Cooperation in Self-Organized Heterogeneous Swarms

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)
im Fachgebiet

Informatik

Vorgelegt

von Diplom-Bioinformatikerin Ruby Louisa Viktoria Moritz

geboren am 8. Juni 1987 in Erlangen

Die Annahme der Dissertation wurde empfohlen von:

1. Professor Dr. Martin Middendorf, Universität Leipzig
2. Professor Dr. Sanaz Mostaghim, Otto-von-Guericke-Universität Magdeburg

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 26.02.2015 mit dem Gesamtprädikat summa cum laude.

Beruflicher Werdegang

Zeitraum	**Abschluss/Tätigkeit, Institution, Ort**
Juli 2006	Abitur, Giebichenstein-Gymnasium “Thomas Müntzer”, Halle (Saale), Deutschland
Oktober 2009	Tutorin, Martin-Luther-Universität Halle-Wittenberg Halle (Saale), Deutschland
September 2011	Bioinformatik Diplom, Martin-Luther-Universität Halle-Wittenberg, Halle (Saale), Deutschland
Februar 2012 bis Dezember 2014	Doktorandin, Universität Leipzig, Leipzig, Deutschland
Oktober 2014	DAAD Stipendiatin (FITweltweit), Advanced Concept Team, ESTEC, ESA, Noordwijk, Niederlande

Acknowledgments

This study has been supported by the European Social Fund (ESF) and the Free State of Saxony within the junior research group 'Schwarm-inspirierte Verfahren zur Optimierung, Selbstorganisation und Ressourceneffizienz'. Further, the German Academic Exchange Service (DAAD) funded an essential part of this study with a 'FITweltweit' scholarship.

My gratitude goes to everyone who helped me in this scientific endeavor, professionally or as moral support. Thank you.

Contents

Part I

Preface

1 On Swarms and Multi-Objectivity

1.1 Artificial Intelligence for Artificial Swarms

Complexity in nature is typically organized by a multitude of decentralized, distributed, and heterogeneous systems. Shaped by natural evolution they can withstand most diverse selective pressures, and the huge plasticity, exceptional flexibility, and high adaptability have resulted in an extraordinary global success. These robust and effective systems serve as inspiration not just for the research of this study. Especially fascinating are the simple mechanisms in swarms of social insects of hundreds sometimes up to millions of individuals, that provide food, maintain a hygienic environment, and safety from predators or competitors. These properties are more rare in human societies than we wish them to be. Yet, where the human brain and decision apparatus is complex, insects neuronal systems are far more simple. Often the higher complexity and potential of neuronal systems is considered as beneficial, i.e. the smarter the better, however, simplicity is cheaper, in nature as well as engineering. If swarms of simple individuals are better in performing simple, but essential, tasks in comparison to centralized systems, then developing strategies and mechanism for swarm behavior is a key discipline.

Although, individuals in an insect swarm appear very similar, we must remember that no two members of a colony are completely alike. In some cases they might look identical on first sight, but they differ in age, physiology, genotype, and hence the phenotype. Indeed, behavioral ecologists identify individual personalities of different ants and bees. These differences render different members of the swarm more or less suitable for specific tasks. A prominent example is the cooperation of leaf cutting ants, where the larger ants carry smaller ants that would otherwise be much slower due to their shorter legs (Hölldobler, 1990). Another example is the task allocation in honeybees. Older honeybees gather food while the younger ones maintain the hive and take care of the brood (Seeley et al., 1985). All these

tasks are essential for the colony, however, leaving the hive to forage is a far more dangerous endeavor and chances are the forager does not return. Leaving the more risky task to the older bees is obviously a better strategy than the other way round, because then there would never be any old bees.

Artificial swarms are generally exposed to multiple tasks just like natural swarms. These tasks require certain skills of the swarm members. Swarms of homogeneous agents may therefore be less efficient than those comprised of heterogeneous agents with individual skills. Some tasks, that are called multi-robot or multi-agent tasks, require the agents to cooperate in order to complete the task efficiently. The problem of assigning single agents to a set of single-agent tasks is profoundly studied in the literature (for an overview see Gerkey and Mataric (2004)). However, the problem of assigning teams of agents to multi-agent tasks is less studied. A formalization of the problem is available (Lau and Zhang, 2003) but only few distributed strategies have been investigated (Shehory and Kraus, 1996).

The central problem of the multi-agent task allocation lies in the detection of a partition of a set of agents with different capabilities into teams or coalitions. These teams are allocated to the available multi-agent tasks. The optimal partition and allocation maximizes some given utility function but it is NP-hard to infer an optimal partition, i.e. it is not possible to solve this problem in polynomial time (if $P \neq NP$).

Another interesting aspect of the coalition formation problem lies within the required communication for cooperation and whether decisions to form groups can be made decentralized or not (e.g. Li and Sycara (2004); Procaccia and Rosenschein (2006)). The benefit of the cooperation usually depends on the cost of communication and the compromises that have to be made. In general, essential information should be communicated if the communication cost is lower than the one for retrieving the information by other means.

1.2 The Trouble with Multi-Objectivity

Most decisions that have to be made in daily life or industrial, engineering, or scientific contexts need to consider multiple aspects. A very common example is the nearly daily conflict we face between price and quality when we make a purchase, or when we want to optimize comfort, time spent, and our health in our choice of transportation. Obviously, we provide these objectives with some personal weights and reduce it to a single objective problem. However, when these weights are not known, such a simplification

is impossible. Formally, a *multi-objective optimization problem* asks for solutions from a solution space X (also called search space) that are optimal with respect to $k > 1$ objectives. Typically, the objectives are conflicting with each other and it is impossible to simultaneously optimize all of them. This discrepancy is covered by the concept of the *Pareto dominance* relation. One solution dominates another one if it is better in at least one objective and not worse in all other objectives. Further, a solution is called *Pareto optimal* if it is not dominated by any other solution from X. The set of all Pareto optimal solutions is denoted as the *Pareto set*. The detection of this set is the main goal in multi-objective optimization. Unfortunately, many discrete multi-objective optimization problems are NP-hard. Continuous multi-objective optimization problems, on the other hand, often have optimization functions that are either a black box or algebraically too complicated, i.e. it is impossible to solve them analytically. To avoid immense computational effort, the goal is often reduced to find a set of non-dominated solutions that are close to the Pareto optimal solutions. Another selective feature of these sets is diversity in their solutions.

A problem of Pareto sets and sets of solutions that are close to the Pareto set is their size, especially if there are many objectives. Typically, the more objectives are to be considered, the more solutions are contained by the Pareto set. This effect has been called the 'curse of dimensionality' (Kukkonen and Lampinen, 2007).

Metaheuristics, e.g. genetic algorithms (GA), particle swarm optimization (PSO), or ant colony optimization (ACO), are typically applied, if the optimal solution is computationally too intense. While these algorithms were initially designed to solve single-objective problems they have been extended to solve multi-objective problems as well. Today, there are a multitude of multi-objective variants of various population- based metaheuristics, e.g. the Vector Evaluation Genetic Algorithm (VEGA) (Schaffer, 1984) or the well known non-dominated sorting genetic algorithm (NSGAII) (Deb et al., 2002) (for overviews refer to Coello Coello (2009); Coello Coello et al. (2005); Fonseca and Fleming (1995)). Aside from these genetic algorithms, there are also multi-objective extensions of the ACO, e.g. by Doerner et al. (2001); Iredi et al. (2001) (for overviews see Angus and Woodward (2009); Leguizamón and Coello Coello (2011)). For problems on continuous solution spaces extensions of the PSO have been proposed, e.g. by Coello Coello and Salazar Lechuga (2002) and Hu and Eberhart (2002) (for an overview see Reyes-Sierra and Coello Coello (2006)).

In population-based metaheuristics new solutions are constructed from a set of previously selected solutions. Often the Pareto dominance relation

is used to select the solutions for the population. Unless the population is allowed to contain an arbitrary number of solutions, the use of the Pareto dominance relation will select more solutions than the population can hold. Therefore, it is necessary to compare and rank elements of sets of (non-dominated) solutions. Clearly, the choice of which solutions are to be kept in the population is particularly important for small population sizes. Small populations give some advantages with respect to memory and time requirements of an algorithm. This is of great importance in restrictive computational environments, e.g. when solutions need to be delivered in real time or the algorithm runs on specific hardware, e.g. as part of an embedded system.

1.3 Multi-Objective Tasks in Swarms

Multi-agent tasks can be multi-objective tasks, as well. When a team is confronted with multi-objective multi-agent tasks and has to decide which one should be performed first, all available tasks need to be ranked by the team members. Either all members use the same ranking method or – more interestingly – they use individual ranking methods. In the latter case the swarm is heterogeneous in the decision apparatus. Hence, the assessment of swarm performance under different distributions of ranking schemes are a major topic of this thesis. While these distributions can be set manually, the agents of the swarm can use evolutionary methods to propagate their own ranking scheme if it is beneficial. To this end I introduce different evolutionary methods that are inspired by natural phenomenons.

While most genetic and evolutionary algorithms presume haploidy in their recombination mechanisms, i.e. both recombination partners provide one copy of their genetic material, there are also a few approaches that consider diploid systems, e.g. by Goldberg and Smith (1987). In diploid systems all individuals have two, usually distinct, sets of chromosomes, i.e. their genetic material. In diploid dominance systems one set is expressed like a haploid set while the second set has no further influence and is solely carried along. This enables the population to maintain a higher diversity and allows for keeping genetic material over many generations until it might become useful again.

While diploid systems are well studied, the concept of haplo-diploidy has been disregarded altogether by researches in evolutionary algorithms and other artificial mechanism, so far. Yet, haplo-diploidy is a phenomenon that is common in many animal systems where males are haploid and females are

diploid, including all Hymenoptera (bees, ants, wasps) and also some beetles and mites (Crozier, 1977). This is due to the fact that females develop from fertilized eggs while males develop from unfertilized eggs.

1.4 Structure of this Thesis

This study is split into three main parts. Part II introduces the required concepts of multi-agent systems and multi-agent tasks (Moritz and Middendorf, 2013, 2014a,b). In Part III the different ranking relations are introduced and theoretically and empirically analyzed (Moritz et al., 2014a,b, 2013). In Part IV I analyze different strategies to solve multi-objective multi-agent tasks in dynamic environments.

Part II

Cooperation in Heterogeneous Swarms

2 Introduction

Multi-agent tasks and reconfigurability

Swarms can consist of a number of identical individuals, however most swarms – natural and artificial – are heterogeneous to some extent. Different members of the swarm show different demeanor and abilities. This is usually the case, because the swarm members were initially designed to perform specific individual tasks. They might be able to solve other tasks as well, but these may be less suited and solved less efficiently. Yet in emergency situations it can be imperative for some swarm members to engage in tasks they have not been designed for, in order to maintain the systems integrity. Further on, I refer to swarm members as *agents*.

Some tasks only require the attention of a single agent. However, the swarm can also be confronted with *multi-agent tasks* that require a collaboration of multiple agents to be completed efficiently. Only teams of agents can process these tasks creating the new task of forming such a team. A formalization for the problem of assigning teams of agents to multi-agent tasks is given in Lau and Zhang (2003).

Given is a set of agents that have different capabilities and have to be partitioned into teams. These teams are then allocated to the available tasks to some benefit defined by a given utility function. The problem to maximize this benefit is an NP-hard problem and there are several heuristics for this problem (for an overview see Vig and Adams (2006)).

2.1 State of the Art

In dynamic environments multi-agent societies that solve multi- agent tasks can benefit from adaptability. Most approaches, as reviewed by Alberola et al. (2014), consider systems where the adaptation process is structured into the following phases: monitoring, design, selection, and evaluation.

One relatively simple approach assumes a (static) social interaction network between the agents where each agent has a single fixed skill from a set of multiple skills. In order to execute a task agents have to form a team that resembles an induced connected subgraph of the given network. Tasks enter the system dynamically and each task is characterized by a vector that describes for each skill how many agents with that skill are required in the team.

Other strategies in decentralized team formation have been studied (Gaston and DesJardins, 2008), however, most studies on this topic are based on either very complex agents or complex cooperation scenarios, e.g. Lichocki et al. (2013); Recchia et al. (2014). The study of a simpler model by Shehory and Kraus (1996) focused on distributed scenarios that are (i) not necessarily super-additive, i.e. merging small groups into larger must not be beneficial, (ii) where the agents behave group rational, i.e. they only form a group if it is profitable for the whole group, and (iii) where the agents differ in their capabilities. An agent is characterized by a vector $(a_1, \ldots, a_k)$ where a_r, $r \in \{1, \ldots, k\}$ gives its capability to perform a some action r. The groups capability is the joint capability of its member agents, where agents can be member of several groups. The groups capability has to satisfy a threshold for each type of action. Shehory and Kraus (1996) set a clear focus on the computational effort of the distributed group formation algorithm in relation to the quality of the resulting groups. In subsequent studies these algorithms were further improved (Rahwan, 2007).

Khalouzadeh et al. (2010) studied coalition formation in mobile agents that execute tasks in different locations. Several agents with different capabilities can be required to form a coalition in order to execute a task. These coalitions can be formed by agents in close proximity to the task. However, In the study of Khalouzadeh et al. (2010) all agents had access to global information about the positions and capabilities of all other agents.

More research in the scope of self-organized coalition formation by mobile agents located within an arena was published by Singh et al. (2010). They proposed a system, where agents only connect to other agents if it is beneficial with respect to some payoff function. Sets of connected agents are considered to be a coalition that has to pay coordination costs. Singh et al. (2010) focused on the influence of different payoff functions and coordination costs in the coalition sizes.

In a system of Abramson (2008) the agents also move within an arena and form clusters as a concept of cooperation. By a trial and error approach the agents decide on their cooperative partners. Their conflict lies in achieving a social optimum in the long term and an individual optimum in

the short term. A special application of coalition formation has been studied by Bölöni et al. (2007) where agents need to traverse dangerous locations in the aftermath of a disaster.

Another interesting aspect of the coalition formation problem lies within the required communication for cooperation and whether decisions to form groups can be made decentralized (e.g. Li and Sycara (2004); Procaccia and Rosenschein (2006)). The amount and type of communication is a determining factor especially in groups containing different types of agents with different capabilities and needs. Agents have to communicate to make compromises and decisions within a group. Hence, the benefit of the cooperation depends on the cost of communication and the compromises that have to be made. Kutanoglu and Wu (2007) studied a possible measure to make local decisions within the coalition to schedule tasks in order to reduce or avoid global communication between agents.

2.2 Adaptive Team Formation in Heterogeneous Swarms

In this study I analyze agents performing a foraging task. A swarm of heterogeneous agents strives to collect a multitude of different resources, e.g. food, construction material, or pollution. The collection of different resource types requires different skills, e.g. cutting, carrying, or digging. The resources are located within a two-dimensional arena in which the agents move and can form groups, i.e. coalitions, of agents that collect in unison. Agents of the same group have access to each others skills. Since the resources in the arena are heterogeneously distributed and dynamic over time, the agents are forced to move and dynamically form groups for efficient foraging. Providing the agents with the ability of reconfiguration increases their adaptability in dynamic scenarios. It also works as a game changer in the coalition formation problem. The focus of the problem moves from the agents capabilities, that can be reconfigured, to the size of the coalition. Of course reconfiguration comes at a cost. It is essential that these costs do not exceed the overall benefit of the group.

This study focuses on systems with limited communication, where agents can only communicate with other agents if they are in close proximity. In real life applications the members of artificial swarms could be designed to perform a certain set of tasks. High-level communication or coordination must not be included in this set of tasks. In general, this study focuses in methods to perform certain tasks with very simple strategies to handle

the swarms coordination. The task of swarm coordination is superimposed and the capabilities of the agents for communication and coordination are assumed to be minimal as it is not assumed that the main tasks of the swarm requires communication or coordination.

While their communication is limited the swarm members are qualified to perform a set of tasks using the tools or hardware they are equipped with. Under the assumption that these tools can be exchanged or the hardware reconfigured, new problems and solutions in the team formation problem arise which are the central issue in this study. The existence of reconfigurable hardware, e.g. field programmable gate arrays (FPGA), and current efforts of producing devices that fit into an Universal Serial Bus (USB) slot and are thus easily switched make this assumption less ambitious than it may seem at first.

I will first introduce the basic model developed to simulate the heterogeneity, resource availability, and agent movement is introduced (Chapter 3). Chapter 4 compromises comprehensive analysis of the group formation problem in a non-reconfigurable system, including experiments and results. An extension of the model to allow reconfigurability and an extensive analysis of different group formation approaches is given in Chapter 5.

Most methods and results presented in this part are published in Moritz and Middendorf (2013, 2014a,b).

3 The Model

Modeling (reconfigurable) agents

The design of the model presented in the following is kept as simple as possible, reducing the number of parameters to a minimum to enable a clear and coherent analysis of the few parameters of interest. The abilities and knowledge of the agents are very limited in order to investigate a general scenario that does not depend on assumptions from specific applications.

The multi-agent systems consist of a set of n agents $A = \{a_1, \ldots, a_n\}$ located in a two-dimensional torus arena of $d \times d$ fields $F = \{f^{0,0}, f^{0,1}, \ldots f^{d,d}\}$ (i.e. the last field in every row and column is adjacent to the corresponding first field in its row, respectively column). The agents are confronted with a resource collection task, with k different types of resources. Each field $f \in F$ contains f_j of resource type $j \in \{1, \ldots, k\}$. However, these values are not fixed and a field can have a high value for f_j at some time t and a low value at another time t', where t and t' are both part of the same simulation. Every agent has the task to collect as many resources as possible. Simulations are turn based, i.e. time discrete.

3.1 Slot Model

Throughout this study the agents capabilities are described by a simple *slot model*. These are commonly used to model FPGAs based on the notion that they have several operational units that can perform simple tasks (examples are Ou and Prasanna (2010); Rullmann and Merker (2011)). In these models, a slot (also called slice or frame) is the smallest relevant unit of the configuration. If the architecture is reconfigurable one slot may be able to perform different tasks at different times, but only one at any specific time.

Each agent a has s slots, and each slot is in one of k possible states, where k is the number of different resources types. The set of resources types is

given by $R = \{r_1, \ldots, r_k\}$. The configuration of an agent assigns a state to every slot. Each slot in state $i \in \{1, \ldots, k\}$ increases the agents capability in the collection of resources of type r_i. Figure 3.1 shows an exemplary configuration for an agent with eight slots in a model with three different types of resources.

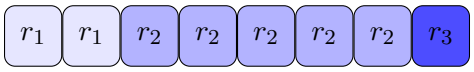


Figure 3.1: Configuration of an agent a_i with $s = 8$ slots; the brightness indicates the current state (1 = light, 2 = medium, 3 = dark); r_i with $i \in \{1, 2, 3\}$ in a slot denotes the resource type whose collection this slot enhances; vector (a_{i1}, a_{i2}, a_{i3}) describes the skills of a_i, i.e. a_{ij} denotes the number of slots in state j.

Let agent a_i have a_{ij} of its s slots in state j in order to collect resource type $r_j \in R$. If agent a_i is located on field $f \in F$ then it can collect an amount of $(a_{ij} f_j)/s$ of resource type r_j per simulation turn. The performance $P(a_i)$ of agent a_i is defined as the total amount of resources the agent can collect in one simulation turn.

$$P(a_i) = \frac{1}{s} \sum_{j=1}^{k} a_{ij} f_j \tag{3.1}$$

Note, that with $f_j \in [0, 1]$, $j \in \{1, \ldots, k\}$, and $\sum_{j=1}^{k} a_{ij} = s$ the value of $P(a_i)$ is in $[0, 1]$. In general, $P(a_i) \in [\min f_j, \max f_j]$, where $P(a_i)$ is a weighted mean of the f_j, $j \in \{1, \ldots, k\}$.

3.2 Capabilities in Groups

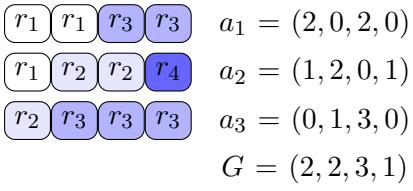


Figure 3.2: Example of a group G with three agents and their capabilities with $k = 4$ resource types and $s = 4$ slots per agent.

In order to increase its performance an agent can join a group or recruit other agents and start a group of its own. While it is said that a group is as strong as its weakest member, here groups are as strong as their strongest

member. This way a small group potentially benefits more from the recruitment of a new agent than a larger group. Formally, the capability a_{ij} of agent a in group G is increased to the maximal value $\max_{a' \in G} a'_j$ of all agents in the group (see Figure 3.2). Hence, if an agent is not better in the collection of at least one resource type than all other agents of the group, the group has no benefit from this agent. Unless, the agent is able to reconfigure its slots to become a profitable member of the group, the group should not recruit this agent. However, with reconfigurable capabilities a group with k members has at some point, through reconfiguration, one agent specialized on one specific resource type for all resource types. Any additional agent does not provide any benefit for the group. Equation (3.2) gives the formal definition of the performance $P(G)$ of a group G.

$$P(G) = \frac{1}{s} \sum_{j=1}^{k} \max_{a_i \in G} a_{ij} f_j \qquad P(G) \in [0, k] \tag{3.2}$$

3.3 Reconfiguration

A reconfigurable agent is able to change the state of its slots by a reconfiguration operation. Following the predicament of simplicity in agent design the reconfiguration is based on a simple random process called *mutation*. This is motivated by the similar mutation process occurring in nature within DNA. Due to natural selection genes essential for survival are conserved in the DNA, while other parts of the DNA with less or no impact on the individuals fitness tend to show higher diversity within a population as more mutations accumulate here. A slot can either be of benefit to the group or not. In the latter case I refer to the slot as *idle*. An idle slot is of no benefit to the group and – like DNA – starts to change its state randomly. Formally, a slot of an agent $a \in G$ that is in state j is called idle if one of the following conditions hold:

1. $f_j = 0$,
2. there is another agent $a' \in G$ with $a_j < a'_j$, or
3. there exists another agent $a' \in G$ with $a_j = a'_j$ and a' joined G before a.

In the first idle simulation turn a slot has a probability of $\mu \in [0, 1]$ to mutate during that simulation turn, μ being called the *mutation strength*. In each subsequent simulation turn that this slot remains idle its mutation

probability increases by μ. After x idle simulation turns a slots mutation probability is $x\mu$. The mutation probability of a slot is reset to zero if the slot (1) becomes active (i.e. it is not idle) or (2) mutates.

A mutation is the change of a slots state into a randomly chosen new state from $\{1, \ldots, k\}$ with a uniform distribution. The process of the mutation interferes with the productivity of the reconfiguring agent. This interference is the *reconfiguration cost* $c_\mu \in [0, 1]$. The cost is payed by a reduced performance such that for a simulation turn where a slot of an agent $a \in G$ mutated its performance is reduced to $(1 - c_\mu)P(G)$. The reconfiguration costs are independent of the number of the slots that mutate, i.e. the costs for two mutating slots are identical to the costs of one mutating slot.

3.4 Movement

Agents have a position and an orientation within the arena (see Figure 3.3). Their position is given by the coordinates of the field they reside on. Their orientation points onto one of the eight fields in the Moore-neighborhood of their current location. During one simulation turn they can either rotate and change their orientation by 45° or move onto the adjacent field in their current orientation. If the agent remains in its current position then (i) if it was rotating in the previous simulation turn it keeps rotating 45° in the same direction or (ii) if it was moving in the previous simulation turn it rotates by 45° in a random direction. This way the agents avoid inefficient back and forth rotations and can avoid 'bad' fields more effectively.

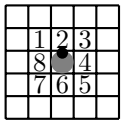


Figure 3.3: Orientation and location of an agent within a small part of the arena facing the field on top of it.

All members of a group are located in the same field where they collect resources collectively. There can not be two agents on the same field unless they are in a group. Note, that the size of a field gives the size of the area an agent or group can process during one simulation turn and not their physical size. The physical size of an agent is considered here to be so small compared to the size of a field that it is negligible. The assumption that at most one group of agents is located on a field is motivated by the fact that several groups on a field would hinder each other in their work or at least would have to coordinate their work. To ensure their common location agents of the same group move together and are oriented in the same direction. If a

new agent has been recruited by the group the new agent adapts itself to the orientation and location of the group.

In order to disrupt the system and especially the group formation process the agents are equipped with a battery that has to be reloaded once its empty. During this reloading time the agents are inactive. They dislodge from their group and stop moving until the battery is fully reloaded. The group continues to move and leaves the inactive agent behind but has to make up for the loss of the agent and its capabilities. This keeps the whole system in a constant struggle to keep effective groups despite regular loss of group members. Inactive agents block their field for other groups, as it is unclear to other agents, when it will reactivate again. Inactive agents can not be member of any group.

4 || Non-Reconfigurable Swarms

The team formation problem

The different configurations of the agents and the synergetic effect of the groups pose an optimization problem where we need to maximize the capabilities of the groups in the absence of superfluous agents. Obviously, this is a problem when agents cannot reconfigure.

Section 4.1 gives a proof of NP-completeness for the *Optimal Partition Problem* which is somewhat related to the Coalition Formation Problem, but based on different assumptions. Section 4.2 presents the methods developed in this study to achieve the formation of balanced and reasonably sized groups of non- reconfigured agents. In Section 4.3 the empiric study and its results are presented.

4.1 The Optimal Partition Problem

The Optimal Partition Problem occurs in systems of non-reconfigurable agents. Under the assumption that two groups in direct proximity can exchange member agents or even merge into one group and that a group is not allowed to contain a superfluous agent, the optimal regrouping is NP-hard. Optimality is defined by the average capabilities – with the synergetic effect – of the grouped agents. In the following formal proof the agents are simplified to binary vectors, where their capabilities for any resource can be at most 1 and is 0 otherwise. During this proof we reduce the well known NP-complete 3-SAT problem to the Optimal Partition Problem.

4.1.1 Definitions and Notations

Given is a set $A = \{a_1, \ldots, a_{|A|}\}$ of binary k-dimensional vectors $a_i \in \{0,1\}^k$, $i \in \{1, \ldots, |A|\}$, $k \in \mathbb{N}$. Let $\mathcal{P}$ be a partition of A. Dimension $j \in \{1, \ldots, k\}$

is *covered* by a set $S \in \mathcal{P}$ if there exists a vector $a_i \in S$ such that $a_{ij} = 1$. The *cover* of S is defined by the number of covered dimensions and is written as $\|S\|$.

$$cover(S) = \|S\| = \sum_{j=1}^{k} s_j \qquad \text{with} \qquad s_j = \begin{cases} 1 & \text{if } j \text{ is covered in } S \\ 0 & \text{otherwise.} \end{cases}$$

The score of S is the size of S times the number of dimensions that are covered by S:

$$score(S) = |S|\|S\|$$

The score of a partition of a set of vectors is the sum of the scores of the sets in the partition. A set $S \in \mathcal{P}$ is valid, if for each vector $a \in S$ there is some $j \in \{1, \ldots, k\}$ such that $a_j = 1$ and for all vectors $a' \in S$, $a \neq a'$, $a' = 0$. In this case vector a is *unique* in j for S.

Hence, a vector can only be part of a valid set S, if its absence reduces the number of dimensions covered by S. A vector a is invalid in a set S if for each position j with $a_j = 1$ there exists another vector $a' \neq a$ in S with $a'_j = 1$. A partition is valid, if all its sets are valid. The Optimal Partition problem is to find a valid partition with a score $\geq \mathsf{K}$.

4.1.2 Proof of NP-Completeness

Theorem 1. *The Optimal Partition problem is NP-complete.*

Proof. Let $\mathcal{I} = (\mathcal{C}, V)$ be an instance of the 3-SAT problem with a set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of clauses of size three over a set $V = \{v_1, \ldots, v_n\}$ of variables. For simplicity the clauses and their literals are indexed such that

$$\chi(3(r-1)+l),\ l \in \{1,2,3\}, r \in \{1, \ldots, m\}$$

is the l-th literal of the r-th clause C_r.

In the following we construct an instance A of the Optimal Partition problem. The size of each vector $a \in \{0,1\}^k$ is $k = 3m + n$. For each clause $C_r \in \mathcal{C}$ there is a vector $a^{C_r} \in A$. Vector a^{C_r} contains exactly three 1-values at positions $(3r-2)$, $(3r-1)$, and $3r$. Observe that no other clause-vector has a 1 in these positions.

Additionally there are two vectors $a^{v_i}, a^{\overline{v_i}} \in A$ for every variable $v_i \in V$. Vector a^{v_i} contains a 1 at every position of a positive occurrence of v_i in $\mathcal{C}$ and at position $3m + i$. We define a^{v_i}, $a^{\overline{v_i}}$ and an additional vector a^{ANTI}

with

$$a_j^{v_i} = \begin{cases} 1 & \text{if } \chi(j) = v_i \text{ or } j = 3m + i \\ 0 & \text{otherwise} \end{cases}$$

$$a_j^{\overline{v_i}} = \begin{cases} 1 & \text{if } \chi(j) = \overline{v_i} \text{ or } j = 3m + i \\ 0 & \text{otherwise} \end{cases}$$

$$a_j^{ANTI} = \begin{cases} 1 & \text{if } j \in \{1, \ldots, 3m\} \\ 0 & \text{otherwise.} \end{cases}$$

The score of the Optimal Partition is $\geq \mathsf{K} = 3m^2 + 7mn + 3m + 2n^2 + 2n$. This concludes the construction of A. Further, we make the following observations.

1. For every position $j = 3(r-1) + l \leq 3m$, $l \in \{1, 2, 3\}$, $r \in \{1, \ldots, m\}$ there are exactly three vectors in A (namely $a^{\chi(j)}$, a^{C_r}, a^{ANTI}) with a 1-value in that position.

2. For every position $3m + i, i \in \{1, \ldots, m\}$ there are exactly two vectors in A (a^{v_i}, $a^{\overline{v_i}}$) with a 1-value in that position.

3. A set S that contains any a^{C_r} and a^{ANTI} is not valid.

4. If all vectors a^{C_r} are in the same set all dimensions from 1 to $3m$ are covered.

5. Apart from the vector a^{ANTI} the only way to make a clause-vector a^{C_r} an invalid member of a set is to add the respective three variable-vectors of the literals occurring in C_r.

Claim 1.1. *For each set S_i in a valid partition $S_1, \ldots, S_h$ of $\mathcal{P}$ with score $\geq \mathsf{K}$ it holds that all k dimension are covered by S_i.*

Proof of Claim 1.1. The claim follows from the facts that

- the number of vectors in A is $|A| = m + 2n + 1$,
- their length is given with $k = 3m + n$, and
- the score K is given by $\mathsf{K} = 3m^2 + 7mn + 3m + 2n^2 + n = |A|k$

□

Claim 1.2. *Any valid partition $\mathcal{P}$ with a score $\geq \mathsf{K}$ consists of at most two sets.*

Proof of Claim 1.2. From Observation 1 and 2 follows that the total number of ones in the vectors of A is $3 \cdot 3m + 2n$. As Claim 1.1 holds there should be $\geq k = 3m + n$ 1-values per set. Clearly

$$\frac{3 \cdot 3m + 2n}{k} = \frac{3 \cdot 3m + 2n}{3m + n} < 3$$

and thus there are at most two sets in any solution with a score $\geq \mathsf{K}$. □

Now we can show that there exists a valid partition of A with a score $\geq \mathsf{K}$, if and only if the given instance of 3-SAT is satisfiable.

⇐: We partition A such that all vectors $a^{C_r} \in A, r \in \{1, \ldots, m\}$ are in S_1 and a^{ANTI} is in S_2. For every variable $v_i \in V$ we put a^{v_i} into S_1 and $a^{\overline{v_i}}$ into S_2, if v_i is assigned with *false* in the satisfiable solution of $\mathcal{I}$. If v_i is assigned with *true* in the solution of $\mathcal{I}$, a^{v_i} is added to S_2 and $a^{\overline{v_i}}$ is added to S_1.

All vectors a^{C_r} are unique in at least one dimension in S_1. Let the l-th literal of C_r be *true* (as the instance of 3-SAT is satisfiable, there is at least one) then a^{C_r} is unique in dimension $(3(r-1)+l)$ in S_1. The other two vectors with a 1-value in that position (namely $a^{\chi(3(r-1)+l)}$, a^{ANTI}) are both in S_2.

The positive variable-vectors a^{v_i} are separated from the respective negative $a^{\overline{v_i}}$ and thus both are unique in $3m+i$ in their respective set and thus valid. Vector a^{ANTI} is valid in S_2 as it is unique in every dimension $j \in \{1, \ldots, 3m\}$ that is covered by S_1 with two vectors. As all vectors are valid, so are the two sets S_1 and S_2 and the partition $\mathcal{P}$.

The resulting sets are completely covered by their vectors: Dimensions 1 to $3m$ by a^{ANTI} in S_2 and by the clause-vectors a^{C_r} in S_1 (Observation 4). Hence the score of the partition $\{S_1, S_2\}$ is $\geq \mathsf{K}$.

⇒: The truth-value assignment for the variables is given by the variable-vectors contained in the set with the vector a^{ANTI}, if there are exactly two sets in the optimal partition.

If it is possible to find the optimal partition in polynomial time for this constructed instance, then the respective algorithm can solve any instance of 3-SAT in polynomial time as well. □

4.1.3 A Restricted Version of the Optimal Partition Problem

In the following all vectors of A have the same number of 1-values. The previous construction fails this restriction, as the vector a^{ANTI} has more 1-values than any other vector. Another problem is that the number of 1-values contained in the vectors a^{v_i} depends on the problem instance. This problem can be addressed with a simplified version of 3-SAT where every variable occurs exactly four times, 3,4-SAT. This version of the 3-SAT problem is NP-complete Tovey (1984).

Theorem 2. *The Optimal Partition problem is NP-complete even when each vector has the same number of 5 ones.*

Proof. Let $\mathcal{I} = (\mathcal{C}, V)$ be an instance of the 3,4-SAT problem with clauses $\mathcal{C} = \{C_1, \ldots, C_m\}$ over the variables $V = \{v_1 \ldots, v_n\}$, where $n = \frac{3}{4}m$. We expand $\mathcal{I}$ to $\mathcal{I}_8 = (\mathcal{C}_8, V_8)$ where each literal v_i and $\overline{v_i}$ occurs exactly 4 times. This is done by adding an extra variable v_i' and four clauses C_1^i, C_2^i, C_3^i, and C_4^i for every variable $v_i \in V$ such that for $l \in \{1, 2, 3, 4\}$

$$C_l^i = (\widetilde{v_i}, v_i', \overline{v_i'}), \text{ with } \widetilde{v_i} = \begin{cases} v_i & \text{if the number of } v_i \in V \text{ is } < l \\ \overline{v_i} & \text{otherwise.} \end{cases}$$

The literals v_i' and $\overline{v_i'}$ occur exactly 4 times, once in each of the new clauses. Note that $n_8 = \frac{8}{3}m_8$. In the following we exclusively consider $\mathcal{I}_8 = (\mathcal{C}_8, V_8)$ and omit the index 8 for simplicity. We construct an instance A of the Optimal Partition problem, where every vector has exactly five 1-values. The size of each vector $a \in \{0, 1\}^k$ is $k = 9n + 2 + 5p$ with $p = 4n^2 + 3$. For each clause $C_r \in \mathcal{C}$ there is a vector $a^{C_r} \in A$ with

$$a_j^{C_r} = \begin{cases} 1 & \text{if } 3(r-1) < j \leq 3r \\ & \text{or } j = 8n + 1 \\ & \text{or } j = 8n + 2 \\ 0 & \text{otherwise} \end{cases}$$

Additionally there are two vectors $a^{v_i}, a^{\overline{v_i}} \in A$ for every variable $v_i \in V$. Vector a^{v_i} contains a 1 at every position of a positive occurrence of v_i in $\mathcal{C}$

and in position $8n + 2 + i$. $a^{\overline{v_i}}$ is analogously defined.

$$a_j^{v_i} = \begin{cases} 1 & \text{if } \chi(j) = v_i \\ & \text{or } j = 8n + 2 + i \\ 0 & \text{otherwise.} \end{cases}$$

$$a_j^{\overline{v_i}} = \begin{cases} 1 & \text{if } \chi(j) = \overline{v_i} \\ & \text{or } j = 8n + 2 + i \\ 0 & \text{otherwise.} \end{cases}$$

Finally there are p vectors $a^{x_l}, l \in \{1, \ldots, p\}$ with:

$$a_j^{x_l} = \begin{cases} 1 & \text{if } j \in \{9n + 2 + 5l - 4, \ldots, \leq 9n + 2 + 5l\} \\ 0 & \text{otherwise.} \end{cases}$$

We conclude the construction of A by setting

$$\mathsf{K} = (9n + 2 + 5p)\left(\frac{11}{3}n + p\right) + 5n^2.$$

This construction leads to the following observations:

1. There are $|A| = m + 2n + p$ vectors in A.
2. There are $5(m + 2n + p)$ 1-values in A.
3. In the $5p$ positions $9n + 2$ to $9n + 2 + 5p$ a 1-value occurs only in a single vector (namely vectors a^x).
4. In the $9n$ positions 1 to $8n$ and $8n + 2$ to $9n + 2$ a 1-value occurs only in two vectors (namely vectors a^{C_r} and a^{v_i}).
5. In the 2 positions $8n + 1$ and $8n + 2$ a 1-values occurs in m vectors (namely vectors a^{C_r}).
6. If all vectors are in one set, the vectors a^{C_r} and a^{v_i} an $a^{\overline{v_i}}$ are not valid. Hence, there are at least two sets in $\mathcal{P}$.

In the following we show how to improve the score of any valid partition of A.

Claim 2.1. *For any partition $\mathcal{P}^{>2} = \{S_1^{>2}, \ldots, S_h^{>2}\}$ with $h > 2$ there is a partition $\mathcal{P}^2 = \{S_1^2, S_2^2\}$ with score$(\mathcal{P}^2) >$ score$(\mathcal{P}^{>2})$.*

Proof of Claim 2.1. Without loss of generality, let $S_1^{>2}$ and $S_2^{>2}$ be the sets with the highest score in $\mathcal{P}^{>2}$. Any vector $v \notin S_1^{>2} \cup S_2^{>2}$ can be added to either $S_1^{>2}$ or $S_2^{>2}$ without violating the validity of the partition. By construction and Observation 3 to 5 it is clear that for every vector $v \in A$ there is at least one position $j \in \{1, \ldots, k\}$ with $v_j = 1$ where at most one

other vector $v' \in A$, $v' \neq v$ with $v'_j = 1$ exists. Vector v is placed in the respective other set than vector v'. It is clear that the final partition into two sets has an overall score that is higher than the starting partition, as the scores of S_1 and S_2 increased and thus each vector is now in a set with a higher (or even) score than they had before. □

Claim 2.2. *For any partition $\mathcal{P} = \{S_1, S_2\}$ with vectors a^x in S_1 and S_2, there is a partition $\mathcal{P}^x = \{S_1^x, S_2^x\}$ where the p vectors a^x are member of S_1 with $score(\mathcal{P}^x) > score(\mathcal{P})$.*

Proof of Claim 2.2. We define

$$S'_1 = \{a^{C_r} \in S_1, a^{v_i} \in S_1\}$$
$$S'_2 = \{a^{C_r} \in S_2, a^{v_i} \in S_2\}.$$

Assume, in S_1 are $p-y$ vectors a^x and y vectors a^x are in S_2, $1 \leq y < k$. Without loss of generality

$$\|S'_1\| + 5|S'_1| \geq \|S'_2\| + 5|S'_2|.$$

The score of $\mathcal{P}$ is determined by

$$\begin{aligned} score(\mathcal{P}) &= \underbrace{(|S'_1| + (p-y))}\underbrace{(\|S'_1\| + 5(p-y))} + \underbrace{(|S'_2| + y)}\underbrace{(\|S'_2\| + 5y)} \\ &= \quad |S_1| \quad\quad\quad \|S_1\| \quad + \quad |S_2| \quad\quad \|S_2\| \\ score(S_1) &= |S'_1|\|S'_1\| + (5|S'_1| + \|S'_1\|)(p-y) + 5(p-y)^2 \\ score(S_2) &= |S'_2|\|S'_2\| + (5|S'_2| + \|S'_2\|)y + 5y^2 \end{aligned}$$

Let $\mathcal{P}^x$ be the partition that is obtained from $\mathcal{P}$ by moving all vectors a^x that are in S_2 into S_1. Hence, $S_2^x = S'_2$. The score of $\mathcal{P}^x$ is determined by:

$$\begin{aligned} score(\mathcal{P}^x) &= \underbrace{(|S'_1| + p)}\underbrace{(\|S'_1\| + 5p)} + |S'_2|\|S'_2\| \\ &= \quad |S_1^x| \quad\quad \|S_1^x\| \quad + |S'_2|\|S'_2\| \\ &= |S'_1|\|S'_1\| + (5|S'_1| + \|S'_1\|)p + 5p^2 + |S'_2|\|S'_2\| \end{aligned}$$

In the following we show that the score of $\mathcal{P}^x$ is always higher than the score of $\mathcal{P}$. Since $p > y$ and $5|S_2'| + \|S_2'\| \leq 5|S_1'| + \|S_1'\|$

$$\begin{aligned}
\Rightarrow \quad & 5|S_2'| + \|S_2'\| - 10(p-y) < 5|S_1'| + \|S_1'\| \\
\Leftrightarrow \quad & -(5|S_1'| + \|S_1'\|) - 10(p-y) + 5|S_2'| + \|S_2'\| < 0 \\
\Leftrightarrow \quad & (5|S_1'| + \|S_1'\|)(-y) - 10y(p-y) + (5|S_2'| + \|S_2'\|)y < 0 \\
\Leftrightarrow \quad & |S_1'|\|S_1'\| + (5|S_1'| + \|S_1'\|)(p-y) + 5p^2 \\
& -10y(p-y) + |S_2'|\|S_2'\| + (5|S_2'| + \|S_2'\|)y < \\
& |S_1'|\|S_1'\| + (5|S_1'| + \|S_1'\|)p + 5p^2 + |S_2'|\|S_2'\| \\
\Leftrightarrow \quad & score(\mathcal{P}) < score(\mathcal{P}^x)
\end{aligned}$$

□

Claim 2.3. *For any partition $\mathcal{P} = \{S_1, S_2\}$ with vectors a^{v_i} and $a^{\overline{v_i}}$ in S_1, there is a partition $\mathcal{P}' = \{S_1', S_2'\}$ with*

$$a^{v_i} \in S_1' \Rightarrow a^{\overline{v_i}} \in S_2' \text{ and } a^{\overline{v_i}} \in S_1' \Rightarrow a^{v_i} \in S_2'.$$

and $score(\mathcal{P}') > score(\mathcal{P})$.

Proof of Claim 2.3. By claim 2.2 assume w.l.o.g. that every vector a^x is in S_1. We transfer every vector $a^{C_r} \in S_2$ into S_1 in a alternating sequence of two steps, without reducing the overall score of $\mathcal{P}$.

Step 1: We show that there is a partition $\mathcal{P}' = \{S_1', S_2'\}$ with $score(\mathcal{P}) < score(\mathcal{P}')$ where S_1' is covered in positions $j \in \{1, \ldots, 8n+2\}$. Assume that S_1 is not covered in positions $j \in \{1, \ldots, 8n+2\}$. Iteratively we take every vector $a^{C_r} \in S_2$ that is valid in S_1 until there is no such vector left in S_2. Let S_1', S_2' be the sets obtained by this from S_1 and S_2 respectively by moving one vector a^{C_r} from S_2 to S_1. The set S_1' is covered from dimensions 1 to $8n+2$, otherwise there would be a vector $a^{C_r} \in S_2'$ that could be added to S_1'. This leaves us with

$$\begin{array}{llll}
|S_1| \geq p, & \|S_1\| \geq 5p, & |S_1'| = |S_1| + 1, & \|S_1'\| \geq \|S_1\| + 1, \\
|S_2| \leq m + 2n, & \|S_2\| \leq 9n + 2, & |S_2'| = |S_2| - 1, & \|S_2'\| \geq \|S_2\| - 5.
\end{array}$$

By the removal of a vector a^{C_r} from S_2 its size decreases by one and it loses coverage in 1 to 5 dimensions. The set S_1 increases its size by one and covers

at least one additional dimension with the transferred vector.

$$
\begin{aligned}
& 6n+1 < p \\
\Rightarrow \quad & 5\left(\frac{19}{8}n\right) + 9n + 2 < p + 5p \\
\Rightarrow \quad & 5|S_2| + \|S_2\| < |S_1| + \|S_1\| \\
\Rightarrow \quad & |S_1|\|S_1\| + |S_2|\|S_2\| < (|S_1|+1)(\|S_1\|+1) + (|S_2|-1)(\|S_2\|-5) \\
\Rightarrow \quad & |S_1|\|S_1\| + |S_2|\|S_2\| < |S_1'|\|S_1'\| + |S_2'|\|S_2'\| \\
\Rightarrow \quad & score(\mathcal{P}) < score(\mathcal{P}')
\end{aligned}
$$

Step 2: Now we separate the vectors a^{v_i} from their counterpart $a^{\overline{v_i}}$, if they are both in S_1. For each vector a^{v_i} and $a^{\overline{v_i}}$ there are at most four vectors a^{C_r} that have a common 1-value position. When a^{v_i} and $a^{\overline{v_i}}$ are both in S_1 at least one of these corresponding four vectors is in S_2 or $\mathcal{P}$ is invalid. When there is a variable $v_i \in V$ with $a^{v_i}, a^{\overline{v_i}} \in S_1$ We move one of them into S_2. Step 1 can than be repeated without decreasing $\|S_1\|$ as afterwards all dimensions $j \in \{1\ldots, 8n+2\}$ are covered once more.

It remains to be shown that the score of the partition does not decrease by an application of Step 2 with a subsequent Step 1. The vector transferred to S_2 is unique in five dimensions in S_2, because all other vectors covering these dimensions are in S_1 to guarantee that it covers all dimensions $j \in \{1\ldots, 8n+2\}$ along with a^{v_i}'s counterpart that is now unique in dimension $8n+2+i$ in S_1.

Let $y \in \{1,2,3,4\}$ be the number of vectors a^{C_r} in S_2 that are added as valid members to S_1 after the transfer of a^{v_i} or $a^{\overline{v_i}}$ from S_1 to S_2 then

$$
\begin{array}{llll}
|S_1| \geq p, & \|S_1\| \geq 8n+2+5p, & |S_1'| = |S_1| + y - 1, & \|S_1'\| \geq \|S_1\|, \\
|S_2| \leq m+2n, & \|S_2\| \leq 9n+2, & |S_2'| = |S_2| - y + 1, & \|S_2'\| \geq \|S_2\| - 3(y-1)
\end{array}
$$

$\|S_2\|$ is reduced by at most $3(y-1)$ for the following two reasons.

1. By the removal of a vector a^{C_r} $\|S_2\|$ is reduced by at most three unless it is the last clause-vector in S_2. If it is the last clause-vector, S_2 ceases to cover dimension $8n+1$ and $8n+2$ and $\|S_2\|$ is reduced by five. As at most one of the removed vectors a^{C_r} is the last to be removed $\|S_2\|$ is reduced by at most $5+3(y-1)$.

2. Because all a^{C_r} that cover at least one dimension that is also covered by the transferred a^{v_i} are in S_1, a^{v_i} is unique in 5 dimensions in S_2. Hence adding a^{v_i} to S_2 increases $\|S_2\|$ by 5.

The upper boundary of $|S_2| \leq m + 2n$ is the number of vectors in A, aside from vectors a^x. The maximum number of dimensions covered by S_2 is bounded by $\|S_2\| \leq 9n + 2$. This boundary supposes that every vector apart from the vectors a^x is in S_2, although such a set is not valid.

The lower boundaries of S_1 consider only the vectors a^x as size of the set. As the set is fully covered in dimensions $\{1, \ldots, 8n+2\}$ and $\{9n+3, \ldots, k\}$ (this was assured in previous steps).

$$
\begin{aligned}
& 3n < p \\
\Rightarrow \quad & 23n + 2 < 8n + 2 + 5p \\
\Rightarrow \quad & 3|S_2| + \|S_2\| < \|S_1\| \\
\Rightarrow \quad & 0 < (y-1)\|S_1\| - 3(y-1)|S_2| - (y-1)\|S_2\| \\
\Rightarrow \quad & |S_1|\|S_1\| + |S_2|\|S_2\| < (|S_1| + y - 1)\|S_1\| + (|S_2| - y + 1)(\|S_2\| - 3(y-1)) \\
\Rightarrow \quad & |S_1|\|S_1\| + |S_2|\|S_2\| < |S_1'|\|S_1'\| + |S_2'|\|S_2'\| \\
\Rightarrow \quad & score(\mathcal{P}) < score(\mathcal{P}')
\end{aligned}
$$

□

Claim 2.4. *For any partition $\mathcal{P} = \{S_1, S_2\}$ of A that has been improved in means of Claim 2.2 and Claim 2.3, it holds that*

$$a^{v_i} \in S_1 \Leftrightarrow a^{\overline{v_i}} \in S_2.$$

and each $a^{C_r} \in S_1$ to have a score $\geq \mathsf{K}$.

Proof of Claim 2.4. The partition $\mathcal{P} = \{S_1, S_2\}$ has for $i \in \{1, \ldots, n\}$ at most one of the vectors a^{v_i} and $a^{\overline{v_i}}$ in S_1. Also every vector a^x is in S_1.

$$
\begin{aligned}
& 1 \leq i \\
\Rightarrow \quad & 4n^2 + 2n < 5i \cdot (4n^2 + 3) \\
\Rightarrow \quad & 4n^2 + 2n < 5i \cdot p \\
\Rightarrow \quad & 4n^2 + 2n < i(18n + 4 + 5p) \\
\Rightarrow \quad & -i(18n + 4 + 5p) + 9n^2 + 2n < 5n^2 \\
\Rightarrow \quad & \underbrace{\left(\frac{11}{3} - i + p\right)}_{|S_1|} \underbrace{(9n + 2 + 5p)}_{\|S_1\|} + \underbrace{(n-i)}_{|S_2|} \underbrace{(9n+2)}_{\|S_2\|} < \underbrace{\left(\frac{11}{3} + p\right)(9n + 2 + 5p) + 5n^2}_{\mathsf{K}} \\
\Rightarrow \quad & score(P) < \mathsf{K}
\end{aligned}
$$

□

Now we can show that there exists a valid partition of A with a score $\geq \mathsf{K}$, if and only if the given instance of 3-SAT is satisfiable.

$\Leftarrow$: Consider a truth-assignment of the variables that satisfies the given problem instance $\mathcal{I}$. A is partitioned such that all vectors $a^{C_r} \in A$, $r \in \{1, \ldots, m\}$ and $a^x \in A$ are in S_1. For every variable $v_i \in V$ we put a^{v_i} into S_1 and $a^{\overline{v_i}}$ into S_2, if v_i is assigned with *false* in the satisfiable solution of $\mathcal{I}$. If v_i is assigned with *true* in the solution of $\mathcal{I}$, a^{v_i} is added to S_2 and $a^{\overline{v_i}}$ is added to S_1.

All vectors a^{C_r} are unique in at least one dimension in S_1. Let the l-th literal of C_r be *true* (as the instance of 3-SAT is satisfiable, there is at least one) then a^{C_r} is unique in dimension $j = 3(r-1)+l$ in S_1. The single other vector with a 1-value in that position, $a^{v_{\chi(i)}}$, is in S_2.

The positive variable-vectors a^{v_i} are separated from the respective negative $a^{\overline{v_i}}$ and thus both are unique in $8n+2+i$ in their respective set and thus valid. As all vectors are valid, so are the two sets S_1 and S_2 and the partition $\mathcal{P}$.

The set S_1 covers every dimension from 1 to k, while S_2 covers $5 \cdot 8n$ dimension. Hence the score of $\mathcal{P}$ is $\geq \mathsf{K}$.

$\Rightarrow$: Let $\mathcal{P}$ be a partition with $score(\mathcal{P}) \geq \mathsf{K}$. By claims 2.2 and 2.4 we can assume that all vectors $a^x \in S_1$, all vectors $a^{C_r} \in S_1$, and $a^{v_i} \in S_1 \Leftrightarrow a^{\overline{v_i}} \in S_2$. A truth-assignment for the underlying 3-SAT problem can be obtained by setting v_i true when $a^{v_i} \in S_2$.

This concludes the proof. □

For any number of ones > 5 one can simply add 1-values in the back of the vectors.

4.2 Groups of Non-Reconfigurable Agents

The Optimal Partition Problem arises whenever two groups can regroup by exchanging agents or merging into one group. Groups are allowed to regroup whenever they are on two adjacent fields and at least one faces the other. One agent of the group is randomly chosen as the groups moderator. The moderator is responsible for the groups movement decisions and ensures that all members of the group move in unison and are always oriented in the same direction. The decisions of the moderator are based on information it

gathers from all member agents of the group. The moderator channels all communication between the group and non-group agents.

The agents perform a gradient walk, i.e. they move on a field f, if f is more beneficial than their current field. Two aspects are considered by the agents to evaluate how beneficial a field is. In the so called *consensus* the increase or decrease of resources an individual agent of the group would endure by the movement is collected. The other considered aspect is the *attraction* from other groups or agents in close proximity. Depending on the capabilities of these agents they can be more or less beneficial as new members of the group and should be approached in order regroup. Agents can read the capabilities of other agents if they are located within each others *sensing range* (see Figure 4.1). A parameter $\alpha \in [0, 1]$ is finally used to weight these two aspects.

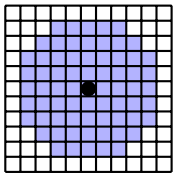


Figure 4.1: Agent (black) with its sensing range (blue).

Consensus Every agent a tries to collect as much resources as possible. If it is located on field f and faces field f', it calculates the total amount of resources P' it can collect by itself on f' and compares it to the total amount P it can collect on f as defined earlier in Equation (3.1)

$$P(a_i) = \frac{1}{s} \sum_{j=1}^{k} a_{ij} f_j \ \in [0, 1]. \tag{3.1}$$

The agent follows the gradient $\Delta_P = (P' - P)/(P' + P) \in [-1, 1]$, with $\Delta_P = 0$ when $P' = P$. If $\Delta_P > 0$, the agent votes to move onto f' to increase its personal gain, else it votes to stay on f and turn. The weight of the vote is Δ_P, such that agents with a higher gain or loss have a stronger influence on the decision. The average of all votes gives the consensus of the group.

$$\mathit{consensus}(G) = \sum_{a_i \in G} \frac{\sum_{j=1}^{k} a_{ij}(f'_j - f_j)}{\sum_{j=1}^{k} a_{ij}(f'_j + f_j)}$$

Attraction Groups can communicate with other groups or agents that are located within their sensing range. They use the gathered information to

decide whether to approach or avoid another group. This decision is based on the capabilities of both groups in their current state and after a possible regrouping event. To quantify the benefit of a regrouping event between two groups G_1 and G_2 we define

$$score(G_1, G_2) = |G_1| \sum_{j=1}^{k} \max_{a \in G_1} a_j + |G_2| \sum_{j=1}^{k} \max_{a \in G_2} a_j$$

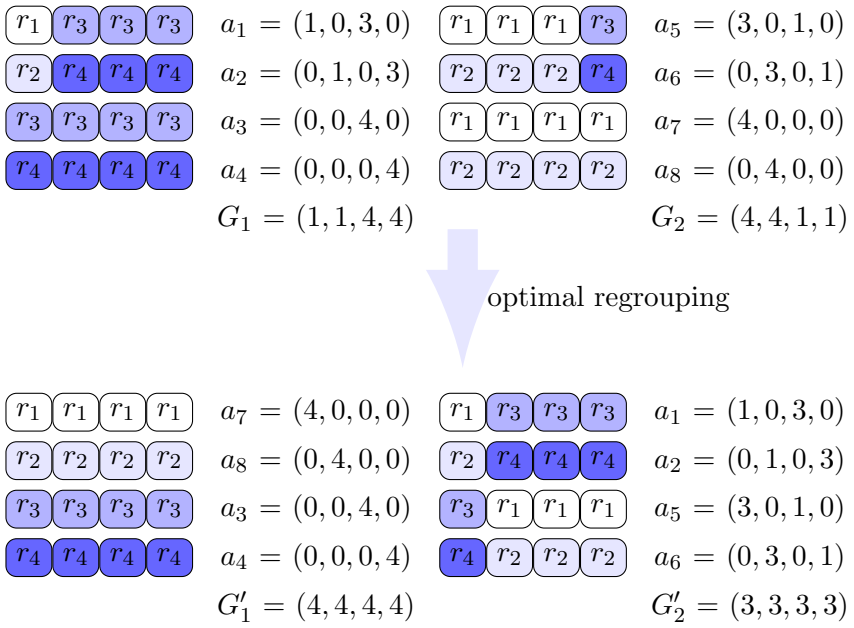


Figure 4.2: Example for an optimal regrouping with the maximal Δ_s with $k = 4$ resource types and $s = 4$ slots. Note, that each agent has to commit the maximal capability for at least one resource type to be a member of a group. Δ_s is 32 for the above example. Mind, that the group sizes are multiplied with the summed capabilities to infer the score of a group.

If the improvement of the score is above a specific threshold t the groups approach each other, otherwise they avoid each other. Let Δ_{score} be the difference between the scores before and after an optimal regrouping event.

$$\Delta_{score}(G_1, G_2) = score(G'_1, G'_2) - score(G_1, G_2)$$

where G'_1 and G'_2 are the potential groups after an optimal regrouping event. The empirically tested upper bound for Δ_{score} is $m = k^2(s-2)$ (see Figure 4.2

for an example). Because the regrouping event can not lower the score (due to the optimality) $\Delta_{score} \in \{0, \ldots, k^2(s-2)\}$. We define

$$attraction(\Delta_{score}) = \frac{\theta(\Delta_{score}) - 1}{\theta(\Delta_{score}) + 1} \in [-1, 1] \text{ with } \theta(x) = \frac{x(t-m)}{t(x-m)}$$

With $\Delta_{score} \in \{0, \ldots, m\}$ we have $attraction(\Delta_{score}) \in [-1, 1]$ just like the *consensus* values. If $\Delta_{score}) > t$ the groups try to approach each other, i.e. $attraction(\Delta_{score}) > 0$. Otherwise they prefer to step onto fields further away from the other group, i.e. $attraction(\Delta_{score}) < 0$.

The Process of Regrouping The distributed feature of a swarm is especially important for foraging tasks to explore the environment effectively, thus many small groups are preferable to few large groups. To this end a group forbids members that do not increase the groups capabilities but solely its size. Thus, every agent a in a group G has to be the only agent of G with the highest capability for at least one resource type r in the group, i.e. for all agents $a' \in G : a'_r < a_r$. A set of agents failing this condition is *invalid*. This requirement leads to the *Optimal Partition Problem* during regrouping events. The limited number of resource types and slots allows an agent to compute an optimal solution in a sufficiently fast manner. For larger k a heuristic approach is advisable.

The Optimal Partition problem is solved with a branch and bound algorithm that finds the optimal partition of $G_1 \cup G_2$ into G'_1 and G'_2. The root of the solution tree represents the unpartitioned set of agents A and has two child nodes. The first child adds a_1 to G'_1, the second child adds a_1 to G'_2. In each further level i of the tree this procedure is repeated for every agent a_i. A branch is pruned if there is an agent a_j, $j > i$ that is invalid in both groups. Every leaf containing a valid partition has a score where the maximal score gives the best solution.

Once the groups G'_1 and G'_2 are determined the agents are set on their respective groups positions that are the exact same positions that prior G_1 and G_2 had. As the agents are allowed to move only one step per simulation turn regrouped agents are immobilized for the rest of the simulation turn. Note, that a regrouping event does not occur if the scores of the groups are already optimal. In that case the groups move on.

Algorithm 1 shows the structure of a simulation turn for an agent moderating a group or only its own movement with the movement decision being made in Line 5.

Algorithm 1: Agent::move() Defines movement of agents and their group.

```
1  if inactive then
2  |  reload batteries
3  else
4  |  determine next field f_next;
5  |  if α · attraction + (1 − α)consensus > 0 and f_next is empty then
6  |  |  move group on f_next
7  |  else
8  |  |  if turned in previous step then
9  |  |  |  turn by 45° in direction of previous step
10 |  |  else
11 |  |  |  turn by 45° in random direction
```

4.3 Experimental Results

In order to infer the properties of this system especially the group formation of the agents several experiments were conducted. In the following the specifics of these experiments are described followed by an extensive analysis.

4.3.1 Distribution of Resources

There are many possibilities to design the distribution of resources in order to simulate a resource collection task. Here, I settled on a design simulating consumption and regeneration of resources.

Each field $f \in F$ has *load* m_f defining the total amount of resources it contains that is initialized with $m_f = m_{max} = 100$ units. The load is harvested and reduced by the group of agents working on f by one unit per agent per simulation turn unless $m_f = 0$, i.e. there are no resources left on f. The lower the load of a field is the harder it is for the agents to effectively collect resources. In each simulation turn n fields are randomly chosen and, if the resource capacity of a chosen field f allows for it (i.e. $m_f < m_{max}$), the load m_f is increased by one unit. As each active agent reduces the load of its respective field by 1 unit in each simulation turn the amount of available resources stabilizes over time.

The availability of a specific resource depends on the load of the field and the distance of the field to the resources *center point*. In the performed experiments are $k = 4$ different resource types. Each resource type $r \in$

$\{r_1, r_2, r_3, r_4\}$ has a center point $(r.x, r.y)$ in normalized coordinates in the arena. The center points of the different resource types are situated in the respective four edges of a square with side length 0.5 that is centered within the arena. The availability of the resources on the other fields is determined by the minimal Euclidean distance $\Delta(f, r)$ of the fields center $(f.x, f.y)$ to the center point of that resource type and the load of the field. Note, that the distance is at most 0.5 since the arena is a torus. The field f with the maximal distance to the center point of resource type r has $f_r = 0$ disregarding the fields load. In general

$$f_r = \frac{m_f(1 - 2\Delta(f, r_i))}{4m_{max}}$$

with

$$\Delta(f, r) = \sqrt{\left(\frac{1}{2} - \left|\,|f.x - r.x| - \frac{1}{2}\right|\right)^2 + \left(\frac{1}{2} - \left|\,|f.y - r.y| - \frac{1}{2}\right|\right)^2} \quad (4.1)$$

When an agent $a \in G$ on a field f collects resources it reduces the load of the field by one unit but can only collect an amount of $\sum_r f_r \max_{a' \in G} a'_r$. The rest of the resource is lost or wasted. The total amount of resources that are collected, wasted, and regenerated are important measures to infer the performance of the system.

4.3.2 Parameter Settings

The threshold t gives the minimal required improvement Δ_{score} by a regrouping event with a visible other group to approach that group. Different thresholds are applied to differently sized groups as shown in Table 4.1 along with all other applied parameter values. A small regrouping event affects at most 4 agents. If there are more agents involved in the regrouping event it is considered to be large. Observe, that large regrouping events affect too many agents to simply merge both groups into one because the maximum size of a single group is $k = 4$. In small regrouping events the score of the affected agents can increase considerably by merging into one group. The maximal improvement for two agents is upper bounded by 24. If $t_{\leq 4} = 25$, two single agents can not attract each other whatever their configurations are. Thus, single agents will constantly try to avoid each other. The threshold is easier to overcome when a single agent meets a pair of already grouped agents.

All results are averaged over the performed runs. In each simulation turn the call sequence of the agents is shuffled.

Table 4.1: Parameter values used in the experiments. Different attraction-threshold values are used depending on the total number of agents in the two groups that do the regrouping. For a total number of ≤ 4 agents a value $t_{\leq 4}$ is used and for ≥ 5 agents a value $t_{\geq 5}$ is used.

	Parameter	Value
$t_{\leq \mathbf{4}}$	t in regrouping events with ≤ 4 agents	$\{5, 25\}$
$t_{\geq \mathbf{5}}$	t in regrouping events with ≥ 5 agents	$\{5, 25\}$
	number of runs	100
	length of one run	2500 simulation turns
	size of the arena	50×50
n	number of agents	50
s	number of slots	12
	sensing range	4 fields
	battery capacity	1000 simulation turns
	reload time	100 simulation turns
α	influence of *attraction*	0.5

4.3.3 Results

Figure 4.3 gives the average groups sizes for the different tested systems. Lower thresholds increase the average group size to approximately 2.8 agents per group. The initial group size is 1, because all agents are initialized on separate fields. Regrouping events increase the average group size during approximately 500 simulation turns very fast. Mind, that with $k = 4$ resource types the maximal group size is restricted to 4.

When $t_{\leq 4} = 25$ single agents have no intent on joining into groups of two. However, half their movement is determined by the gradient of resources they measure and meetings are inevitable on the long run. The two systems with $t_{\leq 4} = 25$ take more time to build larger groups and reach a state where the average group size stabilizes (at least 2000 simulation turns).

Note, that with $t_{\leq 4} = 25$ and $t_{\geq 5} = 25$ has a larger group size than $t_{\leq 4} = 25$ and $t_{\geq 5} = 5$. The higher threshold increases the average group size. One explanation for this phenomenon is the presence of groups with less specialization in systems with $t_{\geq 5} = 5$. They tend to build groups of size 3 that would often become invalid with a fourth agent. Systems with $t_{\geq 5} = 25$ would have more specialized groups that can contain four agents.

The average amount of resources in the system stabilizes after 1000 simulation turns (see Figure 4.4). In each simulation turn $n = 50$ fields are

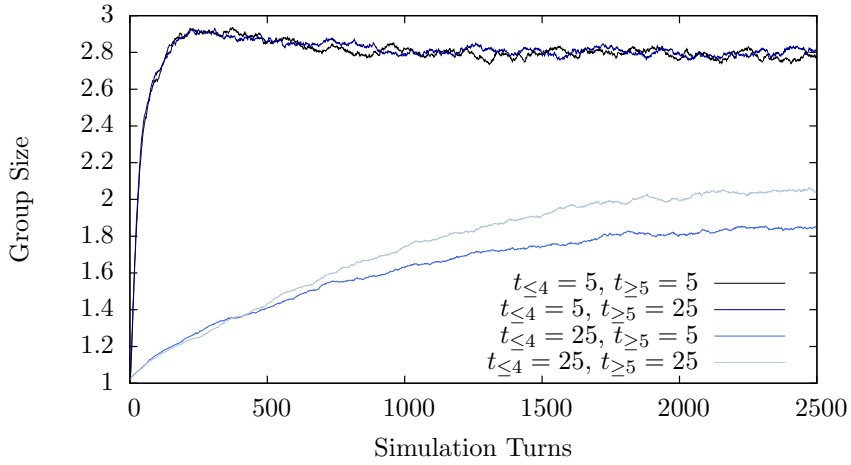


Figure 4.3: Average size of groups per simulation turn.

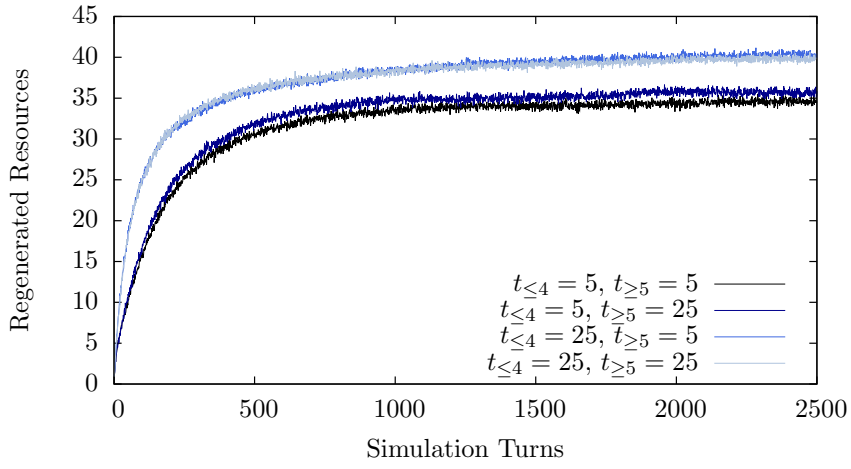


Figure 4.4: Average amount of regenerated load units per simulation turn.

randomly chosen and if their load is below $m_{max} = 100$ their load is increased by one unit. Figure 4.4 shows how many of the 50 random fields have loads below 100 and how their number increases during the simulation runs. This figure implies the distribution of the agents in the arena. The more fields are regenerated the more fields have been visited by groups. A large amount of regenerated fields implies that there are more small groups and single agents that cover more fields of the arena. The systems with $t_{\leq 4} = 25$ are more distributed and up to 80% of the randomly chosen fields have been visited by agents beforehand.

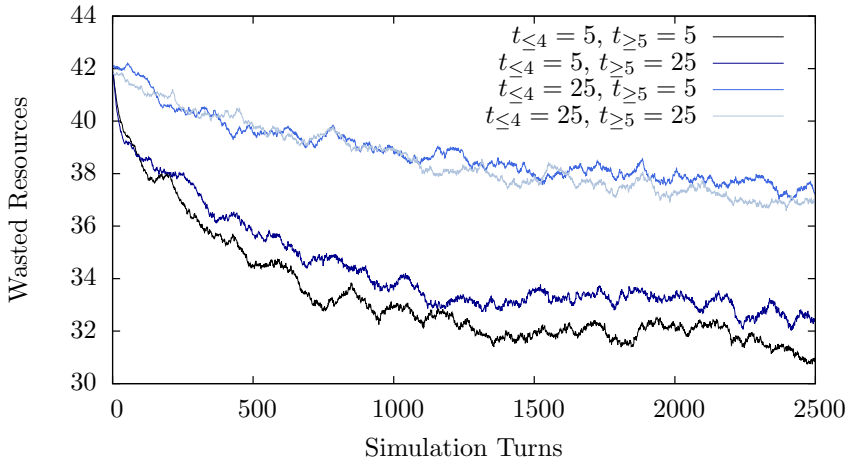


Figure 4.5: Average amount of wasted resources per simulation turn.

Figure 4.5 shows the amount of resources that are wasted by the different systems due to missing capabilities of the groups or low loads of the harvested fields. The larger groups build in systems with $t_{\leq 4} = 5$ waste less resources due to their higher capabilities, while working in general on fields with lower loads than the agents in systems with $t_{\leq 4} = 25$. The system with $t_{\leq 4} = 5$ and $t_{\geq 5} = 5$ produces the least waste.

Finally, Figure 4.6 gives the performance of the compared system. It shows how much resources could be collected by the agents. Indeed the systems with $t_{\leq 4} = 25$ have a higher final performance than the systems with $t_{\leq 4} = 25$. This can be explained by the increased load reduction of the larger groups. They reduce the load of their field to 0 but are unable to find find a consensus and move on. While they can not waste any resources on a field f with $m_f = 0$, they are unable to collect resources as well.

The system with the highest final performance has $t_{\leq 4} = 25$ and $t_{\geq 5} = 5$

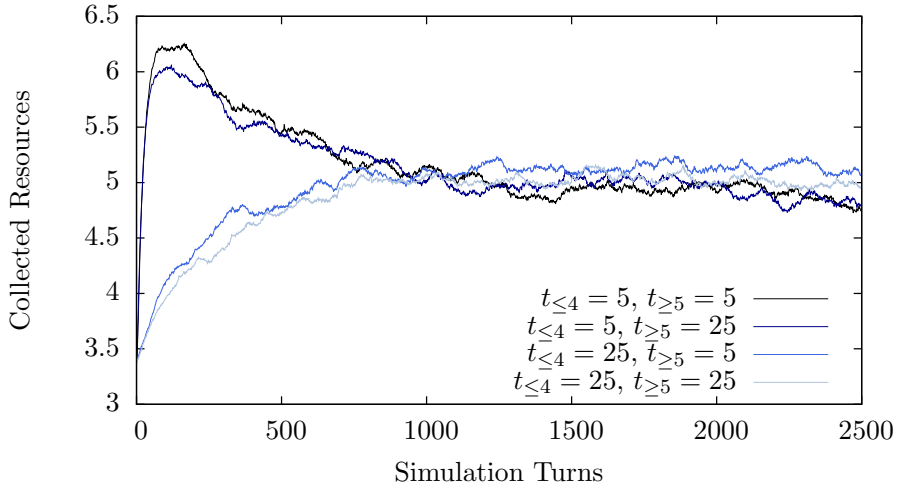


Figure 4.6: Average collected resources per simulation turn.

4.4 Conclusion

In a system of agents that are unable to reconfigure their capabilities the choice of cooperation partners is of crucial importance. Agents should avoid cooperation with small groups, but seek cooperation with large groups. The profit arising from cooperation between two agents does not pay for the loss of efficiency endured by moving towards each other, the immobility during regrouping, and the heightened decrease of mass on the host field.

However, agents should approach larger groups with high capabilities. Being member of a group with specialized agents is worth the effort. In general the system has a higher performance if the agents work in small groups and thus process more fields simultaneously as a swarm on.

5 Reconfigurable Swarms

Controlling group sizes

In a system such as introduced in Chapter 3 with reconfiguring agents the optimal partition problem subsides but other problems arise. In the following I introduce an extension of the model presented in Chapter 3 where the agents are able to adapt themselves to the need of their group by reconfiguration. An accurate description of the agents behavior is given in Section 5.1. The performed experiments and their results are presented in Section 5.2.

5.1 Group Formation of Reconfigurable Agents

The capabilities of the agents are reconfigurable, i.e. the state of a slot can change during a reconfiguration operation. This can be of great benefit to the group containing the reconfiguring agent. The problem in group formation is no longer defined by the search of agents that increase the groups capabilities but lies rather in the decision of how many agents should be contained by a group as those will adapt their capabilities in time. Instead of using regrouping events to get new agents, groups *recruit* single agents that they meet in the arena.

In order to synchronize the system a simulation turn is structured into three phases concatenated by synchronization points, i.e. all agents wait for all other agents to complete the current phase before they start with the next. Figure 5.1 gives an overview of one simulation turn. During the *recruitment phase* recruit new members followed by the *working phase*, where agents move, gather resources, and reconfigure. In the *battery phase* agents change their state of activity depending on the state of their batteries.

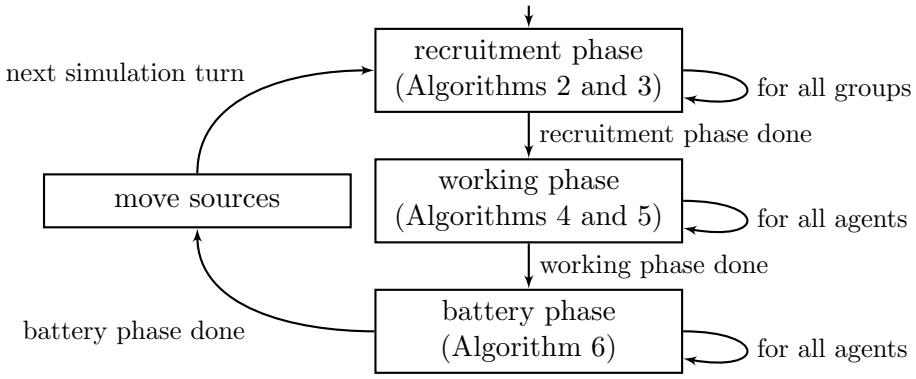


Figure 5.1: Flow diagram of the three phases making up a simulation turn.

5.1.1 Recruitment Phase

A group $G \subset A$ can only recruit an agent $a \in A$ per simulation turn. G has to be oriented towards the field that contains a. It is neither possible to recruit groups of two or more agents, nor to recruit an agent that is member of another group of size ≥ 2, nor to recruit a reloading agent. Note, that these rules avoid conflicts between groups. In this chapter two strategies for recruitment are analyzed: *static recruitment* and *dynamic recruitment*. Assume in the following that a group G is oriented towards a field that contains a single agent $a \in A$ that is not reloading, i.e. a can be recruited by G.

Algorithm 2: Recruitment Phase of the static recruitment version.

```
1 for each group G do
2     f' is the adjacent field in the current orientation
3     if there is a group of size one containing only agent a on f' then
4         draw a random number φ
5         if φ ≤ recruitment rate then
6             recruit a to G
```

Static recruitment. The static recruitment strategy depends on the *recruitment rate*$\in [0, 1]$, i.e. a static parameter. The group G choses a random number $\phi \in [0, 1]$ and recruits a if $\phi <$ *recruitment rate* (see Algorithm 2).

Thus, the group size depends on the frequency the group meets agents that can be recruited and the *recruitment rate*.

Algorithm 3: Recruitment Phase of the dynamic recruitment version.

```
1 for each group G do
2     f' is the adjacent field in the current orientation
3     if there is a group of size one containing only agent a_i on f' then
4         if there are no idle slots then
5             recruit a_i to G
```

Dynamic recruitment. The dynamic recruitment strategy takes current information about the group into account. If at least one of the agents of G has at least one idle slot a is not recruited (see Algorithm 3). Recall from Section 3.3 that a slot of $a' \in G$ in state j is idle on field f if

(i) $f_j = 0$,
(ii) there is another agent $a'' \in G$ with $a'_j < a''_j$, or
(iii) there is another agent $a'' \in G$ with $a'_j = a''_j$ and a'' joined G before a'.

Only if all slots of all agents of G are increasing the performance of G, a is recruited. Note, that this strategy is based solely on information the agents of G already gathered for their reconfiguration processes. The agents neither count the number of agents in their group nor the number of resource types that are available on their current location or their past experience.

5.1.2 Working Phase

During the working phase all reloading agents refill their battery by adding *battery capacity/reload time* units of energy to it. All active agents suffer a loss of one energy unit from their batteries before they move, work, and reconfigure, in this order. Two movement behaviors are considered in the following where groups have to decide whether to move forward on the next field or rotating on their current field.

Random walk. Groups performing a random walk base their movement decision on a random value. The probability to move forward is defined by the *velocity* parameter. A group G choses a random number $\phi \in [0, 1]$ and moves forward if $\phi < velocity$ (see Algorithm 4).

Algorithm 4: Working Phase for the random walk version.

```
1  for each agent a do
2      if a is reloading then
3          reload battery
4      else
5          remove one energy unit from battery
6          if group G with a ∈ G did not move yet then
7              determine next field f'
8              draw a random number φ
9              if φ < velocity and f' is empty then
10                 move G on f'
11             else
12                 if G turned in previous simulation turn then
13                     turn G in previous direction
14                 else
15                     turn G in random direction
16         collect resources and reconfigure idle slots
```

Algorithm 5: Working Phase for the gradient walk version.

```
1  for each agent a do
2      if a is reloading then
3          reload battery
4      else
5          remove one energy unit from battery
6          if group G with a ∈ G did not move yet then
7              determine next field f'
8              if P(G) on f ≤ P(G) on f' and f' is empty then
9                  move G on f'
10             else
11                 if G turned in previous simulation turn then
12                     turn G in previous direction
13                 else
14                     turn G in random direction
15         collect resources and reconfigure idle slots
```

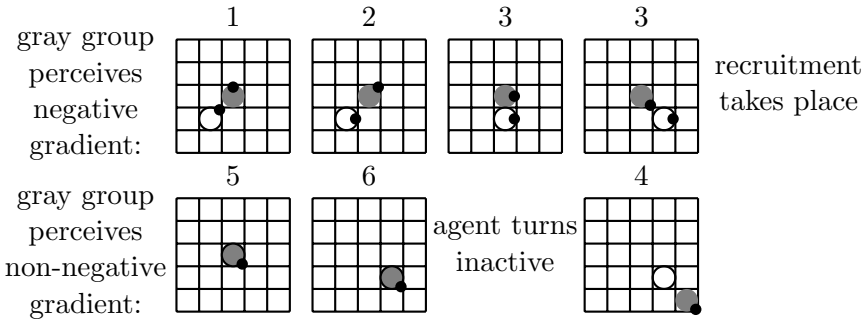


Figure 5.2: Exemplary situation of a collision between a single agent (white) and group of more than one agent (gray);

Gradient walk. The gradient walk movement allows a group to intentionally leave undesirable locations and move towards more beneficial regions of the arena. A group G measures its performance on its current field f and compares it to the performance it would have on the target field f' that lies in the orientation of G. If the performance on the neighbored field is higher or equal the group moves onto f'. Otherwise, the group rotates (see Algorithm 5).

5.1.3 Battery Phase

In the battery phase each agent checks the state of its battery. If the battery fully reloaded during the working phase, i.e. the battery contains *capacity* many energy units, the agent turns active. If the agent is active but the battery went empty during the working phase the agent switches into the reloading state (see Algorithm 6).

Algorithm 6: Battery Phase: Agents set their state of activity.

```
1 for each agent a do
2     a.reload := (a.battery = 0) ∨ (a.reload ∧ a.battery < capacity)
```

Figure 5.2 depicts an exemplary set of simulation turns showing how a group can recruit or lose agents.

5.2 Experimental Results

In the first part of this section the experimental setup is described followed by an analysis of the agent behavior regarding the different movement behaviors and recruitment strategies. An additional analysis shows the performance of the compared systems considering different reconfiguration costs. Finally, a the impact of different parameters and the robustness of the system against changes in them is shown and a conclusion is given.

5.2.1 Distribution of Resources

It is assumed that each resource type has exactly one *center point*, i.e. the place of highest concentration, that is located within the arena. The coordinates of a center point are given in normalized form and are independent of the positions of the fields and their margins. The distribution of the resources are, thus, not bounded by the otherwise discrete properties of the model. The location of the center point of resource type r_j at simulation turn t is denoted by $(r_j.x, r_j.y)(t) = (r_j.x(t), r_j.y(t)) \in [0,1]^2$. The initial location $(r_j.x, r_j.y)(0)$ of each resource type is chosen at random within the arena. In simulation turn $t+1$ the location of the center point of resource type r_j, $j \in \{1 \ldots, k\}$ is relocated to

$$(r_j.x)(t+1) = \begin{cases} r_j.x(t) + \rho_x, & \text{if } r_j.x(t) + \rho_x \in [0,1] \\ r_j.x(t) + \rho_x - 1, & \text{if } r_j.x(t) + \rho_x > 1 \\ r_j.x(t) + \rho_x + 1, & \text{if } r_j.x.(t) + \rho_x < 0 \end{cases}$$

with $\rho_x \in [-v_{max}, v_{max}]$, where $v_{max} \in [0,1]$ is the *maximum velocity* of resources (coordinate $r_j.y$ is changed analogously with $\rho_y \in [-v_{max}, v_{max}]$). Each resource type has an availability radius $r_{resource}$ that is identical for all resource types. The farther away a field $f \in F$ is from the center point of a resource type r_j the smaller is f_j. Let $(f.x, f.y)$ be the center point of f then

$$f_j = \max\left(0, 1 - \frac{\Delta(f, r_j)}{r_{resource}}\right)$$

where $\Delta(f, r_j)$ is the minimal Euclidean distances between between the center of f and the center point of r_j as previously introduced with Equation (4.1). Note, that any field with $\Delta(f, r_j) > r_{resource}$ has $f_j = 0$. With $r_{resource} \geq 0.5$ there is no field $f \in F$ with $f_j = 0$.

In the tested dynamic scenarios not all resource types are available at all times. A resource type can be inactive for several simulation turns. If

resource type r_i, $i \in \{1, \ldots, k\}$, is inactive each field $f \in F$ has $f_i = 0$. The simulation runs have a length of 5000 simulation turns. During the first 1500 simulation turns and the final 1000 simulation turns only two resource types are active. From simulation turn 1501 until simulation turn 3000 ten resource types are active.

5.2.2 Parameters and Measurements

Two kinds of experiments have been performed to infer the properties of this model. The first experiments compared the different movement behaviors and recruitment strategies in detail. The latter experiments tested the influence of the most important system parameters and the systems robustness concerning changes in them.

Table 5.1: Model parameters applied in performance experiments.

Parameter	Definition	Value
velocity	only for random movement	0.6
recruitment rate	only for static recruitment	0.25
d	size of arena	50×50 fields
n	number of agents	100
s	number of slots	10
–	number of simulation turns	4000
k	number of resource types	10
–	capacity of agent batteries	500 simulation turns
–	reload time of empty battery	50 simulation turns
μ	mutation strength	0.1
c_{reconf}	reconfiguration cost	0.5
v_{max}	maximal velocity of resources	0.25
r_{source}	radius of resources	1
c_{reconf}	reconfiguration cost	$1, 0.5, 0$

Table 5.1 gives the parameter values applied in the first experiments. I compare four systems denoted by *random-static* (rs), *random-dynamic* (rd), *gradient-static* (gs), and *gradient-dynamic* (gd) defined by their mode of movement and recruitment strategy. The velocity for the random walk behavior is chosen such that the average number of movements per simulation turn is close to that of the gradient walk agents. Similarly, the relation between positive and negative recruitment decisions in static recruitment and dynamic recruitment systems are nearly identical due to the chosen re-

Table 5.2: With $\mu = 0.1$, let x be the number of idle simulation turns before a slot mutates. $P(x = t)$ gives the probability that the slot mutates after exactly t simulation turns. $P(x \leq t)$ gives the probability that the slot has mutated after t or less simulation turns. $E(x)$ is the expected number of idle simulation turns.

t	$P(x = t)$	$P(x \leq t)$	$E(x)$
1	0.1	0.1	3,66021568
2	0.18	0.28	
3	0.216	0.496	
4	0.2016	0.6976	
5	0.1512	0.8488	
6	0.09072	0.93952	
7	0.042336	0.981856	
8	0.0145152	0.9963712	
9	0.00326592	0.99963712	
10	0.00036288	1.0	

Table 5.3: Model parameters varied in robustness experiments.

Parameter	Definition	Value
velocity	only for random walking agents	$0.1, 0.6, 0.9$
recruitment rate	only for static recruiting agent	$0.1, 0.25, 0.9$
n	number of agents	$50, 100, 50$
μ	mutation strength	$0, 0.1, 0.5$
v_{max}	maximal velocity of resources	$0, 0.25, 1$

cruitment rate. This is done in order to make the systems comparable. The mutation strength of $\mu = 0.1$ ensures that an idle slot mutates within ten simulation turns. Reference Table 5.2 for more stochastic properties of the mutation strength.

In additional experiments the influence of five different parameters is analyzed (see Table 5.3). I define a standard system that shares all parameters given in Table 5.1, applies the gradient walk behavior, and the dynamic recruitment strategy. For each parameter two or three separate experiment were run with a deviation from the standard system only in this parameter.

Each variant of the system was run 50 times to ensure the statistic value of the results. To infer the special characteristics of different variants of the

system I introduce relevant measurements in the following.

Movement Decisions Agents have four types of movement possibilities in a simulation turn. They can (i) moving forward, (ii) rotating due to the movement strategy (i.e. because of randomly deciding to rotate or because the gradient is negative), (iii) rotating to avoid blocked field, (iv) or immobility due to reloading batteries The frequency of these movement possibilities give insight into the mobility the of agents in the different systems.

Group Size In each simulation turn an agent is either member of a group or reloading its batteries. The size of the group of each agent is measured to infer the impact of different numbers of available resource types and behaviors on the group formation.

Idle Slots The number of idle slots of each agent during a simulation turn shows how beneficial the agents are for their group at some point during the simulation.

Collected Resources The average total amount of collected resources per run gives a qualitative measurement for the different systems.

5.2.3 Behavioral Analysis

Mobility I begin with the analysis of the influence of the movement behavior on the system. Figure 5.3 gives a time series on the mobility of the groups in the gradient-static and random-static system. The values are averaged over the performed runs. The respective results for the dynamic recruitment systems are very similar and show no influence of the recruitment strategy on the movement decisions.

Agents in the random-static system move in 53% of all simulation turns onto their respective target field. This is lower than the set velocity of 0.6 as there are reloading times and blocked movements. The average velocity of agents using the gradient movement strategy is 0.51 fields per simulation turn and significantly slower (Mann-Whitney test with p-value < 0.01) than with random movement. Recall, that the velocity parameter has been set for a better comparison such that agents in both system variants have a similar average speed. Due to the reload efficiency of 10 energy units per reloading simulation turn an agent spends $1/11 \approx 0.09$ of its time in the reloading state. This also means that about 9% of the agents are reloading during a

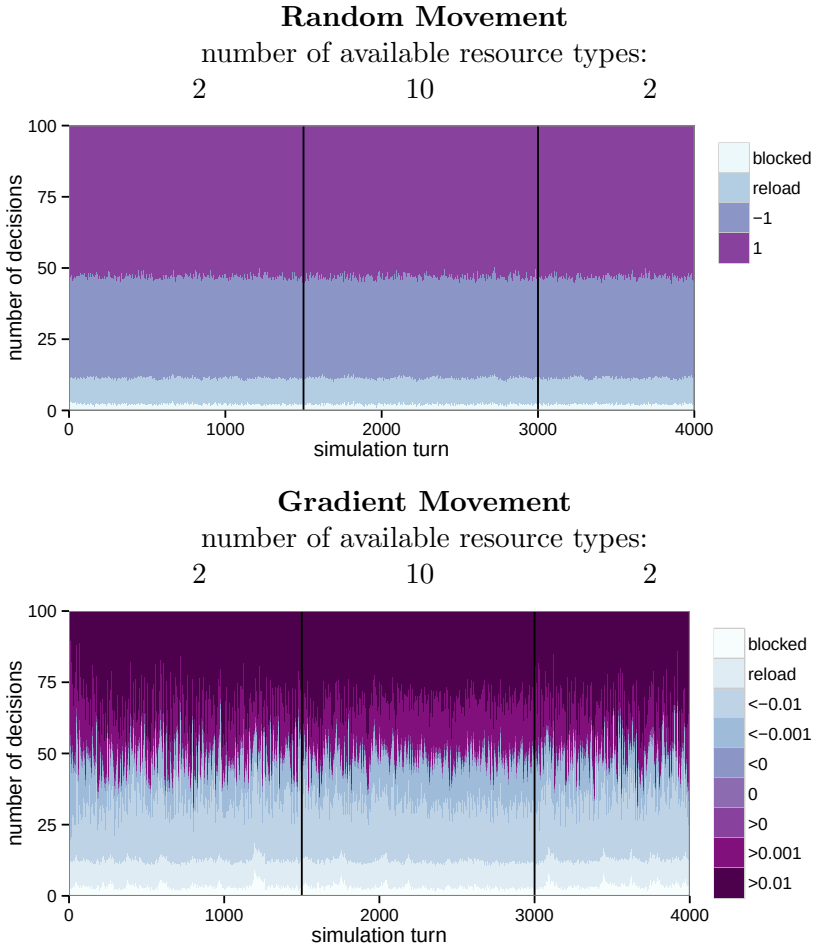


Figure 5.3: Movement behavior of a system with static recruitment: random movement (top) and gradient movement (bottom). Negative numbers indicate a decision to stay and rotate, positive numbers indicate a movement of the group. The lower plot gives margins of the measured gradients. Groups oriented towards fields containing other agents are listed as blocked. Reloading agents are listed as such.

simulation turn. It rarely happens (less than 5%) that the agents movement is blocked by the presence of other agents on the target field, hence, the 100 agents can move relatively freely inside the arena. This ensures that the movement of the agents is barely affected by congestion effects.

The increase or decrease in the number of resource types barely influences the average movement speed in both systems. There are no significant changes with random movement and a slight, but significant (Mann-Whitney test with p-value < 0.01), difference with gradient movement (increase from 0.506 to 0.514 from simulation turn 1001 to 1500 and 2501 to 3000 respectively).

Group Size The recruitment strategy applied by the agents has a great impact on the group formation processes in the system as can be seen in Figure 5.4 which shows a time series of the distribution of group sizes in the random-static, gradient-static, and gradient-dynamic systems. Initially, all agents are placed individually on random locations into the arena with a random battery state. In the random-static system the group sizes stabilize after about 500 simulation turns. About one tenth of the agents are reloading, about one quarter of the agents stay alone (group of size one), slightly more than one quarter of the agents stay in groups of size two, and the remaining agents – approximately one third – are in groups that have at least three members. The average group size for the random-static system is 1.9 independently of the number of available resource types.

The gradient-static system shows a similar group formation behavior. The average group size of 2.0 is slightly, but significantly (Mann-Whitney test with p-value < 0.01), above the one of the random-static system. A possible explanation for this phenomenon is the slightly lower movement rate of agents in the gradient-static system. Slower movement results in a higher chance of meeting other agents in the arena as slower groups rotate more often and are thus more aware of their surroundings. This increases the number of sights of other groups and single agents they can recruit. A hint to this explanation is the higher number of blocked fields in the gradient-static system as seen in Figure 5.3.

The gradient-dynamic system shows a clear adaptation of the group sizes to the changing number of available resource types. After the number of available resource types is increased to ten the average groups size rises from 2.0 (simulation turns 1001 to 1500) to 2.6 (simulation turns 2001 to 2500), which is a significant increase (Mann-Whitney test with p-value $<$ 0.01). The gradient-dynamic system has smaller groups on average than

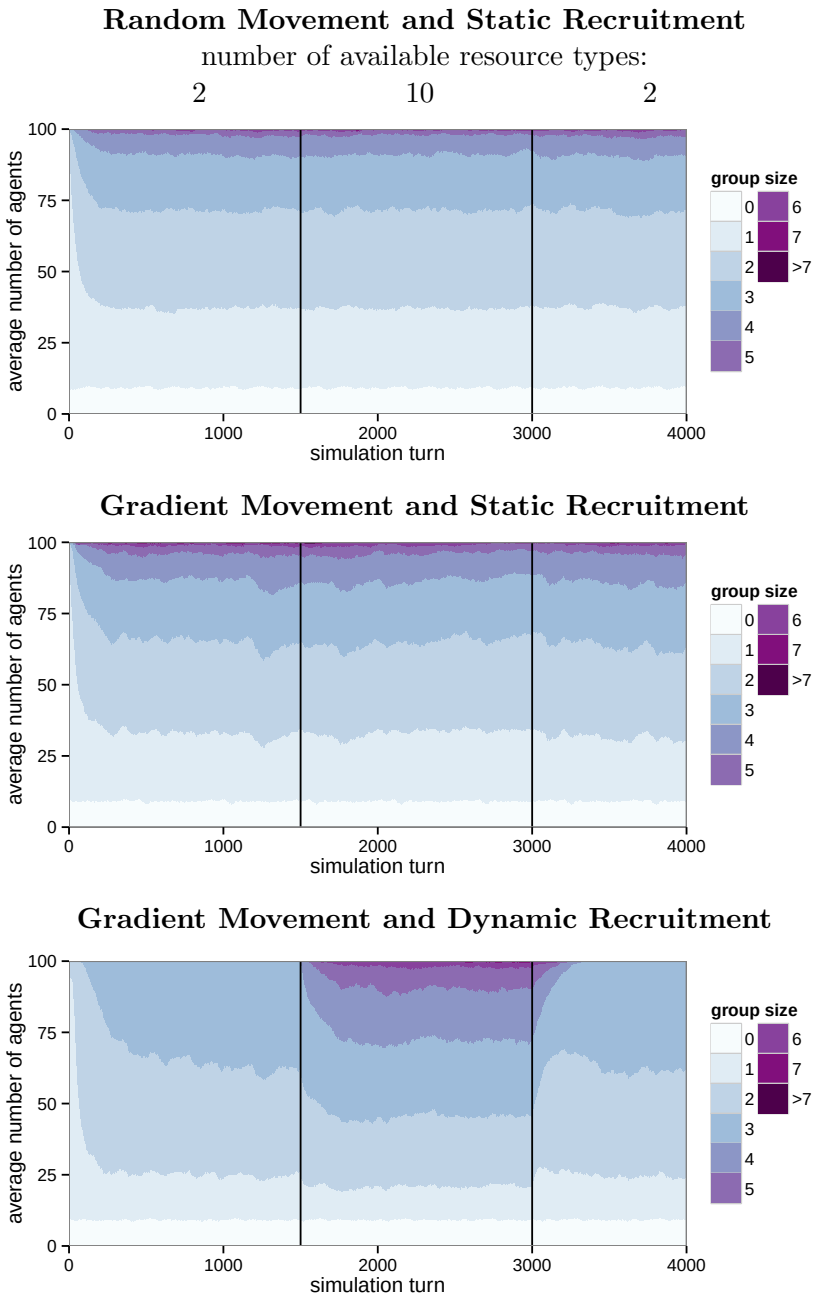


Figure 5.4: The total average number of agents in groups of different sizes: in the random-static system (top), gradient-static system (middle), and gradient-dynamic system (bottom). Reloading agents are listed as agents in a group of size zero.

the gradient-static system during simulation turns with only two available resource types. However, the average group size in the gradient-dynamic system is higher than in the gradient-static system when ten resource types are available. In the gradient-dynamic system most agents are in groups of size two or three when there are two resources types to collect. Once the average group size has adapted to the increase of available resource types – after simulation turn 1500 – over half of the agents are in groups of at least size 3. This adaptation takes approximately 250 simulation turns.

Beginning with simulation turn 3001 the number of active resources is once more reduced to two. The agents of the gradient-dynamic system stop their recruitment immediately, and the system shows a fast decrease in group size. This reaction overshoots and reduces the group size to a point below the systems steady state. The number of groups with less than three agents decreases in favor of a slightly increasing average group size before the system returns to a steady state.

Idle Slots The previous paragraph gave insight on the quantitative properties of the group formation processes. A measure to infer the qualitative properties of a group is the number of idle slots the agents have. Figure 5.5 gives a time series over the number of idle slots in the gradient-static and gradient-dynamic system. During the first 500 simulation turns the system slowly adapts to the two available resources with the random reconfiguration processes. About every eleventh agent reloads and has ten idle slots during that time. Another 15% of the agents have more than seven idle slots, because they are in groups with two other agents that already specialized on the available resource types. From simulation turn 1500 to 3000, with ten available resource types, barely any agent has any idle slot (not counting reloading agents).

Agents in the gradient-dynamic system have an average of 2.5 idle slots when two resource types are available (measured from simulation turn 1001 to simulation turn 1500). In the gradient-static system this number is with 2.8 significantly higher (p-value < 0.01, Mann-Whitney test). With ten available resource types, the average number of idle slots per agent is as low as 0.99 in the gradient-dynamic and 0.97 in the gradient-static system (measured from simulation turn 2501 to 3000). Though the difference between the averages is small, the number of idle slots is significantly higher (Mann-Whitney test with p-value < 0.01) in the gradient- dynamic system than in the gradient-static system. After the number of resource types is reset to two in simulation turn 3000 the groups have a large average number of idle

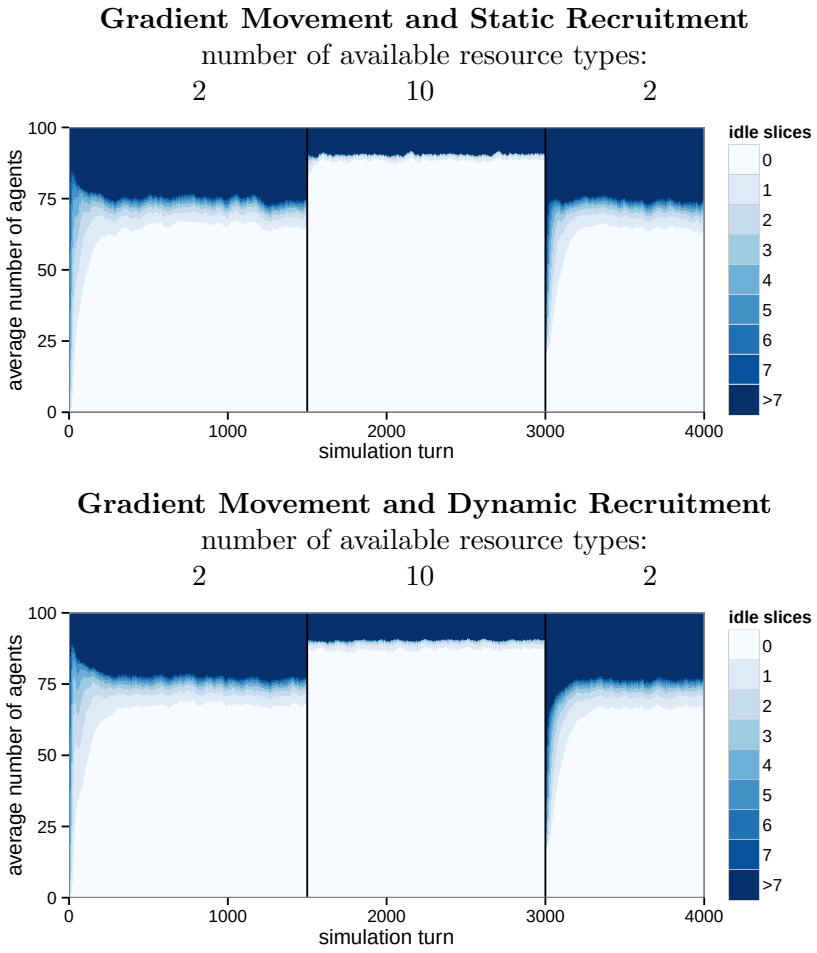


Figure 5.5: Average number of agents with the specified number of idle slots: gradient-static system (top) and gradient-dynamic system (bottom). Reloading agents are listed as agents with ten idle slots.

slots immediately after the change. However, the group sizes of both compared systems adapt to the number of available resource types and reduce the number of idle slots once again. The gradient-dynamic system soon has a lower average number of idle slots than the gradient- static system. The gradient-static recruitment system is slightly faster decreasing the number of idle slot, because the gradient- dynamic system has a higher average group size at the time of the switch. The larger groups have a higher number of idle slots. To adapt to the change, the agents in the gradient-static system only mutate their configuration into a more fitting state, but the average groups sizes do not change. The gradient-dynamic system simultaneously reduces the size of the groups. As the groups can only decrease their size if a member agent becomes inactive, this process takes up to 500 simulation turns.

5.2.4 Performance Analysis

All performance results are significantly different from each other (Friedman test using a Nemenyi post-hoc test with p-value < 0.01), unless stated otherwise.

The performance of a group G is determined by the capabilities of its members and the resource availability at their current location. Recall Equation (3.2)

$$P(G) = \frac{1}{s} \sum_{j=1}^{k} \max_{a_i \in G} a_{ij} f_j \qquad P(G) \in [0, k] \tag{3.2}$$

The actual amount of collected resources can decrease if agents have to pay for their reconfiguration with c_{reconf}, such that the amount of resources collected by agent a in group G is defined by

$$P(a \in G) = \begin{cases} (1 - c_{reconf})P(G) & \text{if a slot of } a \text{ mutated} \\ P(G) & \text{otherwise.} \end{cases}$$

Figure 5.6 shows the performance of the different systems in light of three different reconfiguration costs. The dynamic recruitment systems and the gradient movement systems perform better than their respective counterparts with static recruitment and random movement. Irrespective of the reconfiguration cost, the system with gradient movement paired with dynamic recruitment performs best. However, the improvement achieved by gradient movement is small compared to the positive impact of dynamic recruitment.

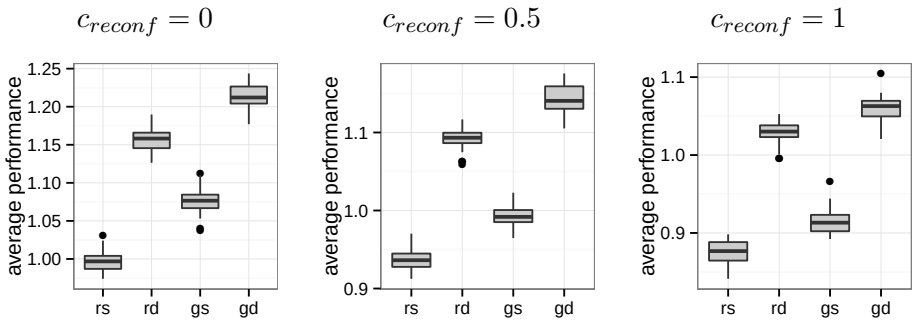


Figure 5.6: Performance of the different systems, r = random movement, g = gradient movement, s = static recruitment, d = dynamic recruitment; Average over all simulation turns and agents in one run.

5.2.5 Robustness

The model has some parameters with considerate influence on the systems performance. Five of them are analyzed in the following. Note, ahead that all presented performance results are significantly different from each other (Friedman test using a Nemenyi post-hoc test with p-value < 0.01), unless stated otherwise to increase readability.

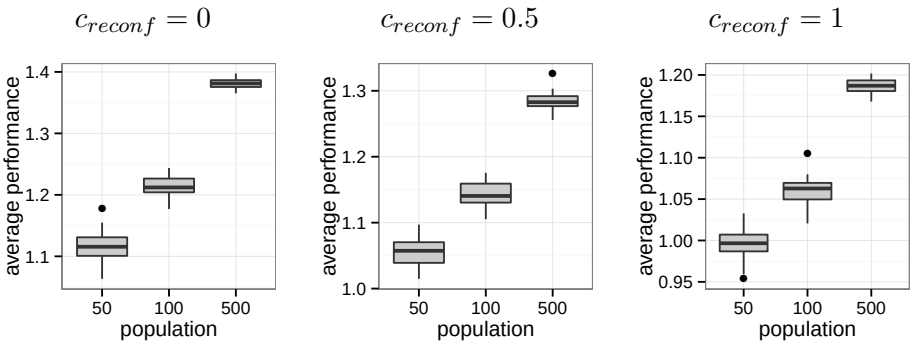


Figure 5.7: Average performance of agents in the gradient-dynamic system with 50, 100, and 500 agents. Average over all simulation turns and agents in one run; boxplot over 50 runs.

Number of Agents Varying the number of agents n in the gradient-dynamic system is similar to changing the size of the arena, as it sets the

density of agents per field. A higher density of agents leads to significantly larger groups (pairwise Mann-Whitney tests with p-values < 0.01) because collisions between agents are more frequent. Their average group sizes are 2.0, 2.2, 2.6 for $n = 50, 100, 500$, respectively. Note, however, that the group sizes never exceed three whenever there are only two resource types available. Figure 5.7 shows the influence of the number of agents on the gradient-dynamic systems average performance. The different reconfiguration costs reduce the average performance but do not change the fact that systems with large n perform better than systems with smaller n.

Velocity of the Agents If the agents are unable to measure the gradient between their current and next field, i.e. in the random-dynamic system, they decide at random whether to move forward or rotate on their current field. How many fields per simulation turn they visit on average is controlled by the velocity parameter. Figure 5.8 shows the influence of different agent velocities on the random-dynamic systems performance. While higher velocities seem to improve the system, the performance of the gradient-dynamic system has a higher performance then any of the systems with random movement. The performance of systems with a velocity of 0.6 and 0.9 are not significantly different.

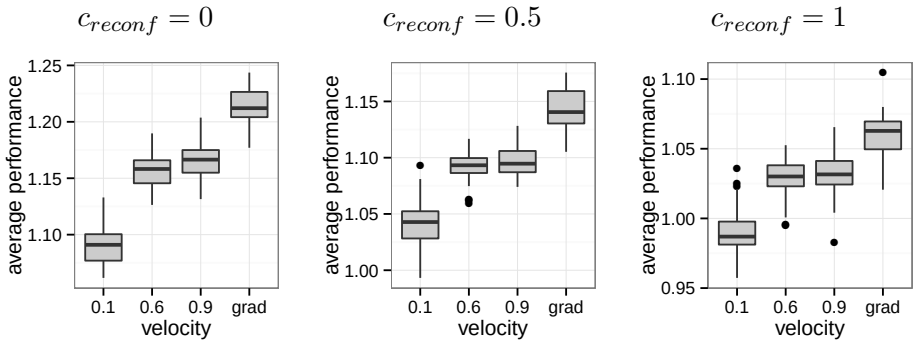


Figure 5.8: Average performance of agents in the random-dynamic system with different velocity values in comparison to the gradient-dynamic system (grad). Average over all simulation turns and agents in one run; boxplot over 50 runs.

Recruitment Rate In the gradient-static system higher recruitment rates increase the average size of the groups. In a large group, with at least one specialized agent for each type of resource, every agent can collect the max-

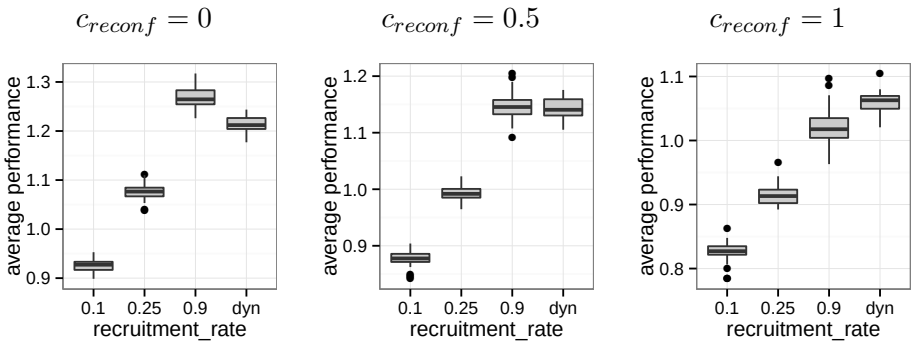


Figure 5.9: Average performance of agents in the gradient-static system with different recruitment rates in comparison with the gradient-dynamic system. Average over all simulation turns and agents in one run; boxplot over 50 runs.

imum amount of resources. Agents that are not better than all other agents for at least one resource type, constantly reconfigure their slots. This can be a problem in the presence of high reconfiguration costs. Therefore, the gradient-static system only outperforms the gradient-dynamic system with high recruitment rates and when there are no reconfiguration costs (see Figure 5.9). With high reconfiguration costs the dynamic recruitment system is better than any static recruitment system. The dynamic recruitment system benefits from the agents ability to adapt their group size. They form large groups when many different resources types are available and smaller groups, otherwise. With medium reconfiguration cost of 0.5 the performance of the best gradient-static system is not significantly different from the gradient-dynamic system.

Mutation Strength The mutation strength μ regulates how fast an agent can adapt its slots into a beneficial state. A high mutation strength reduces the time an agent needs to be proficient in the collection of one specific resource type and can increase the performance of the group. On the other hand it also increases the number of reconfiguration events for agents, that can be considered superfluous in their groups, as all other group members already cover the available resource types with their capabilities.

Figure 5.10 shows the influence of the mutation strength on the standard system. Note, with $\mu = 0$ the system is indeed not reconfigurable. The benefit of the fast adaption of the agents is, even with the highest reconfiguration costs, is higher than the loss of performance by the superfluous agents in the groups. Increasing reconfiguration costs, such that a reconfiguring slot

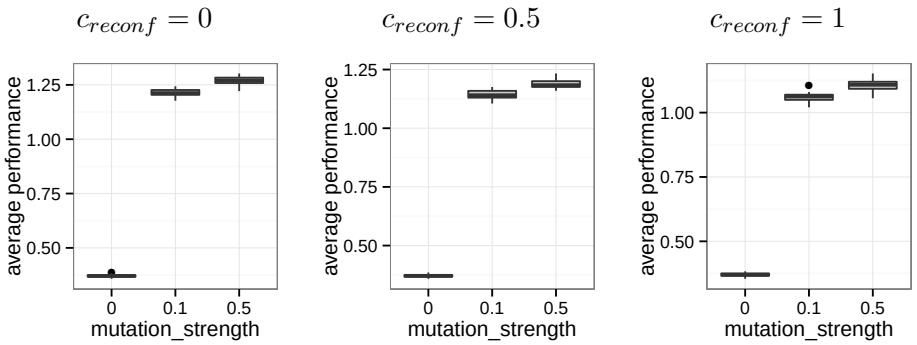


Figure 5.10: Average performance of agents in the gradient-dynamic system with different values for the mutation strength. Average over all simulation turns and agents in one run; boxplot over 50 runs.

disables the agent for multiple simulation turn, will at some point change this phenomenon.

Maximal Velocity of the Resources The velocity of the center points of the resource types influences the performance of the gradient-dynamic system. A slow movement, or even lack thereof, makes it easier for the agents to detect profitable fields and follow the center points. The higher the fluctuation in the environment by faster moving center points, the lower is the performance of the system (see Figure 5.11).

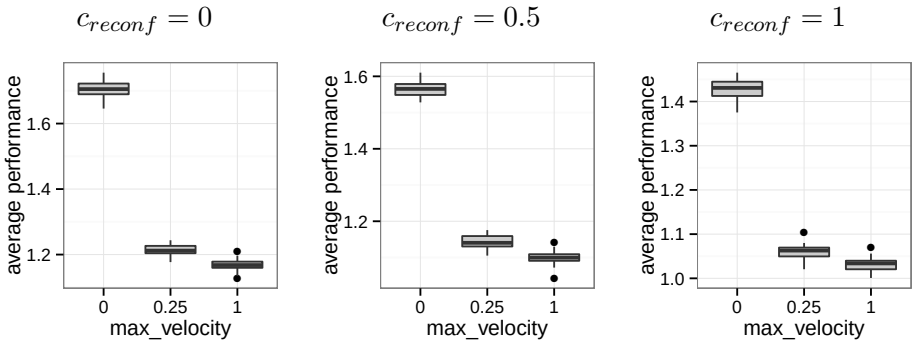


Figure 5.11: Average performance of agents in the standard model with different values for the resources maximal velocity. Average over all simulation turns and agents in one run; boxplot over 50 runs.

5.3 Conclusion

If the benefit of cooperation is bound by some environmental and dynamic influence, a dynamic decision tactic to join into a cooperation can be very profitable. The varying and reconfigurable set of capabilities of the agents provides a very heterogeneous and adaptive swarm of agents. By cooperation through group formation agents can use each others capabilities. It is every agents goal to become member of a group that ideally consists of one specialist for each capability that is currently asked for by the dynamic environment.

The results showed that, when agents base their decisions on their current workload, the group formation of the agents is adaptive and increases the overall performance of the system. In a considerably small amount of time the swarm of agents adapts to larger changes in their environment. The impact of a different movement behavior, i.e. random or gradient movement, had less impact on the performance in such dynamic systems. However, the gradient movement is more beneficial for the agents than the random movement. The system with dynamic recruitment and gradient movement performs best, especially in the presence of reconfiguration costs. The system prove to be robust and behave expectedly to variations in the values of the most crucial parameters.

Part III

Multi-Objective Ranking Relations

6 Introduction

Ranking multi-objective solutions

In daily life, but also in industrial, engineering, and scientific contexts problems with several optimization criteria, e.g. time, price, quality, or ecological concerns, are common. While optimization problems with one objective can be hard to solve optimally at least the simple $<$ relation tells which solution is better than another one. In *multi-objective* optimization problems this is not possible. While the problem asks for solutions from a solution space X that are optimal with respect to $k > 1$ objectives, the objectives are typically conflicting and it is only possible to optimize a (small) subset of the objectives simultaneously. This is where the concept of the *Pareto dominance* relation takes effect. One solution dominates another one if it is better in at least one objective and not worse in all other objectives. Based on this relation a solution is called *Pareto optimal* if it is not dominated by any other solution from X. The set of all Pareto optimal solutions is referred to as Pareto set (solution space) that build the Pareto front (objective space). In multi-objective optimization the goal is to find the Pareto set or at least a subset of it. However, in light of NP-hard problems, determining the Pareto set is rarely an option. A widely accepted alternative to avoid this problem is to settle with a set of solutions that are close to the Pareto optimal solutions, do not dominate each other, and feature a high diversity (see Figure 6.1 for a visualization of the concept).

For the exploration of the solution space the use of metaheuristics maintaining a population of solutions seems beneficial. Therefore, multi-objective variants of various population-based metaheuristics have been developed in recent years. A population is a set of solutions that is used to construct new solutions and to lead the algorithm into more beneficial regions of the solution space. Very popular population-based metaheuristics are genetic algorithms (GAs). Multi-objective variants are being developed since

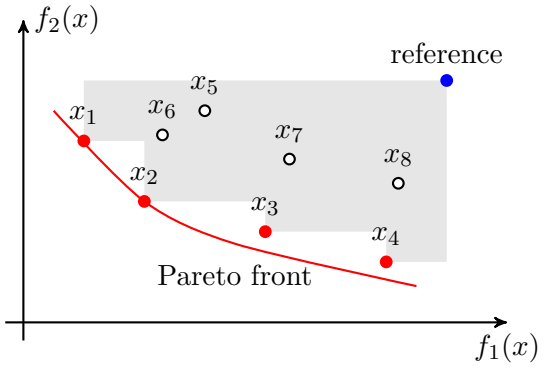


Figure 6.1: Solutions $x_1, \dots x_8$ in their two-dimensional objective space and the actual (usually unknown) Pareto front (red line). Solutions x_1 through x_4 are non-dominated with respect to the Pareto dominance relation. The gray area gives the hypervolume of the non-dominated front (red points) with respect to a reference point (blue).

the eighties of the last century, e.g. the Vector Evaluation Genetic Algorithm (VEGA) (Schaffer, 1984). For overviews on multi-objective GAs see Coello Coello (2009); Coello Coello et al. (2005); Fonseca and Fleming (1995). Another metaheuristic adapted to solve multi-objective problems is the Ant Colony Optimization metaheuristic (ACO) (Dorigo et al., 1999), e.g. by Doerner et al. (2001); Iredi et al. (2001) (for overviews see Angus and Woodward (2009); Leguizamón and Coello Coello (2011)). The population-based ACO (P-ACO) (Guntsch and Middendorf, 2002) has also been extended to handle multi-objective problems (Guntsch and Middendorf, 2003).

All these metaheuristics generate a great number of solutions during their runtime, where the population typically represents the best of all those solutions. It can be beneficial for the algorithms performance to restrict the size of the population and only allow a few choice solutions in it. To chose these solutions from the large set of available solutions constructed in one iteration a ranking relation is required.

6.1 Ranking Relations

The Pareto dominance is the most intuitive relation when it comes to ranking solutions of multi-objective problems. A solution $a \in X$ is only ranked higher than another solution $b \in X$ if a is better or equal than b concerning all objectives, without being equal in all objectives. However, the Pareto dominance relation is not total and many solutions, e.g. all solutions of the non-dominated front, can not be ranked. This is why the Pareto dominance is insufficient to guide heuristics into favorable regions of the solution space, especially when the number of objectives is large. To salvage this situa-

tion several methods to compare, rank, and prune sets of (non-dominated) solutions have been proposed in the past.

An approach proposed by Kukkonen and Deb (2006) is based on clustering the set of solutions and taking one representative solution from each cluster to obtain a large diversity in the set of selected solutions. Another option is the introduction of a ranking relation that provides ranks for solutions within non-dominated sets. The easiest approach are ranking schemes based on aggregation methods such as the use of weighted sums of the objectives (Jakob et al., 1992) or the distance to a target vector of objective values (Wienke et al., 1992). Other ranking methods are more objective based. For example, there are rankings using given priorities for a lexicographic sorting of the objectives or performing a separate optimization of the single objectives in an order that is compliant with these priorities (Cvetkovic and Parmee, 2002). Weights, target vectors, and priorities should be given by the user. These approaches can, however, be difficult or impossible for many applications, e.g. when no reasonable weights are known. We then require methods that do not depend on any predefined weights but are, for example, based on the notion of Pareto dominance. The *dominance rank* counts for each solution $a \in X$ the number of solutions $b \in X$ that are dominated by a, while the *dominance count* gives the number of solutions $c \in X$ dominating a (Fonseca and Fleming, 1993; Zitzler and Thiele, 1999). The *dominance depth* gives the number of times the current set of non-dominated solutions has to be removed from the remaining set of solutions until a becomes non-dominated (Srinivas and Deb, 1994). Note, that in a set of non-dominated solutions these values fail to provide a ranking.

The *Global Detriment method* (Garza-Fabre et al., 2009) uses the sum of the differences of one solution $a \in X$ to all other solutions $b \in X$ over all those objectives where a is inferior to b. The two methods *Profit* and *Distance to the Best Known Solution* (Garza-Fabre et al., 2009) are similar to the Global Detriment method where the first includes a normalization over the different objectives and the second is based on a comparison with the best known objective values. However, methods based on the difference in objectives bear the potential disadvantage that they typically need some normalization between the different objectives (i.e. the objective functions and their derivatives should be in the same range, respectively).

It is possible to avoid any normalization when the differences between the objectives values are mostly ignored and only their sign is taken as information for the ranking. Thus, the magnitude of the differences of the objective values are disregarded. *Relation favor* (Drechsler et al., 2001) counts the number of objectives for which one solution is better or worse

than the other. A solution is preferred over another if it wins, i.e. is better, in more objectives than the other. Several extensions or similar relations have been proposed since then. They give alternatives for the decision of when an objective is counted as won or lost (Laumanns et al., 2002; Sülflow et al., 2007) or require a specific number of won objectives for preference (Farina and Amato, 2004; Zou et al., 2008).

Approaches based on the number of won objectives can also be used to rate a solution with respect to a set of solutions, e.g. (Bentley and Wakefield, 1998; Maneeratana et al., 2006; Mostaghim and Schmeck, 2008). A detailed description of such methods is given in Section 7.2.

Other approaches do not rank single solutions but sets of them. This is commonly done in the indicator function framework, which assigns a real value to a set of solutions. The *binary hypervolume indicator* by Zitzler and Thiele (1998), for example, is defined as the hypervolume of the objective space dominated by one set of solutions but not by a respective other (see Figure 6.1 for a simple example with only two dimensions and one point in the reference set). A good overview and experimental comparison over some ranking methods can be found in Garza-Fabre et al. (2010, 2009).

6.2 Correlation of Objectives

Another aspect of multi-objective optimization that is overlooked far too often is the correlation between different objectives. Objectives are often enough not independent from each other but correlated (Xu et al., 2013). Especially, when the performance of metaheuristics is tested on randomly created problem instances or on instances from benchmark libraries, the objectives of these instances are often independent (in particular when random instances are used) or the correlation is not investigated. The correlation between objectives can influence the correlation between the optima of a multi-objective problem (Knowles and Corne, 2002). The optima are in general very different if their objectives are not positively correlated (Jaszkiewicz, 2002).

In fact, considering that the performance of local search operators for multi-objective problems is strongly influenced by the strength of the correlation between the objectives (Paquete and Stützle, 2006), correlations should also be investigated in the application of metaheuristics. The ACO, for example, performs better with 'less aggressive' strategies (e.g. iteration-best instead of best-so-far pheromone update) when the objectives are highly positively correlated (López-Ibánez et al., 2004). On instances with weakly

or negatively correlated objectives this does not hold. Positively correlated objectives decrease – in contrast to negatively correlated objectivese – the number of solutions in the Pareto front for NK-landscapes (Verel et al., 2011a). This might enable metaheuristics to find a large fraction of them. Past studies showed the influence of the correlation of objectives on hybrid metaheuristics (Garrett et al., 2007) and the co-influence of the correlation of objectives, the objective space dimensions, and the degree of non-linearity on the size of the Pareto set (Verel et al., 2011b).

Ishibuchi et al. (2013a, 2011, 2013b) investigated the influence of correlated objectives on evolutionary multi-objective algorithms for the multi-objective 0/1 Knapsack problem. The performance of MOEA/D is – unlike the performance of the NSGA-II and SPEA2 – severely degraded by an increase in the number of objectives when they were highly correlated (Ishibuchi et al., 2011). The NSGA-II and SPEA2 perform well on multi-objective problems with highly correlated objectives (Ishibuchi et al., 2013a). Also, the search of the hypervolume-based SMS-EMOA is biased toward the region of the Pareto front with good values for duplicated objectives (Ishibuchi et al., 2013b). If objectives are highly positively correlated a dimensionality reduction can help metaheuristics (Brockhoff et al., 2008; Goel et al., 2007). To this end, positively correlated objectives can be aggregated in groups (Murata and Taki, 2010).

6.3 Ranking Relations on Correlated Objectives

Part III of this study focuses on the impact of different ranking methods for metaheuristics by applying them in each iteration to select only a few solutions from the set of already found solutions. These few solutions are then used as population in to generate new solutions in the following iteration. There are some advantages in maintaining only a few solutions in the population, such as memory and runtime requirements of the algorithm. This is of great importance in restrictive computational environments, e.g. when solutions need to be delivered in real time or the algorithm runs on specific hardware. The smaller the population the higher is the influence of a single solution in the population. This increases the influence of the ranking method dramatically.

With the theoretic and empiric analysis of two ranking relations this study shows benefits and pitfalls of different strategies on multi-objective optimization with metaheuristic. Both of these ranking relations refine the Pareto dominance relation and are total preorders. In a comparison

with other state-of-the-art ranking relations they are applied in a P-ACO and a GA solving a multi-objective Flow-Shop-Scheduling and Traveling-Salesperson problem.

Further, the influence of different correlations between the objectives on the performance of the algorithms using the compared ranking relations is studied. To this end, I develop a simple method for the instance generation of problems with multiple differently correlated objectives.

Most methods and results have been published in Moritz et al. (2014a,b, 2013).

7 Ranking Relations

Refinements of the Pareto dominance

7.1 Definitions

In the following I consider a multi-objective optimization problem with a set of solutions X and a vector of objective functions

$$\vec{f}(a) = (f_1(a), \ldots, f_k(a)),$$

where $a \in X$ and $f_i : X \mapsto \mathbb{R}$. A solution of the problem can be a vector of real values in case of continuous optimization problems or a vector of elements from a finite set in case of combinatorial optimization problems to name two common variants. The task at hand is to find solutions from X that minimize the objectives, i.e.

$$\min_{a \in X} \vec{f}(a) = \min_{a \in X}(f_1(a), \ldots, f_k(a)). \tag{7.1}$$

Note, that by using $-f_i(a)$ maximization is also possible. With $d > 1$ there can be multiple minimal solutions that are inferred by the use of the Pareto dominance relation ($\prec$). Let $a, b \in X$, then

$$a \prec b \iff \forall i \in \{1, \ldots, k\} : f_i(a) \leq f_i(b) \wedge \vec{f}(a) \neq \vec{f}(b).$$

Solution a *dominates* b if $a \prec b$. Note that, if $a \prec b$ then there is at least one $i \in \{1, \ldots, k\}$ with $f_i(a) < f_i(b)$. Two solutions $a, b \in X$ are called incomparable if $a \not\prec b \wedge b \not\prec a$ or indifferent in case of $\vec{f}(a) = \vec{f}(b)$. A solution $a \in X$ is called *Pareto optimal* if $\nexists b \in X : b \prec a$. A solution $a \in X$ is called *non-dominated* solution with respect to a subset $X' \subseteq X$ if $\nexists b \in X' : b \prec a$. The set of all Pareto optimal solutions from X is called the *Pareto set* and the corresponding set of objective vectors in $\mathbb{R}^k$ is called the *Pareto front* of X.

Table 7.1: Properties of a relation R on a set X.

Property	Definition
reflexive	$\forall a \in X\colon (a,a) \in R$
irreflexive	$\forall a \in X\colon (a,a) \notin R$
transitive	$\forall a,b,c \in X\colon (a,b),(b,c) \in R \Rightarrow (a,c) \in R$
symmetric	$\forall a,b \in X\colon (a,b) \in R \Rightarrow (b,a) \in R$
asymmetric	$\forall a,b \in X\colon (a,b) \in R \Rightarrow (b,a) \notin R$
antisymmetric	$\forall a,b \in X\colon (a,b),(b,a) \in R \Rightarrow a = b$
total	$\forall a,b \in X\colon (a,b) \in R \vee (b,a) \in R$
preorder on X	R reflexive and transitive
total preorder on X	R is a preorder on X and total
partial order on X	R is reflexive, transitive, and antisymmetric
total order on X	R is a partial order on X and total

Table 7.1 gives an overview over the different properties of relations that are needed in the following.

Let R and S be two relations on a set X. Then S is a *refinement* of R if

$$\forall a,b \in X : (a,b) \in R \wedge (b,a) \notin R \Rightarrow (a,b) \in S \wedge (b,a) \notin S.$$

A refinement of the Pareto dominance relation keeps the order given by the Pareto dominance while it could make solutions comparable that are incomparable with the Pareto dominance. Given an irreflexive and asymmetric relation R over a set W, we define for two vectors $\vec{u}, \vec{v} \in W^k$

$$\begin{aligned}
\mathcal{B}^R(\vec{u},\vec{v}) &= |\{i : (u_i,v_i) \in R, i \in \{1,\dots,k\}\}| \\
\mathcal{E}^R(\vec{u},\vec{v}) &= |\{i : (u_i,v_i) \notin R, (v_i,u_i) \notin R, i \in \{1,\dots,k\}\}| \\
\mathcal{W}^R(\vec{u},\vec{v}) &= |\{i : (v_i,u_i) \in R, i \in \{1,\dots,k\}|
\end{aligned}$$

Due to the asymmetry and irreflexivity each element $i \in \{1,\dots,k\}$ is included in exactly one of the three sets used to compute $\mathcal{B}^R(\vec{u},\vec{v})$, $\mathcal{E}^R(\vec{u},\vec{v})$, and $\mathcal{W}^R(\vec{u},\vec{v})$, i.e.

$$\mathcal{B}^R(\vec{u},\vec{v}) + \mathcal{E}^R(\vec{u},\vec{v}) + \mathcal{W}^R(\vec{u},\vec{v}) = k.$$

When we use this notion with the $<$ relation to compare solutions $a, b \in X$ with respect to their values in the objective space $\vec{f}(a), \vec{f}(b) \in \mathbb{R}^d$, we

have

$$\mathcal{B}^{<}\Big(\vec{f}(a),\vec{f}(b)\Big) = |\,\{i : f_i(a) < f_i(b), i \in \{1,\ldots,k\}\}\,|$$
$$\mathcal{E}^{<}\Big(\vec{f}(a),\vec{f}(b)\Big) = |\,\{i : f_i(a) = f_i(b), i \in \{1,\ldots,k\}\}\,|$$
$$\mathcal{W}^{<}\Big(\vec{f}(a),\vec{f}(b)\Big) = |\,\{i : f_i(b) < f_i(a), i \in \{1,\ldots,k\}\}\,|$$

For better readability we write $\mathcal{B}^R(a,b)$ instead of $\mathcal{B}^R\Big(\vec{f}(a),\vec{f}(b)\Big)$ when we compare two solutions $a, b \in X$. We handle $\mathcal{E}^R(a,b)$ and $\mathcal{W}^R(a,b)$ analogously. The values $\mathcal{B}^{<}(a,b)$, $\mathcal{E}^{<}(a,b)$, and $\mathcal{W}^{<}(a,b)$, respectively, counts the number of objectives where a solution a is better (respectively equal, worse) than a solution b (Drechsler et al., 2001; Farina and Amato, 2004; Sülflow et al., 2007; Zou et al., 2008).

For a solution $a \in X$, a set $X' \subseteq X$ of solutions and an irreflexive and asymmetric relation $R \subseteq X \times X$ a similar notion is introduced:

$$\mathcal{B}^R\big(a,X'\big) = |\{x : (a,x) \in R, x \in X'\}|$$
$$\mathcal{E}^R\big(a,X'\big) = |\{x : (a,x) \notin R, (x,a) \notin R, x \in X'\}|$$
$$\mathcal{W}^R\big(a,X'\big) = |\{x : (x,a) \in R, x \in X'\}|$$

Note, that

$$\mathcal{B}^R\big(a,X'\big) + \mathcal{E}^R\big(a,X'\big) + \mathcal{W}^R\big(a,X'\big) = |X'| \tag{7.2}$$

where $|X|$ is the number of elements in X'. Mind, that $\mathcal{B}^R(a,v)$ and $\mathcal{B}^R(a,\{v\})$ have a different meaning. For $X'', X' \subseteq X$ define $\mathcal{B}^R(X'',X') = \sum_{a\in X''} \mathcal{B}^R(a,X')$. Analogously $\mathcal{E}^R(X'',X')$ and $\mathcal{W}^R(X'',X')$ are defined.

7.2 State of the Art Ranking Relations

In the past two decades several approaches to compare non-dominated solutions based on certain relations between their objective values have been proposed. Those most relevant for this study are reviewed in the following.

7.2.1 The Favor Relation

The *favor relation* by Drechsler et al. (2001) is a basic yet effective relation to compare multi-objective solutions with each other. It simply counts the number of objectives that a solution $a \in X$ is better than a solution $b \in X$

and vice versa. The solution with the higher count is preferred. Using the previously defined notation, we have

$$a \prec_f b \iff \mathcal{B}^<(a,b) > \mathcal{B}^<(b,a)$$

Obviously, $\prec_f$ is an irreflexive and asymmetric relation. It is not transitive (Drechsler et al., 2001) as can be seen in a small example with $\vec{f}(a) = \{2,2,2,3\}$, $\vec{f}(b) = \{3,2,3,1\}$, and $\vec{f}(c) = \{4,2,1,2\}$ where $a \prec_f b$, $b \prec_f c$ but $a \not\prec_f c$. The relation is not total, since ties in the number of respectively better objectives lead to incomparability. Note, that with $k \leq 2$, i.e. for two or one objective, the favor relation and the Pareto dominance relation are equivalent (Drechsler et al., 2001). Indeed, the favor relation is a refinement of the Pareto dominance relation, as shown in the following.

Theorem 3. *The favor relation is a refinement of the Pareto dominance relation on X, i.e. for $a, b \in X$: $a \prec b \wedge b \not\prec a \Rightarrow a \prec_f b \wedge b \not\prec_f a$.*

Proof. It suffices to show that $a \prec b$ implies $a \prec_f b$ as the second part of premise and conclusion are a direct consequence of the asymmetry of $\prec$ and $\prec_f$. By definition of the Pareto dominance relation

$$\begin{aligned} \nexists i \in \{1,\ldots,d\} : f_i(b) < f_i(a), \text{ i.e. } \mathcal{B}^<(b,a) = 0, \\ \text{and } \exists i \in \{1,\ldots,d\} : f_i(a) < f_i(b), \text{ i.e. } \mathcal{B}^<(a,b) > 0 \end{aligned}$$

This implies $a \prec_f b$. □

Proposition 7.2.1 introduces some properties on the combination of the Pareto dominance relation and the favor relation relevant for the following section.

Proposition 7.2.1. *Let $a, b, c \in X$ with $a \prec b$. Then:*
(i) $b \prec_f c \Rightarrow a \prec_f c$,
(ii) $c \not\prec_f b \Rightarrow c \not\prec_f a$,
(iii) $c \prec_f a \Rightarrow c \prec_f b$
(iv) $a \not\prec_f c \Rightarrow b \not\prec_f c$.

Proof. Because $a \prec b$, with $a, b \in X$, for any $c \in X$ it holds that
(1) $\mathcal{B}^<(b,c) \leq \mathcal{B}^<(a,c)$, because $\forall k \in \{i : f_i(b) < f_i(c)\} : f_k(a) \leq f_k(b)$, and
(2) $\mathcal{B}^<(c,b) \geq \mathcal{B}^<(c,a)$, because $\forall k \in \{i : f_i(c) < f_i(a)\} : f_k(a) \leq f_k(b)$.

These inequalities together with the corresponding premise of the implication gives the result.

(i) $(b \prec_f c \Rightarrow a \prec_f c)$ Due to $b \prec_f c$ we have $\mathcal{B}^<(b,c) > \mathcal{B}^<(c,b)$ yielding:

$$\mathcal{B}^<(a,c) \geq \mathcal{B}^<(b,c) > \mathcal{B}^<(c,b) \geq \mathcal{B}^<(c,a)\,.$$

By definition this is equivalent to $a \prec_f c$.

(ii) $(c \not\prec_f b \Rightarrow c \not\prec_f a)$ From $c \not\prec_f b$ we know $\mathcal{B}^<(c,b) \leq \mathcal{B}^<(b,c)$ and thus

$$\mathcal{B}^<(c,a) \leq \mathcal{B}^<(c,b) \leq \mathcal{B}^<(b,c) \leq \mathcal{B}^<(a,c)\,,$$

i.e. $c \not\prec_f a$.

(iii) $(c \prec_f a \Rightarrow c \prec_f b)$ Similarly $c \prec_f a$ gives

$$\mathcal{B}^<(c,b) \geq \mathcal{B}^<(c,a) > \mathcal{B}^<(a,c) \geq \mathcal{B}^<(b,c)\,,$$

i.e. $c \prec_f b$.

(iv) $(a \not\prec_f c \Rightarrow b \not\prec_f c)$ Finally $a \not\prec_f c$ gives

$$\mathcal{B}^<(b,c) \leq \mathcal{B}^<(a,c) \leq \mathcal{B}^<(c,a) \leq \mathcal{B}^<(c,b)\,,$$

i.e. $b \not\prec_f c$.

□

A relation R on X can be represented by a graph $G_R = (V, E)$ where the nodes represent the elements of X, i.e. $V = X$ and the edges are defined by R, i.e. $E = R$. A graph G representing the favor relation is a directed graph because the relation is asymmetric with $(b,a) \in E$ iff $a \prec_f b$. $G_{\prec_f}\left(X, \vec{f}\right)$ is then called the *favor graph* of X with the vector $\vec{f}$ of objective functions. The following lemma shows that every directed graph can be the favor graph of V and some $\vec{f}$.

Lemma 7.2.1. *For every directed graph $G = (V, E)$ that is loop-free (i.e. $\forall a \in V : (a,a) \notin E$) and has at most one edge between two nodes there exists a vector of objective functions $\vec{f}$ on the set $X = V$ such that the favor graph $G_{\prec_f}(X, \vec{f}$ of V and $\vec{f}$ is isomorphic to G.*

Proof. Let $G = (V, E)$ be given with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. For the construction let $\vec{f} = \{f_1, \ldots, f_{3m}\}$ be a $3m$-dimensional vector of objective functions. For each $v_r \in V$ let $v_r^{(h)}$, with $h \in \{1, \ldots, m\}$, be the three dimensional subvector of $\vec{f}(v_r)$ which defines the positions $\{3h-2, 3h-1, 3h\}$ in $\vec{f}(v)$. We define the objective functions such that for each edge $e_h = (v_i, v_j) \in E$ we have

$$v_r^{(h)} = \begin{cases} (1,1,2) & \text{if } k = i \\ (2,2,1) & \text{if } k = j \\ (1,3,1) & \text{otherwise} \end{cases}, \; r \in \{1, \ldots, m\}$$

By the construction, it follows that if there exist an edge $e_h = (v_i, v_j) \in E$ then $v_j^{(h)} \prec_f v_i^{(h)}$ and $v_j^{(l)} \not\prec_f v_i^{(l)} \wedge v_i^{(l)} \not\prec_f v_j^{(l)}$ for $r \in \{1, \ldots, m\}$, $l \neq h$. If two nodes $v_i, v_j \in V$ are not connected by an edge in G then $v_j^{(h)} \not\prec_f v_i^{(h)} \wedge v_i^{(h)} \not\prec_f v_j^{(h)}$ for all $h \in \{1, \ldots, m\}$. For vectors $v_i, v_j \in V$ it follows that the favor relation $v_i \prec_f v_j$ holds iff $(v_j, v_i) \in E$. Hence, the favor graph of V and $\vec{f}$ is isomorphic to G. □

There are several variations of the favor relation. The *ϵ-preference* rates an objective f_i, $i \in \{1, \ldots, k\}$, of a solution as won only if its value is larger by a constant value ϵ_i than the corresponding value of the competitor solution (Sülflow et al., 2007). The *L-dominance* relation (Zou et al., 2008) applies a constant additive tolerance for the number of won objectives by demanding

$$\mathcal{B}^<(a, b) - L = \mathcal{W}^<(a, b)$$

Zou et al. (2008) demanded in addition that a is better than b with respect to some norm, e.g. the sum of the objective values. The *$(1-k)$-dominance* relation Farina and Amato (2004) applies a multiplicative tolerance such that $a \prec_{1-k} b$) iff

$$k \cdot \mathcal{B}^<(a, b) \geq \mathcal{W}^<(a, b)\,, \text{ with } k \in [0, 1]$$

7.2.2 Winning Score

While the favor relation compares solutions directly it is also possible to compare solutions by some score they make in comparison with sets of solutions, e.g. in the domination rank, domination count, and domination depth based ranking methods. The *winning score* (Maneeratana et al., 2006) also compares solutions in respect to a set of solutions $X' \subset X$. For a solution $a \in X'$ the winning score is defined by

$$S(a) = \sum_{b \in X'} (\mathcal{B}^<(a, b) - \mathcal{W}^<(a, b)).$$

Another such method is the *average rank* (Bentley and Wakefield, 1998). The average rank is given by

$$rank(a) = \sum_{i=1}^{k} r_i(a)$$

where $r_i(a)$ is the rank of a with respect to objective i. The winning score and average rank are equivalent, i.e. they induce the same order on the elements of X' (Corne and Knowles, 2007).

Note, that because $a \prec b$ implies that the average rank of a is smaller than the average rank of b, the winning score is a refinement of the Pareto dominance relation. It is due to the use of set scores in combination with $\leq$ a total preorder. The winning score relation is abbreviated as *Win* when it is applied in the experiments as a means of comparison.

7.2.3 Weighted Sum

Similar to the Global Detriment method (Garza-Fabre et al., 2009) the weighted sum relation – called W relation – computes a weighted sum, however only pairwise and with weights for normalization.

$$a \prec_{\mathrm{W}} b \text{ iff } \sum_{i=1}^{k} w_i f_i(a) \leq \sum_{i=1}^{k} w_i f_i(b).$$

The weighted sum relation is a refinement of the Pareto dominance relation as $a \prec b$ implies that the weighted sum of a is smaller than the weighted sum of b. Furthermore, it is a total preorder, since $\leq$ is used as comparison operator.

When this ranking method is applied we use random weights from a uniform distribution that gradually change from one iteration to the next, by averaging them with a random new weight. It is enforced that $w_i \geq 0$ for $i \in \{1, \ldots, k\}$ and $\sum_{i=1}^{k} w_i = 1$.

7.3 Two new Ranking Relations

In the following I introduce two ranking relations first mentioned by Schwarz (2011). The *WL relation* and the *Points relation* are both based on the favor relation. More precisely, they are based on $\mathcal{B}^{\prec_f}(a, A)$, $\mathcal{E}^{\prec_f}(a, A)$, or $\mathcal{W}^{\prec_f}(a, A)$ which give for a solution $a \in X$ the number of solutions in $A \subset X$ compared to which a wins, is incomparable, or loses according to the favor relation, respectively. The following theorem relates Pareto dominance to these values.

Theorem 4. *Let $A \subseteq X$ and $a, b \in X$. It holds that*

$$a \prec b \Rightarrow \mathcal{B}^{\prec_f}(a, A) \geq \mathcal{B}^{\prec_f}(b, A) \wedge \mathcal{W}^{\prec_f}(a, A) \leq \mathcal{W}^{\prec_f}(b, A)$$

If $a, b \in A$ both inequalities are strict.

Proof. Assume $a \prec b$. By Proposition 7.2.1 for every x it holds that (i) if $b \prec_f x$ then $a \prec_f x$ and (ii) if $x \prec_f a$ then $x \prec_f b$. Hence, the first sentence of the theorem holds. Furthermore, by Theorem 3 we have $a \prec_f b$. Thus, it follows tha $\mathcal{B}^{\prec_f}(a, A) \geq \mathcal{B}^{\prec_f}(b, A) + 1$ and $\mathcal{W}^{\prec_f}(a, A) - 1 \leq \mathcal{W}^{\prec_f}(b, A)$ when $a, b \in A$. □

7.3.1 The Win-Lose Relation

The Win-Lose (WL) relation is similar to score systems in tournaments where all participants, i.e. the solutions, compete against each other. A ranking is established by comparing the number of won competitions and in case of equality the number of lost competitions. Formally, we have for a set $A \subseteq X$ and solutions $a, b \in X$, a relation ${\prec_{\mathrm{WL}}}^A$ defined on X such that

$$\begin{aligned} a \,{\prec_{\mathrm{WL}}}^A\, b \Leftrightarrow\ & \mathcal{B}^{\prec_f}(a, A) > \mathcal{B}^{\prec_f}(b, A) \vee \\ & \big(\mathcal{B}^{\prec_f}(a, A) = \mathcal{B}^{\prec_f}(b, A) \wedge \mathcal{W}^{\prec_f}(a, A) \leq \mathcal{W}^{\prec_f}(b, A)\big) \end{aligned}$$

If the context is clear we may omit A in ${\prec_{\mathrm{WL}}}^A$ in the notation for better readability. Note, that the WL relation induces a mixed lexicographic order on X where it first maximizes over $\mathcal{B}^{\prec_f}(a, A)$ and secondly minimizes over $\leq \mathcal{W}^{\prec_f}(a, A)$. It is – unlike common lexicographic orders – not a partial order because antisymmetry does not hold, i.e. $a \prec_{\mathrm{WL}} b$ and $b \prec_{\mathrm{WL}} a$ does not imply $a = b$ (see Example 7.3.1), but it is a partial preorder.

Example 7.3.1. *Let* $X = \{a, b, c\}$ *with*

$$\begin{aligned} \vec{f}(a) &= (1, 2, 3) & & & a &\prec_f c \\ \vec{f}(b) &= (3, 1, 2) & & \textit{i.e.} & b &\prec_f a \\ \vec{f}(c) &= (2, 3, 1), & & & c &\prec_f b, \end{aligned}$$

$$\begin{aligned} \textit{then}\ \ & \mathcal{B}^{\prec_f}(a, X) = \mathcal{B}^{\prec_f}(b, X) = \mathcal{B}^{\prec_f}(c, X) = 1 \\ & \mathcal{W}^{\prec_f}(a, X) = \mathcal{W}^{\prec_f}(b, X) = \mathcal{W}^{\prec_f}(a, X) = 1 \end{aligned}$$

Hence, $a \prec_{WL} b \prec_{WL} c \prec_{WL} a$. *Since,* a, b, c *are pairwise unequal the* $\prec_{WL}$ *relation is not antisymmetric.*

Theorem 5. *For* $A \subseteq X$ *the WL relation* ${\prec_{WL}}^A$ *is a total preorder on* X.

Proof. A total preorder is reflexive, transitive, and total. In the following it is shown the WL relation ${\prec_{\mathrm{WL}}}^A$ is (i) transitive and (ii) total (which implies it is reflexive).

(i) Let $a, b, c \in X$ with $a \prec_{\mathrm{WL}}{}^{A} b$ and $b \prec_{\mathrm{WL}}{}^{A} c$ then $a \prec_{\mathrm{WL}}{}^{A} c$ follows from the transitivity of $>$, $=$, and $\leq$.

(ii) Under the assumption of $a \not\prec_{\mathrm{WL}}{}^{A} b$, with $a, b \in X$, there are two possible cases:

(a) $\mathcal{B}^{\prec_f}(a, A) < \mathcal{B}^{\prec_f}(b, A)$ and

(b) $\mathcal{B}^{\prec_f}(a, A) = \mathcal{B}^{\prec_f}(b, A) \wedge \mathcal{W}^{\prec_f}(a, A) > \mathcal{W}^{\prec_f}(b, A)$.

In each case the definition of the WL relation implies $b \prec_{\mathrm{WL}}{}^{A} a$.

□

These properties ensure the possibility of sorting a set of solutions according to the WL relation with sorting algorithms based on pairwise comparisons, e.g. merge sort. The following proposition is needed to show that the WL relation is a refinement of the Pareto dominance.

Proposition 7.3.1. *Let $A \subseteq X$ and $a, b \in A$. Then $a \prec b \Rightarrow b \not\prec_{WL}{}^{A} a$.*

Proof. Due to Theorem 4 $a \prec b$ implies $\mathcal{B}^{\prec_f}(a, A) > \mathcal{B}^{\prec_f}(b, A)$. Thus, by definition $b \not\prec_{\mathrm{WL}}{}^{A} a$ holds. □

Theorem 6. *The WL relation $\prec_{WL}{}^{A}$ is a refinement of the Pareto dominance relation on $A \subseteq X$, i.e. for $a, b \in A$:*

$$a \prec b \wedge b \not\prec a \Rightarrow a \prec_{WL}{}^{A} b \wedge b \not\prec_{WL}{}^{A} a.$$

Proof. By Proposition 7.3.1 $a \prec b$ implies $b \not\prec_{\mathrm{WL}}{}^{A} a$. Because the $\prec_{\mathrm{WL}}{}^{A}$ relation is total (Theorem 5) $a \prec_{\mathrm{WL}}{}^{A} b$ holds. □

The WL relation is not a refinement of the favor as $a \prec_{\mathrm{WL}} b$ does not implies $a \prec_f b$ or vice versa. We extend the WL relation in the following for sets $A, B, C, D \subseteq X$

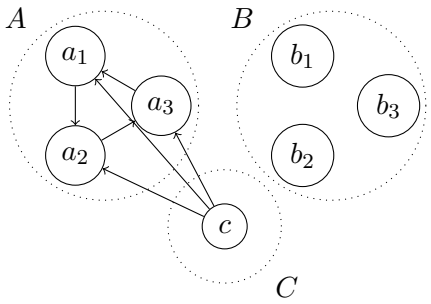
$$A \prec_{\mathrm{WL}}{}^{C,D} B \Leftrightarrow \mathcal{B}^{\prec_f}(A, C) > \mathcal{B}^{\prec_f}(B, D) \vee \left(\mathcal{B}^{\prec_f}(A, C) = \mathcal{B}^{\prec_f}(B, D) \wedge \mathcal{W}^{\prec_f}(A, C) \leq \mathcal{W}^{\prec_f}(B, D)\right)$$

If $C = D$ we simply write $A \prec_{\mathrm{WL}}{}^{C} B$. On this basis we define property $(*)$.

Definition 7.3.1. *A relation R with operator $\prec_R$ on X suffices* ***property $(*)$****, iff for all subsets $B, C \subseteq A \subset X$ with $|B| = |C|$:*

$$B \prec_R^A C \Rightarrow B \prec_R^{A-B, A-C} C.$$

Note, that property $(*)$ enforces that a set of solutions $B \subseteq A$ would get the same rank among equally sized sets when comparisons inside the set are ignored. Thus, it should be impossible for a set to increase its rank among other equally sized sets by generating a high score from comparisons between its own solutions.



$$\mathcal{B}^{\prec_f}(a_i, X) = 1 \quad \mathcal{B}^{\prec_f}(a_i, X - A) = 0 \quad \mathcal{W}^{\prec_f}(a_i, X - A) = 1 \text{ for } i \in \{1, 2, 3\}$$
$$\mathcal{B}^{\prec_f}(b_i, X) = 0 \quad \mathcal{B}^{\prec_f}(b_i, X - B) = 0 \quad \mathcal{W}^{\prec_f}(b_i, X - B) = 0 \text{ for } i \in \{1, 2, 3\}$$
$$\mathcal{B}^{\prec_f}(c, X) = 3 \quad \mathcal{B}^{\prec_f}(c, X - C) = 3 \quad \mathcal{W}^{\prec_f}(c, X - C) = 0$$

Figure 7.1: Example graph G and the values it produces.

The following example shows that property $(*)$ does not hold in general for the WL relation. Consider a directed graph $G = (V, E)$ with $V = \{a_1, a_2, a_3, b_1, b_2, b_3, c\}$. Let $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, and $C = \{c\}$. There are edges $E = \{(a_1, a_2), (a_2, a_3), (a_3, a_1), (c, a_1), (c, a_2), (c, a_3)\}$ (details see Figure 7.1). As shown in Lemma 7.2.1 there exists an objective function $\vec{f}$ such that G is the favor graph of $X = V$. Then $\mathcal{B}^{\prec_f}(b, X) = 0$ for each node $b \in B$, $\mathcal{B}^{\prec_f}(a, X) = 1$ for each node $a \in A$, and $\mathcal{B}^{\prec_f}(c, X) = 3$. Thus, c is the highest ranked node followed by the nodes of set A while the nodes in B are of lowest rank with respect to the WL relation. Therefore, $A \prec_{\text{WL}}{}^{X} B$. But $B \prec_{\text{WL}}{}^{X-B, X-A} A$ since $\mathcal{B}^{\prec_f}(b, X - B) = \mathcal{B}^{\prec_f}(a, X - A) = 0$ and $\mathcal{W}^{\prec_f}(b, X - B) = 0$ but $\mathcal{W}^{\prec_f}(a, X - A) = 1$, for respectively every $b \in B$ and $a \in A$.

7.3.2 The Points Relation

The *Points relation* does fulfill property $(*)$ under certain conditions by accounting for ties between solutions. Such ranking relations are actually used in many tournament rules where draws are common. The Points relation

defines a *point score* of a solution $a \in X$ with respect to a set of solutions $A \subseteq X$. or sets $A, B \subseteq X$

$$S(a, A) = w\mathcal{B}^{\prec_f}(a, A) + \mathcal{E}^{\prec_f}(a, A)$$
$$S(B, A) = \sum_{a \in B} S(a, A),$$

where $w \geq 1$ is the amount of points rewarded for a won comparison. The Points relation for $a, b \in X$ and $A \subseteq X$ and subsets $A, B, C \subseteq X$ is then defined by

$$a \prec_{\mathrm{Pt}}{}^{A} b \iff S(a, A) \geq S(b, A)$$
$$B \prec_{\mathrm{Pt}}{}^{A} C \iff S(B, A) \geq S(C, A),$$

where $b \prec_{\mathrm{Pt}}{}^{A} c \Leftrightarrow \{b\} \prec_{\mathrm{Pt}}{}^{A} \{c\}$. The reference set A is omitted in the notation if its usage is clear from the context.

Theorem 7. *For $A \subseteq X$ the Points relation $\prec_{Pt}{}^{A}$ is a total preorder on X.*

Proof. Transitivity and totality are a consequence from $\geq$ being transitive and total. Reflexivity is a consequence of totality. □

The theorem ensures that it is possible to use the Points relation to sort a set of solutions with a generic sorting algorithm, e.g. merge sort. Note, that the Points relation is not a partial order since antisymmetry does not hold, i.e. $a \prec_{\mathrm{Pt}} b$ and $b \prec_{\mathrm{Pt}} a$ does not imply $a = b$. This is the case because unequal solutions can get the same score. The setting of Example 7.3.1 also gives a counter-example for antisymmetry of the Points relation. Further, using Proposition 7.3.2 it can be shown that the Points relation is a refinement of the Pareto dominance.

Proposition 7.3.2. *Let $A \subseteq X$ and $a, b \in A$ then $a \prec b \Rightarrow b \not\prec_{Pt}{}^{A} a$.*

Proof. Assume $a \prec b$. By Theorem 4 we have $\mathcal{B}^{\prec_f}(a, A) > \mathcal{B}^{\prec_f}(b, A)$. Any $x \in A$ which contributes to $\mathcal{E}^{\prec_f}(b, A)$ contributes one point to $S(b, A)$ and it holds that $b \not\prec_f x \wedge x \not\prec_f b$. Since $x \not\prec_f b$ we have by Proposition 7.2.1 that $x \not\prec_f a$. Hence, $x \prec_f a$ does not hold. Thus, x contributes at least one point to $S(a, A)$ (one point in case of $a \not\prec_f x$ and w points in case of $a \prec_f x$). Altogether, it follows that $S(a, A) > S(b, A)$ and the theorem holds. □

Theorem 8. *The Points relation $\prec_{Pt}{}^{A}$ is a refinement of $\prec$ on A, i.e.*

$$\forall a, b \in A : a \prec b \wedge b \not\prec a \Rightarrow a \prec_{Pt}{}^{A} b \wedge b \not\prec_{Pt}{}^{A} a.$$

Proof. The premise $a \prec b$ implies by Proposition 7.3.2 $b \not\prec_{\mathrm{Pt}}{}^{A} a$. Since the Points relation is total (Theorem 7) $a \prec_{\mathrm{Pt}}{}^{A} b$ follows. □

Mind, that with $w > 1$ there are scenarios where the Points relation gives a higher rank to a solution $a \in A$ that is dominated by some solution $c \in A$ than it gives to a solution $b \in A$ that is in the non-dominated set of A. As an example consider a set $A = \{a, b, c\} \cup D$, with $|D| \geq 2$, $c \prec a$ and $\forall d \in D : c \prec d \wedge a \prec d$ (and all other pairs are incomparable). Hence, $S(a, X) = 2 + w(|D|)$, $S(b, X) = |X| = 3 + |D|$, $S(c, X) = 2 + w(|D| + 1)$. Thus, a is ranked better than b if $w > 1$.

In tournaments $w = 2$ is often used when draws are rare. However, when draws are a more common event a higher w-value is used, e.g. $w = 3$, to encourage the participants even more in securing victories.

Theorem 9. *With $w = 2$, property $(*)$ holds for the Points relation, i.e.*

$$\forall A, B, C \subset X \text{ with } |B| = |C| : B \prec_{Pt}{}^{A} C \longrightarrow B \prec_{Pt}{}^{A-B, A-C} C.$$

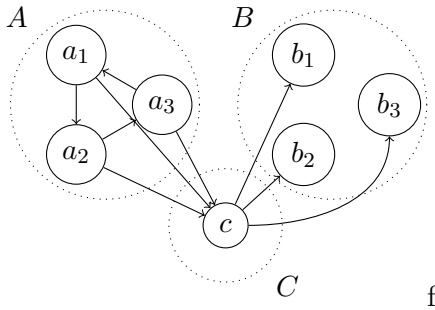
Proof. Note, that for the Points relation with $w = 2$ every comparison rules exactly two points and for any subset $A \subset X$ we have $S(A, A) = |A|^2$. With $|B| = |C| = p$ and $S(B, A) \geq S(C, A)$ we have

$$\begin{aligned} S(B, A - B) = S(B, A) - S(B, B) &= S(B, A) - p^2 \\ &\geq S(C, A) - p^2 = S(C, A) - S(C, C) = S(C, A - C). \end{aligned}$$

□

For $w \neq 2$ property $(*)$ does not hold for the Points relation. Consider the following two examples as proof.

Example 7.3.2. *With $w = 4$ the scenario of Figure 7.1 shows that property $(*)$ does not hold. We have $G = (V, E)$ with $V = \{a_1, a_2, a_3, b_1, b_2, b_3, c\}$ and $E = \{(a_1, a_2), (a_2, a_3), (a_3, a_1), c, a_1), (c, a_2), (c, a_3)\}$. Let $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, and $C = \{c\}$. As shown in Lemma 7.2.1 there exists an objective function $\vec{f}$ such that G is the favor graph of $X = V$. Then $S(b, X) = 7$ for each node $b \in B$, $S(a, X) = 8$ for each node $a \in A$, and $S(c, X) = 16$. Thus, regarding the Points relation, c is the highest ranked node followed by the nodes of set A while the nodes in B are of lowest rank. Therefore, $A \prec_{Pt}{}^{X} B$. However, $B \prec_{WL}{}^{X-B, X-A} A$ since $S(b, X - B) = 4$ and $S(a, X - A) = 3$, for respectively every $b \in B$ and $a \in A$.*



for $a \in A$ and $b \in B$:

$$\mathcal{B}^{\prec_f}(a, X) = 2 \quad \mathcal{E}^{\prec_f}(a, X) = 3 \quad \mathcal{B}^{\prec_f}(a, X-A) = 1 \quad \mathcal{E}^{\prec_f}(a, X-A) = 3$$
$$\mathcal{B}^{\prec_f}(b, X) = 0 \quad \mathcal{E}^{\prec_f}(b, X) = 6 \quad \mathcal{B}^{\prec_f}(b, X-B) = 0 \quad \mathcal{E}^{\prec_f}(b, X-B) = 3$$

Figure 7.2: Example graph G and the values it produces.

Example 7.3.3. *With $w = 1$ the scenario of Figure 7.2 shows that property $(*)$ does not hold. We have $G' = (V, E)$ with $V = \{a_1, a_2, a_3, b_1, b_2, b_3, c\}$ and $E = \{(a_1, a_2), (a_2, a_3), (a_3, a_1), (c, b_1), (c, b_2), (c, b_3), (a_1, c), (a_2, c), (a_3, c)\}$. Let $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, b_3\}$, and $C = \{c\}$. Due to Lemma 7.2.1 there exists an objective function $\vec{f}$ such that G' is the favor graph of $X = V$. Then $S(b, X) = 6$ for each node $b \in B$, $S(a, X) = 5$ for each node $a \in A$, and $S(c, X) = 4$. Thus, regarding the Points relation, the nodes of B are the highest ranked nodes followed by the nodes of set A while c has the lowest rank. Therefore, $B \prec_{Pt}{}^{X} A$. However, $A \prec_{WL}{}^{X-A,X-B} B$ since $S(b, X - B) = 3$ and $S(a, X - A) = 4$, for respectively every $b \in B$ and $a \in A$.*

Table 7.2 gives an overview over the discussed ranking relations.

Table 7.2: Summary of the characteristics of the relations.

Relation	Refinement of $\prec$	Characteristics
Favour	yes (Theorem 3)	irreflexive, asymmetric, not transitive, not total (Drechsler et al., 2001)
WL	yes (Theorem 6)	total preorder (Theorem 5)
Points	yes (Theorem 8)	total preorder (Theorem 7)
W	yes (Section 7.2.3)	total preorder (Section 7.2.3)
Win	yes (Section 7.2.2)	total preorder (Section 7.2.2)

8 Problems and Metaheuristics

Objectives and algorithms

Before we embark in the empiric analysis of the different ranking relations let me introduce two multi-objective problems which will be used later. The Flow-Shop Scheduling problem (FSP) and the Traveling Salesperson problem (TSP) are both well known NP-hard problems. This chapter describes these two problems in detail. This includes an analysis of their objectives and correlations and a method for the generation of multi-objective TSP instances. Finally, the applied metaheuristics are introduced.

8.1 Flow Shop Scheduling Problem

The Flow Shop Scheduling problem denotes the problem of scheduling a set of n jobs that have to be processed by m machines. All machines are in a fixed order and pass a job they successfully processed to their respective predecessor. Every job has to be processed by all machines, starting with the first machine, then from the i-th machine to the $i+1$-st machine, $i \in \{1, \dots m\}$, until it is finished by the m-th machine. A machine can only process one job at a time and a job can only be processed by one machine at a time. We assume that all jobs are available at time 0.A matrix $P \in \mathbb{N}^{m \times n}$ defines the processing times of the jobs, such that p_{ij} is the time machine j required to process job i successfully. A test instance, such as the 15 instances applied in this study (see Table 8.1) provides such a matrix P. The Flow Shop Scheduling problem is NP-hard (Garey et al., 1976). The considered multi-objective version of the Flow Shop Scheduling Problem has four objectives that are to be minimized.

1. The *Total Time*, T_{total}, is defined by the completion time of the last job on the last machine.
2. The *Flow Time*, T_{flow}, is defined by the sum of the processing and idle

Table 8.1: Used data sets and their properties.

Instance	from	n	m	Pareto set available
car1	Carlier (1978)	11	5	✓
hel1	Heller (1960)	100	10	×
reC37,39,41	Reeves (1995)	75	20	×
ta101-ta105	Taillard (1993)	200	20	×
ta111-ta115		500	20	×

Table 8.2: Pearson correlation (all p-values < 0.01) for the Pareto set and a random set of solutions (502 solutions each) for the *car1* instance. Total Time (T_{total}), Flow Time (T_{flow}), Total Idle Time Jobs ($T_{i.j.}$), and Total Idle Time Machines ($T_{i.m.}$)

	T_{flow}		$T_{i.m.}$		$T_{i.j.}$	
	Pareto	random	Pareto	random	Pareto	random
T_{total}	0.512	0.675	-0.404	0.382	0.179	0.289
T_{flow}			-0.813	-0.128	0.676	0.771
$T_{i.m.}$					-0.846	-0.325

times of the jobs.

3. The *Total Idle Time of the Jobs*, $T_{i.j.}$, is defined by the sum of all waiting times of all jobs after the job was finished on a machine and before it is processed on the next machine.
4. The *Total Idle Time of the Machines*, $T_{i.m.}$, is defined by the sum of all idle times of all machines where a machine has an idle time when it is unoccupied between the processing of two jobs.

These four objectives are correlated (see Table 8.2). When we compare the correlations of the objective of the solutions in the Pareto set (for problem instance *car1*) the correlations are all closer to -1 than the respective correlations in sets of random solutions. This effect changes the sign of the respective correlation of objectives T_{total} and $T_{i.m.}$ from a positive correlation in the random set to a negative correlation in the Pareto set. In the Pareto set the strongest absolute correlation is the (negative) correlation between the total idle times of the machines $T_{i.m.}$ and of the jobs $T_{i.j.}$. The second strongest absolute correlation is the also negative correlation between the flow time T_{flow} and $T_{i.m.}$. The two strongest positive correlations are between T_{flow} and $T_{i.m.}$ and between T_{total} and T_{flow}. In the random set the

strongest correlations are the two positive correlations between T_{flow} and $T_{i.j.}$ and between the objectives T_{total} and T_{flow}.

The correlations between the objectives is a typical phenomenon in many application problems. They are the results of the problems special characteristics and show that solutions of the Pareto set have different characteristics than random solutions. The mixture of strong and weak as well as positive and negative correlations highlights the difficulty of the provided multi objective optimization problem.

8.2 Traveling Salesperson Problem

The Traveling Salesperson problem is characterized by the search for a round trip among a set of n cities defined by a permutation of $\{1, \ldots, n\}$. A matrix $C \in \mathbb{R}^{n \times n}$ provides the distances between the cities. Here, we only consider symmetric problems, i.e. $c_{ij} = c_{ji}$. The objective value of a permutation π where $\pi(i)$ is the position of the i-th city in π is then defined by

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\phi(i,j), \text{ with } \phi(i,j) \quad = \begin{cases} 1 & \text{if } \pi(j) - 1 = \pi(i) \mod n, \\ & \text{i.e. } j \text{ follows on } i \text{ in } \pi \\ 0 & \text{otherwise.} \end{cases}$$

Note, that the first element in the permutation is the successor of the last element in the permutation. The TSP is NP-hard (Michael and David, 1979). A multi-objective TSP instance with k objectives has a set of these matrices $\{C^1, \ldots, C^k\}$, where c^l_{ij} is the cost of objective l when city j follows directly after city i. These objectives can be time, fuel, distance, or safety, for example.

The benchmark instances are typically single-objective problems. Thus, the generation of multi-objective instances is a prerequisite to analyze the performance of multi-objective metaheuristics on the TSP. The central task in the instance generation for a multi-objective TSP is the generation of multiple matrices that should consider correlations between the objectives.

A good starting point in instance generation can be a given matrix, e.g. from a real world instance. Knowles and Corne (2003, 2002) applied such an approach for the multi-objective Quadratic Assignment problem (QAP) which is based on k flow matrices. The first matrix is the given matrix. The j-th matrix, $j \in \{2, \ldots, k\}$ is filled with random values that correlate with the respective values of the first matrix. This correlation is regulated with some parameter $\alpha_j \in [-1, 1]$. Furthermore, the ratio of random and

correlated entries in the generated matrix can be set. This QAP instance generator has been applied to study the influence of correlation mostly in bi-objective problems on the optimization behavior of metaheuristics and local search operators (Garrett et al., 2007; Liefooghe et al., 2011; López-Ibánez et al., 2004, 2006; Paquete and Stützle, 2006). Note, that each of the objectives 2 to k has a defined correlation with objective 1. But the correlation between two of the objectives from 2 to k might be different.

Alternatively the generation of an instance can be based on a random instance. Corne and Knowles (2007) proposed a method for the multi-objective TSP where for each pair of cities (i, j) k distance values $d_h(i, j)$, $h \in \{2, \ldots, k\}$ were created with

$$d_h(i, j) = c \cdot d_{h-1}(i, j) + (1 - c)rand \tag{8.1}$$

where the values $d_1(i, j)$ are chosen uniformly at random from $[0, 1]$, $c \in [-1, 1]$ regulates the correlation, and *rand* is a uniform random number from $[0, 1]$. Observe, that Equation (8.1) can be somewhat problematic: (i) with $c < 0$ the distance values can become > 1 even for original distance values from $[0, 1]$. This might lead to an uneven influence of different objectives, (ii) the distance values are randomized to a different extent for c and $-c$. (iii) Objective $h \in \{2, \ldots, k\}$ has a defined correlation with objective $h - 1$. The correlations between pairs of objectives with $|i - j| \geq 2$ is less well defined.

A third approach, as proposed by Verel et al. (2011a), is the use of a correlation matrix to define the correlation between the objectives. In particular, NK-landscapes have been investigated, where the same correlation strength was set to each pair of objectives. The method could be applied to other problems as well.

For their study of evolutionary multi-objective algorithms Ishibuchi et al. (2013a, 2011, 2013b) proposed a method to generate instances for the multi-objective n-item 0/1 Knapsack problem. The n-item 0/1 Knapsack problem is the task to allocate n items with k knapsacks with capacities $c_1, \ldots, c_k$. For each item knapsack pair (i, j) a weight w_{ij} and profit p_{ij} with $i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\}$, is defined. A bi-objective problem with the objectives functions is extended with objectives $g_i(x)$ of a solution x such that

$$\begin{aligned} g_1 &= f_1, & g_2 &= f_2, \\ g_{2h+1} &= \alpha f_{2h+1} + (1 - \alpha) f_1, & g_{2h+2} &= \alpha f_{2h+2} + (1 - \alpha) f_2, \end{aligned}$$

Table 8.3: Used data sets and their properties.

Instance	from	n	Optimal tour available
att48		48	✓
eil51	Padberg and Rinaldi (1991)	51	✓
berlin52		52	✓

for $h \in \{1, \ldots, k/2 - 1\}$ and $\alpha \in [0, 1]$. or

$$
\begin{aligned}
g_1 &= f_1, & g_2 &= f_2,\\
g_3 &= f_1 + \alpha f_2, & g_4 &= f_2 + \alpha f_1,\\
g_5 &= f_1 - \alpha f_2, & g_6 &= f_2 - \alpha f_1,\\
g_7 &= f_1 + \beta f_2, & g_8 &= f_2 + \beta f_1,\\
g_9 &= f_1 - \beta f_2, & g_{10} &= f_2 - \beta f_1
\end{aligned}
$$

for $\alpha, \beta \in [0, 1]$. This approach induces pairwise different correlations between the objectives.

Of all above approaches, the only one inducing a homogeneous correlation structure, i.e. where all pairwise correlations between different objectives are equal, are the NK-landscape instances from Verel et al. (2011a). Yet, to the best of my knowledge, there are no such approaches for application oriented multi-objective problems, e.g. the TSP. However, considering the analysis of the influence of correlated objectives on the performance of metaheuristics, homogeneous correlations appear to be the most basic case. In empiric studies the most basic case is of great importance as comparison for other, more complicated, scenarios. In the following, I introduce a simple, yet effective, method to induce homogeneous correlations on a multi-objective TSP.

Similar to the approach from Knowles and Corne (2003, 2002), it is based on a single initial matrix. These are taken from a standard benchmark library, specifically the TSPLIB (Reinelt, 1991). Table 8.3 gives an overview over the the used instances. Note, that once we transform an instance into a multi-objective problem we do not have a Pareto set for any of the problems.

Initially, all distance values of the given matrix are normalized with the maximal distance such that they are in $[0, 1]$ and are denoted by d. Then, k distance functions $d_1, \ldots, d_k$ are defined with a correlation factor $c \in [-1, 1]$

for $h \in \{1, \ldots, k\}$ with

$$d_h(i,j) = \begin{cases} c\, d(i,j) + (1 - \ c\ )rand & \text{for} \quad 0 \leq c \leq 1 \\ |c|(1 - d(i,j)) + (1 - |c|)rand & \text{for} \ -1 \leq c < 0 \end{cases}$$

Observe, that this addresses the concerns we formulated for the method of Equation (8.1), i.e. (i) the distance values remain in $[0,1]$ for all objectives, (ii) the random influence for c is equal to that of $-c$, (iii) and the correlations between all pairs of objectives are clearly defined.

Expected Correlation The parameter c induces a correlation but is different from the actual Pearson correlation. In the following we derive a formula for the expected Pearson correlation between the objectives. Let the initial distances be given by a matrix $M = [m_{ij}] \in [0,1]^{n \times n}$. The random values are provided by a random matrix $R = [rand_{ij}] \in [0,1]^{n \times n}$ where $rand_{ij}$ is chosen uniformly at random from $[0,1]$, for $i \neq j$ and $rand_{ii} = 0$. For a symmetric TSP additionally $d_{ij} = d_{ji}$ and $rand_{ij} = rand_{ji}$. As $R \in [0,1]^{n \times n}$ is uniformly distributed we have the expectation $E(R) = \frac{1}{2}$ and variance $Var(R) = \frac{1}{12}$. The distribution of $M \in [0,1]^{n \times n}$ is unknown with the expected value $E(M)$ and variance $Var(M)$. We define

$$\begin{aligned} M^+ &= cM + (1-c)R \in [0,1]^{n \times n} && \text{for } c \in [0,1] \\ M^- &= |c|(1-M) + (1-|c|)R \in [0,1]^{n \times n} && \text{for } c \in [-1,0] \end{aligned}$$

where M^+ is positively and M^- is negatively correlated with M. To derive the expected correlation between two matrices the Pearson correlation coefficient r is applied. Note, that for the symmetric TSP only the relevant entries, e.g. over indices $i, j \in \{1, \ldots, n\}$ with $i < j$. It can be shown that for $M' \in \{M^+, M^-\}$

$$\begin{aligned} r(M,M') &= \frac{\sum (m_{ij} - \overline{m})(m'_{ij} - \overline{m'})}{\sqrt{\sum (m_{ij} - \overline{m})^2 \sum (m'_{ij} - \overline{m'})^2}} \ = \frac{E(MM') - E(M)E(M')}{\sqrt{Var(M)Var(M')}} \\ r(M,M') &= \frac{cVar(M)}{\sqrt{c^2 Var(M)^2 + \frac{(1-|c|)^2 Var(M)}{12}}} \end{aligned} \tag{8.2}$$

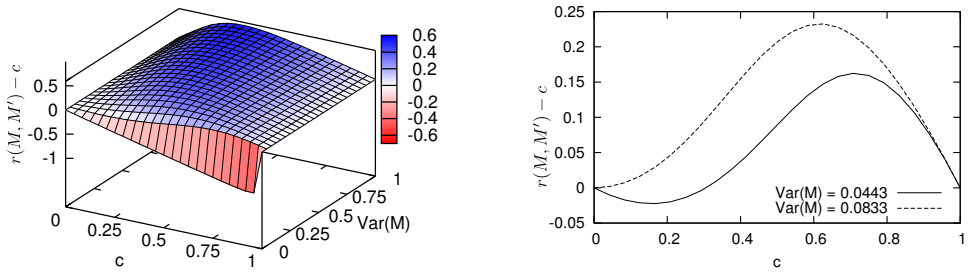


Figure 8.1: Pearson correlation coefficient $r(M, M') - c$ (left); for two specific variances of M (right)

The last equation can be obtained after deriving the following equations

$$\begin{aligned} E(M') &= E(cM + (1-c)R) = cE(M) + (1-c)E(R) \qquad (8.3)\\ &= cE(M) + \frac{(1-c)}{2} \\ Var(M') &= Var(cM + (1-c)R) \qquad (8.4)\\ &= c^2 Var(M) + (1-c)^2 Var(R) + cov(M, R) \\ &= c^2 Var(M) + \frac{(1-c)^2}{12} \\ E(MM') &= E(M(cM + (1-c)R)) = E(cM^2 + (1-c)MR) \\ &= cE(M^2) + (1-c)E(MR) \\ &= c(Var(M) + E(M)^2) + (1-c)(E(M)E(R) + cov(M, R)) \\ &= c(Var(M) + E(M)^2) + \frac{(1-c)E(M)}{2} \end{aligned}$$

Note, $cov(M, R) = 0$ because M and R are independent random variables.

Figure 8.1 visualizes the difference of the expected Pearson correlation coefficient and the correlation parameter c, i.e. $r(M, M') - c$, for different values of $Var(M)$ and $c \in [0, 1]$. Note, that if M is uniformly distributed in $[0, 1]$, we have $E(M) = 1/2$ and $Var(M) = 1/12$. This gives

$$r(M, M') = \frac{c}{\sqrt{c^2 + (1-c)^2}}$$

The variance measured in the test instances was $Var(M) \approx 0.0043$. In order to obtain a specific Pearson correlation coefficient $r(M, M') = r$ the correlation factor c has to be chosen as

$$c = \frac{1}{1 + 2\sqrt{3}\sqrt{(\frac{1}{r} - 1)Var(M)}}.$$

The above analysis referred to the correlation of the initial matrix M and one of the generated matrices M'. However, the final problem instance does not contain M as an objective matrix. In the following we infer the correlation between two matrices M_1 and M_2, that are (i) both derived with the same correlation parameter c, or (ii) derived with c and $-c$ respectively from the initial matrix M. We have

$$\text{(i)} \qquad r(M_1, M_2) = \frac{c^2 Var(M)}{c^2 Var(M) + \frac{(1-|c|)^2}{12}}$$

$$\text{(ii)} \qquad r(M_1, M_2) = -\frac{c^2 Var(M)}{c^2 Var(M) + \frac{(1-|c|)^2}{12}}.$$

To infer these correlations we use Equation (8.3) and 8.4. For $E(M_1, M_2)$ the following can be determined when R_1 and R_2 are the random values used in the generation.

$$\begin{aligned}
E(M_1M_2) &= E((cM + (1-c)R_1)(cM + (1-c)R_2)) \\
&= E(c^2M^2 + c(1-c)M(R_1 + R_2) + (1-c)^2 R_1R_2) \\
&= c^2E(M^2) + c(1-c)(E(MR_1) + E(MR_2)) + (1-c)^2E(R_1R_2) \\
&= c^2(Var(M) + E(M)^2) + c(1-c)(E(M)E(R_1) + cov(M, R_1) \\
&+ E(M)E(R_2) + cov(M, R_2)) + (1-c)^2(E(R_1)E(R_2) + cov(R_1, R_2) \\
&= c^2\left(Var(M) + E(M)^2\right) + c(1-c)E(M) + \frac{(1-c)^2}{4}
\end{aligned}$$

The expected correlation between M_1 and M_2 is then

$$\begin{aligned}
r(M_1, M_2) &= \frac{E(M_1M_2) - E(M_1)E(M_2)}{\sqrt{Var(M_1)Var(M_2)}} = \frac{E(M_1M_2) - E(M_1)^2}{\sqrt{Var(M_1)^2}} \\
&= \frac{E(M_1M_2) - E(M_1)^2}{Var(M_1)} \\
&= \frac{c^2(Var(M) + E(M)^2) + c(1-c)E(M) + \frac{(1-c)^2}{4} - \left(cE(M) + \frac{(1-c)}{2}\right)^2}{c^2 Var(M) + \frac{(1-c)^2}{12}} \\
&= \frac{c^2 Var(M)}{c^2 Var(M) + \frac{(1-c)^2}{12}}
\end{aligned}$$

Figure 8.2 shows $r(M_1, M_2) - c$ for the first case and two choice values of $Var(M)$ and c. In order to obtain a specific Pearson correlation coefficient

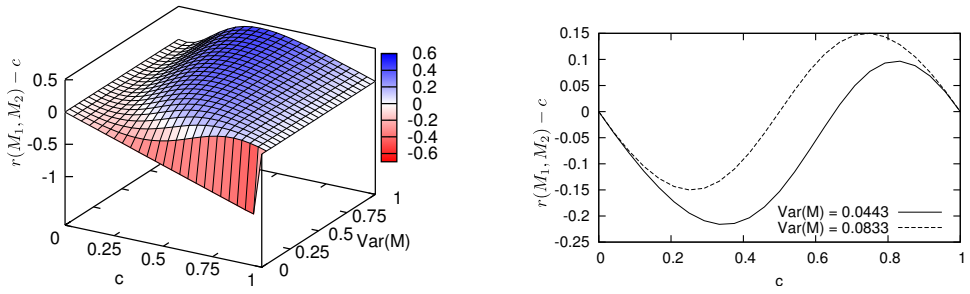


Figure 8.2: Expected Pearson correlation coefficient $r(M_1, M_2) - c$ (right) for two specific variances of M (right)

$r = r(M_1, M_2)$ the correlation parameter we set c such that

$$c = \pm \left(1 + \sqrt{12 Var(M) \frac{1 - r(M_1, M_2)}{r(M_1, M_2)}} \right)^{-1}.$$

8.3 Population-Based Ant Colony Optimization

Solving NP-hard problems optimally is usually not an option due to time constraints. Instead metaheuristics can be applied. One of these metaheuristics that specifically searches for solutions of combinatorial problems is the Ant Colony Optimization (ACO) algorithm (Dorigo et al., 1999). It is based on a pheromone matrix $\tau = [\tau_{ij}]$, $i, j \in \{1, \ldots, n\}$ where τ_{ij} is the pheromone value for placing item j as follower of item i in the permutation. The algorithm iterates over the matrix and constructs a set L of l solutions, i.e. permutations, by performing roulette wheel decisions in each row of the matrix. Good solutions are allowed to imprint themselves on the matrix by increasing the pheromone values in the matrix cells that have been chosen during their construction. After each iteration pheromone evaporates. The interested reader is referred to an overview on the ACO algorithms by Dorigo and Stützle (2004).

The Population-based ACO (P-ACO) (Guntsch and Middendorf, 2002) stores a set of p solutions in its population P that are solely responsible for the values of τ. In the version of the P-ACO applied here (see Algorithm 7) the solutions can remain in the population over several iterations and are only replaced if a new and better solutions has been found, i.e. the population is the set of the best solutions found to date. The original P-ACO algorithm chose the best solution from the set of new solutions L and added

Algorithm 7: Multi objective P-ACO with ranking

```
P ← {};
Initialize pheromone matrix [τ_ij] ← τ_init
repeat
    L ← {}
    foreach of the l ants do
        Construct a solution S using pheromone matrix [τ_ij]
        L ← L ∪ {S}
    Sort the solutions in P ∪ L with a ranking method
    Let P be the set of p best solutions from L ∪ P
    Compute a pheromone matrix [τ_ij] from P
until t iterations done;
return non-dominated solutions in P ∪ L
```

it to P while removing the oldest solution from P in every iteration. But other strategies for the removal of solutions from P have also been proposed (Guntsch and Middendorf, 2002), e.g. to remove the worst solution. Mind, that this version of the P-ACO only requires a multi-objective ranking relation to be applied to multi-objective optimization problems, no additional matrices, populations,or type of ants are required. The population is employed to compute the pheromone information, where each pheromone value τ_{ij} is defined by

$$\tau_{ij} = \tau_{init} + \frac{\tau_{max} - \tau_{init}}{p} \sum_{r=1}^{p} \phi(r, i, j), \text{ with}$$

$$\phi(r, i, j) = \begin{cases} 1 & \text{if } j \text{ follows on } i \text{ in solution } r \\ 0 & \text{otherwise.} \end{cases}$$

The values τ_{init} and τ_{max} are parameters of the algorithm giving the minimal and maximal pheromone values. The solutions are constructed as they are in the traditional ACO algorithm. Iteratively, item j is selected after item i in a permutation with probability $p_{ij} = \tau_{ij} / \sum_{a \in \mathcal{S}} \tau_{ia}$, where $\mathcal{S}$ is the set of items that are still selectable, i.e. the set of items that have not already been placed in the permutation. Using the pheromone information like this is particularly suitable for scheduling problems (Merkle and Middendorf, 2014).

In this study six variants of the P-ACO are compared, where each variant uses a different ranking method as listed in Table 8.4.

Table 8.4: The six variants of the P-ACO that are applied and compared in this work.

Algorithm	Ranking Scheme	Definition
Pt-P-ACO	Points relation	Section 7.3.2
WL-P-ACO	WL relation	Section 7.3.1
Win-P-ACO	Wining Score	Section 7.2.2
W-P-ACO	W relation	Section 7.2.3
Std-P-ACO	Standard Scheme	Section 8.3
Cr-P-ACO	Crowding Scheme	Section 8.3

Std-P-ACO The Standard P-ACO is motivated by Guntsch and Middendorf (2003). It computes the non dominated set S from the solutions in $P \cup L$ in each iteration. The new population for the next iteration is determined by selecting a solution $s \in S$ at random and then choosing the $p-1$ solutions closest to s in the objective space as the new population. Note, that this version of the P-ACO is solely based on the Pareto dominance relation.

Cr-P-ACO The Crowding P-ACO was introduced by Angus (2007). The pheromone update of a solution in the Crowding P-ACO is inversely proportional to the dominance depth of the corresponding solution (Deb et al., 2000), unlike the pheromone update in all other compared P-ACOs of this study. The selection of the solutions for the population is based on a crowding scheme. In each iteration for each newly generated solution $a \in L$ a random subset $P' \subset P$ is chosen. Let $b \in P'$ be the solution of P' with the highest similarity to a, where similarity is measured as the number of common adjacent jobs, i.e. in the solution space. If a dominates b, i.e. $a \prec b$, b is replaced by a. Otherwise, a is discarded. Note, that this is different from the crowding measure used by Deb et al. (2000), where similarity was defined with respect to the objective space.

The Crowding P-ACO applies a slightly different pheromone update. Each solution $a \in P$ increases the pheromone value of its respective cells in the pheromone matrix by $1/(r(a)+1)$, where $r(a)$ is the inverse rank of a, i.e. the best solution $a' \in P$ has $r(a') = 0$ and the worst solution $a'' \in P$ has $r(a'') = p$.

Algorithm 8: Multi objective GA with ranking

```
1  P ← {p random solutions}
2  repeat
3      L ← {};
4      foreach solution a ∈ P do
5          foreach solution b ∈ P, b ≠ a do
6              Construct a solution b by performing a crossover with a
               and b
7              L ← L ∪ {c}
8          construct a solution c by performing a mutation on a
9          L ← L ∪ {c}
10     Sort the solutions in P ∪ L with a ranking method
11     Let P be the set of p best solutions from L ∪ P
12 until t iterations done;
13 return non dominated solutions in P ∪ L
```

8.4 Genetic Algorithm

Genetic algorithms (GA) are based on the notion of crossover and point mutation processes in DNA . They also keep a population P of p solutions that they use to produce so-called 'offspring', i.e. a set L of l new solutions (for an introduction see Mitchell (1996)). The version applied in this study (Algorithm 8) defines the two types of mutations as described in the following.

A crossover mutation between two solutions $a, b \in P$ generates a new solution c by picking a random interval with start and end point $t_s, t_e \in \{1, \ldots, n\}, t_s \leq t_e$, where $c_i = a_i$ for $i \in \{t_s, \ldots, t_e\}$ and c_i respectively a_i denote the item in ith position of the respective solution. Positions $c_i, i \in \{0, \ldots, t_s - 1, t_e + 1, \ldots, n\}$ are filled with the remaining items in the order they appear in solution b. All pairs of solutions in P are used to generate $p(p-1)$ new solutions by crossover.

A point mutation denotes a random exchange of two neighbored items in the permutation. For each pair of neighbors a random number is drawn and a mutation occurs if that number is below the *mutation probability* μ. Each solution from P is used to create a new solution by mutation, such that there are p new solutions generated with this method.

Overall, in each iteration $l = p(p-1) + p = p^2$ new solutions are gener-

Table 8.5: The four variants of the GA that are applied and compared in this work.

Algorithm	Ranking Scheme	Definition
Pt-GA	Points relation	Section 7.3.2
WL-GA	WL relation	Section 7.3.1
Win-GA	Wining Score	Section 7.2.2
W-GA	W relation	Section 7.2.3

ated. These new solutions are then ranked with the solutions of the population to infer the population for the next iteration. In this process a solution $a \in P$ can only be replaced by a solution $b \in L$, if b is better than a regarding the applied ranking method. Equality in quality does not suffice.

Four variants of the GA are compared in this study (see Table 8.5).

9 Experimental Results

Solution set quality and convergence

This chapter includes a description of the performed experiments and analysis of their results. The first section gives a comparison of the ranking methods when applied on the FSP. In Section 9.2 the influence of different correlations on the behavior of the ranking relations is analyzed applying the TSP instance generation method introduced in Section 8.2.

9.1 Influence of the Ranking Relations

In this section the six P-ACOs and four GAs are compared in their performance regarding the different FSP instances presented in Section 8.1. The specific parameters of the algorithms are summarized in Table 9.1. Two types of experiments were performed to measure different aspect of the algorithms. The first experiments consisted of 25 runs with 50 000 iterations each for all algorithms. The results were used to analyze the properties of the final non-dominated sets of solutions of the different algorithms. The second type of experiments were run 10 times for 100 000 iterations each and only on the four instances *car1*, *hel1*, *reC37*, and *ta111*. The results were used to analyze the convergence of the different P-ACO algorithms regarding the different ranking schemes.

The population size of 5 is small as such small populations have been shown to perform well (Guntsch and Middendorf, 2002, 2003; Oliveira et al., 2011). Only the crowding scheme of the Cr-P-ACO and Cr-GA require a larger population. The size of the chosen subset that is used in the crowding scheme is 5 to increase comparability (i.e. each new solution is compared with 5 solutions of the population in order to decide whether the solution is discarded or submitted to the new population). All algorithms generate 25 new solutions in each iteration. The parameter of the pheromone update of

Table 9.1: Parameters used in experiments. (*) For the Cr-P-ACO and Cr-GA $p = l = 25$, otherwise $p = 5$.

Symbol	Definition	Values for experiments to infer Performance	and Convergence
t	iterations	50 000	100 000
	runs	25	10
p	population size	5 (25*)	5 (25*)
l	number of ants	25	25
τ_{init}	minimal pheromone	$1/n$	$1/n$
τ_{max}	maximal pheromone	$\tau_{init} + 24$	$\tau_{init} + 24$
μ	mutation probability	$1/(n-1)$	$1/(n-1)$
	solution evaluations	1 250 000	2 500 000
	instances	all	*car1*, *hel1*,*reC37*, *ta111*

the P-ACO were set to $\tau_{init} = 1/n$ and $\tau_{max} = \tau_{init} + 24$ as suggested by the results of Guntsch and Middendorf (2003) but without any influence of the instance dependent parameter n on the decision probabilities. For the GA the mutation operator had a mutation probability of $1/(n-1)$ for each position.

In the experimental analysis we compare the quality of the final $p + l$ solutions of all ten algorithms and adjoin an analysis of the convergence behavior of the P-ACO algorithms. To evaluate the quality of the solutions competitively we analyze how many solutions of one algorithm dominates the solutions of another algorithm. Additionally we infer which algorithm scored best regarding each of the four objectives separately. As the Pareto set is only available for one instance (namely *car1*) we are unable to check how many Pareto optimal solutions were found by the respective algorithms and how close they generally were.

9.1.1 Analysis of the Non-Dominated Sets

The first type of experiments consisted of 25 runs with 50 000 iterations. In the final iteration of each of these runs there are p solutions in the population of the algorithm and additional l solutions that were generated anew with the exception of the Crowding algorithms that have l solutions in the population.

Table 9.2 presents the average number of non-dominated solutions in these final sets of solutions. The GAs have generally smaller final non-dominated sets than the P-ACOs especially on the larger *ta* instances. GAs

Table 9.2: Average size of the non-dominated sets.

	P-ACO						GA			
	{ Pt	WL	W	Win	Cr	Std }	{ Pt	WL	W	Win }
car1	4.8	4.7	4.2	4.4	9.4	8.2	4.6	4.3	4.4	4.1
car8	2.8	3.0	1.5	2.3	6.0	6.5	3.4	3.6	1.6	2.2
hel1	4.9	4.6	5.0	4.7	5.9	5.6	1.8	1.9	1.8	2.0
reC37	4.8	4.6	5.0	4.9	5.6	6.2	2.4	2.3	2.8	2.2
reC39	4.9	4.9	5.5	4.9	6.3	7.4	2.0	2.4	2.1	2.4
reC41	4.4	4.4	4.5	4.7	7.0	6.2	2.3	2.0	1.7	2.0
ta101	5.1	5.0	5.4	4.9	6.7	6.2	2.0	2.0	2.1	2.0
ta102	4.7	4.6	5.8	4.9	6.1	6.2	1.8	2.0	2.1	2.0
ta103	5.0	4.5	5.1	5.0	6.7	6.0	1.9	1.8	2.0	1.9
ta104	4.8	4.6	5.0	4.9	6.1	5.3	1.8	2.0	1.8	1.7
ta105	5.0	4.7	5.5	5.0	6.0	7.0	1.8	2.5	2.4	1.8
ta111	4.8	4.5	6.0	4.8	6.5	6.2	1.7	1.8	2.0	2.2
ta112	5.0	4.9	6.2	4.9	7.5	6.4	1.8	1.8	2.3	1.9
ta113	4.8	4.5	4.8	4.8	5.8	5.6	1.6	2.2	1.7	1.8
ta114	4.8	4.6	4.6	5.1	5.6	5.7	1.7	1.7	2.2	1.7
ta115	4.9	4.6	6.0	5.1	5.4	6.7	1.7	1.9	1.8	1.7
∅	4.7	4.5	5.0	4.7	6.4	6.3	2.1	2.3	2.2	2.1

Table 9.3: Fractions of solutions in the combined non-dominated set: with Pt (upper part) or with WL (lower part); absolute average size of the set with standard deviation; values are averages over all problem instances.

	Parts in combined set of					Average absolute size
	{ Pt	W	Win	Cr	Std }	± standard deviation
P-ACO	0.396	0.261	0.335	0.013	0.050	72.1 ± 17.0
GA	0.353	0.342	0.411			31.3 ± 14.4
	{ WL	W	Win	Cr	Std }	
P-ACO	0.376	0.269	0.342	0.013	0.048	65.6 ± 15.9
GA	0.362	0.360	0.415			31.4 ± 15.5

require heterogeneous populations to construct new solutions by crossover. Performing a crossover on two similar solutions yields yet one more similar solution. The small populations of only five solutions, however, seem to miss this heterogeneity. Cr- and Std-P-ACO have larger non-dominated sets. Note, that if less than p solutions are in the non-dominated set most – if not all – solutions generated in the final iteration are dominated by the solutions of the population. This is often the case for the Pt-P-ACO, WL-P-ACO, and Win-P-ACO algorithms indicating a high quality of the solutions in the population.

In the following we separately analyze the results of the Points and WL relation. To evaluate the quality of the final solution sets of the different algorithms all final sets of all respective algorithms are merged. The non-dominated solutions from this merged set is denoted by the *combined non-dominated set*. A good set of solutions should dominate a lot of the solutions of other sets and provide a big part of the combined non-dominated set. In Table 9.3 the fractions of the non-dominated sets are given with the average absolute size of the combined non-dominated sets. Note, that some solutions in the combined non-dominated set were detected by several of the algorithms, i.e. the sum of the fractions within each class can be > 1.

In case of the P-ACO the Points and WL relation could secure the largest part the non-dominated set for their respective algorithms. The Crowding scheme and Standard P-ACO have barely any solutions in the combined non-dominated sets. The combined sets of the GAs indicate a qualitative lead of the Win-GA before the WL-, respectively, Pt-GA that are represented by approximately the same amount of solutions as the W-GA.

Table 9.4: Fraction of solutions that rank in the top 10% of all final solutions from the respective type of algorithms (P-ACO respectively GA) concerning one of the objectives with standard deviation; Total Time(T_{total}), Flow Time (T_{flow}), Total Idle Time Jobs ($T_{i.j.}$),Total Idle Time Machines ($T_{i.m.}$); values are averages over all problem instances; best values are shown in grey.

Combined front of		{ Pt	W	Win	Cr	Std }	{ WL	W	Win	Cr	Std }
P-ACO	T_{total}	58.4	6.1	34.5	1.0	0.0	53.1	6.5	39.5	0.9	0.0
	T_{flow}	42.5	25.1	26.2	0.0	6.2	43.4	28.9	23.0	0.0	4.7
	$T_{i.m.}$	53.3	3.7	34.4	2.8	5.9	55.0	3.6	32.8	2.8	5.9
	$T_{i.j.}$	5.0	87.3	2.7	2.5	2.5	3.8	88.4	1.5	3.1	3.1
GA	T_{total}	40.6	5.3	50.9			51.30	5.1	40.0		
	T_{flow}	30.1	39.8	28.0			27.3	30.1	41.0		
	$T_{i.m.}$	35.1	1.0	61.9			44.5	1.2	53.0		
	$T_{i.j.}$	2.1	92.9	2.1			7.0	89.80	1.6		

9.1.2 Objective Oriented Analysis

While the above analysis showed which algorithms final solution set is close to the (unknown) Pareto set another quality aspect is the heterogeneity of the solutions. An algorithm producing a homogeneous set of solutions is typically prone to optimize towards one specific objective. A set with solutions that show good values in all objectives has a higher quality concerning this aspect of diversity.

In order to analyze this property all final non-dominated solution sets are merged and sorted by the respective objective value the single solutions obtained. A good algorithm should be represented in the top tenth of such a sorting with many solutions and regarding all objectives (see Table 9.4). It is quite obvious that the algorithms using a weighted ranking scheme optimize mostly for $T_{i.j.}$ and its positively correlated objective T_{flow}. With 0.676 (Pareto set) and 0.771 (random set) these two objectives have the highest positive correlation of all pairs of objectives (see Table 8.2). The Pt- and WL-P-ACO respectively have the largest fraction of solutions in the top tenth of all objectives but $T_{i.j.}$ where they rank second. The Pt-GA is outperformed by the Win-GA in T_{total} and $T_{i.m.}$ and by W-GA in T_{flow}. WL-GA is outperformed by both, Win-GA and W-GA, in T_{flow} and Win-GA in $T_{i.j.}$. Cr- and Std-P-ACO are barely represented by their solutions.

Aside from being good in one objective heterogeneous sets of solutions also contain solutions that are good, though not best, in two or more ob-

jectives. This is best described with the previously defined value $\mathcal{B}^<(a,b)$ that counts the objectives in which solution a is better than solution b. To infer how solutions a of one algorithm compete against solutions b of the other algorithms in the non-dominated set, we refer to the $\mathcal{B}^<(a,b)$ values of all these comparisons. Figure 9.1 depicts the fraction of comparisons of each algorithm that ended with a specific $\mathcal{B}^<(a,b)$ value. These values can only be from $\{0,1,2,3\}$ as $\mathcal{B}^<(a,b) = 4$ would imply $a \prec b$ and violate the properties of the non-dominated set. $\mathcal{B}^<(a,b) = 0$ implies $\vec{f}(a) = \vec{f}(b)$.

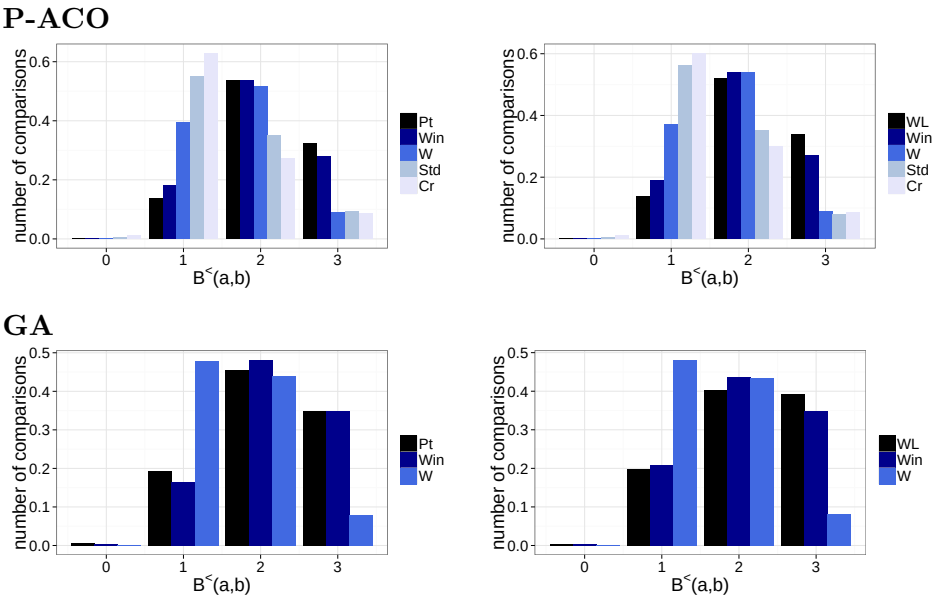


Figure 9.1: Histogram of the number of $\mathcal{B}^<(a,b)$ values of all solutions a in the combined non-dominated sets of one algorithm compared with all the solutions b of all respective other algorithms; for the group of algorithms with Pt-P-ACO (left) and the group of algorithms with WL-P-ACO (right).

The solutions of Pt-P-ACO and WL-P-ACO have the largest fractions of solution comparison with $\mathcal{B}^<(a,b) \geq 2$ closely followed by Win-P-ACO. The W-P-ACO still has over half its comparison with $\mathcal{B}^<(a,b) \geq 2$, while the comparisons with Cr-P-ACO and Std-P-ACO mostly have $\mathcal{B}^<(a,b) = 1$.

The respective results with the genetic algorithms are based on only three sets of solutions each. Here, the Wining Score relation induces slightly higher $\mathcal{B}^<(a,b)$ values than the Points relation. The WL relation induces more $\mathcal{B}^<(a,b) \geq 2$ than both compared ranking schemes.

9.1.3 Convergence Behavior

Besides the quality of the final set of solutions an algorithm acquires with a specific ranking scheme it is also interesting to see how the ranking schemes influence the general behavior of the algorithm during the runtime. Most importantly, this analysis focuses on the convergence behavior of the algorithm. The convergence of an algorithm is described by the increase of quality of the solutions in the population and the ones just generated throughout all iterations. Typically, there is a strong initial increase in quality that slows down when the algorithm is unable to generate solutions better than those already in its population.

Figure 9.2 shows how the different P-ACO algorithms converge during 100 000 iterations on four exemplary instances. These are *car1*, *hel1*, *reC37*, and *ta111*. The hypervolume indicator is applied to measure the quality of the sets of solutions with a reference point defined by the maximal values of all objective values occurring throughout all experiments. It is measured every ten iterations.

The Cr- and Std-P-ACO algorithms fail to improve the quality of their solutions to the extend the other compared algorithms do within the first 50 000 iterations. The Std-P-ACO, however, performs better than the Cr-P-ACO. In all four compared instances the Pt-P-ACO achieves the lowest hypervolume indicator, closely followed by WL-P-ACO and Win-P-ACO that are both twice second and third. W-P-ACO is clearly better than Cr- and Std-P-ACO but is also clearly worse than Pt-, WL-, and Win-P-ACO.

Note, as the Pareto set for the *car1* instance is computable in justifiable time we can compare the hypervolume indicator of random subsets of the Pareto set with the solution sets of the compared algorithm. Mind, the Pareto set contains 502 solutions and its hypervolume indicator is obviously far smaller than a set of only 30 solutions can obtain. From these 30 solutions a small number is actually non-dominated and accounted for in the computation of the hypervolume indicator. For means of a better understanding of the presented values we give the average hypervolume indicator of 1000 random subsets of the Pareto set with 30 (red) and 5 (green) solutions, respectively.

The bad performance of Cr- and Std-P-ACO and apparently random behavior with respect to the dominated hypervolume can be explained by the small population and missing archive of the P-ACO that was used in the original algorithm by Guntsch and Middendorf (2002). The sparse pheromone matrix, built solely by the few solutions in the population, appears to be insufficient for algorithms that rank solutions solely by Pareto dominance.

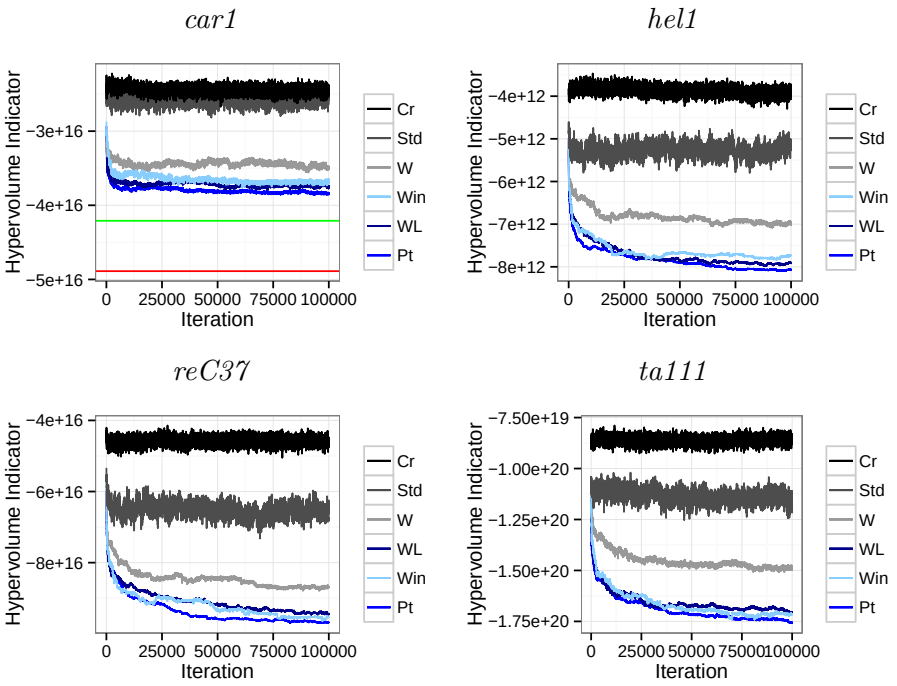


Figure 9.2: Average hypervolume indicator during 10 test runs with 100000 iterations measured every 10 iterations on four different instances: *car1* (top, left), *hel1* (top, right), *reC37* (bottom, left), *ta111* (bottom, right); reference point is the combination of all maximal values for each objective measured in the experimental runs; green (upper) and red (lower) horizontal lines for *car1* give the average hypervolume indicator for 1000 samples with 5 respectively 30 randomly chosen solutions from the true Pareto front.

Table 9.5: Parameters used in experiments. (*)For the Cr-P-ACO and Cr-GA $p = l = 25$, otherwise $p = 5$.

Symbol	Definition	Values for experiments to infer Performance	and Convergence
t	iterations	25 000	50 000
	runs	50	10
p	population size	5 (25*)	5 (25*)
l	number of ants	25	25
τ_{init}	minimal pheromone	$1/n$	$1/n$
τ_{max}	maximal pheromone	$\tau_{init} + 24$	$\tau_{init} + 24$
μ	mutation probability	$1/(n-1)$	$1/(n-1)$
	solution evaluations	625 000	1 250 000
	instances	all	*att48*
c	correlation parameter	$\{0.1, 0.5, 0.9\}$	$\{0.1, 0.9\}$

9.2 Influence of Correlations

The experiments on the correlated TSP instances were run with the same parameters that were used on the FSP instances (see Table 9.5). Since there is no standard heuristic for multi-objective TSP no heuristic was applied in order to avoid dependencies induced by the use of specific heuristics.

The TSP test instances were generated with $k = 4$ objectives each, where we distinguish between homogeneous and heterogeneous instances. Homogeneous instances are generated such that for each objective $h \in \{1, 2, 3, 4\}$ we construct a distance matrix d_h using the initial matrix d such that

$$d_h(i, j) = c \cdot d(i, j) + (1 - c)rand. \tag{9.1}$$

A heterogeneous instance applies Equation (9.1) for the first three objectives $h \in \{1, 2, 3\}$ and defines

$$d_4(i, j) = c(1 - d(i, j)) + (1 - c)rand.$$

The parameter c is set to 0.1 to induce a low correlation between the objectives, to 0.5 to induce a medium correlation, and to 0.9 to induce a strong correlation between the objectives. With three instances (*att48*, *berlin52*, *eil51*), two instance generation types (homogeneous and heterogeneous), and three values for c (0.1, 0.5, 0.9) we have $3 \cdot 2 \cdot 3 = 18$ problem instances. These instances are identical over all simulation runs in order to compare the different algorithms.

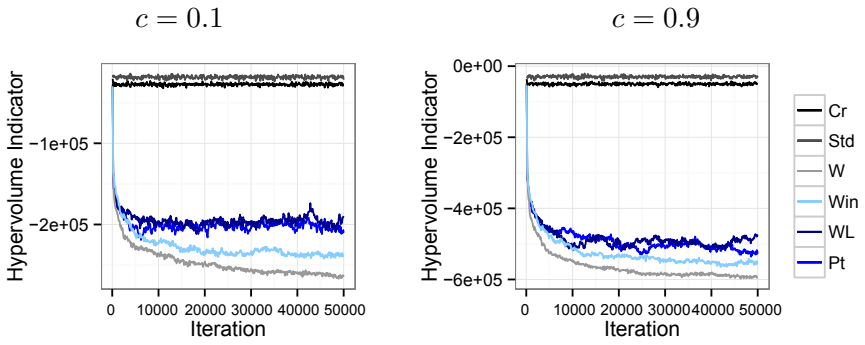


Figure 9.3: Convergence behavior of the P-ACO variants on two homogeneous instances derived from the *att48* instance; hypervolume indicators of the solutions generated every hundredth iteration and the solutions of respective population; averaged values over 10 runs

Convergence behavior The influence of the correlation on the convergence behavior of the P-ACO variants is shown at the example of two instances derived from the *att48* problem instance in Figure 9.3. Both instances are heterogeneous with $c = 0.1$ and $c = 0.9$. Note, that the P-ACO with the weighted ranking scheme is better suited for these TSP instances where all objective values are in the same range and have the same order of magnitude in improvement, unlike the previously studied FSP problem. The Pt- and WL-P-ACO perform similar and are slightly worse than the Win-P-ACO. The convergence behavior of the Cr- and Std-P-ACO fail to improve their solutions throughout the run where the Cr-P-ACO has a slightly lower hypervolume indicator due to their larger population.

Figure 9.3 shows that the algorithms converge in the first 25 000 iterations while the strongest quality increase occurs in the initial 5000 iterations. The higher correlation in the instance with $c = 0.9$ shows a slightly steeper decrease of the hypervolume indicator and hints on a faster convergence.

The convergence behavior of the GA on the same instance is shown in Figure 9.4. While the W-GA clearly achieves the lowest hypervolume indicator with a low correlation, the results for the higher correlation does not favor any particular ranking scheme over another. The initial strong decrease of the hypervolume indicator subsides furing the first 5000 iterations, followed by a slower decrease.

Final solution sets Table 9.6 shows that the final non-dominated solution sets are larger when the algorithms solve these TSPs then when solving the

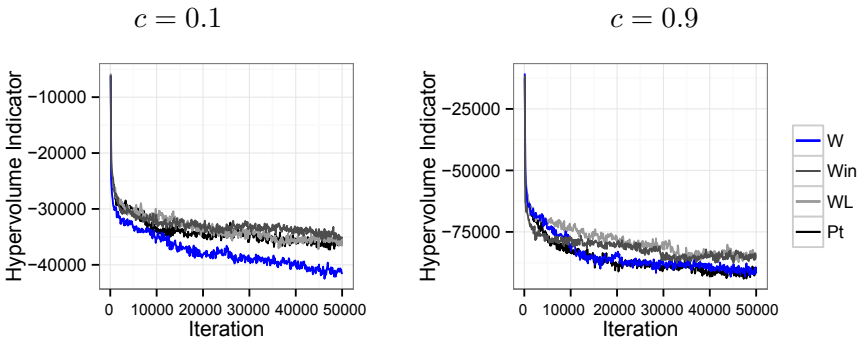


Figure 9.4: Convergence behavior of the GA variants on two homogeneous instances derived from the *att48* instance; hypervolume indicators of the solutions generated every hundredth iteration and the solutions of respective population; averaged values over 10 runs

Table 9.6: Average size of the non-dominated sets.

	P-ACO						**GA**			
c_4	{ Pt	WL	W	Win	Cr	Std }	{ Pt	WL	W	Win }
0.1	16.8	16.8	16.7	16.6	24.9	30.0	15.9	16.5	16.1	16.1
0.5	16.9	16.9	16.3	16.5	24.9	29.9	16.2	15.9	16.4	16.0
0.9	16.6	16.8	16.7	16.5	24.9	29.9	16.7	16.2	16.5	16.2
−0.1	16.7	17.1	16.7	17.3	24.9	29.9	16.6	16.0	16.0	16.4
−0.5	16.7	16.6	16.8	17.1	24.9	30.0	15.7	16.4	16.0	16.5
−0.9	17.2	16.9	16.7	16.9	24.9	29.9	15.7	16.1	16.2	15.9

FSPs. The different correlations strengths have no further influence on the number of final solutions. Note, that the five solutions in the population should have a higher quality than the 25 solutions generated in the final iteration, unless the algorithms fails to optimize the problem like the Std-P-ACO. The algorithms need larger populations or even archives of solutions in order for the different correlations to have an effect on the size of the final solution set. This behavior of the algorithms is visible in both types of algorithms. The influence of the correlations becomes more apparent in the data shown in Tables 9.7 and 9.8. In this comparison we select for each objective the 10% best of all final solutions of all compared algorithms (shown in the second line of the tables). We then analyze the distribution of this solution set regarding to their algorithm of origin.

Table 9.7: Fractions of solutions in the top tenth of all final solutions regarding single objectives with Pt; values are averages over all problem instances.

		P-ACO					**GA**		
c_4		{ Pt	W	Win	Cr	Std}	{ Pt	W	Win }
0.1	f_1	0.226	0.415	0.337	0	0	0.218	0.365	0.417
	f_2	0.226	0.462	0.298	0	0	0.413	0.334	0.253
	f_3	0.164	0.485	0.333	0	0	0.307	0.436	0.258
	f_4	0.152	0.533	0.302	0	0	0.302	0.452	0.246
0.5	f_1	0.269	0.445	0.277	0	0	0.437	0.265	0.297
	f_2	0.247	0.388	0.356	0	0	0.272	0.374	0.354
	f_3	0.241	0.436	0.314	0	0	0.438	0.331	0.231
	f_4	0.171	0.467	0.357	0	0	0.213	0.583	0.204
0.9	f_1	0.304	0.336	0.346	0	0	0.306	0.384	0.311
	f_2	0.262	0.445	0.278	0	0	0.265	0.451	0.284
	f_3	0.288	0.418	0.284	0	0	0.383	0.370	0.247
	f_4	0.275	0.484	0.230	0	0	0.462	0.270	0.268
−0.1	f_1	0.255	0.424	0.300	0	0	0.294	0.433	0.272
	f_2	0.257	0.366	0.357	0	0	0.142	0.528	0.330
	f_3	0.176	0.430	0.376	0	0	0.330	0.452	0.218
	f_4	0.186	0.507	0.288	0	0	0.461	0.234	0.306
−0.5	f_1	0.295	0.387	0.299	0	0	0.268	0.418	0.315
	f_2	0.255	0.420	0.305	0	0	0.270	0.268	0.462
	f_3	0.231	0.401	0.354	0	0	0.368	0.342	0.290
	f_4	0.130	0.556	0.302	0	0	0.365	0.383	0.252
−0.9	f_1	0.346	0.382	0.251	0	0	0.347	0.301	0.352
	f_2	0.357	0.314	0.304	0	0	0.414	0.255	0.331
	f_3	0.380	0.333	0.262	0	0	0.307	0.417	0.277
	f_4	0.049	0.643	0.299	0	0	0.214	0.543	0.243

In Table 9.7 we compare the Pt-P-ACO and Pt-GA with the state-of-the-art algorithms. Apparently the Cr-P-ACO and Std-P-ACO have no solutions whose objective values range in the best tenth of all solutions. The W-P-ACO and W-GA provide most of the solutions for almost all objectives and correlation variations but are closely followed the respective Pt- and Win algorithms. Note, that with negative correlations the W-P-ACO has most of the best solutions in the negatively correlated objective, i.e. f_4. The Pt-P-ACO appears to focus its optimization effort onto the three positively correlated objectives. The data received from the respective genetic algorithms does not show this behavior.

The respective results for the WL ranking scheme is shown in Table 9.7. The data is similar to that of the Pt ranking scheme. The correlation clearly influences the behavior of the different algorithms. Mind, that the values of f_4 in Win-P-ACO with $c_4 = -0.9$ somewhat imply that f_4 is favored over the other objectives by W-P-ACO. Otherwise, Win-P-ACO should have a higher value as it also benefits from Pt- and WL-P-ACO mostly ignoring f_4. This benefit however subsides, because W-P-ACO is more prominent.

9.3 Conclusion

The different types of results showed that the Points and WL ranking relations are of benefit to the P-ACO and induce a competitive performance in GAs when solving problems with very heterogeneous objectives, i.e. the objective values are different in their ranges and their power of improvement. However, they are still competitive when applied on homogeneous problems where weighted schemes perform very well. Ranking schemes based solely on the Pareto dominance are insufficient with small populations.

The correlation of objectives influences the time of convergence of all applied algorithms and had a notable impact on their behavior. The Points and WL ranking relations clearly prioritized positively correlated objectives unlike the weighted ranking scheme that optimized towards single objectives with negative correlations. The Winning Score remained a strong competitor for both proposed methods. While the Points and WL ranking relations showed better results for the FSP, the Winning Score was better when applied to solve the TSPs.

Table 9.8: Fractions of solutions in the top tenth of all final solutions regarding single objectives with WL; values are averages over all problem instances.

		P-ACO					**GA**		
c		{ WL	W	Win	Cr	Std }	{ WL	W	Win }
0.1	f_1	0.236	0.429	0.332	0	0	0.343	0.372	0.431
	f_2	0.258	0.444	0.310	0	0	0.291	0.397	0.316
	f_3	0.207	0.486	0.330	0	0	0.346	0.418	0.250
	f_4	0.153	0.567	0.320	0	0	0.333	0.409	0.217
0.5	f_1	0.249	0.455	0.287	0	0	0.553	0.201	0.247
	f_2	0.252	0.381	0.359	0	0	0.269	0.388	0.344
	f_3	0.161	0.479	0.354	0	0	0.321	0.403	0.275
	f_4	0.246	0.436	0.312	0	0	0.228	0.566	0.206
0.9	f_1	0.231	0.368	0.394	0	0	0.367	0.354	0.279
	f_2	0.328	0.412	0.254	0	0	0.252	0.457	0.291
	f_3	0.276	0.425	0.292	0	0	0.318	0.415	0.267
	f_4	0.278	0.482	0.234	0	0	0.415	0.303	0.282
−0.1	f_1	0.205	0.468	0.319	0	0	0.345	0.404	0.251
	f_2	0.186	0.410	0.396	0	0	0.220	0.490	0.290
	f_3	0.148	0.449	0.392	0	0	0.429	0.397	0.175
	f_4	0.175	0.515	0.297	0	0	0.353	0.262	0.384
−0.5	f_1	0.220	0.437	0.337	0	0	0.412	0.325	0.263
	f_2	0.242	0.429	0.319	0	0	0.278	0.280	0.442
	f_3	0.190	0.423	0.377	0	0	0.372	0.331	0.297
	f_4	0.092	0.581	0.319	0	0	0.322	0.418	0.259
−0.9	f_1	0.301	0.412	0.270	0	0	0.328	0.325	0.347
	f_2	0.332	0.326	0.322	0	0	0.447	0.230	0.324
	f_3	0.334	0.363	0.286	0	0	0.318	0.379	0.303
	f_4	0.077	0.629	0.288	0	0	0.260	0.514	0.226

Part IV

Cooperative Swarms Solving Multi-Objective Tasks

10 Introduction

Evolutionary multi-agent systems

Heterogeneous swarms of agents that solve multi-agent tasks as introduced in Part II might need to take multi-objective decisions as well. When confronted with multi-objective multi-agent tasks, for example, agents require ranking schemes, like those introduced in Part III, to decide efficiently which task is preferable at the time of decision. Though, different ranking schemes might lead to a better mid term strategy depending on environmental factors. Those environmental factors can change over time, e.g. the priority of the objectives or their correlations. In such dynamic situations it could be beneficial for the agents to base their decision on different ranking schemes and react to changes in their environment. In order to enable such adaptivity in the swarm of agents, multiple ranking schemes need to be known by the agents. One approach to maintain multiple ranking schemes in the swarm is the use of its distributed and heterogeneous properties. When each agent uses its individual ranking scheme the system can indeed contain as many types of ranking schemes as there are agents. Inspired by evolutionary processes that lead to adaptivity in species we introduce an evolutionary mechanism to achieve adaptivity in the swarm of agents.

Evolutionary adaptive mechanisms have been studied in the field of Multi-Agent learning (Panait and Luke, 2005). Evolutionary processes like recombination, crossover, and mutation have been used in evolutionary algorithms such as genetic algorithms for a long time and are also part of evolutionary multi-agent systems, e.g. in studies from Pieter and de Jong (2004); Shibata and Fukuda (1993); Whitacre et al. (2010). In evolutionary robotics the shape, movement, or interaction between robots is designed using evolutionary processes. For a recent review on the topic the reader is referred to Bongard (2013). However, most of these studies either focus on homogeneous swarms or systems where each agent or robot evolves by

itself using neural networks or genetic algorithms. In this study I am more interested in transfer of genetic material from one agent to another one such that agents can learn from the experiences of other agents.

Evolutionary processes are often based on haploid mechanisms where all individuals contain one copy of their genetic material, i.e. chromosomal sets in biology or solutions in genetic algorithms. Haploidy is very straight forward in the elimination of lethal genes or bad solutions, as they are immediately removed from the population by the death of the individual and, therefore, its failure to reproduce.

In contrast, diploid individuals carry two different copies of genetic material. In case of a dominance recessive relation between two parts, or rather alleles, of these copies the dominant allele defines the phenotype of the individual while the recessive allele remains dormant. The individuals fitness is defined by the benefits of the dominant alleles but when it reproduces, the recessive alleles will be multiplied as well. This mechanism allows 'bad' genetic material to remain in the population but also increase the diversity of the population. In dynamic systems this enhanced diversity can make the significant difference towards higher adaptability.

The concept of diploidy is not new in genetic algorithms (Goldberg and Smith, 1987; Yukiko and Nobue, 1994). It has been shown that the higher diversity in the population can improve results by avoiding a premature convergence of the algorithm (Herrera and Lozano, 1996; Leung et al., 2001; Mauldin, 1984; Potts et al., 1994). A study of diploidy in neural networks showed that diploids had lower average fitness but a higher peak fitness in fixed environments and were able to keep a genetic 'memory' that they could fall back on in changing environments with repetitive conditions (Calabretta et al., 1996). The benefit of diploidy in dynamic multi-agent system has been studied by (Bowers and Sevinç, 2006).

A third reproduction mechanism that occurs in nature is haplo-diploidy. In haplo-diploid species the males that hatch from unfertilized eggs are haploid while the females that origin from fertilized eggs and are diploid. In nature we can find this phenomenon for example in Hymenoptera (e.g. bees, ants, wasps). While the diploid females provide a high diversity in the population the haploid males eliminate lethal alleles from the population – by dying. So far and to the best of my knowledge, the concept of haplo-diploidy has been altogether disregarded for evolutionary algorithms and evolutionary multi-agent systems. However, it is interesting to see how the above three concepts compete against each other in an evolutionary system.

In Chapter 11 an evolutionary model of agents solving multi-objective multi-agent tasks is described followed by its analysis in Chapter 12.

11 Model

Solving multi-objective multi-agent tasks

This model is designed to be as minimalistic as possible with a small number of variable parameters to enable a clear and coherent analysis of certain aspects of the system. The agents have limited abilities and knowledge in order to investigate a general scenario that does not depend on assumptions from specific applications. The model is designed to analyze the general problem that is described in the following.

The model provides a dynamic environment for a swarm of individual agents that are tasked with collection of distributed and rare resources. The collection of the resources requires a specific number of agents for a specific amount of time in a specific location. The number of required agents and the required time to collect the resources deviates locally. Each agent has only very limited information about the environment. However, an agent can asses information about its current location and has the ability to memorize information for a limited number of other locations which it has visited before. Additionally, an agent can communicate with other agents in its vicinity and can build a team with them to cooperatively collect resources. The agents try to collect as many resources as possible. For this, a decentralized strategy is needed due to the limited sensory ranges and the restricted communication skills of the agents.

This approach provides each agent with two possible behavioral states: the *single agent state* or *exploration state* and the *team state* or *exploitation state*. In the exploration state an agent walks randomly around in order to find locations with good properties concerning the collection of resources. It will memorize the best locations it has found. In the exploitation phase a team targets the best locations its members have found and memorized during their respective former exploration phase. Once a team has collected all resources at a location that was found by a member of the team, that team

member leaves the team, erases its memory, and returns into the exploration state.

We investigate here whether the decentralized strategy enables the agents to gather information effectively by swarming the environment and to subsequently use this information to successfully collect resources in teams. Obviously, the general strategy raises a couple of specific design problems in the agents behavior. We found the following to be most interesting:

1. Which locations should be memorized during the exploration phase?
2. How long should an agent explore?
3. Gathering more agents can be essential for the success of a team. When two teams meet and they both require more agents, which team is able to take agents from the other team?
4. Each team has several potential target locations to chose from, i.e. the locations which are memorized by its members. By which means should the team decide which location to target next?

Since we investigate dynamic scenarios where there might not be one best strategy the system should adapt or evolve the strategies that the agents use. Each simulation turn consists of three phases: the team formation phase, the working phase, and the reproduction phase (for a graphical overview see Figure 11.1). The details of the agent model are introduced in the following.

11.1 Environment

The environment consists of a two-dimensional arena F that is a torus with $d \times d$ fields. Thus, the last field in every row and column is adjacent to the corresponding first field in its row respectively column. $A = \{a_1, \ldots, a_n\}$ is the set of agents that are located in the arena. The size of a field defines the sensory range of an agent with respect to the resources. Hence, an agent on a field $f \in F$ can sense the amount of resources on f but it cannot sense the amount of resources on other fields. For collecting resources a team of at least two or more agents has to be formed, i.e. a single agent cannot collect resources. All agents of a team are located on the same field. On each field at most one team of agents can collect resources at a time. Each team of agents can communicate with a team or agent that is located on the same field or one of the eight adjacent fields (Moore neighborhood) but not to teams/agents on other fields, i.e. the communication range includes

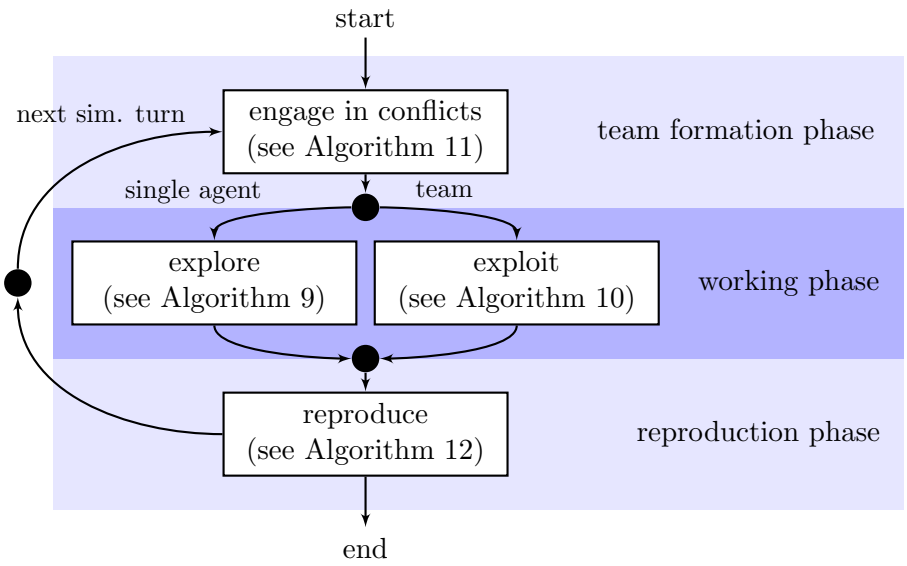


Figure 11.1: Flow diagram of the three phases making up a simulation turn. Black dots are points of synchronization.

only the current field and its adjacent fields. Each agent and each team has an orientation which points to one of the eight fields that are adjacent to its current field (see Figure 11.2).

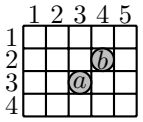


Figure 11.2: Two agents a and b in each others sensing range in a small part of the arena.

Each simulation is a sequence of simulation turns. During each simulation turn an agent or a team of agents can move from its current field to an adjacent field (see Figure 11.2). Additional to its coordinates $(x(f), y(f))$ each field $f \in F$ has the following attributes

1. $r(f)$ denotes the amount of resources on f,
2. $a(f)$ is the required minimum size of a team to collect the resources on f,
3. $t(f)$ is the number simulation turns that a team has to remain on f in order to collect the resources on f.

The attribute values of each field are chosen randomly. When all resources of a field have been collected, the field is initialized anew.

11.2 Working Phase

During the working phase single agents move to explore their environment in search for good fields while teams move either towards their target field or, if they are on their target field, collect resources.

11.2.1 Exploration State

A single agent in the exploration phase gathers up-to-date information from the environment. The best fields that an agent visits can be saved in its *field memory*. For each memorized field, the memory contains the location, the attribute values, and the time of visit. Whenever an agent moves onto a field f it can add information about f to its memory, but might need to discard information about another field due to the limited capacity of the memory. There are four quality measures to be considered by an agent. Hence, the agent is confronted with a multi-objective optimization problem. It should consider

1. the amount of resources $r(f)$,
2. the number of required agents $a(f)$,
3. the resource collection time $t(f)$, and
4. the age of the gathered information, i.e. the time of the visit.

Objective 1 is to be maximized while all other objectives are to be minimized. The agents can use different ranking methods to infer which information on which field is to be kept and which is to be discarded. The specific ranking methods are described in Section 11.5. Algorithm 9 gives an overview over the working phase for agents in the exploration state.

After an agent changes into the exploration state it will refuse to become member of a team in order to gather and filter enough information on fields. However, after its *exploration time*, defined by a specific number of simulation turns, has passed the agent tries to become member of a team to put its gathered knowledge to use. Before an agents has finished its exploration time it is called *blocked agent* and otherwise it is called a *free agent* because it is free to join a team.

11.2.2 Exploitation State

All members of a team share their location such that the team takes just as much space in the arena as a single agent. To ensure their common location agents of the same team move together. Teams of agents no longer explore their environment but decide on a target field and move there on

Algorithm 9: Working phase: exploration state

```
chose random number ρ ∈ [0, 1]
if ρ < rotation probability then
    change orientation by 45°
move forward in the current orientation onto the adjacent field f′
if field memory M is full then
    rank all fields from M ∪ {f′}
    discard the field with lowest rank and place all other in M
else
    add f′ to M
```

the shortest path disregarding their orientation. Once they arrived at their target field they either engage in the resource collection task or fail to do so because the team is too small or another team is currently working on the field. If the team can work on the field it remains there for the required time to collect the available resources and then leave, otherwise they leave immediately. Leaving a target field f consists of three steps: (1) part from the agent that provided the information of f, (2) decide on a new target field f', (3) and move one field into the direction of f'. See Algorithm 10 for details.

Algorithm 10: Working phase: exploitation state

```
if team T is located on its target field f_T then
    if no other team works on f_T and |T| ≥ a(f_T) and
    counter < t(f_T) then
        work on target
        counter := counter + 1
        if counter = t(f_T) then
            increase personal and global collected resources by r(f_T)
            counter := 0
    else
        owner of f_T in T leaves the team
        chose next target field
        counter := 0
else
    move toward target field
```

11.3 Team Formation Phase

Whenever two teams (or free agents) T and T' are in each others sensing range, i.e. their location fields are in each others Moore neighborhood, they can engage in a team formation event. Team T will try to get agents from T' if T misses some agents to process its target field successfully. However, no team wants to give away any of its agents, as they provide possibly relevant field information. If both teams have enough agents they rather avoid a conflict, but in any other case they engage in a competition over their member agents (see Algorithm 11).

Algorithm 11: Team formation during conflicts

```
1  if T's exploration time has ended and |T| < a(f_T) and T has not
   been part of a conflict in this simulation turn then
2      chose a team or agent T' randomly from all those  teams and
       agents with the following properties:
3        1. T' is in the Moore-neighborhood of T
4        2. T' is not working on a field
5        3. T' 's exploration time has ended
6        4. T' was in no conflict during the current simulation turn
7      chose a random number ρ ∈ [0, 1]
8      if ρ ≤ power(T)/(power(T) + power(T')) then
9          T takes min(|T'|, a(f_T) − |T|) agents from T'
10     else
11         T' takes min(|T|, a(f_T) − |T'|) agents from T
```

Of the many possible strategies to decide this conflict the agents use a very archaic approach. Each agent a has a power $pow(a)$. The power $pow(T)$ of a team T is the sum of the power of its members, i.e. $pow(T) = \sum_{a \in T} pow(a)$. In a conflict between team T and T' team T wins if

$$\rho \leq \frac{pow(T)}{pow(T) + pow(T')}$$

where $\rho \in [0, 1]$ is a random number from a uniform distribution. The additive power favors large teams over small teams. This enables larger teams that are unable to work on their target field with their current size to grow more easily.

Let $x = \min\left(a(f_T) - |T|, |T'|\right)$, where f_T is the target field of T and $|T|$, $|T'|$ are the sizes of the teams T and T' respectively. Then x agents from

T' are transfered to T if T wins. If $a(f_T) - |T| \leq |T'|$, T' is completely absorbed by T. A worst case scenario for T' is the loss of the agent that provided T' with the information of $f_{T'}$. Then T' is forced to chose another target field after it already invested time to move towards $f_{T'}$.

11.4 Choosing a Target Field

A target field has to be chosen in the following situations:

(A) Agent a just finished its exploration phase and is in a conflict with team T. If a wins it gets agents from T and builds its own team. In this case a choses one of the fields in its own field memory as target field. It ignores the information of the newly acquired agents, as any other team would when it got new agents after a victory. Note that a leaves the team after the resources from its proposed field are collected, thus, returning into the exploration phase as soon as possible.

(B) Team T reached its target field. Independent of T's success in collecting resources on that field, it evicts the agent that provided the information and has to chose a new target field.

(C) Team T lost a conflict and the agent providing the information on the previous target field left the team.

In the latter two cases, team T choses its new target field among all fields memorized by all member agents. There are four objectives to consider in this decision.

1. The **amount of resources** $r(f)$ on the field f.
2. The probability $P(|T| \geq a(f_T))$ to have the **required team size** by the time the field is reached by the agents. This value is estimated by the team with a simple learning mechanism for the expected change of the size of the team per simulation turn P_{growth}. In each simulation turn P_{growth} is computed anew as follows

$$P_{growth} = 0.8P_{growth} + 0.2\Delta$$

where Δ is the change of the size of the team during the last simulation turn. For example, if a team has lost two agents in the last simulation turn, then $P_{growth} = 0.8P_{growth} - 0.2 \cdot 2$. Further, P_{growth} is used to infer the probability to reach the needed team size $a(f)$ during the time of travel. The required time of travel is inferred by the use of Equation (11.1) where f_c is current location and f_T the target field of team T then

$$\Delta(f_c, f_T) = \left\| \begin{pmatrix} \frac{d}{2} - \left| |x(f_c) - x(f_T)| - \frac{d}{2} \right| \\ \frac{d}{2} - \left| |y(f_c) - y(f_T)| - \frac{d}{2} \right| \end{pmatrix} \right\| \tag{11.1}$$

where $\|x\|$ is the Euclidean norm of x. Note, that the torus arena gives four Euclidean distances between two fields and that Equation (11.1) determines the minimum Euclidean distance between f_c and f_T. The agents derive an approximated probability for team T to have at least size $a(f_T)$ with

$$P\left(|T| \geq a(f_T)\right) = \max\left(0, \min\left(1, 1 - \frac{a(f_T) - P_{growth}\Delta(f_c, f_T)}{2|T|}\right)\right)$$

Note that $|T| + P_{growth}\Delta(f_c, f_T)$ is the expected size of $|T|$ at the time is reaches f_t. Mind, that if the expected team size equals the number of required agents $a(f)$, then $P\left(|T| \geq a(f_T)\right) = 0.5$. If the expected number of agents added to the team during the transfer ($P_{growth}\Delta(f_c, f_T)$) equals $a(f)$, then $P\left(|T| \geq a(f_T)\right) = 1.0$

3. The **required time** for a target field f_T is $\Delta(f_c, f_T) + t(f_T)$, i.e. the time to reach it and collects it resources afterwards.
4. Once the resources of a field have been collected it is regenerated. However, agents are not informed whether any of their memorized fields regenerated since their **time of visit**. The more time has passed since they memorized the information the higher is the possibility that the information is deprecated because the field regenerated in the mean time.

The objectives should be considered by the team when it choses a new target field, as they all influence the general uptake of resources. In a team each agent ranks all fields in the field memory of all member agents. These ranks are average to infer the highest ranking field and chose it as new target field.

11.5 Ranking Methods

Each agent applies exactly one the following ranking methods at a time. Note that the field with the highest rank is considered the 'best' field.

Dominance Depth Ranking The dominance depth ranking is based on the Pareto dominance relation. One field f is Pareto dominant over another field f' if f is not worse than f' in any objective and in at least one objective better then f'. The Pareto dominance relation is not total thus it is possible to have two fields and neither dominates the other. A dominance depth ranking sorts the set of fields such that no field is dominated by any field of lower rank where all ranks are minimized. In case of incomparability the order is set randomly.

Weighted Ranking The weighted ranking uses a weight vector $w \in [0,1]^k$, with $\sum_{i\in\{1,\dots,k\}} w_i = 1$, that is individual for each agent. It is used to compute a weighted sum over all objectives. The field with the lowest weighted sum has the highest rank. Note, that the negative value of the first objective is applied, as it is the only objective that needs to be maximized. For a set of fields M where $f^{(j)}$ is the objective value of objective $j \in \{1,\dots,k\}$ that needs to be minimized we have for a field $f \in M$ with a weight vector w

$$s(f) = \sum_{j=1}^{k} \frac{w_i \left(f^{(j)} - \min_{f'\in M} f'^{(j)}\right)}{\max_{f'\in M} f'^{(j)} - \min_{f'\in M} f'^{(j)}}, \quad f \in M.$$

The lower $s(f)$ the higher is the rank of f.

WL Ranking This ranking method conducts pairwise comparisons of all fields in the set. Each comparison can have a winner, a loser, or a tie. The field that has better values in more objectives than its competitor wins the comparison and the competitor loses the comparison. If the number of objectives in which they are better is equal to the number of objectives in which they are worse, both fields are in a tie. A field is prioritized before another field if it either wins in more comparisons or, in case of equal number of won comparisons, loses less comparisons.

Points Ranking Similar to the WL ranking, all fields are compared to each other. The Points ranking gives a field two points for a victory and one point for a tie. The number of points is used to rank the fields. If two or more fields have an equal amount of points, their order is set randomly.

11.6 Evolutionary Processes

Instead of tuning the critical parameters in an exhaustive analysis and to avoid over fitting in dynamic environments some of the critical parameters are instead learned by evolutionary mechanisms. These mechanisms are inspired by evolutionary processes that prove over millions of years to provide adaptability and efficiency of systems.

Each agent has at least one set of genetic material in shape of a chromosome set. These chromosomes contain the agents behavioral properties (see Table 11.1). Haploid agents contain one chromosome set, while diploid agents contain two chromosome sets where only one is expressed and one is

Table 11.1: Chromosomes that are contained by one chromosome set of an agent.

Chromosome	Definition
Exploration Time	minimum number of simulation turns an agent spends exploring the environment
Power	increases the chance of an agent to win a conflict
Ranking Scheme	ranking scheme the agent applies to chose one field over another in a multi-objective decision problem
Weights	weights vector of agents using the weighted ranking scheme.

repressed. Further polyploidy, i.e. more than two chromosome sets is not analyzed in this study.

11.6.1 Reproduction and Death

Reproduction and death are the central motives in this evolutionary multi-agent system. Agents reproduce once they collected a certain amount of resources, defined by the *reproduction threshold* (see Algorithm 12). Death is realized such that the number of agents remains constant and each reproduction goes along with the death of a random other agent. Three different reproductive mechanisms are considered in the following.

Haploid Reproduction A haploid agent a has one chromosome set. If a reaches the reproduction threshold it randomly choses agent a' and overwrites the chromosome set of a'. During the copy process the chromosome set can mutate, such that a' chromosome set is not identical to that of a. The chromosome set of a does not mutate. In this case a' is considered to have fallen prey to some environmental influence and died (see Figure 11.3). It is, however, immediately replaced by a's offspring. Mind, that a and a' can be the same agent.

Figure 11.3: Haploid reproduction were C_1 replaces C'_1

Algorithm 12: Reproduction of agent a with chromosome sets C_{a1} (haploid) and C_{a2} (diploid, haplo-diploid).

```
1  if collected resources overstep threshold then
2      if haploid reproduction then
3          chose a random agent a'
4          a' replaces its chromosome set with C_a1
5          a' mutates
6      if diploid reproduction then
7          chose two random agents a' and a''
8          chose two random numbers ρ, σ ∈ {1, 2}
9          replace chromosome sets of a'' with C_aρ and C_a'σ
10         replace C_a'σ of a' with C_a(ρ mod 2)+1
11         a' and a'' recombine their chromosomes into two new sets
12         a' and a'' mutate
13     if haplo-diploid reproduction then
14         foreach chromosome set C_ai do
15             chose a random agent a'
16             chose a random number ρ ∈ {1, 2}
17             a' replaces C_a'ρ with C_ai
18             a' recombines its chromosomes into two new sets
19             a' mutates
20     reset collected resources to 0
```

Diploid Reproduction Diploid agents have two chromosome sets and use one of them actively. An agent a that reaches the reproduction threshold choses a reproductive partner a' and a second agent a'' at random. The chromosome sets of a and a' replace those of a' and a'', such that a' and a'' have one chromosome set of a and a', respectively (see Figure 11.4). The new chromosome sets of a' and a'' are shuffled, i.e. the genes are reallocated to the chromosome sets at random to simulate recombination. Afterwards the chromosome sets mutate and both a' and a'' chose their active chromosome set. This is, w. l. o. g. due to the recombination process, the first chromosome set. The genotype of a remains unchanged throughout the whole procedure. In this scenario a and a' both reproduce and a' and a'' die. The genetic information of a'' is lost.

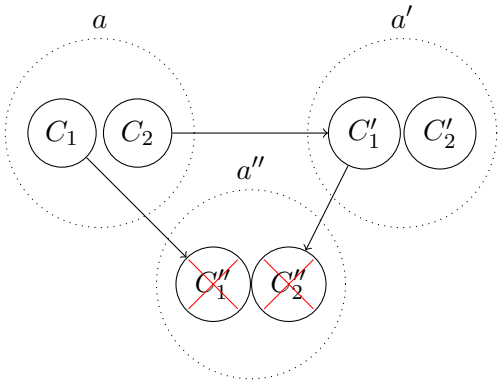


Figure 11.4: Exemplary diploid reproduction were a and a' reproduce. Their offspring replaces a' and a''.

Haplo-Diploid Reproduction Haplo-diploid agents have two two chromosome sets and represent the diploid female individuals. If an agent a reaches the reproduction threshold it choses two other agents a' and a'' at random. Of these two agents, each overwrites one of their chromosome sets (see Figure 11.5), shuffles the new and old chromosome sets, and mutates them. In this case a sends drones to a' and a'' who reproduce and are replaced by their own offspring.

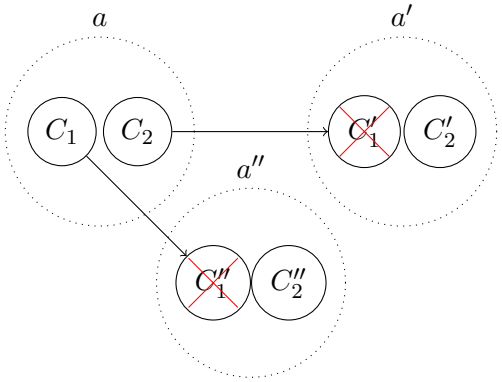


Figure 11.5: Exemplary haplo-diploid reproduction were a reproduces with a' and a''. Their offspring replaces a' and a''.

In all three variants of reproduction death is a random event. Instead of eliminating the weakest member of the swarm, a random agent is chosen and some of its genetic material leaves the gene pool. However, the best agents collect more resources and can reproduce more frequently. This is similar to actual natural selection where those with sufficient resources reproduce but death typically strikes at random.

11.6.2 Mutation

The values for the traits *power*, *exploration time*, and the weights for the weighted ranking scheme can mutate. The mechanism behind these mutations is inspired from the mutation of microsatellites in DNA (for an overview see Ellegren (2004)). Microsatellites are repeats of short sequences (1-4 bp). Their length are very specific on an individual basis and are generally used to infer close relatives and identify individuals in research and forensics. Copying such a repetitive structure is an error prone work for the DNA-polymerases. Since, microsatellites are non-coding regions the copy errors are not as critical as copy errors in other parts of the DNA can be. Pieces of the repeated segment are thus added or lost in copying process. Quite like microsatellites, the length of the values for the traits *power*, *exploration time*, and the weights for the weighted ranking scheme fluctuate from one generation to the next. This fluctuation is achieved by choosing a random number ρ from a normal (Gaussian) distribution $N(0, \sigma)$, where 0 is the mean value and σ the variance. The current value of the trait is then increased by *rho*. Note, that by varying σ the strength of the mutations is set. The only additional rule for this mutation scheme is that values can only decrease as low as zero. Formally, we write

$$x' = \max(0, x + \rho), \text{ with } \rho \in N(0, \sigma)$$

where x is the old value of a parameter and x' the new value.

12 || Empiric Analysis

Diversity and efficiency

This chapter gives a description of the performed experiments and their results. Multiple scenarios are tested to analyze the agents behavior and the evolutionary processes under different environmental conditions.

12.1 Experimental Setting

The initial field properties $r(f)$, $a(f)$, and $t(f)$ are randomly. The respective random values origin either from an exponential distribution with $\lambda = 1$ or from an even distribution depending on the applied scenario (see Table 12.1). The values are chosen such that $r(f) \in [0, 1]$, $a(f) \in \{2, \ldots, 10\}$, $t(f) \in \{1, \ldots, 10\}$. When a field is regenerated its values are not reset with new random values but swapped with the values of another field. This procedure maintains the respective distributions of the parameters in the environment. Otherwise, the distributions of a parameter could change. In Scenario 1, for example, the resource values $r(f)$ are exponentially distributed. The agents have a preference to work on the few fields with high a high abundance of resources. Fields with high $r(f)$, thus have a higher chance of regeneration

Table 12.1: Different applied scenarios with the distributions that are used to set the field values and the regeneration rate.

	$r(f)$	$a(f)$	$t(f)$	regeneration rate
1	exponential	even	even	0
2	even	exponential	even	0
3	even	even	exponential	0
4	even	even	even	100

Table 12.2: Fixed parameters chosen for all experiments.

Parameter	Definition	Values
	number of performed runs	50
	arena size	50×50
	number of simulation turns	50 000
k	objectives	4
n	agents	100
	rotation probability	0.25
	reproduction threshold	1
	memory size	5
	initial exploration time	10 sim. turns
pow	initial power	10
w	initial weights	random in $[0, 500]$
σ	mutation strength	1

than fields with low $r(f)$. If the values of these fields would be replaced by random values from an exponential distribution, the frequency of fields with low resources increases, while fields with more resources would get rarer and rarer.

This phenomenon is avoided by swapping the values of a regenerating field with those of an existing other field. Thus, distributions are maintained while the swarm is confronted with an environment of constant change and deprecating information. An additional *regeneration rate* gives the number of fields that are picked at random in each simulation turn to be regenerated.

At first we analyze the four scenarios in Section 12.2. In Section 12.3 we inspect the adaptability of the swarm in dynamic environments. Table 12.2 gives an overview of the fixed parameter values applied in all experiments. Note, that the weights for the objectives are in $[0, 500]$ instead of $[0, 1]$. The weights are normalized for the ranking process but otherwise have these rather large values for their mutation process. Initially, the weights have a very large variance and it is interesting to see how the different reproduction schemes handle this situation.

The initial power and exploration time, though prone to the same mutation strength, have much smaller initial values. Here it is more interesting to see how fast the values for the power increase or decrease. All agents start with the same initial power and exploration time in each experiment.

In the second type of experiments we analyze dynamic environments. A dynamic environment is defined by a periodic rotation of Scenarios 1 to 4.

Three dynamic scenarios are defined by the time period between a change of scenario, i.e. 100, 1000, or 10 000 simulation turns.

All results are based on measurements performed every 100 simulation turns. These values include means and variances of the parameters prone to evolutionary mechanisms and the amount of collected resources collected in the previous 100 simulation turns. Further, the objective values of all fields in all agents field memory at the time of measurement are used to infer correlations between the objectives.

12.2 Analysis of the Scenarios

Objective Correlations The four scenarios provide different environments for the agents. The distributions of the values for the parameters $r(f)$, $t(f)$, and $a(f)$ induce different strengths in the objective correlations (see Table 12.3). While the correlation between the required time and the available resources is low or not significant, there are stronger correlations between the required time and the age of information. This phenomenon can be explained by the circumstance that new information is derived from fields that were recently visited by members of the team and that are still in close proximity. The older the information is the larger can be the distance between the team and the respective field.

Table 12.3: Correlations between objectives in different scenarios denoted by numbers in black rectangles; the objectives are (1) required time, (2) available resources, (3) required agents, and (4) the age of information as measured in the haploid system. All presented values are statistically significant Pearson correlations with p-values < 0.01.

1	2	3	4	**2**	2	3	4	**3**	2	3	4	**4**	2	3	4
1	–	-0.15	0.37	1	0.09	-0.11	0.35	1	0.13	0.01	0.38	1	0.02	-0.07	0.30
2		0.11	0.17	2		0.09	0.18	2		0.10	0.20	2		0.02	-0.01
3			0.12	3			0.12	3			0.13	3			0.12

In most scenarios the correlation between the required time and the probability to have a sufficient team size is negative. This is due to the fact that a longer distance to the target field gives a team more possibilities to recruit more agents on their way there. This negative correlation subsides if the actual processing time $t(f)$ of a field has a larger impact on the required time than the distance the team has to travel because the values of $t(f)$ are

exponentially distributed, i.e. in Scenario 3.

The higher regeneration rate in Scenario 4 turns the otherwise positive correlation between required time and required agents into a slightly negative correlation. In Scenarios 1 to 3 the active choosing process of the agents inflicts a positive correlation – they prefer fields where both values are good over those where only one value is good. In Scenario 4, however, the agents have different priorities as will become apparent in the further analysis.

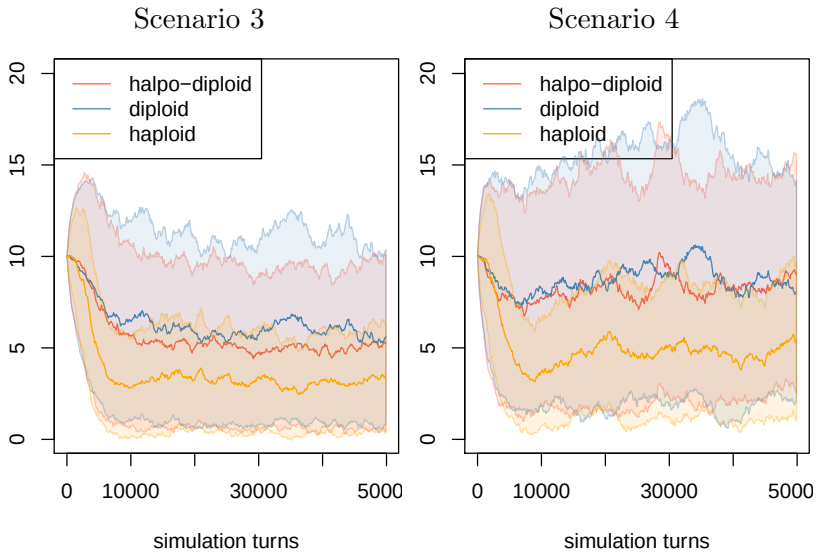


Figure 12.1: Exploration time in different scenarios with average values and standard deviations.

Exploration Time The exploration time gives the minimal time an agent spends to explore its environment and detect beneficial fields in its surroundings. This parameter is prone to evolutionary processes and its average values and standard deviations can be seen for two scenarios in Figure 12.1. Initially, all agents have an exploration time of 10 simulation turns. The following development is similar in all four scenarios. The average exploration time decreases during the first 10 000 simulation turns then remains on this lower level. Note, that the haploid reproduction mechanism results in a lower diversity, i.e. lower standard deviation, implying a more straight forward approach where the swarm rapidly decides on an exploration time that is lower (below 5 simulation turns) than the exploration times favored by the haplo-diploid and diploid approach. Scenario 4 appears to require

slightly longer exploration times than the other three scenarios that show very similar developments and are represented by Scenario 3 in Figure 12.1.

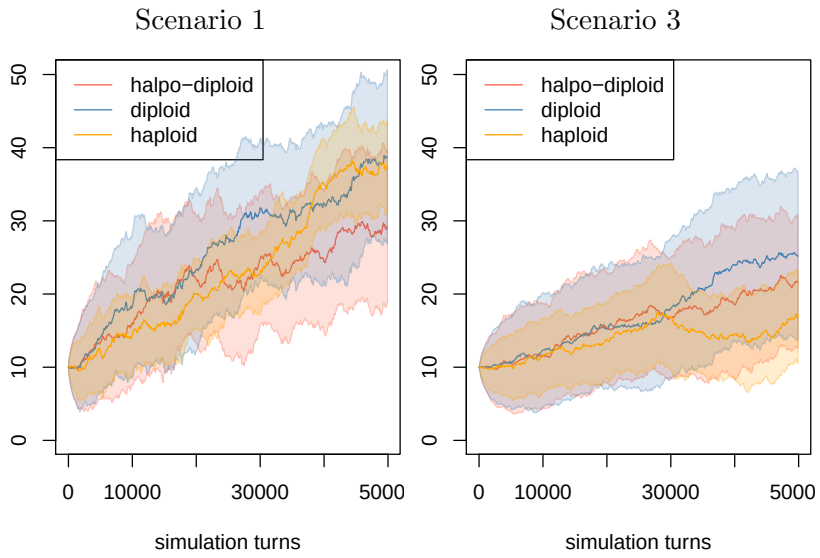


Figure 12.2: Power in different scenarios with average values and standard deviations.

Power Another parameter whose values are prone to evolutionary processes is the power. The power of the agents is used to decide which team of agents wins in case of a conflict and can assimilate agents of the other team into itself. Figure 12.2 shows how the average values of the power parameter develop in the different scenarios with the three reproduction mechanisms.

Initially, all agents have a power of ten. In course of the simulation run the values for the power increase, as could be suspected from this parameters properties. More powerful agents can force other agents to join their team and have a higher chance of having a sufficiently sized team when they reach their target field. This is obviously beneficial.

The diploid and haplo-diploid reproduction mechanisms have a higher standard deviation than the haploid reproduction mechanism. All three mechanism increase the values of the power with similar speed. Scenario 1 appears to give a higher selective pressure on the agents power, that rises to average values of up to 40, while the values grow slower in the other scenarios (from 20 to 30) as represented by Scenario 3 in Figure 12.2.

Ranking schemes In Scenarios 1, 2, and 3 and the Dominance Depth ranking scheme dominates all systems regardless of their reproduction mechanisms by making up more than half of all ranking scheme chromosomes. The Weighted ranking is often a second followed by the WL ranking scheme. The Points ranking scheme goes always extinct in systems with haplo-diploid and haploid reproduction in the first three scenarios. In diploid reproduction systems they have a marginally higher survival rate but are, on average over all runs, never more frequent then the Dominance Depth or Weighted ranking scheme. After 25 000 simulation turns the frequencies of the ranking schemes barely fluctuate and remain on their respective levels. An exemplary time series can be seen in Figure 12.3 which shows the abundances of the four different ranking schemes during the simulations with diploid reproduction in Scenario 2.

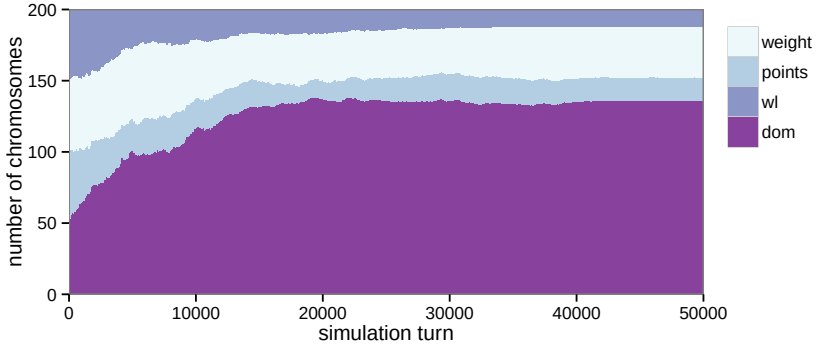


Figure 12.3: Distribution of the different ranking schemes in the haplo-diploid system for Scenario 2; Dominance Depth is abbreviated by 'dom'.

Scenario 4 induces a very different development (see Figure 12.4). None of the ranking schemes goes always extinct and no ranking scheme is clearly the most dominant in this scenario. However, in the following analysis it will become more apparent that these systems are rarely mixed but typically dominated by one the ranking schemes.

12.3 Diversity and Adaptability Analysis

A special focus in this analysis lies within the maintained diversity and efficiency of the different evolutionary approaches. As measure of diversity we applied the Shannon-Weaver index (Shannon, 2001), which is commonly

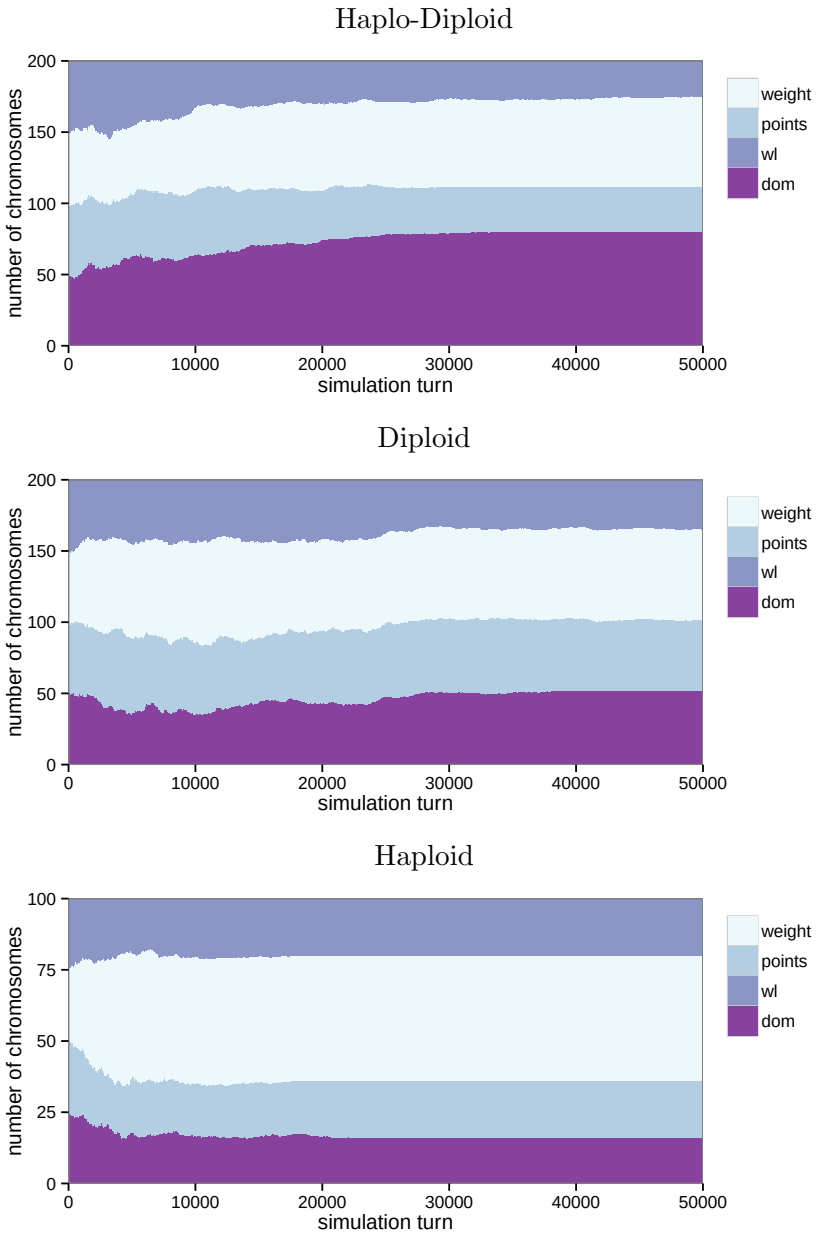


Figure 12.4: Distribution of the different ranking schemes in the different systems for Scenario 4; Dominance Depth is abbreviated by 'dom'.

applied in population studies, on the frequencies of the ranking schemes.

$$H' = \sum_{i=1}^{R} p_i \ln p_i,$$

where p_i is the proportional abundance of class $i \in \{1, \ldots, R\}$ with a total of R classes. The initial Shannon-Weaver index of the different ranking schemes is

$$H' = \sum_{i=1}^{R} \frac{1}{4} \ln \frac{1}{4} \approx 1.386,$$

Figure 12.5 shows the development of the Shannon-Weaver index during the simulations of the scenarios. Once the index is zero the system only contains one ranking scheme. For all scenarios the index decreases very fast during the first 10 000 simulation turns. Scenario 1 shows the steepest decrease in diversity compared to the other scenarios, while Scenario 4 shows the slowest decrease in diversity. Only in scenarios 3 and 4 there are simulation runs that maintain multiple ranking schemes up to the end of the simulation runs. The haploid systems have the lowest diversity and the fastest decrease. The diploid and haplo-diploid systems show no considerable difference.

Figure 12.6 shows the development of the diversity in the dynamic settings. A change of scenario every 1000 or 10 000 simulation turns results in a decrease of diversity and promotes systems based on only one ranking scheme. With more frequent changes of scenarios the process of complete loss of diversity is prolonged. However, besides the haplo-diploid system in the dynamic setting with a change of scenario every 1000 simulation turns, none of the different reproduction mechanism maintain more the one ranking scheme in the system till the end of the simulation. The haploid system is the fastest to converge and settle on a single ranking scheme, while the diploid and haplo-diploid system are slower.

Figure 12.7 gives the efficiency of the different reproduction mechanisms in the different dynamic systems. A Friedman-Nemenyi post-hoc analysis implies that the only significant differences in the performance of the compared dynamic systems (p-value < 0.01) lies in the system where the scenarios changes every 10 000 simulation turns and the diploid system is significantly better than the haploid system. A comparison of the efficiencies of the systems in the different scenarios shows that in Scenario 1 the diploid system has a significantly higher efficiency than the haploid system and in Scenario 3 both the diploid and haplo-diploid system are significantly better than the haploid system.

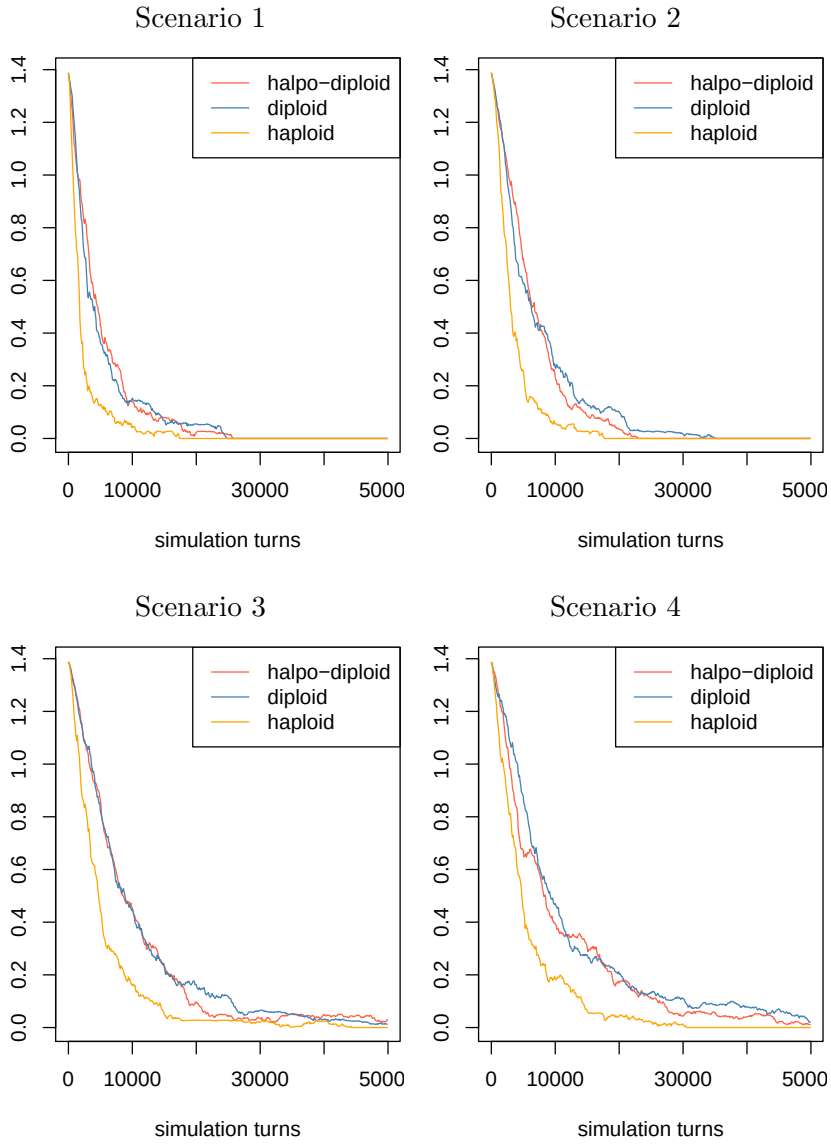


Figure 12.5: Diversity in different scenarios with average values and standard deviations.

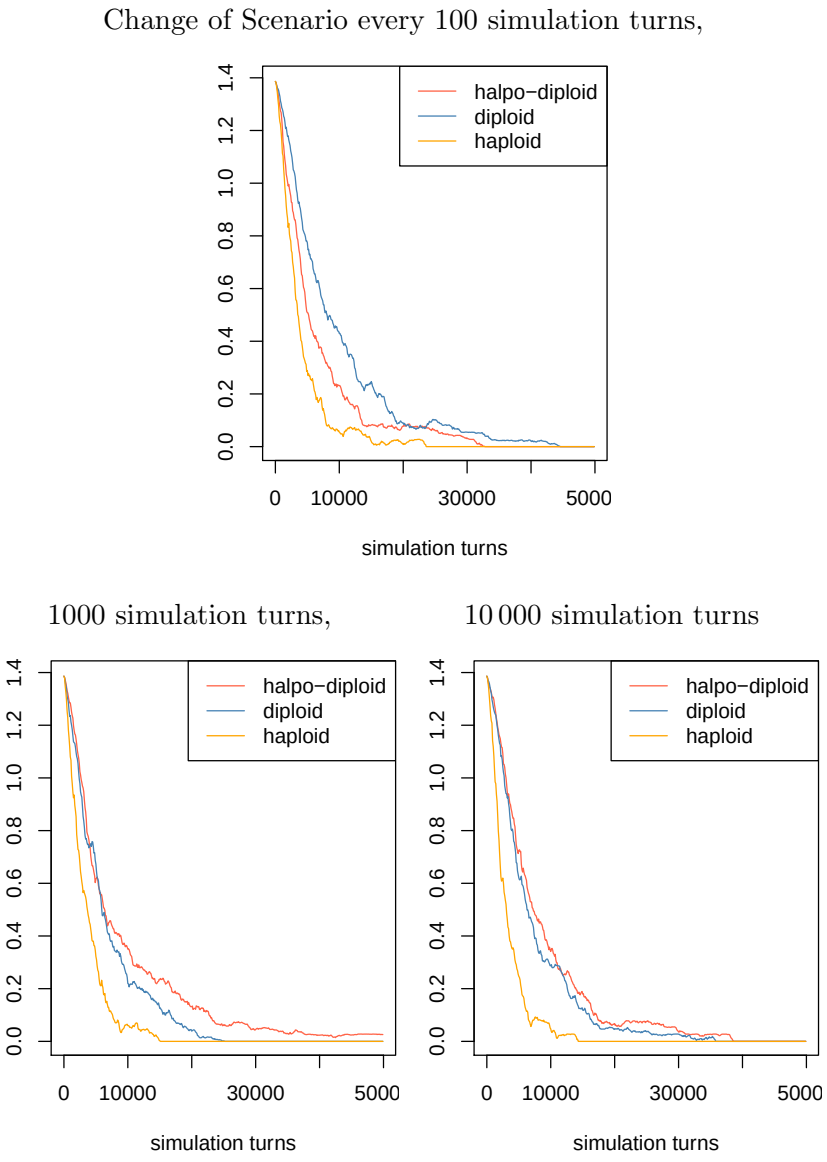


Figure 12.6: Diversity in different dynamic systems with average values and standard deviations.

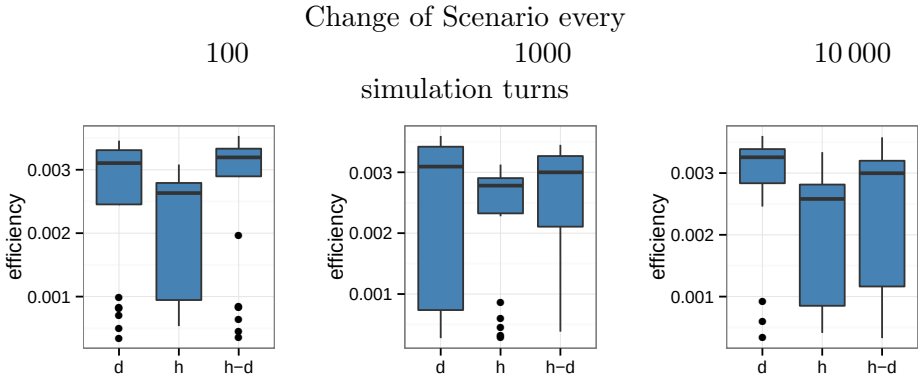


Figure 12.7: Efficiency in different dynamic systems with average values and standard deviations.

12.4 Conclusion

In this part we compared three different reproduction approaches: haploid, diploid, and haplo-diploid reproduction. We applied them on a multi-agent system with agents solving multi-objective multi- agent tasks and evolved some of the most crucial parameters of the system. The quality of the evolutionary mechanisms was inferred with a focus on the ranking schemes the agents applied to chose which task to prioritize at which time.

While, the diploid and haplo-diploid approaches could maintain a higher genetic diversity in the system they also showed tendencies to perform better than the haploid system. However, in the final state of the systems in nearly all experimental runs only one ranking scheme 'survived'. The fact that these final dominant ranking schemes are incoherently different ones in the different simulation runs implies that a mixed system might be more beneficial, nevertheless.

Part V

Synthesis

13 Summary and Outlook

Swarms occur in nature as well as in artificial scenarios and pose interesting challenges for science in both fields. The similarities between natural and artificial swarms incite researchers to seek inspiration in the natural phenomena of swarm organization and behavior for the development of respective traits in artificial swarms. However, the differences between both kinds of swarms lead most engineers to rather apply centralized concepts for the organization of swarms instead of decentralized and distributed concepts as those observed by biologists in nature. And thus, in robotics, telecommunication, or traffic, to name a few examples, distributed systems are typically controlled by centralized mechanisms. In order to further promote decentralized strategies and show their beneficial qualities, I studied the means of cooperation, team formation, synergistic effects in heterogeneous, reconfigurable multi-agent systems with a special focus on the difficulty of solving multi-objective problems.

Further, I developed methods that show how simple distributed strategies can induce adaptability in dynamic environments and how specific ranking schemes can increase the performance of metaheuristics and multi-agent systems.

13.1 Team Formation

In heterogeneous multi-agent systems where agents are generally different from each other concerning their capabilities or behavior, cooperation can be essential. Especially, in light of multi-agent tasks, multiple agents are required to cooperate by forming teams and processing tasks together. While it is shown in this study that the team formation process can be NP-hard, in some cases it is more important to solely focus on the size of the teams. Considering scenarios with reconfigurable agents this is especially important. If there are costs in the organization of teams, that rise with the team size, the agents need to form teams of sufficient size but remaining as small as

possible. If the optimal team size is unknown and the agents are forced to infer it by trial-and-error, it suffices to provide them with a very simple decision mechanism to control the team size. The system performance is significantly increased if agents are able to dynamically infer the optimal team size by checking whether a team member is partly idle at the moment.

When the teams are confronted with multi-objective multi-agent tasks they need to infer which of the available tasks should be processed first. In order to rank these tasks the agents need to apply ranking relations for multi-objective problems.

13.2 Ranking Schemes

Multi-objective ranking schemes are better studied in the field of multi-objective optimization than in multi-agent systems. In the mechanisms of population-based metaheuristics the problem of ranking multi-objective solutions to determine a few 'best' solutions among them, is very similar to the problem of the agents solving multi-objective multi-agent tasks. In a thorough analysis I compared several ranking schemes and their performance considering the population-based Ant Colony Optimization algorithm and a genetic algorithm solving two different combinatorial problems. Ranking schemes counting objectives where one solution is better, equal, or worse than another solution, outperform ranking schemes solely based on the concept of Pareto dominance. It also became quite apparent that the correlation between the objectives had a significant influence on the algorithms and the obtained solutions. Different types of problems favored different ranking schemes, a notion that implies hybrid algorithms applying multiple ranking schemes could be more beneficial.

13.3 Evolutionary Mechanisms

While the metaheuristics apply one ranking scheme at a time it is more interesting to see how heterogeneous multi-agent system perform if they can use multiple ranking schemes simultaneously. Instead of forcing a specific distribution of the ranking schemes onto the system, I applied and compared three different evolutionary mechanisms to let the agents evolve their most crucial parameter values and the type of ranking scheme they apply. This dynamic system showed that different scenarios favored different ranking schemes. Although, a diploid and haplo-diploid approach maintained a higher diversity in the system than a compared haploid approach, typically

the systems evolved into a homogeneous system where all agents applied the same ranking scheme. The diploid and haplo-diploid systems showed a tendency to have a higher efficiency than the haploid system.

In the course of this study, I was able to show that simple mechanisms often suffice in multi-agent systems to acquire a specific adaptive behavior and that swarm members should not only work together but also learn from each other.

13.4 Outlook

The problem of cooperatively solving multi-objective tasks in multi-agent systems has many interesting aspects. This study gave more insight into some of them. However, to convince the community of engineers confronted with this kind of problems to rather apply distributed and decentralized instead of centralized methods, far more research needs to be conducted. This includes the study of specific applied scenarios that can profoundly benefit from decentralized mechanisms. Current advances in the fields of robotics show tendencies towards autonomy by providing machines with sensors and the means to base their actions on information gathered from their environment in real time, e.g. autonomous cars. However, most research conducted in the field of autonomy develops and studies mechanisms for single units and their mobility. While (crash free) mobility is typically a prerequisite for mobile autonomous units of any kind, communication and cooperation between multiple of such units will be necessary for many problems arising after robots learned to walk, fly, swim, drive, or crawl. Some of these problems can be solved by sharing the right bit of information, at other times, it might be beneficial if a robot updated software or even changed some of its hardware according to the settings of another robot. Whether these autonomous units are transport drones, cars, mars rovers, satellites, or just smart phones – there are heterogeneous swarms that can benefit in many scenarios from decentralized mechanisms and cooperative strategies.

Several animal societies apply such strategies successfully but the precise mechanisms are often not fully understood. We need to further interdisciplinary investigations to grasp the complex nature of biological swarms and infer how these natural mechanisms can be beneficial in robot societies.

A possible scenario is the autonomous distribution of tasks among heterogeneous agents. For example, satellites of different missions and built could cooperatively distribute tasks among them that might arise after natural disasters and require immediate response. As satellites are typically

not geostationary, their positions relative to earth surface are constantly changing but can be adapted by using their limited amount of fuel. The three-dimensional flight maneuvers that are required to perform a formation flight are similar to the flocking behavior of birds.

Another scenario, that could benefit from distributed self-organized control is the routing of autonomous cars to avoid traffic jams during rush hour, where we can draw inspiration from the behavior of ants that maintain a fluent transport network inside their colonies despite their great numbers and limited space. Such a decentralized control requires no central node and could be more secure from third parties or other unforeseeable events that require a fast and adaptive reaction.

Any task based on the exploration of unmapped territory can benefit of the mechanisms social insects use for foraging or nest site location. Among such scenarios are the exploration of caves, deep see or deep space, but also collapsed buildings. Swarms of robots can explore faster but need to communicate with each other to perform an efficient search. Further, artificial swarms can benefit from mechanisms inspired by the food transfer in social insects, i.e. trophallaxis, to share common resources, e.g. their battery power, to increase their performance.

These are just three examples of scenarios that could benefit from self-organized decentralized mechanisms like those found in nature. The potential of nature inspired swarm mechanisms needs to be further investigated to keep in pace with the current engineering process towards autonomous systems.

Bibliography

Abramson, M. (2008). Coalition formation of cognitive agents. In *Proc. of the Second International Conference on Computational Cultural Dynamics*, page pp. 7.

Alberola, J. M., Julian, V., and Garcia-Fornes, A. (2014). Challenges for adaptation in agent societies. *Knowledge and Information Systems*, 38(1):1–34.

Angus, D. (2007). Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In *IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making*, pages 333–340.

Angus, D. and Woodward, C. (2009). Multiple objective ant colony optimisation. *Swarm Intelligence*, 3:69–85.

Bentley, P. and Wakefield, J. (1998). Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms. In Chawdhry, P., Roy, R., and Pant, R., editors, *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer.

Bölöni, L., Khan, M. A., and Turgut, D. (2007). Agent-based coalition formation in disaster response applications. *International Journal of Intelligent Control and Systems*, 12:107–117.

Bongard, J. C. (2013). Evolutionary robotics. *Communications of the ACM*, 56(8):74–83.

Bowers, R. I. and Sevinç, E. (2006). Preserving variability in sexual multiagent systems with diploidy and dominance. In *Engineering Societies in the Agents World VI*, pages 184–202. Springer.

Brockhoff, D., Saxena, D., Deb, K., and Zitzler, E. (2008). On handling a large number of objectives a posteriori and during optimization. In

Knowles, J., Corne, D., Deb, K., and Chair, D., editors, *Multiobjective problem solving from nature*, Natural Computing Series, pages 377–403. Springer.

Calabretta, R., Galbiati, R., Nolfi, S., and Parisi, D. (1996). Two is better than one: a diploid genotype for neural networks. *Neural Processing Letters*, 4(3):149–155.

Carlier, J. (1978). Ordonnancements-a contraintes disjonctives. *RAIRO*, 12(4):333–351.

Coello Coello, C. (2009). Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. *Frontiers of Computer Science in China*, 3(1):18–30.

Coello Coello, C., Pulido, G., and Montes, E. (2005). Current and future research trends in evolutionary multiobjective optimization. In *Information Processing with Evolutionary Algorithms*, Advanced Information and Knowledge Processing, pages 213–231. Springer.

Coello Coello, C. A. and Salazar Lechuga, M. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. In *Congress on Evolutionary Computation*, volume 2, pages 1051–1056.

Corne, D. W. and Knowles, J. D. (2007). Techniques for highly multiobjective optimisation: some nondominated points are better than others. In *Genetic and Evolutionary Computation Conference*, pages 773–780.

Crozier, R. (1977). Evolutionary genetics of the hymenoptera. *Annual review of entomology*, 22(1):263–288.

Cvetkovic, D. and Parmee, I. (2002). Preferences and their application in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 6(1):42–57.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN*, number 1917 in LNCS, pages 849–858. Springer.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

Doerner, K. F., Hartl, R. F., and Reimann, M. (2001). Are COMPETants more competent for problem solving? - the case of a routing and scheduling problem. In *Genetic and Evolutionary Computation Conference*, page 802.

Dorigo, M., Caro, G., and Gambardella, L. (1999). Ant algorithms for discrete optimization. *Artificial life*, 5(2):137–172.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge.

Drechsler, N., Drechsler, R., and Becker, B. (2001). Multi-objective optimisation based on relation favour. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 154–166.

Ellegren, H. (2004). Microsatellites: simple sequences with complex evolution. *Nature reviews genetics*, 5(6):435–445.

Farina, M. and Amato, P. (2004). A fuzzy definition of "optimality" for many-criteria optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 34(3):315–326.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *ICGA*, pages 416–423. Morgan Kaufmann.

Fonseca, C. M. and Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16.

Garey, M. R., Johnson, D. S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129.

Garrett, D., Dasgupta, D., Vannucci, J., and Simien, J. (2007). Applying hybrid multiobjective evolutionary algorithms to the sailor assignment problem. In Jain, L. C., Palade, V., and Srinivasan, D., editors, *Advances in Evolutionary Computing for System Design*, volume 66 of *Studies in Computational Intelligence*, pages 269–230. Springer.

Garza-Fabre, M., Toscano Pulido, G., and Coello Coello, C. (2010). Alternative fitness assignment methods for many-objective optimization problems. In Collet, P., Monmarché, N., Legrand, P., Schoenauer, M., and

Lutton, E., editors, *Artifical Evolution*, number 5975 in LNCS, pages 146–157. Springer.

Garza-Fabre, M., Toscano Pulido, G., and Coello Coello, C. A. (2009). Ranking methods for many-objective optimization. In *Proc. Advances in Artificial Intelligence (MICAI)*, volume 5845 of *LNCS*, pages 633–645.

Gaston, M. E. and DesJardins, M. (2008). The effect of network structure on dynamic team formation in multi-agent systems. *Computational Intelligence*, 24(2):122–157.

Gerkey, B. P. and Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23:939 – 954.

Goel, T., Vaidyanathan, R., Haftka, R. T., Shyy, W., Queipo, N. V., and Tucker, K. (2007). Response surface approximation of Pareto optimal front in multi-objective optimization. *Comput. Method. Appl. M.*, 196(4):879–893.

Goldberg, D. E. and Smith, R. E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *ICGA*, pages 59–68.

Guntsch, M. and Middendorf, M. (2002). A population based approach for ACO. In Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., and Raidl, G. R., editors, *Applications of Evolutionary Computing*, number 2279 in LNCS, pages 72–81. Springer.

Guntsch, M. and Middendorf, M. (2003). Solving multi-objective permutation problems with population based ACO. In Fonseca, C. M., Fleming, P. J., Zitzler, E., Thiele, L., and Deb, K., editors, *Evolutionary Multi-Criterion Optimization*, number 2636 in LNCS, pages 464–478. Springer.

Heller, J. (1960). Some numerical experiments for an $m \times j$ flow-shop and its decision-theoretical aspects. *Operations Research*, 8(2):178–184.

Herrera, F. and Lozano, M. (1996). Adaptation of genetic algorithm parameters based on fuzzy logic controllers. *Genetic Algorithms and Soft Computing*, 8:95–125.

Hölldobler, B. (1990). *The ants*. Harvard University Press.

Hu, X. and Eberhart, R. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Congress on Evolutionary Computation*, volume 2, pages 1677–1681.

Iredi, S., Merkle, D., and Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 359–372.

Ishibuchi, H., Akedo, N., and Nojima, Y. (2013a). A study on the specification of a scalarizing function in MOEA/D for many objective knapsack problems. In *Learning and Intelligent Optimization Conference*, number 7997 in LNCS, pages 231–246. Springer.

Ishibuchi, H., Akedo, N., Ohyanagi, H., and Nojima, Y. (2011). Behavior of EMO algorithms on many-objective optimization problems with correlated objectives. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1465–1472. IEEE.

Ishibuchi, H., Yamane, M., and Nojima, Y. (2013b). Effects of duplicated objectives in many-objective optimization problems on the search behavior of hypervolume-based evolutionary algorithms. In *Computational Intelligence in Multi-Criteria Decision-Making (MCDM), 2013 IEEE Symposium on*, pages 25–32. IEEE.

Jakob, W., Gorges-Schleuter, M., and Blume, C. (1992). Application of genetic algorithms to task planning and learning. In *Parallel Problem Solving from Nature*, pages 293–302. Elsevier.

Jaszkiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71.

Khalouzadeh, L., Nematbakesh, N., and Zamanifar, K. (2010). A decentralized coalition formation algorithm among homogeneous agents. *Journal of Theoretical and Applied Information Technology*, 22:35 – 42.

Knowles, J. and Corne, D. (2003). Instance generators and test suites for the multiobjective quadratic assignment problem. In Fonseca, C. M., Fleming, P. J., Zitzler, E., Thiele, L., and Deb, K., editors, *Evolutionary Multicriterion Optimization*, number 2632 in LNCS, pages 295–310. Springer.

Knowles, J. D. and Corne, D. (2002). Towards landscape analyses to inform the design of hybrid local search for the multiobjective quadratic assignment problem. *HIS*, 87:271–279.

Kukkonen, S. and Deb, K. (2006). A fast and effective method for pruning of non-dominated solutions in many-objective problems. In *Parallel Problem Solving from Nature*, volume 4193 of *LNCS*, pages 553–562.

Kukkonen, S. and Lampinen, J. (2007). Ranking-dominance and many-objective optimization. In *IEEE Congress on Evolutionary Computation*, pages 3983–3990.

Kutanoglu, E. and Wu, S. D. (2007). Coalitions in coordinated multi-agent production scheduling: A computational study. *Journal of Manufacturing Systems*, 26:12 – 21.

Lau, H. C. and Zhang, L. (2003). Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 346–350. IEEE.

Laumanns, M., Thiele, L., Deb, K., , and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282.

Leguizamón, G. and Coello Coello, C. A. (2011). Multi-objective ant colony optimization: A taxonomy and review of approaches. In *Integration of Swarm Intelligence and Artificial Neural Network*, chapter 3, pages 67–94. World Scientific, Singapore.

Leung, K.-S., Duan, Q.-H., Xu, Z.-B., and Wong, C. (2001). A new model of simulated evolutionary computation-convergence analysis and specifications. *Evolutionary Computation, IEEE Transactions on*, 5(1):3–16.

Li, C. and Sycara, K. (2004). A stable and efficient scheme for task allocation via agent coalition formation. In *Algorithms for Cooperative Systems*, page pp. 20. World Scientific.

Lichocki, P., Wischmann, S., Keller, L., and Floreano, D. (2013). Evolving team compositions by agent swapping. *IEEE Transactions on Evolutionary Computation*, 17(2):282–298.

Liefooghe, A., Paquete, L., Simões, M., and Figueira, J. (2011). Connectedness and local search for bicriteria knapsack problems. In Merz, P. and Hao, J.-K., editors, *Evolutionary Computation in Combinatorial Optimization*, number 6622 in LNCS, pages 48–59. Springer.

López-Ibánez, M., Paquete, L., and Stützle, T. (2004). On the design of ACO for the biobjective quadratic assignment problem. In Dorigo, M., Birattari, M., Blum, C., Gambardella, L., Mondada, F., and Stützle, T., editors, *Ant Colony Optimization and Swarm Intelligence*, number 3172 in LNCS, pages 214–225. Springer.

López-Ibánez, M., Paquete, L., and Stützle, T. (2006). Hybrid population-based algorithms for the bi-objective quadratic assignment problem. *Journal of Mathematical Modelling and Algorithms*, 5(1):111–137.

Maneeratana, K., Boonlong, K., and Chaiyaratana, N. (2006). Compressed-objective genetic algorithm. In *Parallel Problem Solving from Nature*, volume 4193 of *LNCS*, pages 473–482. Springer.

Mauldin, M. L. (1984). Maintaining diversity in genetic search. In *AAAI*, pages 247–250.

Merkle, D. and Middendorf, M. (2014). Swarm intelligence. In Burke, E. K. and Kendall, G., editors, *Search Methodologies*, pages 213–242. Springer US.

Michael, R. G. and David, S. J. (1979). Computers and intractability: a guide to the theory of NP-completeness. *WH Freeman & Co., San Francisco*.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge.

Moritz, R. L. and Middendorf, M. (2013). Self-organized cooperation between agents that have to solve resource collection tasks. In *Proc. IEEE Swarm Intelligence Symposium*, page pp. 12.

Moritz, R. L. and Middendorf, M. (2014a). Decentralized and dynamic group formation of reconfigurable agents. *Memetic Computing*, pages 1–15.

Moritz, R. L. and Middendorf, M. (2014b). Self-adaptable group formation of reconfigurable agents in dynamic environments. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, pages 287–301. Springer.

Moritz, R. L., Reich, E., Bernt, M., and Middendorf, M. (2014a). The influence of correlated objectives on different types of P-ACO algorithms. In Blum, C. and Ochoa, G., editors, *Evolutionary Computation in Combinatorial Optimisation*, volume 8600 of *Lecture Notes in Computer Science*, pages 230–241. Springer Berlin Heidelberg.

Moritz, R. L., Reich, E., Schwarz, M., Bernt, M., and Middendorf, M. (2014b). Refined ranking relations for selection of solutions in multi objective metaheuristics. *European Journal of Operational Research*.

Moritz, R. L. V., Reich, E., Schwarz, M., Bernt, M., and Middendorf, M. (2013). Refined ranking relations for multi objective optimization and application to P-ACO. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 65–72.

Mostaghim, S. and Schmeck, H. (2008). Distance based ranking in many-objective particle swarm optimization. In *Parallel Problem Solving from Nature*, volume 5199 of *LNCS*, pages 753–762. Springer.

Murata, T. and Taki, A. (2010). Examination of the performance of objective reduction using correlation-based weighted-sum for many objective knapsack problems. In *Hybrid Intelligent Systems (HIS), 10th International Conference on*, pages 175–180. IEEE.

Oliveira, S. M., Hussin, M. S., Stützle, T., Roli, A., and Dorigo, M. (2011). A detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 13–14.

Ou, J. and Prasanna, V. K. (2010). *Energy efficient hardware-software co-synthesis using reconfigurable hardware*. CRC Press.

Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100.

Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434.

Paquete, L. and Stützle, T. (2006). A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *Eur. J. Oper. Res.*, 169(3):943–959.

Pieter, J. and de Jong, E. D. (2004). Evolutionary multi-agent systems. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 872–881. Springer.

Potts, J. C., Giddens, T. D., and Yadav, S. B. (1994). The development and evaluation of an improved genetic algorithm based on migration and artificial selection. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(1):73–86.

Procaccia, A. D. and Rosenschein, J. S. (2006). The communication complexity of coalition formation among autonomous agents. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 505 – 512.

Rahwan, T. (2007). *Algorithms for Coalition Formation in Multi Agent Systems*. PhD thesis, University of Southhampton.

Recchia, T., Chung, J., and Pochiraju, K. (2014). Performance of heterogeneous robot teams with personality adjusted learning. *Biologically Inspired Cognitive Architectures*, 7(70):87–97.

Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, 22(1):5–13.

Reinelt, G. (1991). TSPLIB–A traveling salesman problem library. *ORSA Journal on Computing*, 3(4).

Reyes-Sierra, M. and Coello Coello, C. A. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308.

Rullmann, M. and Merker, R. (2011). A cost model for partial dynamic reconfiguration. In *Transactions on High-Performance Embedded Architectures and Compilers IV*, pages 370–390. Springer.

Schaffer, J. D. (1984). *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN.

Schwarz, M. (2011). Rangbestimmung in Pareto-Mengen als Methode zur mehrdimensionalen Optimierung. Diploma thesis (Diplomarbeit), University of Leipzig, Germany.

Seeley, T. D. et al. (1985). *Honeybee ecology: a study of adaptation in social life.* Princeton University Press.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55.

Shehory, O. and Kraus, S. (1996). Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165 – 200.

Shibata, T. and Fukuda, T. (1993). Coordinative behavior by genetic algorithm and fuzzy in evolutionary multi-agent system. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 760–765. IEEE.

Singh, V. K., Husaini, S., and Singh, A. (2010). Self-organizing agent coalitions in distributed multi-agent systems. In *2010 International Conference on Computational Intelligence and Communication Networks*, pages 650 – 655.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Sülflow, A., Drechsler, N., and Drechsler, R. (2007). Robust multi-objective optimization in high dimensional spaces. In *International Conference on Evolutionary Multi-Criterion Optimization*, volume 4403 of *LNCS*, pages 715–726. Springer.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285.

Tovey, C. A. (1984). A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85 – 89.

Verel, S., Liefooghe, A., Jourdan, L., and Dhaenens, C. (2011a). Analyzing the effect of objective correlation on the efficient set of MNK-landscapes. In Coello, C. A., editor, *Learning and Intelligent Optimization*, number 6683 in LNCS, pages 116–130. Springer.

Verel, S., Liefooghe, A., Jourdan, L., and Dhaenens, C. (2011b). Pareto local optima of multiobjective NK-landscapes with correlated objectives. In Merz, P. and Hao, J.-K., editors, *Evolutionary Computation in Combinatorial Optimization*, number 6622 in LNCS, pages 226–237. Springer.

Vig, L. and Adams, J. A. (2006). Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22:637 – 649.

Whitacre, J. M., Rohlfshagen, P., Bender, A., and Yao, X. (2010). The role of degenerate robustness in the evolvability of multi-agent systems in dynamic environments. In *Parallel Problem Solving from Nature, PPSN XI*, pages 284–293. Springer.

Wienke, D., Lucasius, C., and Kateman, G. (1992). Multicriteria target vector optimization of analytical procedures using a genetic algorithm: Part I. theory, numerical simulations and application to atomic emission spectroscopy. *Analytica Chimica Acta*, 265(2):211–225.

Xu, Y., Qu, R., and Li, R. (2013). A simulated annealing based genetic local search algorithm for multi-objective multicast routing problems. *Ann. Oper. Res.*, 206(1):527–555.

Yukiko, Y. and Nobue, A. (1994). A diploid genetic algorithm for preserving population diversity – Pseudo-Meiosis GA. In *Parallel Problem Solving from Nature–PPSN III*, pages 36–45. Springer.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms – a comparative case study. In *Parallel problem solving from naturePPSN V*, pages 292–301. Springer.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

Zou, X., Chen, Y., Liu, M., and Kang, L. (2008). A new evolutionary algorithm for solving many-objective optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(5):1402–1412.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbstständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, 2. März 2015

(Unterschrift)