implicit* consideration of constraints, respectively. Since both is required to solve the analog layout problem in its entirety, the technical aim of the SWARM methodology is to devise a new automation approach that supports both an explicit *and* implicit consideration of constraints.

For that purpose, this thesis attempts to combine two basic automation strategies: *optimization algorithms* (which can consider constraints only explicitly) and *procedural generators* (able to consider constraints only implicitly). As a scientific look reveals, these two automation strategies follow two fundamentally different *automation paradigms* denoted as *top-down automation* and *bottom-up automation*. The asset herein is that their respective strengths and weaknesses complement each other, which suggests that a coalescence of *bottom-up meets top-down* has much more potential than optimization-based or generator-based approaches alone. However, the scientific challenge is posed by the question if and how the two different automation paradigms can be brought together.

To address this challenge, SWARM pursues a very interdisciplinary approach building upon the idea of *decentralization* and touching various fields related to *multi-agent systems*. Simple *analog basic circuits* are realized as autonomous, mobile, self-interested, parameterized layout modules that interact with each other as agents inside an increasingly tightened layout zone. By provoking a flow of *self-organization*, a compact and constraint-compliant layout *outcome* is meant to emerge from the module interaction, reflecting the phenomenon of *emergence* as can be found in nature. SWARM pays respect to several principles of self-organization such as operating near the *edge of chaos* and promoting *synergy*, and draws parallels to several other types of decentralized systems subsumed under the term *artificial life* (e.g., cellular automata and agent-based models of collective motion). The SWARM methodology consists of three core concepts that are correlated with each other:

Responsive modules realize the simple analog basic circuits in the layout (e.g., Current Mirrors and Differential Pairs). They are implemented as procedural generators whose natural *introversive behavior* is enhanced with an *extroversive behavior*. In short, the major traits are:

- The responsivity is achieved through an *interface fabric* that equips the modules with the ability to read and modify their design context.
- Based on this *context awareness*, a responsive module can manage other components in the layout as a *governing module* (e.g., positioning the components and wiring them).
- Governing modules can be joined and/or imposed onto each other to build hierarchical *module associations*.
- Each module covers a certain amount of *intrinsic variability* (spanning the module layout's *degrees of freedom*) which –in the case of a hierarchical module association– evaluates into a *cumulative variability*. This variability is then exploited during the module interaction.

Module interaction refers to the concept behind the decision-making by which the responsive modules interact with each other as *participants* during the self-organization phase of a

SWARM run. Every participant adheres to a common *action scheme* consisting of four *measures*:

- Assessing the participant's *condition*, which depends on five *influencing factors* (interference, turmoil, protrusion, wounds, and noncompliance).
- Perceiving its *free peripheral space* (i.e., the –presumably– vacant, rectangular area around the participant).
- Exploring and evaluating possible *actions*, which involves nine native actions (such as *Budging* and *Swapping*) but may also revert to custom actions (e.g., *Imitation*).
- Executing the *preferred* action (or staying idle) according to a distinct *comparison metric*. By executing an action, the participant can move, rotate, and deform itself (i.e., assume another layout variant as featured by its cumulative variability).

Interaction control is exerted by a control organ responsible for carefully steering the module interaction towards the desired outcome. This is done in an indirect way, by successively modifying the size of the layout zone's outline:

- Beginning with an initial constellation of *primitive devices* (upon which the governing modules are instantiated), the user-defined layout zone is set and *enlarged* to a significantly greater size (that easily accommodates all modules).
- Then, several *rounds* of interaction are performed until all participants have *settled* (i.e., no participant has executed an action within the same round).
- If all participants are in a valid location, the interaction control organ *tightens* the layout zone according to a certain *tightening policy* (in order to induce another *settlement*).
- Multiple *tightening-settlement cycles* are performed until the layout zone reaches the user-defined size (or the participants fail to settle in a viable constellation).
- Throughout the interaction, solid and volatile *comfort padding* can be wrapped around a participant. The latter changes with the zone size in favor of a fluent self-organization. The former preserves a fix amount of layout space that can be used by an inter-module routing step to complete the layout in a subsequent finalization phase.

In view of the current implementation, applying the SWARM methodology to a plain placement problem has shown that even *globally optimal* outcomes can emerge from the module interaction. Contemplating that the responsive modules do not cooperate (but compete) with each other, and that the modules do not survey the problem as a whole (but only have a limited viewpoint and selfishly pursue –above all– their personal desires), this observation is quite remarkable and substantiates the decentralized self-organization approach taken by SWARM.

With the power to support *full variability* (whereby the aspect ratio of a layout module – representing a black box– can change within a continuous range) as well as nonrectangular (but

rectilinear) outlines of the layout zone, SWARM is also suitable for the task of *floorplanning*. Therein, SWARM not only minimizes the total layout area but also the distances of floorplan blocks connected with each other or to the periphery. Two floorplanning examples have been given.

The primary purpose of SWARM –i.e., tackling practical *place-and-route* problems– has been demonstrated on a Symmetric P-Input OTA and a Folded Cascode P-Input OTA circuit. The outline of the available layout space as well as the basic module arrangement (predefined via a placement template) are considered explicitly during the interaction while each module simultaneously takes care of its innate design requirements implicitly. Another mentionable attribute of SWARM is that it refrains from randomization and works completely *deterministic*.

Producing various OTA layouts with different aspect ratios depicts that SWARM combines the versatility of optimization algorithms with the layout quality of procedural generators. For circuits of such magnitude (denoted as *advanced modules*), it is of practical interest that SWARM not only goes strong with regard to layout quality but also surpasses generator-based and optimization-based automation in terms of *design productivity*. In particular, SWARM's benefit increases when targeting multiple different circuit classes, because the basic set of governing modules is largely the same and thus profits from *re-use*.

By teaming the *bottom-up* capabilities of procedural generators with the *top-down* perspective of algorithmic optimization, SWARM's multi-agent approach represents one veritable way of merging the two automation paradigms towards a *bottom-up meets top-down* design flow. With this philosophy in mind, the presented SWARM methodology paves the way to a novel branch of automation, inspiriting entirely new opportunities for future works in EDA research and development.

Research work on SWARM continues as it is meant to become part of a holistic IC design flow on module level, from the functional circuit specification across the generation of a sized schematic diagram to the creation of the physical layout (thus covering the three different *design domains* of the well-known *Y diagram*). Convinced by the results already obtained so far, SWARM itself is about to be integrated into an industrial design flow for ASICs in automotive applications. Further conceptual enhancements of SWARM that have been envisaged are:

- adaptive tightening policies (correlated with the fluency of the self-organization),
- multiple concurrent interaction control organs (handling different circuit parts),
- hierarchically nested interaction flows (for tackling higher-level layout blocks),
- governing modules with learning aptitude (to train expedient interaction maneuvers),
- improving performance and convergence (runtime reduction and independence of start conditions),
- parallelization of module activity via multi-threading (quite obvious for a multi-agent approach),

- user intervention to facilitate real-time human-machine collaboration (as pursued in Industry 4.0).

This continuation of the work on SWARM is strategically motivated by the belief that –in the long run– *bottom-up meets top-down* approaches may turn out to be one essential key for finally closing the persistent *automation gap* in analog layout design.

Vocabulary

One should use common words
to say uncommon things.
Arthur Schopenhauer (German philosopher)

Mathematical Operators

Calculation	
$\sum$	sum
$\prod$	product
$\lvert \ldots \rvert$	absolute value of a scalar
$\lfloor \ldots \rfloor$	floor (rounding down a scalar)
$\lceil \ldots \rceil$	ceiling (rounding up a scalar)
$\min$	smallest of the given values
$\max$	greatest of the given values
Set Theory	
$\{\ldots\}$	set brackets
$\emptyset$	empty set
$\in$	membership
$\notin$	non-membership
$\subseteq$	subset
$\subset$	proper subset
$\supseteq$	superset
$\supset$	proper superset
$\cap$	set intersection
$\cup$	set union
$\setminus$	set complement

$\triangle$	symmetric difference
$\times$	Cartesian product
$\lvert \ldots \rvert$	cardinality of a set
Logic	
$\forall$	universal quantification
$\exists$	existential quantification
$\nexists$	non-existential quantification
$\exists!$	uniqueness quantification
$\wedge$	logical conjunction (AND)
$\vee$	logical disjunction (OR)
$\oplus$	exclusive disjunction (XOR)
$\Leftrightarrow$	if and only if
$\square$	quod erat demonstrandum (end of proof)
Thesis-specific	
$\diamond$	commutative order of two morphisms
$\triangleright$	noncommutative order of two morphisms
$\succ$	succession of two morphisms
$\Leftarrow$	mapping from parameter domain to layout variability
$\dotplus$	set plus single element
$\dotminus$	set minus single element

Geometrical Operators

Domain	
$\mathbb{U}$	entire layout plane (geometrical *universe*)
$\varnothing$	geometrical void ($\varnothing = \overline{\mathbb{U}}$)
Membership	
$G_1 \sqsubseteq G_2$	G_1 is enclosed by G_2 (geometrical inclusion)
$G_1 \sqsubset G_2$	G_1 is properly enclosed by G_2 (strict geometrical inclusion)
$G_1 \sqsupseteq G_2$	G_1 encloses G_2 (geometrical containment)
$G_1 \sqsupset G_2$	G_1 properly encloses G_2 (strict geometrical containment)
Construction	
$G = ((\check{x}, \check{y}), (\hat{x}, \hat{y}))$	rectangle from south-western vertex $(\check{x}, \check{y})$ to north-eastern vertex $(\hat{x}, \hat{y})$
$G = (N_1, N_2, \dots)$	polygonal shape, given as a polygonal chain with sequential nodes $N_1, N_2, \dots$
$G_1 \sqcap G_2$	intersection of G_1 and G_2 (geometrical AND)
$G_1 \sqcup G_2$	union of G_1 and G_2 (geometrical OR)
$G_1 \hookleftarrow G_2$	relative complement of G_2 in G_1 (geometrical difference)
$\overline{G}$	absolute complement of G ($\overline{G} = \mathbb{U} \hookleftarrow G$)
Outline Functions	
$\vdash G$	horizontal coordinate of the leftmost point in G
$\dashv G$	horizontal coordinate of the rightmost point in G
$\top G$	vertical coordinate of the uppermost point in G
$\bot G$	vertical coordinate of the lowermost point in G
$\boxdot G$	rectangular bounding box of geometrical shape G
$\boxdot \mathcal{G}$	rectangular bounding boxes of geometrical shapes $\mathcal{G}$
$\circledcirc_\varepsilon G$	enlargement of G's contour in the four cardinal directions by ε (*grow operator*)
$\circledast_\xi G$	relative shrinking of G's contour by a factor of ξ (*contraction operator*)
Measurement	
$\leftrightarrow G$	width of G
$\updownarrow G$	height of G
$[G]$	area of G
$\lvert G \rvert$	number of vertices of G
$\overline{L_1 L_2}$	Euclidean distance between two points L_1 and L_2

Symbols

α	angle in the orientation of a design component (counterclockwise)	(p. 46)
a	general index	(p. 44)
A	adoption process of a governing module	(p. 36)
Ab	absorption operation of a module's adoption process	(p. 37)
Ad	adaptation operation of a module's adoption process	(p. 37)
Am	amendment operation of a module's adoption process	(p. 37)
As	assimilation operation of a module's adoption process	(p. 37)
β	exponentiation base in a logistic regressive tightening policy	(p. 135)
b	general index	(p. 44)
B	bounding box around the part of a participant inside the layout zone	(p. 75)
$\mathbb{B}$	Boolean domain $\{0, 1\}$	(p. 73)
γ	number of clashes between two participants	(p. 61)
c_s	solid comfort padding amount	(p. 147)
c_ξ	volatile comfort padding factor	(p. 148)
c_v	volatile comfort padding share	(p. 148)
C	connection between two participants	(p. 62)
$\mathcal{C}$	set of connections of a participant	(p. 65)
d	ductility of a design component	(p. 54)
Δ	uppercase delta (absolute difference)	(p. 79)
δ	lowercase delta (relative change)	(p. 101)
D	deformation morphism of a module transformation	(p. 43)
$\mathbb{D}_{P,I}$	parameter domain of an input parameter I for a module P	(p. 51)
ε	arbitrarily small positive number	(p. 77)
ϵ	absolute edge displacement due to a tightening or enlargement of the layout zone	(p. 112)
e	emphasis of a connection between two participants	(p. 62)
E	edge of a rectangle or polygon	(p. 75)
$\mathtt{f}$	number of fingers parameter of a procedural generator for a MOS transistor	(p. 51)
f	number of fingers value of a procedural generator instance for a MOS transistor	(p. 51)
F	constellation frame around all participants	(p. 111)
g	(introversive) behavior of a procedural generator	[1]
G	geometrical shape in the layout design	(p. 60)
$\mathcal{G}$	set of geometrical shapes	(p. 60)
η	number of connections of a participant	(p. 62)
$\bar{h}$	horizontal flipping in the orientation of a design component	(p. 46)
h	height of an object	(p. 83)
H	hard constraint (strict confinement) imposed on a set of constraint members	(p. 73)
$\mathcal{H}$	set of hard constraints imposed on a set of constraint members	(p. 74)
ϑ	aversion between two participants	(p. 60)
θ	tension contributing to the turmoil of a participant	(p. 65)

Θ	turmoil of a participant	(p. 65)
i	general counter	(p. 55)
I	input parameter of a procedural generator	[1]
$\mathcal{I}$	set of input parameters of a procedural generator	[1]
j	general counter	(p. 55)
$\varkappa$	kickoff enlargement multiplier (desired zone size divided by intrinsic minimum)	(p. 113)
k	general counter	(p. 55)
K	corridor for a participant's perception of its free peripheral space	(p. 75)
$\mathcal{K}$	set of corridors for a participant's perception of its free peripheral space	(p. 75)
λ	leeway coefficient for the calculation of a connection's relaxation threshold	(p. 63)
ℓ	layout layer in a semiconductor technology	(p. 60)
Λ	set of layout layers in a semiconductor technology	(p. 60)
l	length (Euclidean distance between two points)	(p. 64)
L	location (x, y) of a design component / point in the layout plane	(p. 46)
$\mathcal{L}$	set of locations	(p. 83)
$\mathbb{L}$	the universe of layout designs	[1]
m	minimal movement distance to prevent infinitesimal actions	(p. 84)
M	movement morphism of a module transformation	(p. 43)
$\mathcal{M}$	constraint members (set of components on which a certain constraint is imposed)	(p. 73)
n	general counter	(p. 55)
N	node (vertex) of a geometrical shape	(p. 108)
$\mathbb{N}^0$	set of natural numbers (including zero)	(p. 70)
$\mathbb{N}^+$	set of positive natural numbers (excluding zero)	(p. 51)
ξ	contraction amount for a relative tightening or enlargement of the layout zone	(p. 112)
Ξ	set of contraction amounts	(p. 124)
o	orientation parameter of a procedural generator	(p. 51)
O	orientation of a design component	(p. 46)
$O_{\bar{h}}$	orientation of a design component in horizontal notation	(p. 46)
O_v	orientation of a design component in vertical notation	(p. 47)
$\mathbb{O}$	set of orientations	(p. 108)
ϖ	pressing rate (to define the rigorousness of the layout zone tightening policy)	(p. 117)
p	positioning parameter of a procedural generator	(p. 51)
P	parameterized design component / procedural generator / participant	(p. 52)
$\mathcal{P}$	set of parameterized design components / procedural generators / participants	(p. 55)
$\mathcal{P}_\iota$	participants that are involved in an action	(p. 79)
q	contraction quotient in an exponential progressive tightening policy	(p. 131)
Q	square geometrical shape (regular quadrilateral)	(p. 63)
ϱ	relaxation threshold of a connection between two participants	(p. 63)
ρ	region of a wound on a participant	(p. 70)
r	radius of a circle	(p. 63)
R	rotation morphism of a module transformation	(p. 43)

$\mathbb{R}$	set of real numbers	(p. 62)
ς	severity of a wound on a participant	(p. 70)
s	strength of a connection between two participants	(p. 62)
S	free peripheral space of a participant	(p. 75)
$\mathbb{S}$	the universe of schematic circuits	[1]
τ	trouble contributing to the interference of a participant	(p. 59)
t_H	constraint type of a hard design constraint H	(p. 73)
T	transformation of a participant	(p. 79)
$\mathcal{T}$	set of transformations (i.e., an action involving multiple participants)	(p. 103)
$\mathbb{T}$	set of actions (where each action is given as a set of transformations)	(p. 108)
υ	vertical flipping in the orientation of a design component	(p. 47)
Υ	interference of a participant	(p. 59)
u	number of minor tightenings in a linear progressive tightening policy	(p. 127)
$\mathcal{U}$	set of obstacles	(p. 75)
φ	conciliation quota for the decrease of the aversion between two participants	(p. 62)
Φ	morphism (movement, rotation, or deformation) of a module transformation	(p. 43)
$v_t(\mathcal{M})$	verification function for a constraint of type t imposed on constraint members $\mathcal{M}$	(p. 73)
V	layout variant of a design component	(p. 80)
$\mathcal{V}$	variability of a design component	(p. 50)
$\acute{\mathcal{V}}$	intrinsic variability of a design component	(p. 50)
$\widetilde{\mathcal{V}}$	cumulative variability of a design component	(p. 51)
$\mathfrak{w}_{\text{ch}}$	total channel width of a MOS transistor	(p. 53)
$\mathfrak{w}_{\text{min}}$	minimum channel width of a MOS transistor	(p. 53)
w	width of an object	(p. 83)
W	wound of a participant	(p. 70)
$\mathcal{W}$	set of wounds of a participant	(p. 72)
χ	projection divisor for tightening or enlarging a nonrectangular layout zone	(p. 124)
x	horizontal coordinate of a component's location or of a point in the layout plane	(p. 46)
X	exploration scenario in a participant's action exploration plan	(p. 108)
$\mathcal{X}$	exploration plan for a participant's action exploration	(p. 108)
ψ	protrusion extent of a participant's protrusion	(p. 68)
Ψ	protrusion of a participant	(p. 67)
ω	overlap between two participants	(p. 60)
y	vertical coordinate of a component's location or of a point in the layout plane	(p. 46)
Y	yielding region around a participant, used to determine a *Yielding* action	(p. 91)
ζ	number of a participant's unrelaxed connections	(p. 65)
z, z^*	zone size quotients in a logistic regressive tightening policy	(p. 135)
Z	outline of the user-defined layout zone	(p. 31)

References

A reader lives a thousand lives before he dies.
The man who never reads lives only one.
George R. R. Martin (US-American writer)

Bibliography

[1] Daniel Marolt, “Layout Automation in Analog IC Design with Formalized and Nonformalized Expert Knowledge”, Ph.D. Thesis, University of Stuttgart, Dec. 2018, DOI: 10.18419/opus-10231, URL: http://elib.uni-stuttgart.de/handle/11682/10248.

[2] Francis Heylighen / Carlos Gershenson, “The Meaning of Self-Organization in Computing”, *IEEE Intelligent Systems*, pp. 72–75, May 2003, URL: http://pcp.vub.ac.be/Papers/IEEE.Self-organization.pdf.

[3] John Horgan, “From Complexity to Perplexity”, *Scientific American*, vol. 272, no. 6, Jun. 1995.

[4] Werner Ebeling / Jan Freund / Frank Schweitzer, “Komplexe Strukturen: Entropie und Information”, B. G. Teubner Stuttgart, Leipzig, 1998, ISBN: 978-3-322-85167-3.

[5] Carlos Gershenson / Francis Heylighen, “How Can We Think the Complex?”, *Managing Organizational Complexity: Philosophy, Theory and Application* (edited by Kurt Richardson), ch. 3, pp. 47–61, Information Age Publishing, 2005, URL: https://arxiv.org/abs/nlin/0402023.

[6] John T. Emlen Jr., “Flocking Behavior in Birds”, *The Auk*, vol. 69, no. 2, pp. 160–170, Apr. 1952, DOI: 10.2307/4081266.

[7] C. C. Trowbridge / H. K. Job, “On the Origin of the Flocking Habit of Migratory Birds”, *The Popular Science Monthly*, vol. 84, pp. 209–217, Mar. 1914, Public Domain content, URL: http://www.archive.org/stream/popularsciencemo84newy#page/214/mode/1up.

[8] Randal S. Olson / Arend Hintze / Fred C. Dyer / David B. Knoester / Christoph Adami, "Predator Confusion is Sufficient to Evolve Swarming Behaviour", *Journal of the Royal Society Interface*, vol. 10, no. 85, pp. 1–8, Aug. 2013, DOI: 10.1098/rsif.2013.0305.

[9] Steven J. Portugal / Tatjana Y. Hubel / Johannes Fritz / Stefanie Heese / Daniela Trobe / Bernhard Voelkl / Stephen Hailes / Alan M. Wilson / James R. Usherwood, "Upwash Exploitation and Downwash Avoidance by Flap Phasing in Ibis Formation Flight", *Nature*, vol. 505, pp. 399–402, Jan. 2014, DOI: 10.1038/nature12939.

[10] Simeon Andrew Ning, "Aircraft Drag Reduction Through Extended Formation Flight", Ph.D. Thesis, Stanford University, Department of Aeronautics and Astronautics, Aug. 2011, URL: `https://purl.stanford.edu/dp147ff0571`.

[11] Mitchell G. Ash, "Gestalt Psychology in German Culture, 1890–1967: Holism and the Quest for Objectivity", Cambridge University Press, Cambridge, United Kingdom, 1995, ISBN: 0-521-47540-6.

[12] Russ Dewey, "Psychology: An Introduction – Chapter Four: Senses (The Whole is Other than the Sum of the Parts)", *cites F. Heider (1977)*, 2007, URL: `http://www.intropsych.com/ch04_senses/whole_is_other_than_the_sum_of_the_parts.html`.

[13] Harold J. Morowitz, "The Emergence of Everything: How the World Became Complex", Oxford University Press, USA, 2002, ISBN: 978-0-19-513513-8.

[14] "Self-Organizing Systems: The Emergence of Order" (edited by F. Eugene Yates / Alan Garfinkel / Donald O. Walter / Gregory B. Yates), Series: Life Science Monographs, First Edition, Plenum Press, New York and London, 1987, ISBN: 978-1-4612-8227-3, DOI: 10.1007/978-1-4613-0883-6.

[15] Carl Anderson, "Self-Organization in Relation to Several Similar Concepts: Are the Boundaries to Self-Organization Indistinct?", *Biological Bulletin*, vol. 202, no. 3, pp. 247–255, Jun. 2002.

[16] "Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering" (edited by Giovanna Di Marzo Serugendo / Anthony Karageorgos / Omer F. Rana / Franco Zambonelli), Series: Lecture Notes in Computer Science, vol. 2977, Springer, Berlin Heidelberg, 2004, ISBN: 978-3-540-21201-0, DOI: 10.1007/b95863.

[17] P. Massioni / M. Verhaegen, "Distributed Control for Identical Dynamically Coupled Systems: A Decomposition Approach", *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 124–135, Jan. 2009, DOI: 10.1109/TAC.2008.2009574.

[18] Mikhail Prokopenko, "Guided Self-Organization", *Human Frontier Science Program Journal*, vol. 3, no. 5, pp. 287–289, Oct. 2009, DOI: 10.2976/1.3233933.

[19] Mitchel Resnick, "Decentralized Modeling and Decentralized Thinking", *Modeling and Simulation in Precollege Science and Mathematics* (edited by W. Feurzeig / N. Roberts), pp. 114–137, Springer, New York, 1999.

[20] Yaneer Bar-Yam, “A Mathematical Theory of Strong Emergence Using Multiscale Variety”, *Complexity*, vol. 9, no. 6, pp. 15–24, Aug. 2004, DOI: 10.1002/cplx.20029.

[21] Mark A. Bedau, “Artificial Life: Organization, Adaptation and Complexity From the Bottom Up”, *Trends in Cognitive Sciences*, vol. 7, no. 11, pp. 505–512, Nov. 2003, DOI: 10.1016/j.tics.2003.09.012.

[22] “The MIT Encyclopedia of the Cognitive Sciences” (edited by Frank C. Keil / Robert Andrew Wilson), MIT Press, Massachusetts, 2001, ISBN: 978-0-262-73144-7.

[23] Aristotle, “Metaphysics, Book H 1045a 8–10”.

[24] George Henry Lewes, “Problems of Life and Mind”, First Series, vol. 2, Trübner, London, 1875, ISBN: 1-4255-5578-0.

[25] Stuart A. Kauffman, “Reinventing the Sacred”, Basic Books, New York, NY, USA, 2008, ISBN: 978-0-465-00300-6.

[26] Francis Heylighen, “Self-Organization, Emergence and the Architecture of Complexity”, *Proc. of 1st European Conference on System Science*, pp. 23–32, Paris, 1989.

[27] Philip W. Anderson, “More and Different: Notes from a Thoughtful Curmudgeon”, World Scientific, London, 2011, ISBN: 978-981-4350-12-9.

[28] James P. Crutchfield, “Is Anything Ever New? Considering Emergence”, *Santa Fe Institute Studies in the Sciences of Complexity* (edited by G. Cowan / D. Pines / D. Melzner), Series: Integrative Themes, vol. XIX, Addison-Wesley, Reading, MA, 1994.

[29] Steven B. Johnson, “Only Connect”, *online*, Oct. 2001, URL: `http://www.theguardian.com/books/2001/oct/15/society`.

[30] Mark A. Bedau, “Downward Causation and the Autonomy of Weak Emergence”, *Principia*, vol. 6, no. 1, pp. 5–50, 2002, published by NEL – Epistemology and Logic Research Group, Federal University of Santa Catarina, Brazil.

[31] Jochen Fromm, “Types and Forms of Emergence”, *online*, Jun. 2005, URL: `http://arxiv.org/abs/nlin/0506028`.

[32] Robert B. Laughlin, “A Different Universe: Reinventing Physics from the Bottom Down”, Basic Books, New York, NY, USA, Mar. 2005, ISBN: 978-0-465-03828-2.

[33] Mark A. Bedau, “Weak Emergence”, *Philosophical Perspectives: Mind, Causation, and World* (edited by James Tomberlin), vol. 11, pp. 375–399, Blackwell Publishers, Oxford, 1997.

[34] William Edward Seager, “Emergence and Supervenience”, *online*, URL: `http://www.utsc.utoronto.ca/~seager/emsup.pdf`.

[35] Gerald E. Marsh, “The Demystification of Emergent Behavior”, *online*, 2009, URL: `http://arxiv.org/ftp/arxiv/papers/0907/0907.1117.pdf`.

[36] Ernst Mayr, “The Growth of Biological Thought: Diversity, Evolution, and Inheritance”, Harvard University Press, 1982, ISBN: 0-674-36446-5.

[37] Anil K. Seth, “Measuring Autonomy and Emergence via Granger Causality”, *Artificial Life*, vol. 16, no. 2, pp. 179–196, 2010, DOI: 10.1162/artl.2010.16.2.16204.

[38] Paul C. W. Davies, “Emergent Biological Principles and the Computational Properties of the Universe”, *Complexity*, vol. 10, no. 2, pp. 11–15, Nov. 2004, DOI: 10.1002/cplx.20059.

[39] David J. Chalmers, “Strong and Weak Emergence”, *The Re-Emergence of Emergence: The Emergent Hypothesis from Science to Religion* (edited by Philip Clayton / Paul Davies), Oxford University Press, Oxford, 2006, ISBN: 978-0199287147.

[40] Noam Miller / Robert Gerlai, “From Schooling to Shoaling: Patterns of Collective Motion in Zebrafish (Danio Rerio)”, *PLoS ONE*, vol. 7, no. 11, pp. 1–6, Nov. 2012, DOI: 10.1371/journal.pone.0048865.

[41] Frank Fraser Darling, “A Herd of Red Deer: A Study in Animal Behaviour (Wild Lives)” (edited by Walter Stephen), Luath Press Limited, Edinburgh, Jun. 2008, ISBN: 978-1-906307-42-4.

[42] Chad M. Topaz / Maria R. D’Orsogna / Leah Edelstein-Keshet / Andrew J. Bernoff, “Locust Dynamics: Behavioral Phase Change and Swarming”, *PLoS Computational Biology*, vol. 8, no. 8, pp. 1–11, Aug. 2012, DOI: 10.1371/journal.pcbi.1002642.

[43] Bolei Zhou / Xiaogang Wang / Xiaoou Tang, “Understanding Collective Crowd Behaviors: Learning a Mixture Model of Dynamic Pedestrian-Agents”, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2871–2878, Jun. 2012, DOI: 10.1109/CVPR.2012.6248013.

[44] Gordon Firestein (Seacology USA), “School of Jacks”, *online*, Jan. 2005, licensed under CC BY-SA 3.0, URL: `https://commons.wikimedia.org/w/index.php?curid=22766988`.

[45] Shyamvs78 (own work), “Spotted Deer Group in Jim Corbett National Park (India)”, *online*, Feb. 2008, licensed under CC BY 3.0, URL: `https://commons.wikimedia.org/w/index.php?curid=3521093`.

[46] Waugsberg (own work), “Swarm of Bees in the Air Shortly Before Landing on a Tree”, *online*, May 2007, licensed under CC BY-SA 3.0, URL: `https://commons.wikimedia.org/w/index.php?curid=2133292`.

[47] Matt Morgen, “Crowd Driven from Tompkins Square by the Mounted Police in the Tompkins Square Riot of 1874”, *Frank Leslie’s Illustrated Newspaper*, Jan. 1874, Public Domain content, URL: `https://commons.wikimedia.org/w/index.php?curid=2186990`.

[48] L. David Mech, “Alpha Status, Dominance, and Division of Labor in Wolf Packs”, *Canadian Journal of Zoology*, vol. 77, no. 8, pp. 1196–1203, 1999, DOI: 10.1139/z99-099.

[49] Manuele Brambilla / Eliseo Ferrante / Mauro Birattari / Marco Dorigo, “Swarm Robotics: A Review from the Swarm Engineering Perspective”, *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, Mar. 2013, DOI: 10.1007/s11721-012-0075-2.

[50] Mehmet Karatay (own work), “Safari Ants on the Chogoria Route of Mount Kenya”, *online*, May 2007, licensed under CC BY-SA 3.0, URL: `https://commons.wikimedia.org/w/index.php?curid=2179109`.

[51] S. Goss / S. Aron / J.-L. Deneubourg / J. M. Pasteels, “Self-Organized Shortcuts in the Argentine Ant”, *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989, DOI: 10.1007/BF00462870.

[52] Claire Detrain / Jean-Louis Deneubourg, “Self-Organized Structures in a Superorganism: Do Ants ‘Behave’ Like Molecules?”, *Physics of Life Reviews*, vol. 3, no. 3, pp. 162–187, Sep. 2006, DOI: 10.1016/j.plrev.2006.07.001.

[53] Jon Nelson, “Origin of Diversity in Falling Snow”, *Atmospheric Chemistry and Physics*, vol. 8, pp. 5669–5682, Sep. 2008.

[54] Kenneth G. Libbrecht, “The Physics of Snow Crystals”, *Reports on Progress in Physics*, vol. 68, pp. 855–895, Mar. 2005, DOI: 10.1088/0034-4885/68/4/R03.

[55] Charles Schmitt (own work), “Stellate Snowflake”, *online*, Mar. 2014, licensed under CC BY-SA 4.0, URL: `https://commons.wikimedia.org/w/index.php?curid=44338386`.

[56] A. V. Getling, “Rayleigh-Bénard Convection: Structures and Dynamics”, Advanced Series in Nonlinear Dynamics, vol. 11, World Scientific Publishing, Mar. 1998, ISBN: 978-981-02-2657-2.

[57] A. V. Getling / O. Brausch, “Cellular Flow Patterns and Their Evolutionary Scenarios in Three-Dimensional Rayleigh-Bénard Convection”, *Physical Review E*, vol. 67, no. 4, pp. 1–4, Apr. 2003, DOI: 10.1103/PhysRevE.67.046313.

[58] Heinz Georg Schuster / Wolfram Just, “Deterministic Chaos: An Introduction”, Fourth, Revised and Enlarged Edition, John Wiley & Sons, Weinheim, Jan. 2005, ISBN: 978-3-527-40415-5.

[59] R. B. Levien / S. M. Tan, “Double Pendulum: An Experiment in Chaos”, *American Journal of Physics*, vol. 61, no. 11, pp. 1038–1044, Nov. 1993, DOI: 10.1119/1.17335.

[60] Andy Martin / Kristian Helmerson, “Emergence: The Remarkable Simplicity of Complexity”, *online*, Oct. 2014, URL: `http://theconversation.com/emergence-the-remarkable-simplicity-of-complexity-30973`.

[61] “The Science of Fractal Images” (edited by Heinz-Otto Peitgen / Dietmar Saupe), Springer Verlag, 1988, ISBN: 978-1-4612-8349-2.

[62] Solkoll (own work), “Koch Snowflake”, *online*, Feb. 2005, Public Domain content, URL: `https://commons.wikimedia.org/w/index.php?curid=59048`.

[63] Douglas H. Werner / Suman Ganguly, “An Overview of Fractal Antenna Engineering Research”, *IEEE Antennas and Propagation Magazine*, vol. 45, no. 1, pp. 38–57, Feb. 2003.

[64] Tom Addiscott, “Emergence or Self-organization? Look to the Soil Population”, *Communicative and Integrative Biology*, vol. 4, no. 4, pp. 469–470, Jul. 2011, DOI: 10.4161/cib.4.4.15547.

[65] Tom De Wolf / Tom Holvoet, “Emergence Versus Self-Organisation: Different Concepts But Promising When Combined”, *Engineering Self-Organising Systems: Methodologies and Applications* (edited by S. Brückner *et al.*), Series: Lecture Notes in Computer Science, vol. 3464, pp. 1–15, Springer, Berlin Heidelberg, 2005.

[66] Jürgen Appelo, “Self-Organization vs. Emergence”, *online*, Oct. 2009, URL: `http://noop.nl/2009/10/self-organization-vs-emergence.html`.

[67] Andy Brandt, “The Triangle of Self-Organization”, *online*, Jul. 2013, URL: `http://pragmaticleader.net/blog/2013/7/3/the-triangle-of-self-organization`.

[68] Deborah M. Gordon, “From Division of Labor to the Collective Behavior of Social Insects”, *Behavioral Ecology and Sociobiology*, pp. 1–8, Dec. 2015, DOI: 10.1007/s00265-015-2045-3.

[69] Steven Berlin Johnson, “Emergence: The Connected Lives of Ants, Brains, Cities, and Software”, Scribner, New York, NY, USA, 2001, ISBN: 978-0-684-86876-9.

[70] Humberto R. Maturana / Francisco J. Varela, “Autopoiesis and Cognition: The Realization of the Living”, Series: Boston Studies in the Philsosophy and History of Science, vol. 42, D. Reidel Publishing, Dordrecht, Holland, 1980, ISBN: 978-9-027-71016-1.

[71] Niklas Luhmann, “Die Gesellschaft der Gesellschaft”, Suhrkamp, Frankfurt am Main, 1997, ISBN: 3-518-58240-2.

[72] Niklas Luhmann, “Operational Closure and Structural Coupling: The Differentiation of the Legal System”, *Cardoso Law Review*, vol. 13, pp. 1419–1441, 1992.

[73] “Luhmann Observed: Radical Theoretical Encounters” (edited by Anders la Cour / Andreas Philippopoulos-Mihalopoulos), Palgrave Macmillan, Jun. 2013, ISBN: 978-1-137-01528-0.

[74] Christopher G. Langton, “Computation at the Edge of Chaos: Phase Transitions and Emergent Computation”, *Physica D: Nonlinear Phenomena*, vol. 42, no. 1–3, pp. 12–37, Jun. 1990, DOI: 10.1016/0167-2789(90)90064-V.

[75] James P. Crutchfield / Karl Young, “Computation at the Onset of Chaos”, *Entropy, Complexity, and the Physics of Information* (edited by W. Zurek), Series: SFI Studies in the Sciences of Complexity, vol. 8, pp. 223–269, Addison-Wesley, Reading, Massachusetts, 1990.

[76] Jochen Fromm, “Edge of Chaos”, *online*, May 2009, URL: http://www.wiki.cas-group.net/index.php?title=File:Edge_of_Chaos.png.

[77] Roger Lewin, “Complexity: Life at the Edge of Chaos”, Second Edition, University of Chicago Press, London, UK, 1999, ISBN: 0-226-47654-5.

[78] Katrina Schwartz, “On the Edge of Chaos: Where Creativity Flourishes”, *online*, May 2014, URL: http://ww2.kqed.org/mindshift/2014/05/06/on-the-edge-of-chaos-where-creativity-flourishes/.

[79] Robert M. Bilder / Kendra S. Knudsen, “Creative Cognition and Systems Biology on the Edge of Chaos”, *Frontiers in Psychology*, vol. 5, no. 1104, Sep. 2014, DOI: 10.3389/fpsyg.2014.01104.

[80] Gary William Flake, “The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation”, MIT Press, Cambridge, Massachusetts, 1998, ISBN: 978-0-262-56127-3.

[81] Elisabeth Göbel, “Theorie und Gestaltung der Selbstorganisation”, Series: Betriebswirtschaftliche Forschunsergebnisse, vol. 111, Duncker & Humblot, Berlin, 1998, ISBN: 978-3-428-49434-7.

[82] Heinz von Foerster, “Principles of Self-Organization – In a Socio-Managerial Context”, *Self-Organization and Management of Social Systems: Insights, Promises, Doubts, and Questions* (edited by Hans Ulrich / Gilbert J. B. Probst), Springer Series in Synergetics, vol. 26, pp. 2–24, Springer-Verlag, Berlin Heidelberg, 1984, ISBN: 978-3-642-69764-7.

[83] F. G. Varela / H. R. Maturana / R. Uribe, “Autopoiesis: The Organization of Living Systems, Its Characterization and a Model”, *Biosystems*, vol. 5, no. 4, pp. 187–196, May 1974, DOI: 10.1016/0303-2647(74)90031-8.

[84] Scott Camazine / Jean-Louis Deneubourg / Nigel R. Franks / James Sneyd / Guy Theraulaz / Eric Bonabeau, “Self-Organization in Biological Systems”, Series: Princeton Studies in Complexity, Princeton University Press, 2003, ISBN: 978-0-691-11624-2.

[85] Francis Heylighen, “The Science of Self-Organization and Adaptivity”, *Knowledge Management, Organizational Intelligence and Learning, and Complexity* (edited by L. Douglas Kiel), Series: The Encyclopedia of Life Support Systems, pp. 1–26, EOLSS Publishers Company Limited, United Kingdom, Jan. 2009, ISBN: 978-1-848-26913-2.

[86] Pierre Paul Grassé, “La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. La théorie de la stigmergie: Essai d’interprétation du comportement des termites constructeurs”, *Insectes Sociaux*, vol. 6, no. 1, pp. 41–83, Mar. 1959.

[87] Leslie Marsh / Christian Onof, “Stigmergic Epistemology, Stigmergic Cognition”, *Cognitive Systems Research*, vol. 9, no. 1–2, pp. 136–149, Mar. 2008, DOI: 10.1016/j.cogsys.2007.06.009.

[88] Marco Dorigo / Mauro Birattari / Thomas Stützle, “Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique”, *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[89] Carlos Gershenson, “Self-Organizing Traffic Lights”, *Complex Systems*, vol. 16, no. 1, pp. 29–53, 2005.

[90] H. van Dyke Parunak, “Making Swarming Happen”, *Proc. of Conference on Swarming and Network Enabled Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance*, pp. 26–40, Jan. 2003.

[91] Ralph Beckers / Owen E. Holland / Jean-Louis Deneubourg, “From Local Actions to Global Tasks: Stigmergy and Collective Robotics”, *Prerational Intelligence: Adaptive Behavior and Intelligent Systems Without Symbols and Logic* (edited by Holk Cruse / Jeffrey Dean / Helge Ritter), Series: Studies in Cognitive Systems, vol. 26, pp. 1008–1022, Springer Netherlands, 2000, DOI: 10.1007/978-94-010-0870-9_63.

[92] Peter A. Corning, “Synergy and Self-Organization in the Evolution of Complex Systems”, *Systems Research*, vol. 12, no. 2, pp. 89–121, 1995, DOI: 10.1002/sres.3850120204.

[93] Hermann Haken, “Synergetics: Introduction and Advanced Topics”, Springer-Verlag, Berlin Heidelberg, 2004, ISBN: 978-3-540-40824-6.

[94] “International Encyclopedia of Ergonomics and Human Factors” (edited by Waldemar Karwowski), Second Edition, vol. 3, CRC Press, Taylor & Francis Group, Boca Raton, FL, 2006, ISBN: 978-0-415-30430-6.

[95] Peter A. Corning, “What is Life? Among Other Things, It’s a Synergistic Effect!”, *Cosmos and History: The Journal of Natural and Social Philosophy*, vol. 4, no. 1–2, pp. 233–243, 2008, ISSN: 1832-9101.

[96] Angus Stevenson, “Oxford Dictionary of English”, Third Edition, Oxford University Press, Aug. 2010, ISBN: 978-0199571123.

[97] Robley Dunglison, “Medical Lexicon: A Dictionary of Medical Science”, Ninth Edition, Blanchard and Lea, Philadelphia, 1853.

[98] Carlos Gershenson, “A General Methodology for Designing Self-Organizing Systems”, *ECCO Working Paper*, May 2005, URL: `http://arxiv.org/abs/nlin/0505009v3`.

[99] Peter A. Corning, “Nature’s Magic: Synergy in Evolution and the Fate of Humankind”, Cambridge University Press, May 2003, ISBN: 978-0-521-82547-4.

[100] “Innovations and Advanced Techniques in Computer and Information Sciences and Engineering” (edited by Tarek Sobh), Springer, Dordrecht, 2007, ISBN: 978-1-4020-6268-1.

[101] W. D. Hamilton, “Geometry for the Selfish Herd”, *Journal of Theoretical Biology*, vol. 31, no. 2, pp. 295–311, May 1971, DOI: 10.1016/0022-5193(71)90189-5.

[102] Friedrich Böhringer (own work), “Schafherde in Schoren”, *online*, Oct. 2009, licensed under CC BY-SA 2.5, URL: https://commons.wikimedia.org/w/index.php?curid=8694470.

[103] Yoav Shoham / Kevin Leyton-Brown, “Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations”, Cambridge University Press, 2009, ISBN: 978-0-521-89943-7.

[104] Adam Smith, “The Wealth of Nations”, Series: Oxford World’s Classics, Oxford University Press, May 2008, ISBN: 978-0-19-953592-7.

[105] William Ross Ashby, “An Introduction to Cybernetics”, Fourth Impression, Chapman & Hall Ltd, London, 1961, ISBN: 978-0-416-68300-4.

[106] Wordpress, “The First Law of Cybernetics”, *online*, Oct. 2011, URL: https://firstlaw.wordpress.com/2011/10/18/ashbys-law/.

[107] William Ross Ashby, “Principles of the Self-Organizing System”, *Principles of Self-Organization: Transactions of the University of Illinois Symposium* (edited by H. von Foerster / G. W. Zopf Jr.), pp. 255–278, Pergamon Press, London, UK, 1962.

[108] William Ross Ashby, “Requisite Variety and Its Implications for the Control of Complex Systems”, *Cybernetica*, vol. 1, no. 2, pp. 83–99, 1958.

[109] Stafford Beer, “The Heart of Enterprise”, Series: Managerial Cybernetics of Organization, vol. 2, Wiley, 1979, ISBN: 978-0-471-27599-2.

[110] “Artificial Life: An Overview” (edited by Christopher G. Langton), Series: A Bradford Book – Complex Adaptive Systems, MIT Press, Cambridge, Massachusetts, 1997, ISBN: 978-0-262-62112-0.

[111] John von Neumann, “The General and Logical Theory of Automata”, *Cerebral Mechanisms in Behavior – The Hixon Symposium* (edited by L. A. Jeffress), pp. 1–31, John Wiley & Sons, New York, NY, USA, 1951, presented September 20, 1948, in Pasadena.

[112] John von Neumann / Arthur W. Burks, “Theory of Self-Reproducing Automata”, University of Illinois Press, Urbana and London, 1966.

[113] Martin Gardner, “Mathematical Games – The Fantastic Combinations of John Conway’s New Solitaire Game ‘Life’”, *Scientific American*, vol. 223, pp. 120–123, Oct. 1970, ISBN: 0-89454-001-7.

[114] Elwyn R. Berlekamp / John H. Conway / Richard K. Guy, “Winning Ways for Your Mathematical Plays”, Second Edition, vol. 4, Taylor & Francis, 2004, ISBN: 978-1-568-81144-4.

[115] Christopher G. Langton, “Studying Artificial Life with Cellular Automata”, *Physica D: Nonlinear Phenomena*, vol. 22, no. 1–3, pp. 120–149, Oct. 1986, DOI: 10.1016/0167-2789(86)90237-X.

[116] A. Gajardo / A. Moreira / E. Goles, “Complexity of Langton’s Ant”, *Discrete Applied Mathematics*, vol. 117, no. 1–3, pp. 41–50, Mar. 2002, DOI: 10.1016/S0166-218X(00)00334-6.

[117] A. K. Dewdney, “Computer Recreations – The Cellular Automata Programs That Create Wireworld, Rugworld and Other Diversions”, *Scientific American*, vol. 262, pp. 146–149, 1990.

[118] “Cellular Automata: 10th International Conference on Cellular Automata for Research and Industry” (edited by Georgios C. Sirakoulis / Stefania Bandini), Series: Lecture Notes in Computer Science, vol. 7495, Springer, Berlin, Heidelberg, 2012, ISBN: 978-3-642-33349-1.

[119] Konrad Zuse, “Rechnender Raum”, Series: Schriften zur Datenverarbeitung, Springer Fachmedien, Wiesbaden, 1969, ISBN: 978-3-663-00810-1, DOI: 10.1007/978-3-663-02723-2.

[120] Stephen Wolfram, “A New Kind of Science”, Wolfram Media, Jun. 2002, ISBN: 978-1-57955-008-0.

[121] Steve Tadelis, “Game Theory: An Introduction”, Princeton University Press, Princeton, New Jersey, Jan. 2013, ISBN: 978-0-691-12908-2.

[122] John Nash, “Non-Cooperative Games”, *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, Sep. 1951, DOI: 10.2307/1969529.

[123] Roman L. Weil Jr., “The N-Person Prisoner’s Dilemma: Some Theory and a Computer-Oriented Approach”, *Systems Research and Behavioral Science*, vol. 11, no. 3, pp. 227–234, May 1966, DOI: 10.1002/bs.3830110310.

[124] Robert W. Rosenthal, “Games of Perfect Information, Predatory Pricing and the Chain-Store Paradox”, *Journal of Economic Theory*, vol. 25, no. 1, pp. 92–100, Aug. 1981, DOI: 10.1016/0022-0531(81)90018-1.

[125] John von Neumann / Oskar Morgenstern, “Theory of Games and Economic Behavior”, Sixtieth Anniversary Edition, Princeton University Press, Princeton, New Jersey, 2007, ISBN: 978-0-691-13061-3.

[126] Michael Wooldridge, “An Introduction to Multiagent Systems”, Second Edition, John Wiley & Sons, Jun. 2009, ISBN: 978-0-470-51946-2.

[127] Stan Franklin / Art Graesser, “Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents”, *Proc. of Workshop on Intelligent Agents, Agent Theories, Architectures, and Languages*, pp. 21–35, Springer, 1996, ISBN: 3-540-62507-0.

[128] Stuart Russell / Peter Norvig, “Artificial Intelligence: A Modern Approach”, Third Edition, Pearson Education Limited, Aug. 2013, ISBN: 978-1-292-02420-2.

[129] Jochen Fromm, “The Emergence of Complexity”, Kassel University Press, Kassel, 2004, ISBN: 978-3-89958-069-3.

[130] Jacques Ferber, “Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence”, Addison-Wesley Longman Publishing, Boston, MA, USA, Feb. 1999, ISBN: 978-0-201-36048-6.

[131] Riccardo Poli, “Analysis of the Publications on the Applications of Particle Swarm Optimisation”, *Journal of Artificial Evolution and Application*, vol. 2008, no. 4, pp. 1–10, Jan. 2008, DOI: 10.1155/2008/685175.

[132] Shiyong Wang / Jiafu Wan / Daqiang Zhang / Di Li / Chunhua Zhang, “Towards Smart Factory for Industry 4.0: A Self-Organized Multi-Agent System with Big Data Based Feedback and Coordination”, *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 101, no. C, pp. 158–168, Jun. 2016, DOI: 10.1016/j.comnet.2015.12.017.

[133] Onn Shehory / Katia Sycara / Gita Sukthankar / Vick Mukherjee, “Agent Aided Aircraft Maintenance”, *Proc. of 3rd Annual Conference on Autonomous Agents*, pp. 306–312, 1999, DOI: 10.1145/301136.301216.

[134] Vivek Kumar / S. Srinivasan, “A Review of Supply Chain Management Using Multi-Agent System”, *International Journal of Computer Science Issues*, vol. 7, no. 5, pp. 198–205, Sep. 2010, ISSN: 1694-0814.

[135] T. Logenthiran / Dipti Srinivasan / Ashwin M. Khambadkone, “Multi-Agent System for Energy Resource Scheduling of Integrated Microgrids in a Distributed System”, *Electric Power Systems Research*, vol. 81, no. 1, pp. 138–148, Jan. 2011, DOI: 10.1016/j.epsr.2010.07.019.

[136] Muaz A. Niazi / Amir Hussain, “Agent-Based Computing from Multi-Agent Systems to Agent-Based Models: A Visual Survey”, *Scientometrics*, vol. 89, no. 2, pp. 479–499, Nov. 2011, DOI: 10.1007/s11192-011-0468-9.

[137] Nigel Gilbert, “Agent-Based Models”, Series: Quantitative Applications in the Social Sciences, vol. 153, Sage Publications, 2008, ISBN: 978-1-4129-4964-4.

[138] Craig W. Reynolds, “Flocks, Herds, and Schools: A Distributed Behavioral Model”, *Proc. of 14th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 25–34, Jul. 1987, DOI: 10.1145/37401.37406.

[139] Craig W. Reynolds, “Boids – Background and Update”, *online*, Jul. 2007, URL: `http://www.red3d.com/cwr/boids/`.

[140] András Czirók / Mária Vicsek / Tamás Vicsek, “Collective Motion of Organisms in Three Dimensions”, *Physica A: Statistical Mechanics and its Applications*, vol. 264, no. 1–2, pp. 299–304, Feb. 1999, DOI: 10.1016/S0378-4371(98)00468-3.

[141] M. Ballerini / N. Cabibbo / R. Candelier / A. Cavagna / E. Cisbani / I. Giardina / V. Lecomte / A. Orlandi / G. Parisi / A. Procaccini / M. Viale / V. Zdravkovic, “Interaction Ruling Animal Collective Behavior Depends on Topological Rather than Metric Dis-

tance: Evidence from a Field Study", *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1232–1237, Jan. 2008, DOI: 10.1073/pnas.0711437105.

[142] Yuki Sakamoto / Tatsuji Takahashi, "Metric and Topological Neighborhoods in Flocking Models", *Proc. of 8th International Conference on Bioinspired Information and Communications Technologies*, pp. 118–121, 2014, DOI: 10.4108/icst.bict.2014.258036.

[143] Christopher Hartman / Bedřich Beneš, "Autonomous Boids", *Computer Animation and Virtual Worlds*, vol. 17, no. 3–4, pp. 199–206, Jul. 2006, DOI: 10.1002/cav.123.

[144] Carlos Delgado-Mata / Jesus Ibanez Martinez / Simon Bee / Rocio Ruiz-Rodarte / Ruth Aylett, "On the Use of Virtual Animals with Artificial Fear in Virtual Environments", *New Generation Computing*, vol. 25, no. 2, pp. 145–169, Feb. 2007, DOI: 10.1007/s00354-007-0009-5.

[145] Xiaoyuan Tu / Demetri Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception, Behavior", *Proc. of 21st Annual Conference on Computer Graphics and Interactive Techniques*, pp. 43–50, Jul. 1994, DOI: 10.1145/192161.192170.

[146] Iain D. Couzin / Jens Krause / Richard James / Graeme D. Ruxton / Nigel R. Franks, "Collective Memory and Spatial Sorting in Animal Groups", *Journal of Theoretical Biology*, vol. 218, pp. 1–11, 2002, DOI: 10.1006/yjtbi.3065.

[147] Kai Nagel / Michael Schreckenberg, "A Cellular Automaton Model for Freeway Traffic", *Journal de Physique I*, vol. 2, no. 12, pp. 2221–2229, Dec. 1992, DOI: 10.1051/jp1:1992277.

[148] Tamás Vicsek / András Czirók / Eshel Ben-Jacob / Inon Cohen / Ofer Shochet, "Novel Type of Phase Transition in a System of Self-Driven Particles", *Physical Review Letters*, vol. 75, no. 6, pp. 1226–1229, Aug. 1995, DOI: 10.1103/PhysRevLett.75.1226.

[149] Yue-Xian Li / Ryan Lukeman / Leah Edelstein-Keshet, "Minimal Mechanisms for School Formation in Self-Propelled Particles", *Physica D: Nonlinear Phenomena*, vol. 237, no. 5, pp. 699–720, May 2008, DOI: 10.1016/j.physd.2007.10.009.

[150] Cynthia Nikolai / Gregory Madey, "Tools of the Trade: A Survey of Various Agent Based Modeling Platforms", *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 2, pp. 1–37, Mar. 2009, ISSN: 1460-7425.

[151] Fabio Luigi Bellifemine / Giovanni Caire / Dominic Greenwood, "Developing Multi-Agent Systems with JADE", Wiley Series in Agent Technology, vol. 7, John Wiley & Sons, West Sussex, England, Mar. 2007, ISBN: 978-0-470-05840-4.

[152] Daniel Marolt / Jürgen Scheible / Göran Jerke / Vinko Marolt, "SWARM: A Multi-Agent System for Layout Automation in Analog Integrated Circuit Design", *Agent and Multi-Agent Systems: Technology and Applications (10th KES International Conference, KES-AMSTA 2016)* (edited by Gordan Jezic / Yun-Heh Jessica Chen-Burger / Robert J. Howlett / Lakhmi C. Jain), Series: Smart Innovation, Systems and Technologies, vol. 58, ch. 2, pp. 15–31, Springer, Jun. 2016, DOI: 10.1007/978-3-319-39883-9_2.

[153] Thomas Burdick / Peter Herth / Göran Jerke / Christel Bürzele / Daniel Marolt / Vinko Marolt, “FR PCells”, *Patent Application*, Dec. 2015, title obscured for reasons of confidentiality.

[154] Vinko Marolt, “AE PCell-Based Layout Generators”, *Robert Bosch GmbH*, Jun. 2007, unpublished presentation, image recreated for reasons of confidentiality.

[155] Göran Jerke / Vinko Marolt / Christel Bürzele / Daniel Marolt / Peter Herth / Thomas Burdick, “Advanced Application of Cadence PCell Designer”, *CDNLive! EMEA*, May 2014, Session CUS06, URL: `https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/company/Events/CDNLive/Secured/Proceedings/EU/2014/CUS06.pdf`.

[156] Daniel Marolt / Jürgen Scheible / Göran Jerke, “The Application of Layout Module Generators upon Circuit Structure Recognition”, *Proc. of CDNLive! EMEA*, 6 pages, May 2011, Session AC13.

[157] Karl Sperl, “Ein neuartiger Automatisierungsansatz zur Kombination von Modulgeneratoren mit parametrisierten Verdrahtungs-Zellen im analogen Layoutentwurf”, M.Sc. Thesis, Reutlingen University, Mar. 2014.

[158] Kevin Kelly, “The Bottom Is Not Enough”, *online*, Feb. 2008, URL: `http://kk.org/thetechnium/the-bottom-is-n/`.

[159] G. H. Meisters, “Polygons Have Ears”, *The American Mathematical Monthly*, vol. 82, no. 6, pp. 648–651, Jun./Jul. 1975, DOI: 10.2307/2319703.

[160] Brian W. Kernighan / Shen Lin, “An Efficient Heuristic Procedure for Partitioning Graphs”, *The Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, Feb. 1970, DOI: 10.1002/j.1538-7305.1970.tb01770.x.

[161] C. M. Fiduccia / R. M. Mattheyses, “A Linear-Time Heuristic for Improving Network Partitions”, *Proc. of 19th Design Automation Conference*, pp. 175–181, Jun. 1982, DOI: 10.1109/DAC.1982.1585498.

[162] A. H. Farrahi / D. J. Hathaway / M. Wang / M. Sarrafzadeh, “Quality of EDA CAD Tools: Definitions, Metrics and Directions”, *Proc. of 1st International Symposium on Quality Electronic Design*, pp. 395–405, Mar. 2000, DOI: 10.1109/ISQED.2000.838903.

[163] Mohammad Tehranipoor, “CAD Algorithms – Placement”, *online*, Nov. 2008, URL: `www.engr.uconn.edu/~tehrani/teaching/cad/15_placement.pdf`.

[164] Ralph Otten, “Complexity and Diversity in IC Layout Design”, *Proc. of IEEE International Symposium on Circuits and Computers*, pp. 764–767, Oct. 1980, URL: `https://slideplayer.com/slide/13156322/`.

[165] D. W. Jepsen / C. D. Gelatt Jr., “Macro Placement by Monte Carlo Annealing”, *Proc. of IEEE International Conference on Computer Design*, pp. 495–498, Nov. 1983.

[166] Florin Balasa, “Device-Level Topological Placement with Symmetry Constraints”, *Analog Layout Synthesis – A Survey of Topological Approaches* (edited by Helmut E. Gräb),

ch. 1, pp. 3–60, Springer, New York Dordrecht Heidelberg London, 2011, ISBN: 978-1-4419-6931-6, DOI: 10.1007/978-1-4419-6932-3.

[167] Maqsood Yaqub / Ronald Boellaard / Marc A. Kropholler / Adriaan A. Lammertsma, "Optimization Algorithms and Weighting Factors for Analysis of Dynamic PET Studies", *Physics in Medicine and Biology*, vol. 51, no. 17, pp. 4217–4232, Aug. 2006, DOI: 10.1088/0031-9155/51/17/007.

[168] Carl Sechen / Alberto Luigi Sangiovanni-Vincentelli, "TimberWolf3.2: A New Standard Cell Placement and Global Routing Package", *Proc. of 23rd Design Automation Conference*, pp. 432–439, Jun./Jul. 1986, DOI: 10.1109/DAC.1986.1586125.

[169] Neil R. Quinn Jr., "The Placement Problem as Viewed from the Physics of Classical Mechanics", *Proc. of 12th Design Automation Conference*, pp. 173–178, 1975.

[170] Robert Hooke, "Lectures De Potentia Restitutiva, or of Spring Explaining the Power of Springing Bodies to which are added some Collections", John Martyn, Royal Society London, 1678, URL: `http://name.umdl.umich.edu/A44322.0001.001`.

[171] Jürgen Scheible, "Ein Softwarepaket zur vollautomatischen Plazierung von Bauteilen auf Leiterplatten für den industriellen Einsatz", Ph.D. Thesis, Universität Karlsruhe, 1992, ISBN: 3-18-145920-8.

[172] Elena Lodi / Fabrizio Luccio / Cristina Mugnai / Linda Pagli, "On Two-Dimensional Data Organization I", *Annales Societatis Mathematicae Polonae*, Series IV: Fundamenta Informaticae II, pp. 211–226, 1979.

[173] C. V. Deutsch / X. H. Wen, "An Improved Perturbation Mechanism for Simulated Annealing Simulation", *Mathematical Geology*, vol. 30, no. 7, pp. 801–816, Oct. 1998, DOI: 10.1023/A:1021722508504.

[174] Carl Sechen, "Chip-Planning, Placement, and Global Routing of Macro/Custom Cell Integrated Circuits Using Simulated Annealing", *Proc. of 25th Design Automation Conference*, pp. 73–80, Jun. 1988, DOI: 10.1109/DAC.1988.14737.

[175] D. F. Wong / H. W. Leong / C. L. Liu, "Simulated Annealing for VLSI Design", Kluwer Academic Publishers, Boston, Lancaster, Dordrecht, 1988, ISBN: 978-1-4612-8947-0.

[176] Thomas H. Cormen / Charles E. Leiserson / Ronald L. Rivest / Clifford Stein, "Introduction To Algorithms", Second Edition, MIT Press, Cambridge, Massachusetts, 2001, ISBN: 0-262-03293-7.

[177] Jørgen Bang-Jensen / Gregory Gutin / Anders Yeo, "When the Greedy Algorithm Fails", *Discrete Optimization*, vol. 1, no. 2, pp. 121–127, Nov. 2004, DOI: 10.1016/j.disopt.2004.03.007.

[178] J. D. Conway / G. G. Schrooten, "An Automatic Layout Generator for Analog Circuits", *Proc. of European Conference on Design Automation*, pp. 513–519, Mar. 1992, DOI: 10.1109/EDAC.1992.205989.

[179] Lihong Zhang / Ulrich Kleine / Yingtao Jiang, “An Automated Design Tool for Analog Layouts”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 8, pp. 881–894, Aug. 2006, DOI: 10.1109/TVLSI.2006.878475.

[180] Nuttorn Jangkrajarng / Sambuddha Bhattacharya / Roy Hartono / C.-J. Richard Shi, “IPRAIL – Intellectual Property Reuse-Based Analog IC Layout Automation”, *Integration*, vol. 36, no. 4, pp. 237–262, Nov. 2003, DOI: 10.1016/j.vlsi.2003.08.004.

[181] John K. Ousterhout, “Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 3, no. 1, pp. 87–100, Jan. 1984, DOI: 10.1109/TCAD.1984.1270061.

[182] Timothy J. Barnes, “SKILL: A CAD System Extension Language”, *Proc. of 27th Design Automation Conference*, pp. 266–271, Jun. 1990, DOI: 10.1109/DAC.1990.114865.

[183] Daniel Marolt / Jürgen Scheible / Göran Jerke / Vinko Marolt, “SWARM: A Self-Organization Approach for Layout Automation in Analog IC Design”, *International Journal of Electronics and Electrical Engineering*, vol. 4, no. 5, pp. 374–385, Oct. 2016, DOI: 10.18178/ijeee.4.5.374-385.

[184] William Beebe, “Edge of the Jungle”, The Star Series, Henry Holt and Company, New York, 1921, URL: `http://ia600203.us.archive.org/25/items/edgejungle00beebgoog/edgejungle00beebgoog.pdf`.

[185] Theodore Christian Schneirla, “A Unique Case of Circular Milling in Ants, Considered in Relation to Trail Following and the General Problem of Orientation”, *American Museum Novitates*, no. 1253, pp. 1–26, Apr. 1944, URL: `http://digitallibrary.amnh.org/handle/2246/3733`.

[186] Daniel Marolt / Jürgen Scheible / Göran Jerke / Vinko Marolt, “A Self-Organization Approach for Layout Floorplanning Problems in Analog IC Design”, *Proc. of 12th Conference on Ph.D. Research in Microelectronics and Electronics*, pp. 1–4, Jun. 2016, DOI: 10.1109/PRIME.2016.7519454.

[187] Daniel Marolt / Jürgen Scheible / Göran Jerke / Vinko Marolt, “CAPABLE: A Layout Automation Framework for Analog IC Design”, *Proc. of MPC-Workshop*, vol. 54, pp. 49–59, Jul. 2015, ISSN: 1868-9221.

[188] Eduard Raines, “Rapid Analog Prototyping (RAP): Circuit Prospector and Generic Modgen”, *CDNLive! Boston*, Sep. 2015, Session CUS102, URL: `https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/company/Events/CDNLive/Secured/Proceedings/MA/2015/CUS102.pdf`.

[189] Daniel Marolt / Jürgen Scheible / Göran Jerke / Vinko Marolt, “Analog Layout Automation via Self-Organization: Enhancing the Novel SWARM Approach”, *Proc. of 7th IEEE Latin American Symposium on Circuits and Systems*, pp. 55–58, Feb./Mar. 2016, DOI: 10.1109/LASCAS.2016.7451008.

[190] Cadence Design Systems, Inc., “Virtuoso Unified Custom Constraints User Guide, Chapter 1: The Constraint Manager Assistant”, *Virtuoso Layout Suite*, Aug. 2015, Product Version 6.1.6.

[191] Sherif M. Saif / Mohamed Dessouky / M. Watheq El-Kharashi / Hazem Abbas / Salwa Nassar, “Pareto Front Analog Layout Placement Using Satisfiability Modulo Theories”, *Proc. of Design, Automation and Test in Europe Conference*, pp. 1411–1416, Mar. 2016, ISBN: 978-3-9815370-7-9.

[192] Göran Jerke / Vinko Marolt / Christel Bürzele / Peter Herth / Thomas Burdick, “Visual PCell Programming with Cadence PCell Designer”, *CDNLive! EMEA*, May 2013, Session CUS04, URL: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/company/Events/CDNLive/Secured/Proceedings/EU/2013/CUS04.pdf.

[193] Larry Stockmeyer, “Optimal Orientations of Cells in Slicing Floorplan Designs”, *Information and Control*, vol. 57, no. 2–3, pp. 91–101, May 1983, DOI: 10.1016/S0019-9958(83)80038-2.

[194] Daniel D. Gajski / Robert H. Kuhn, “Guest Editors’ Introduction: New VLSI Tools”, *IEEE Computer*, vol. 16, no. 12, pp. 11–14, Dec. 1983, DOI: 10.1109/MC.1983.1654264.

[195] Daniel Marolt / Thomas Burdick / Göran Jerke / Peter Herth / Vinko Marolt / Jürgen Scheible, “HIPE: Hierarchical Instance Parameter Editing of Parameterized Modules in Analog IC Design”, *Proc. of edaWorkshop 2016*, pp. 18–23, May 2016, ISBN: 978-3-86460-453-9.

[196] Göran Jerke / Vinko Marolt / Christel Bürzele / Daniel Marolt / Andreas Krinke / Peter Herth / Thomas Burdick, “Hierarchical Module Design with Cadence PCell Designer”, *CDNLive! EMEA*, Apr. 2015, Session CUS02, URL: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/company/Events/CDNLive/Secured/Proceedings/EU/2015/CUS02.pdf.

[197] Daniel Marolt / Matthias Greif / Jürgen Scheible / Göran Jerke, “PCDS: A New Approach for the Development of Circuit Generators in Analog IC Design”, *Proc. of 22nd Austrian Workshop on Microelectronics (Austrochip)*, pp. 1–6, Oct. 2014, DOI: 10.1109/Austrochip.2014.6946310.

[198] Matthias Greif / Daniel Marolt / Jürgen Scheible, “gPCDS: An Interactive Tool for Creating Schematic Module Generators in Analog IC Design”, *Proc. of 12th Conference on Ph.D. Research in Microelectronics and Electronics*, pp. 1–4, Jun. 2016, DOI: 10.1109/PRIME.2016.7519539.

[199] Göran Jerke / Vinko Marolt / Christel Bürzele / Peter Herth / Thomas Burdick, “Schematic and Symbol PCell Development with Cadence PCell Designer”, *CDNLive!*

EMEA, May 2017, Session CUS10, URL: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/company/Events/CDNLive/Secured/Proceedings/EU/2017/CUS10.pdf.

[200] Andreas Gerlach / Thoralf Rosahl / Frank-Thomas Eitrich / Jürgen Scheible, "A Generic Topology Selection Method for Analog Circuits with Embedded Circuit Sizing Demonstrated on the OTA Example", *Proc. of Design, Automation and Test in Europe Conference*, pp. 898–901, Mar. 2017, DOI: 10.23919/DATE.2017.7927115.

[201] Florian Leber / Jürgen Scheible, "Eine domänenspezifische Sprache für die prozedurale Dimensionierung im analogen IC Entwurf", *Proc. of Informatics Inside 2017* (edited by Uwe Kloos / Natividad Martínez / Gabriela Tullius), pp. 119–120, May 2017, ISBN: 978-3-00-056455-0.

Further Sources

The bald head in Figure 3.1 [1], Figure 3.2 [1], Figure 3.13 [1], and Figure 3.2 (here) is a derivative of https://www.iconfinder.com/icons/628287/anonym_avatar_default_head_person_unknown_user_icon (created by Anna Litviniuk and licensed as free for commercial use).

The schematic and layout images were generated with the Virtuoso Custom IC Design Environment from Cadence Design Systems, using the Generic Process Design Kit (GPDK).

This thesis was typeset with TeXworks 0.6.2 (implemented by Jonathan Kew), using the document preparation software LaTeX (originally developed by Leslie Lamport), based on the typesetting system TeX (designed by Donald E. Knuth). The source text was written in Notepad++ 7.5.6 (devised by Don Ho). Drawings were created by the author using GIMP 2.6.8 (started by Spencer Kimball and Peter Mattis) and Inkscape 0.48.0 r9654 (initially forked from Sodipodi by Ted Gould, Bryce Harrington, Nathan Hurst, and "MenTaLguY"). The final PDF file was produced via pdfTeX-1.40.19 in November 2020.

Die Reihen der Fortschritt-Berichte VDI:

1 Konstruktionstechnik/Maschinenelemente
2 Fertigungstechnik
3 Verfahrenstechnik
4 Bauingenieurwesen
5 Grund- und Werkstoffe/Kunststoffe
6 Energietechnik
7 Strömungstechnik
8 Mess-, Steuerungs- und Regelungstechnik
9 Elektronik/Mikro- und Nanotechnik
10 Informatik/Kommunikation
11 Schwingungstechnik
12 Verkehrstechnik/Fahrzeugtechnik
13 Fördertechnik/Logistik
14 Landtechnik/Lebensmitteltechnik
15 Umwelttechnik
16 Technik und Wirtschaft
17 Biotechnik/Medizintechnik
18 Mechanik/Bruchmechanik
19 Wärmetechnik/Kältetechnik
20 Rechnerunterstützte Verfahren (CAD, CAM, CAE CAQ, CIM ...)
21 Elektrotechnik
22 Mensch-Maschine-Systeme
23 Technische Gebäudeausrüstung

ISBN 978-3-18-347520-9