ALEKSANDER BYRSKI

SOCIO-COGNITIVE METAHEURISTIC COMPUTING

WYDAWNICTWA AGH
KRAKOW 2018

ALEKSANDER BYRSKI

SOCIO-COGNITIVE METAHEURISTIC COMPUTING

WYDAWNICTWA AGH
KRAKOW 2018

Published by AGH University of Science and Technology Press

Reviewers:
assoc. prof. Zuzana Komínková Oplatková, Tomas Bata University in Zlín, Czech Republic
prof. dr hab. Maciej Paszyński, AGH University of Science and Technology, Poland

Author's affiliation:
AGH University of Science and Technology
Faculty of Computer Science, Electronics and Telecommunications
Department of Computer Science
al. A. Mickiewicza 30, 30-059 Krakow, Poland

Desktop publishing: *Aleksander Byrski*

Technical editor: *Joanna Ciągała*

Cover design: *Agata Wajer-Gądecka*

ISBN 978-83-66016-42-2

AGH University of Science and Technology Press
al. A. Mickiewicza 30, 30-059 Krakow, Poland
tel. 12 617 32 28, 12 636 40 38
e-mail: redakcja@wydawnictwoagh.pl
www.wydawnictwa.agh.edu.pl

To the women of my life:
my wife Bogusia
and my daughter Ola

Contents

ALEKSANDER BYRSKI
Socio-cognitive metaheuristic computing

Abstract

Nature-inspired metaheuristics are very popular these days; their creation is usually justified based on the "no free lunch" theorem by Wolpert and MacReady. However, the creation of novel metaheuristics should be realized with care, not only for the sake of creation (cf. Sörensen reports on superficial metaheuristics); in other words, the inspiration should be solid and well-intended (it would be ideal if such methods were formally verified, however this happens very seldom, because of their complexity). In the case of the metaheuristics presented in this monograph, the inspiration comes from the works of Albert Bandura, a renowned Canadian-American psychologist. One of his most important contributions to contemporary science is the theory of social cognitive learning, showing that people do not only learn from their experiences (trial and error) but also by perceiving other people and (fortunately) their trials and errors. This saves a lot of effort, allowing us to utilize the knowledge gathered by perceiving others in order to build humankind's self-knowledge. This inspiration leads to the proposal of a socio-cognitive metaheuristic paradigm consisting of the introduction or enhancement of the cognitive properties of particular metaheuristics. The most important achievements in this area are socio-cognitive Ant Colony Optimization and socio-cognitive Particle Swarm Optimization. The introduction of cognitive features into such computing algorithms allows us to reach better efficiency in solving selected hard benchmark problems. In this work, the above-mentioned novel algorithms are presented along with selected experimental results. Moreover, the socio-cognitive computing paradigm is defined, and the relationship of the selected metaheuristic algorithm to this paradigm is discussed. This metaphor is also considered as a reference for the selected classic and agent-based metaheuristics. These algorithms are identified by relating them to the literature background, and the possibilities of enhancing them with socio-cognitive features are discussed. Certain examples of further research are also identified. This monograph is meant to introduce a novel perspective on the selected metaheuristics, defining the socio-cognitive computing paradigm and providing guidance in this area for readers who are interested in such nature-inspired computing methods.

ALEKSANDER BYRSKI
Socjokognitywne obliczenia metaheurystyczne

Streszczenie

Metaheurystyki inspirowane naturą należą do popularnych obecnie metod rozwiązywania trudnych problemów optymalizacyjnych. Metody obliczeniowe prezentowane w tej monografii czerpią inspiracje z prac Alberta Bandura, znanego kanadyjsko-amerykańskiego psychologa. Do jego najważniejszych sukcesów należy opracowanie teorii socjalno-kognitywnego uczenia się, zgodnie z którą ludzie uczą się nie tylko na podstawie własnych doświadczeń (najczęściej prób i błędów), ale również obserwując innych ludzi, a co za tym idzie – ich próby i błędy. Gromadzenie wiedzy o innych pozwala na budowanie samoświadomości i znacznie ułatwia proces uczenia się. Inspiracje wspomnianą teorią doprowadziły do zaproponowania socjokognitywnego paradygmatu obliczeniowego polegającego na wprowadzeniu lub rozszerzeniu własności kognitywnych poszczególnych metaheurystyk. Najważniejszym osiągnięciem prezentowanym w niniejszej monografii jest opracowanie dwóch algorytmów socjokognitywnych: algorytmu mrówkowego oraz roju cząstek. Rozszerzenie kognitywności agentów będących podstawową jednostką działającą we wspomnianych algorytmach umożliwiło osiągnięcie lepszej skuteczności w rozwiązywaniu wybranych trudnych problemów benchmarkowych. Wspomniane algorytmy zostały zaprezentowane w niniejszej monografii wraz z wybranymi rezultatami eksperymentów. Przedstawiono analizę zaprezentowanych algorytmów w kontekście teorii socjalno-kognitywnego uczenia się. W ten sam sposób przeanalizowano kilkanaście popularnych metaheurystyk, które były oceniane pod względem możliwości wprowadzenia do nich socjokognitywności bądź rozszerzenia jej istniejących elementów. Przedysktuwane zostały również możliwości dalszego rozwoju algorytmów pasujących do proponowanego paradygmatu. Celem niniejszej monografii jest wprowadzenie nowej perspektywy postrzegania wybranych metaheurystyk, propozycja wprowadzenia socjokognitywnego paradygmatu obliczeniowego oraz umożliwienie czytelnikowi zainteresowanemu obliczeniami inspirowanymi naturą poszerzenia wiedzy na ten temat.

Preface

Starting from the early 1970s with the works of John Holland [1] and his proposal of genetic algorithms, the metaheuristic computing era began. His seminal work paved the way for the development of other metaheuristic algorithms, such as Evolution Strategies by Rechenberg and Schwefel [2], Genetic Programming by Koza [3], Memetic Algorithms by Moscato [4], and a huge number of their hybrids and modifications. Nowadays, new metaheuristics are still being proposed and compared with state-of-the-art algorithms, remembering the famous "no free lunch theorem" by Wolpert and MacReady [5] that drives researchers in the pursuit of novel global optimization methods.

Optimization heuristics (particularly these biologically-inspired techniques) have been gaining attention for more than three decades. Such approaches are supposed to be universal, though critics point to the higher computation times and larger complexities of the algorithms. However, when facing difficult problems, it is usually effective to switch from deterministic approaches to stochastic searches and optimization methods [5], which may justify the additional costs.

Thus, difficult problems will always require novel metaheuristics; therefore, the search for new inspirations is always needed and attractive from the scientific point of view. The landscape of metaheuristics is rich, and many of them can be perceived as universal optimization algorithms (see, for example, the works of Michael Vose on genetic algorithm [6]).

Recently, there has been an increase in the synergistic interaction between biological and cognitive systems on one hand and computational systems on the other. A number of metaphors inspired from natural systems (ant colonies, bird flocks, bee swarms, and so on) have become the bases for constructing interesting metaheuristics and new optimization techniques, thereby affecting the field of computing. As long as the metaheuristics are not merely relabeling terms in existing algorithms [7], they can lead to novel approaches that outperform classic metaheuristics.

During the period of 2015–2017, a team consisting of cognitive psychologists and computer scientists worked on a Polish-Belgian joint research project realized

under the agreement on scientific cooperation between the Polish Academy of Sciences and the Wallonia-Brussels Federation of Belgium: "Modeling the emergence of social inequality in multi-agent system". During the work (besides the considerations related to simulation and modeling), a novel approach to the construction of computing algorithms belonging to the so-called swarm intelligence (Ant Colony Optimization (ACO) [8] and Particle Swarm Optimization (PSO) [9]) was proposed. In this way, two novel hybrids were proposed, both utilizing many populations of the individuals (ants or particles) and introducing certain relationships among these "species." As these hybrids utilized inspirations that come from psychology (like perspective taking and the Social Cognitive Theory by Albert Bandura [10]), they were named socio-cognitive metaheuristics. The research lead us to explore the parameters of those algorithms as well as the possibility of automatically adapting the population structure, for example. The proposed metaheuristics turned out to be good tools for dealing with complex multi-dimensional problems, which encouraged us to extend the research (treating them as a starting point).

Later, after rethinking the features of these algorithms, it turned out that starting from the so-called social algorithms (usually based on a society of individuals) with some relationships between them, the proposed approach can be treated as an enhancement of the cognitive features of the individuals (agents) taking part in the computing. This was not only perceived from but also processed in the light of social-cognitive theory, therefore leading to not only the construction of another two metaheuristics but also the proposal of a new metaheuristic-oriented computing paradigm; namely, socio-cognitive computing. This paradigm can be at least partially recognized in many existing computing systems (hybrids of ACO and PSO, Evolutionary Multi-Agent Systems (EMAS), selected memetic computing systems, etc.).

The proposed metaheuristics are well-rooted in psychology-related inspirations (namely, Social Cognitive Theory and perspective taking), and the whole proposed paradigm goes beyond the definition of several metaheuristics. During the process of design and implementation, the seminal work of Sörensen [7] criticizing superficial metaheuristics was one of the reference points; his observations were used as a way to avoid such pitfalls.

This monograph focuses on introducing of the socio-cognitive computing paradigm into the rich nature-inspired computing world, showing its roots in psychology and cognitive sciences and presenting its way of inception starting from social metaheuristics and describing their enhancements towards increasing their cognitivity. The two important factors considered are increasing the cognitivity of both generation- and non-generation-based algorithms (evolutionary-like vs. swarm intelligence). The knowledge gathered is passed between generations in the former case and remains within the generation in the latter one.

This book starts with general deliberations on metaheuristics, describing those that can be classified as "social-metaheuristics" and introducing the reader to the substance and motivation for the presented perspective on nature-inspired computing methods (see Chapter 1). The next chapter is devoted to introducing the socio-cognitive metaheuristic class, being an enhancement of social-algorithms by introducing (or enhancing) their cognitive abilities. This chapter is rooted in psychological inspirations, mostly the Social Cognitive Theory by Bandura (see Chapter 2).

After giving the reference point, the first socio-cognitive metaheuristics (namely, Socio-cognitive Ant Colony Optimization [11] and Socio-cognitive Particle Swarm Optimization [12]) are described, along with selected experimental results and references to the published literature (see Chapter 3). The chapter is summarized with deliberations on the relationships between the introduced metaheuristics and the Social Cognitive Theory features. The agency features are of special interest, giving the proper background for the formation of sound inspiration in the construction of new metaheuristics.

Next, selected classic metaheuristics either related to socio-cognitive ideas or having the strong potential for building such algorithms are identified (see Chapter 4). Moreover, the hybrids of EMAS [13] that fit very well into this class are sketched out. In the end, the relevance of these algorithm to the elements of social cognitive learning is discussed.

In the end, the book is summarized, and an assessment of the relevance of all of the algorithms discussed to the Social Cognitive Theory elements is given, showing the reader the future possibilities of introducing new metaheuristics related to the socio-cognitive computing paradigm proposed in this monograph.

⋆ ⋆ ⋆

Hereby, I would like to thank all the people without whom this work may not have been written. Let me first thank Marek Kisiel-Dorohinicki and Wojciech Turek, my closest everyday collaborators and good friends who realize endeavors with me that are impossible to be done in such an extent – in such a short time. My thanks also go to the colleagues who have helped lay the cornerstones of the socio-cognitive computing paradigm: Dana Samson, Henryk Bukowski, Tom Lenaerts, Bipin Indurkhya, Ann Nowé, Mateusz Sękara, Michał Kowalski, Ewelina Świderska, Jakub Łasisz, Iwan Bugajski, and Piotr Listkiewicz. I would also like to thank all my excellent current and past M.Sc. and Ph.D. students who always contribute to an efficacious, dynamic research team, helping my colleagues and me in the realization of our common ideas. I am thankful for all of my colleagues working at the Department of Computer Science, AGH University of Science and Technology, for the friendly atmosphere that encouraged me to conduct my research.

I would like to offer special thanks to Professor Edward Nawarecki and Professor Krzysztof Cetnarowicz. Although they are no longer with us, they continue to inspire by their example and dedication to the research and support given to every person – including myself – they served over the courses of their careers. When asked, they never declined to give anyone a helping hand. They will always remain in our memory.

I would like to express my heartfelt gratitude to my parents Maria and Mieczysław for raising me, making my education possible, and supporting me in every possible way right from the start, never letting me down.

Finally, my greatest gratitude is to be expressed to my wife Bogumiła and daughter Aleksandra for creating a family full of partnership and love, making my home a place to which I long to return whenever I am away.

Aleksander Byrski
Kraków, February–September 2018

1. Social metaheuristics

Tackling difficult search problems calls for the application of unconventional methods. This necessity is imposed by having little or no knowledge of the intrinsic features of the problem, topology of the search space, etc. In such cases, approximate techniques like metaheuristics become the methods of last resort.

Having a plethora of metaheuristics to choose from, those population-based (as opposed to single solution-oriented) methods seem to be a very good choice, both at the algorithmic and implementation levels; as they process more than one solution at a time, they can evade the local extrema easier than single-solution approaches. Moreover, it is usually easy to implement them efficiently using ubiquitous parallel systems such as multi-core processors (see, e.g. [14]), graphical processing units (see, e.g. [15]), clouds (see, e.g. [16], and grids (see, e.g. [17]).

Metaheuristics are very often inspired by natural phenomena like evolution or swarm movements. From the point of view of this monograph, the most interesting metaheuristics are social ones (i.e., population-based ones) that realize a search by creating and controlling a number of individuals. Certain interactions arise between them; in other words, mimicking the social phenomena like communication or competition. Based on such algorithms as swarm-intelligence or agent-based computing, the Social Cognitive Theory will be utilized later in the monograph in order to propose a new perspective on such algorithms; namely, socio-cognitive systems. However, we must first discuss social metaheuristics in this chapter, building a reference for further deliberations.

In the course of this chapter, the basic information about optimization with metaheuristics is given. Later, the social inspirations and agency-related features are discussed, as agency is usually an inherent feature of social metaheuristics. Next, selected particular social computing algorithms are described in detail; namely, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), evolutionary algorithms, the Estimation of Distribution Algorithm (EDA), and the Evolutionary Agent-based System (EMAS). Finally, the possibilities of enhancing the discussed algorithms are sketched-out.

1.1. Metaheuristics as methods of last resort

Popular real-world search problems usually consist of finding a set of parameters of a certain model according to specific criteria [18]. These criteria are usually expressed as certain functions of the mentioned parameters.

Definition 1.1.1. *Optimization problem consists in finding all global minimizers* $\arg\min\{\Phi(x)\}, x \in \mathcal{D}$ *of a static objective function (called very often goal function, quality function and in evolutionary metaheuristics "fitness"):* $\Phi : \mathcal{D} \to [0, M]$, *where* $\mathcal{D} \subset \mathbb{R}^N$, $N \in \mathbb{N}$ *stands for the admissible sufficiently regular set of solutions,* $\mathbb{R}_+ \ni M < +\infty$.

Heuristic (*gr. heuresis: to find*) search methods provide "good-enough" solutions without concern as to whether they may be proven to be correct or optimal [19]. It may be said that these methods sacrifice precision, quality, accuracy, and execution time in favor of being able to deal with with difficult problems. Very often they are being referred to as methods of last resort (see, e.g., [20]).

One of the simplest heuristics is a greedy search algorithm that randomly generates solutions and accepts only those that fulfill the predefined criteria. More-sophisticated examples of heuristics are Monte Carlo [21] search methods. Heuristics may also be hybridized in order to improve the effectiveness of other search methods (as A* algorithm or alpha-beta pruning in a tree-search [22]).

Using common sense and remembering Ockham's razor, one should apply complex search techniques solely to difficult problems. Therefore, this study does not include a number of popular tasks such as the optimization of convex functions or linear programming (with their reliable techniques) [23].

In [19], Michalewicz and Fogel propose several reasons why a problem may be considered *difficult* (e.g., the number of possible solutions is too great to perform an exhaustive search for the best answer; the problem is so complex that, in order to provide any feasible answer, a simplified model must be used; the evaluation function describing the quality of the solution is noisy or varies with time, so many solutions are sought).

Certain search problems that fall into the description given above are perceived to be difficult *per se* because their domains are very hard or even impossible to be described and explored using conventional analytical methods (see, e.g., combinatorial optimization problems [24]). The setting of such problems is sometimes called a "black-box scenario" [25].

According to the Definition 1.1.1, let us assume that a meta-algorithm exists that covers all randomized search heuristics working on finite search space $\mathcal{D}$. The functions to be optimized are all functions that may be described as $f : \mathcal{D} \to [0, M]$. Now, the "black-box" scenario is defined as follows [25, Algorithm 1]:

1. Choose some probability distribution p on $\mathcal{D}$ and produce a random search point $x_1 \in \mathcal{D}$, according to p. Compute $f(x_1)$.

2. In step t, stop if the stopping criterion is fulfilled. Otherwise, depending on up-to-date candidate solutions $I(t) = (x_1, f(x_1), \ldots, x_{t-1}, f(x_{t-1}))$, choose some probability distribution $p_{I(t)}$ on $\mathcal{D}$ and produce a random search point $x_t \in \mathcal{D}$ according to $p_{I(t)}$. Compute $f(x_t)$.

If a certain problem can be solved with this scenario only (in a reasonable amount of time), it can be called a "black-box problem". In other words, this notion encompasses all problems whose candidate solutions may be sampled randomly, but there are no means of deriving them on the basis of the existing knowledge of search space $\mathcal{D}$. To sum up, such problems may only be solved using general-purpose algorithms (i.e., heuristics), taking into consideration little or no information from a problem domain.

Complex approaches that may be used to solve such difficult problems (see, e.g., [26]) somehow relieve the user of a deep understanding of the intrinsic relationships among the different features of the problem itself, instead constituting "clever" and "general" computing systems. No one can claim that the Holy Grail of search techniques has been found thinking about these universal techniques, as the well-known "no free lunch theorem" must be kept in mind. Wolpert and MacReady prove that all search and optimization techniques are statistically identical when compared for all problems described in a certain domain [5, 27]. So, there is still much to be done to adapt the parameters of these techniques to solve certain problems.

Without providing such details as the particular problem, accurate definition of a search space, or operators, a general definition of a heuristic algorithm is called a *metaheuristic*. In this way, a simple heuristic algorithm such as a greedy search may be defined, for example, as an "iterative local improvement of a solution based on random sampling" without going into the details of the nature of the random sampling or explored space. Therefore, metaheuristics are usually defined as general-purpose nature-inspired search algorithms [28].

Blum and Roli [29] provide a summary of the metaheuristic properties: they are approximate and usually non-deterministic; their goal is to efficiently explore the search space, seeking (sub-)optimal solutions; they "guide" the search process; they may incorporate mechanisms dedicated to avoiding being trapped in the local extrema; they are not problem-specific; they can utilize search experience (usually implemented as some kind of memory mechanism) to guide the search.

Another class of heuristic algorithms is the so-called hyper-heuristics, which utilizes more-advanced mechanisms (e.g., from the domain of machine learning) to optimize the parameters of the search or even select an appropriate lower-level search method [30].

A simple but effective classification of metaheuristics (cf. [31, 32]) that gives sufficient insight into a problem for the purpose of this monograph is as follows:

- Single-solution metaheuristics work on a single solution to a problem, seeking to improve it in some way. The examples are local search methods (such as local-search, greedy heuristic, tabu search, or simulated annealing) [33].
- Population-based metaheuristics explicitly work with a population of solutions and put them together in order to generate new solutions. Some examples are evolutionary algorithms [34], immunological algorithms [35], particle swarm optimization [9], ant colony optimization [8], memetic algorithms [4], and other similar techniques.

These techniques are usually nature-inspired and follow the different phenomena observed in biology, sociology, culture, or physics (for example).

1.2. Social inspirations and agency in metaheuristics

The sociological definition of a *social system* describes a patterned network of relationships that make a whole existing between certain individuals, groups, and institutions [36]. This describes a formal structure where individuals having roles and statuses, forming a stable group. An individual may belong to many social systems; examples of such systems include communities, cities, nations, corporations, etc. The organization and definition of the groups within a social system depend on various characteristics such as location, socioeconomic status, race, religion, and societal function (for example) [37].

Agent-oriented systems are good examples of computing systems that are socially inspired. Their basic idea is to decompose a task to be solved into smaller parts (subtasks) in order to solve them separately and later integrate the solution (cf. distributed problem solving [38]), all while maintaining a significant level of autonomy.

Intelligent and autonomous software agents have been widely applied in various domains, such as power system management [39], flood forecasting [40], business process management [41], intersection management [42], or solving difficult optimization problems [43], to mention a few. The key to understanding the concept of a multi-agent system (MAS) is its intelligent interaction (like coordination, cooperation, or negotiation). Thus, multi-agent systems are ideally suited for representing problems that have many solving methods, involve many perspectives, and/or may be solved by many entities [44]. This is why one of the major application areas of multi-agent systems is large-scale computing [45, 46].

According to one of the most popular definitions proposed by Wooldridge, an agent is a computer system situated in an environment that is capable of undertaking

independent autonomous actions in this environment in order to fulfill tasks on behalf of its user [47]. Autonomy is perceived as one of the most crucial features of the agent.

It seems that the definition of the agent introduces a new name for some existing well-known programming techniques. At the same time, intelligent agents that are parts of complex systems bring a new quality, crossing the borders of previously existing computer systems and enhancing the notion of an object or process with additional important features; e.g., [47] helping other agents fulfill their goals:

- reactivity: agents may perceive their environment and react to changes in that environment,
- pro-activity: agents may perform tasks based on their own initiatives,
- social ability: agents are able to interact with other agents (and with users).

It is noteworthy that fulfilling a goal becomes a *raison d'être* for an agent. This is the most important determining factor in the agent's undertaking of actions in an environment.

The notion of an agent system is based directly on the notion of an agent. Generally speaking, an agent system is a system in which a key abstraction is that of an agent. Therefore, a multi-agent system is one that consists of a group of agents that interact with one another [48, 49].

Agents act within their environment, and different groups of agents may perform their tasks in different parts of their environments. In particular, their activities may overlap. As an example, the possibility of communication between agents that are "close" in the environment may be given (of course, their closeness strongly depends on the notion of a neighborhood, if such a notion was implemented), or direct interaction with the environment (e.g., only one agent-robot at a time may pass through a door) [44].

The main features of multi-agent systems are as follows: distribution, decentralization, interaction, organization, situatedness, openness, emergency, and adaptation.

Although the notion of an agent was originally used to construct systems consisting of truly autonomous beings connected with dedicated protocols, embedded in an environment (e.g., a network, or more generally, the Internet), and capable of retrieving and processing information, the idea of agency spans to other fields of computer science (in particular, to the domain of computing).

Based on the already autonomous perceptions of certain individuals of metaheuristics, one should note the apparent construction of social structures and social interaction between the individuals. For example, the individuals are connected into groups, demes, and species (like in the island model of an evolutionary algorithm [50] or in co-evolutionary algorithms [51]). Moreover, the individuals interact among

themselves, cooperating towards reaching a common goal (at the same time focusing on their own individual goals, c.f. agency [47]), as in ant colony optimization or particle swarm optimization.

One of two basic classes of metaheuristics tackled in this monograph is a social metaheuristic being a sub-class of a population-based metaheuristic. The following definition roots in population-based metaheuristics and will be further enhanced towards social-cognition later in this book:

Definition 1.2.1. *A social metaheuristic is a computing algorithm that belongs to a class of population-based metaheuristics that conducts searches in a way that mimics the social phenomena occurring in real populations.*

Two firm examples of such an algorithm are Ant Colony Optimization and Particle Swarm Optimization. Moreover, algorithms like the Estimation of Distribution Algorithm or Differential Evolution can be perceived as ones inspired by the processing and exchange of information according to the memetic systems defined by Pablo Moscato. Moreover, another good example of a social metaheuristic is EMAS; surely, this list is not exhaustive. A counter-example might be Scatter Search – a very successful population-based metaheuristic algorithm, yet not inspired by real-world populations nor phenomena.

Later in this book, we will focus on finding a way to introduce or enhance the cognitivity of the above-mentioned algorithms. Now, let us now focus on several social metaheuristics in order to build a starting point for further deliberations.

1.3. Ant Colony Optimization

The Ant System algorithm introduced in 1991 by Marco Dorigo to be applied to solving graph problems is a progenitor of all ACO techniques [52]. The classic ACO algorithm is an iterative process during which a certain number of agents (ants) create a solution step by step [53, 54]. The main goal of the ants is to traverse the graph, finding the path with the lowest cost (usually the shortest distance, but it can be the least fuel consumption or other factors).

Each step of any particular ant consists of choosing a subsequent component of the solution (that is, a graph edge) with a certain probability. This decision may be affected by the interaction among the ants based on the levels of *pheromones* that may be deposited into the environment (on the edges of the graph) by some ants and perceived by other ants (representing the so-called attractiveness for the observed edges in order to choose the next step). This interaction is guided by stigmergic relationships (communication among individuals by means of an environment instead of by direct contact) according to the rules proposed in [52]) (see Fig. 1.1). Each

ant can be treated as an agent; the algorithm can then be easily enhanced by adding more autonomy to each agent. The communication via stigmergy relieves the user from complex and technologically inefficient point-to-point and broadcast communications – instead, the pheromone table becomes a single point of contact for the system. The computation is finished when a feasible solution is found due to the cooperative efforts of all of the ants.

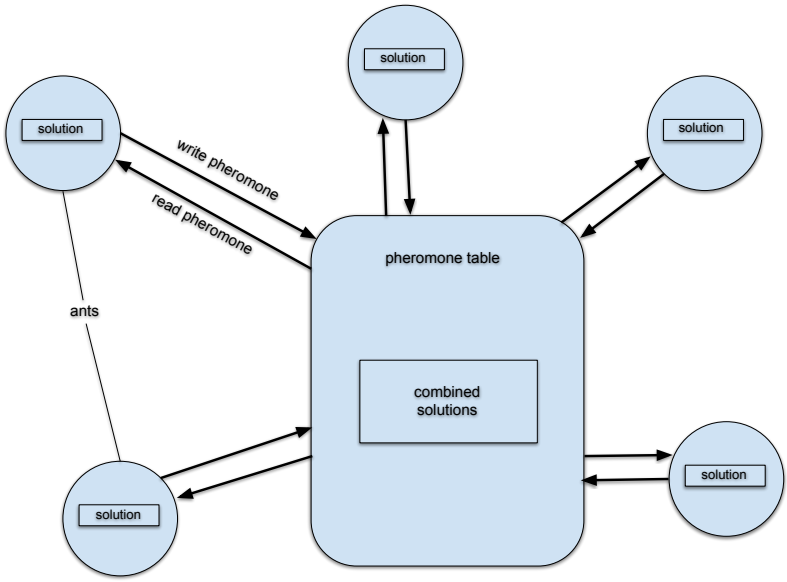

Figure 1.1. Relationships among ants in classic ACO

The ants constitute a society of agents (their basic design is fully reactive; however, with a little effort, more sophisticated agent features can be implemented into each individual ant) exchanging information by stigmergic relationships, utilizing the dedicated pheromone table (implementation of the environment); therefore, the inclusion of ACO-type algorithms into social metaheuristics is natural (cf. Definition 1.2.1).

In the classic ACO algorithm [55], the individuals (ants) are deployed in a graph consisting of a set of vertices ($V = i, j, \ldots; i, j \in \mathbb{N}$) and a set of edges ($E$). It is to note that each edge is associated with a certain distance. Each ant receives a randomly chosen starting graph node and searches for a cycle by moving between the nodes (always choosing the next one, never coming back). While choosing which node to visit next, the ant must evaluate the attractiveness factor for all of the possible edges that can be followed from the present node.

In the ACO algorithm's every iteration, each agent (ant) creates a complete solution. Each ant starts in some initial vertex and travels through the graph according to a probabilistic decision rule. The rule assigns a probability of choosing a move from component i to j based on the edges' costs and pheromone trails. In the basic ACO algorithm – Ant System (AS, [55]), the rule is defined for ant k in iteration t as follows:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^\alpha \cdot [\eta_{ik}(t)]^\beta} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

where $\tau_{ij}(t)$ is the amount of a pheromone on edge (i, j) and $\eta_{ij}(t)$ is the visibility (or attractiveness) of an edge (which can be defined as an inverse distance between cities in the case of TSP). $allowed_k$ is a set of possible transitions for ant k in its current state. Finally, α and β are parameters that express the relative priority of the pheromones as well as their attractiveness. After the solution-construction process, the ant updates the pheromones on the solution path. In the classic AS version, an update is performed at the end of a single iteration according to the following formula:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \sum_{k=1}^{m} \Delta \tau_{ij}^k \tag{1.2}$$

where $\rho \in [0, 1)$ is a pheromone persistence coefficient and m is the number of ants. The pheromone update value for each ant is defined as follows:

$$\Delta \tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if } k\text{th ant uses edge } (i, j) \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

Since the first description of the ACO algorithm, a lot of variants have been developed. Below, we provide the descriptions of several basic algorithms and their most important modifications.

A potential drawback of the AS algorithm already noticed in [55] is that, during the search process, the full information about the best solutions found so far is lost. A possible improvement proposed in the same paper was to introduce an additional number of e "elitist" ants that, during each iteration, report as if they have traversed the best solution found so far by the algorithm. This modification of the AS algorithm is called Elitist Ant System (EAS).

Rank–based Ant System (ASRank, [56]) is an extension of EAS. It uses the elitist ants, but it also modifies the way regular ants update the pheromone trails. The solutions found in each iteration are ordered by their cost, and only the m-best

solutions are taken into account in the pheromone-update process. The update value is weighed according to the solution's rank (μ). Usually, $m = e - 1$.

Another variant of the ACO algorithm, Max-Min Ant System (MMAS, [57]), introduces a few changes to the AS algorithm:

- the values of the pheromones are limited to interval $[\tau_{min}, \tau_{max}]$
- the initial values of the pheromones are set to τ_{max}
- after each iteration, only one ant updates the pheromone values – either the best from the iteration or the best found so far by the algorithm.

The performance of the MMAS algorithm was further improved by the *pheromone trail smoothing* mechanism, which increases the pheromone trails proportionally to their difference to τ_{max} (when the algorithm is very close to convergence). This mechanism can be effectively applied to other elitist ant systems.

Ant Colony System (ACS, [58]) differs from the previous variations because of three aspects: the extended state transition rule, the global updating rule (applied only to edges from the global best solution), and the additional local updating rule.

The state transition rule favors short edges and edges with a large amount of pheromones. In order to make the search more directed, only the global best solution is reinforced by applying the standard formula after each iteration in the *Global Updating* stage. After each state transition, each ant applies the *Local Update*; this changes the pheromone to a smaller extent than in the case of *Global Updating*.

1.4. Particle Swarm Optimization

Particle swarm optimization [9] is an iterative algorithm commonly used for the mathematical optimization of certain problems. PSO belongs to a set of algorithms called metaheuristics – these algorithms do not guarantee finding the most optimum solution but can yield a solution close to it. This fact makes PSO suitable for solving problems where there is either no known algorithm or the execution of an exact algorithm consumes too much time or too many resources. Moreover, PSO does not require that the function being optimized be differentiable, regular, or constant over time.

Particle swarm optimization was originally proposed for simulating social behavior and was used for simulating the group movement of schools of fish, flocks of birds, and so on. However, the algorithm was also found to be useful for performing a mathematical optimization after some simplification. The social background of the algorithm and the direct inspiration from the behavior of flying animals interacting among themselves makes it a very good example of social metaheuristics (cf. Definition 1.2.1).

In PSO, the particles roam around the search space with a certain velocity and direction and adapt their direction according to simple predefined rules based on their history and the history of their neighbors (see Fig. 1.2).

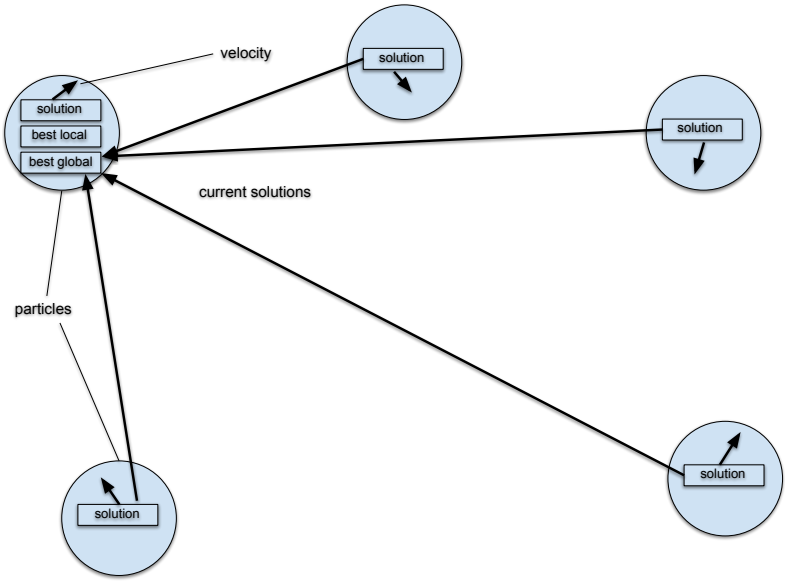

Figure 1.2. Relationships among particles in classic PSO

Although the notion of agency is quite visible in this algorithm, mostly sequential implementations of PSO exist (similar to ACO). This relieves the user from the problem of implementation of dedicated broadcast communication among the agents.

In the basic particle swarm optimization [9] implementation, the potential solutions are located in a subspace of n-dimensional Euclidean space R^n limited in each dimension (usually an n-dimensional hypercube). The search space $D = \mathbb{R}^n$ is a domain of optimized quality function $f : \mathbb{R}^n \to \mathbb{R}$, $n \in \mathbb{N}$.

A particle is a candidate solution described by three n-dimensional vectors: position $X = (x_1, x_2, \ldots, x_n)$; velocity $V = (v_1, v_2, \ldots, v_n)$; and best-known position $A_p = (x_1, x_2, \ldots, x_n)$. A swarm is a set of m particles. The swarm is associated with n-dimensional vector $A_s = (x_1, x_2, \ldots, x_n)$, which is the swarm's best-known position (the solution with the currently highest quality), $n \in \mathbb{N}$.

The execution of the algorithm begins by initializing the start values. Each particle P belonging to swarm S and in neighborhood N is initialized with the following values:

- position X of particle P is initialized with a random vector belonging to search space D,

- the best-known position is initialized with current particle's position: $A_p := X$
- velocity V of particle P is initialized with a random vector belonging to search space D,
- the swarm's best position is updated: $if\ f(A_p) < f(A_s)\ then\ A_s := A_p$.

Once all of the particles are initialized and uniformly distributed in the search space, the main part of the algorithm starts executing. During each iteration of the algorithm, the steps in the Pseudocode 1 are executed until a termination criterion is met.

Pseudocode 1 Pseudocode of Particle Swarm Optimization algorithm

for `each particle P in swarm S` **do**
  `update particle's position:`
    $X \leftarrow X + V$
  `update particle's velocity:`
    $V \leftarrow a(A_s - X) + c(A_p - X) + \omega V; a, b, c, \omega \in [0, 1]$
    where ω is the inertia factor
  `update global best positions:`
    $if f(A_p) < f(A_s)\ then\ A_s \leftarrow A_p$
end for

The most common termination criteria for the particle swarm optimization are as follows:

- the number of executed iterations reaches a specified value,
- the swarm's best position exceeds a specified value,
- the algorithm finds a global optimum,
- the swarm's best positions in two subsequent iterations are the same.

Perceiving PSO as a social metaheuristic is quite natural (as it is in the case of ACO). This is clearly caused by the common classification encompassing ACO and PSO, calling them swarm intelligence algorithms. Thus, PSO employs a number of individuals that share their knowledge, observing their individual and global best-so-far results. The global knowledge (global best solution) must be shared by all of the individuals; however, in the basic implementation of PSO, the individuals are given the necessary information without any particular attention given to the "agency" of the algorithm (although PSO is clearly agent-related) as the individuals undertake the decision of how to move in the solution space on their own. A certain level of cognition can be assigned to PSO as the ants improve their solutions based on the observation of other particles – although not individuals ones, the knowledge about the global best solution is provided by the system and each particle can utilize this information when modifying its trajectory and velocity.

1.5. Evolutionary computing

The origins of the evolutionary algorithms may be found in the 19th-century works of Gregor Mendel – the first person to state the baselines of heredity from parents to offspring – who demonstrated that the inheritance of certain traits in pea plants follows particular patterns (now referred to as the laws of Mendelian inheritance). Later in 1859, Charles Darwin formulated his Theory of Evolution [59]. These theories inspired several independent groups of researchers to create different schools of evolutionary algorithms during the second half of the 20th century:

- John Holland [1, 60] modeled the process of evolution of individuals constructed with the use of binary code in 1975. He was the first researcher to utilize the predefined operators used to change genotypes, which were similar to crossover and mutation. He found that the average fitness of this population tends to increase. A similar algorithm under the name of **genetic algorithm** was later popularized by David Goldberg [34].
- Ingo Rechenberg [2] and Hans-Paul Schwefel [61] researched the optimization of mechanical devices by permuting randomly-generated solutions. Having observed certain similarities to the biological evolution process in their approach, they invented methods known by the name of **evolution strategies** [62].
- Lawrence Fogel [63] tried to model the process of the inception of artificial intelligence upon an approach based on self-organization. He evolved finite automata aimed at understanding a predefined language [64]. This approach was called **evolutionary programming**; after further adaptation, it became a popular technique in optimization [63].
- John Koza tried to work on the automatic generation of computer programs using evolutionary algorithms. His research focused on evolving LISP program structures using a tree-based encoding, which is natural for this language. In this way, a technique called **genetic programming** was devised [3].

  A detailed survey of evolutionary techniques can be found in [65].

Generally speaking, evolutionary metaheuristics process populations of individuals representing exemplary solutions to certain problems. The general goal of this process is to find an optimal solution (or solutions) to a problem by maximizing a predefined goal function (usually called the "fitness function") that is used to evaluate the individuals belonging to the processed population.

It is noteworthy that the individuals contain a genotype that is an encoded solution to a given problem. The genotype consists of genes describing different features of a solution. Different representations are applied to different problems; e.g., optimization problems require binary- or real-value-based representation, while combinatorial problems usually require permutation representations.

In evolutionary algorithms, the population is processed in steps called "generations." One generation consists of several phases that introduce changes into the population. These phases, which are executed in the presented order, are as follows [66, 34, 67]:

1. Initialization: randomly generating individuals to fulfill predefined constraints.
2. Evaluation: computing the value of the fitness function for all individuals.
3. Selection: determining a so-called mating pool comprised of individuals who will become the parents of the next population.
4. Crossover: producing the offspring of parents belonging to a mating pool.
5. Mutation: introducing additional random changes into the newly generated individuals.

Evolutionary algorithms utilize a social metaphor per-se by modeling a group of individuals that cooperate in an attempt to find new solutions based on the genetic repertoire present. The search is guided in an intelligent way (i.e., they are not completely random, as in Monte Carlo methods) by the variation operators (see Figure 1.3).

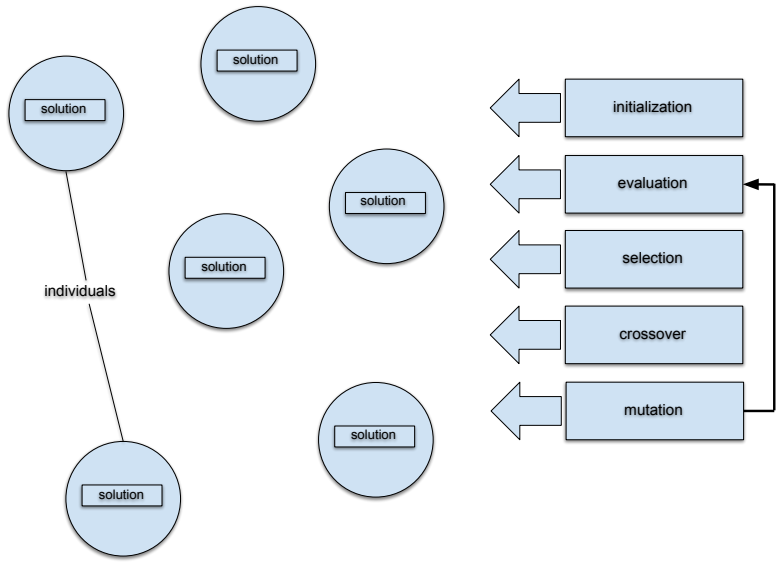

Figure 1.3. Scheme of classic Evolutionary Algorithm

Social inspirations are easily visible in EAs; in particular, the selection operator works on a society, ordering it, choosing better or worse, and generally comparing the individuals and constructing the mating pool according to some selection scheme (cf. Definition 1.2.1).

1.6. Evolutionary agent-based computing

Bäck et al. [68] and others introduced evolutionary algorithms as common metaheuristics useful for solving difficult search and optimization problems. Although they have been widely applied to diverse problems, efficiency remains their main drawback. Besides, researchers induced by the *no-free-lunch* theorem are constantly investigating novel metaheuristics, often hybridizing diverse approaches. One of the concepts that turned out to be useful in terms of evolutionary computing is agency.

There is no official common definition of an agent; however, according to the most widely accepted one, an agent is an autonomous pseudo-intelligent computer system situated in some environment that is able to act reactively (i.e., by reacting to changes in the environment), pro-actively (i.e., by undertaking autonomous actions), or based on interactions with other agents (e.g., cooperation, communication, negotiation, etc.) in order to fulfill a common goal [47].

A Multi-Agent System (MAS) is an open, distributed, and decentralized system in which the main part consists of a group of agents that interact in their common environment [48, 49, 69]. A concept of agent-based systems is derived from the need to decompose a main task into smaller parts to be solved by distributed individuals.

In 1996, Krzysztof Cetnarowicz proposed the concept of an Evolutionary Multi-Agent System (EMAS) [70]. The basis of this agent-based metaheuristic are agents – entities that bear appearances of intelligence and are able to make decisions autonomously. Following the idea of population decomposition and evolution decentralization, the main problem is decomposed into sub-tasks, each of which is entrusted to an agent. One of the most-important features of EMAS is the lack of global control – agents co-evolve independently of any superior management. Another remarkable advantage of EMAS over classic-based algorithms is the parallel ontogenesis – agents may die, reproduce, or act at the same time.

A schematic illustration of EMAS is presented in Figure 1.4. Each agent possesses a genotype that represents an exemplary solution of the tackled problem. Agents are situated on evolutionary islands, where they interact with each other. It is noteworthy that such a structure corresponds to the distributed character of computations and facilitates algorithm parallelization and implementation in a distributed environment.

The quality of each agent's solution is expressed by its *energy* – a non-renewable resource acquired or lost during its lifetime. Energy is exchanged between agents in the process of so-called *meetings* that take place between two agents. An agent with a higher fitness (i.e., its genotype represents a solution of higher quality) acquires some portion of another agent's energy. The mechanism of selection is based on the level of agent energy – agents with less energy are more likely to be removed from the system, as they are assumed to represent poor-quality solutions.

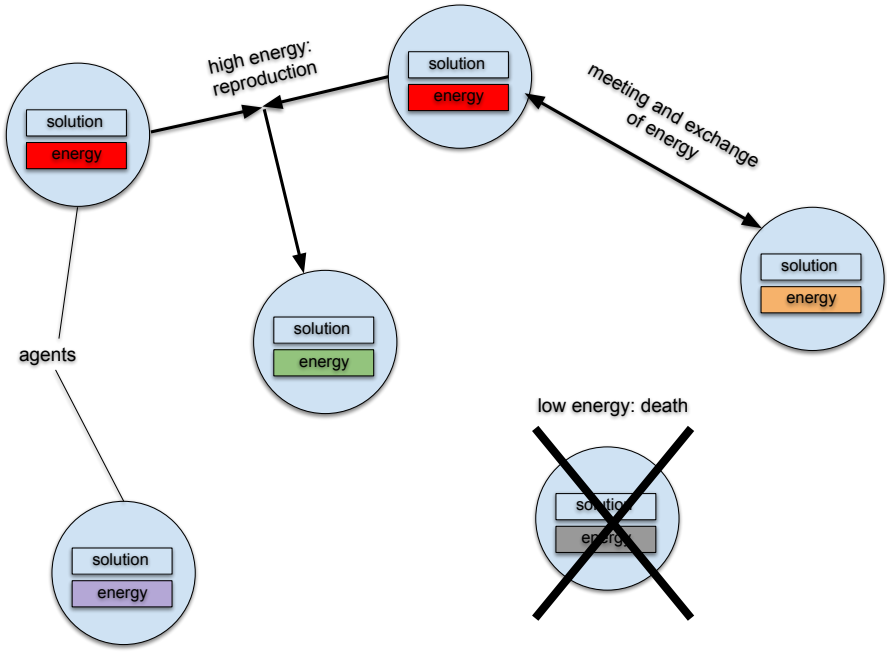

Figure 1.4. Evolutionary Multi-Agent System (EMAS)

Two core phenomena of evolution (i.e., inheritance and the aforementioned selection) are modeled by *reproduction* and the *death* of agents (respectively).

Reproduction is accomplished between two agents who each own a high-enough level of energy (i.e., their genotypes represent high-quality solutions). The information encoded in each parent's genotype is inherited with the use of variation operators – mutation and recombination. Since a newly created agent receives some initial portion of energy and the global amount of this resource must remain constant, parental levels of energy are adequately decreased.

If an agent's energy falls below a certain level, it dies and is removed from the system. This mechanism corresponds to the evolutionary phenomenon of selection, as agents with low levels of energy are supposed to bear low-quality solutions.

Additionally, an agent may change the island on which it is located – if its level of energy is high enough, it can migrate to other evolutionary islands. The mechanism of migration provides an exchange of information and resources throughout the system [71].

Pseudocode 2 presents an algorithm performed by an EMAS agent during each evolutionary step. During each iteration, an agent communicates with a neighbor provided by its parent (i.e., the aggregate agent – also known as island – that encapsulates it). Then, they may execute the reproduction action, doing crossover and mutation, if their levels of energy are high enough (which is verified in the *canReproduce* method). Otherwise, the mechanism of meeting is launched. In the end, the agent may migrate to another island (if its energy stands at a proper level, which is verified in the *canMigrate* method) or make a move on its current island.

Pseudocode 2 Pseudocode of EMAS agent's evolutionary step

```
1:  function STEP
2:      neighbor ← parent.getNeighbor()
3:      if canReproduce(this, neighbor) then
4:          reproduce(this, neighbor)
5:      else
6:          meet(this, neighbor)
7:      end if
8:      if canMigrate(this) then
9:          migrate(this)
10:     else if shouldMove(this) then
11:         move(this)
12:     end if
13: end function
```

Pseudocode 3 illustrates reproduction mechanism in EMAS. Each parent donates half of a descendant's energy (the agent's initial energy is a common global parameter). In this way, the sum of all of the agents' energy levels remains constant. A new agent's genotype is created by a crossover operator – its functioning depends on the provided implementation. Next, the newly-created agent is mutated (again, the particular mutation strategy is dependent on the implementation) and evaluated. Finally, the agent is added to the evolutionary island of its parents.

The process of meetings between two agents is presented in Pseudocode 4. The agents determine which one of them has a lower level of energy; this agent should then give some portion of its energy to the "stronger" one. Finally, it should be verified whether this agent should be removed from the system (in case its energy level reaches the minimal value) by a mechanism of death that models evolutionary selection.

Pseudocode 3 Pseudocode of EMAS reproduction action

```
1: global newbornEnergy
2: function REPRODUCE(agent1, agent2)
3:     agent1.energy ← agent1.energy − newbornEnergy/2
4:     agent2.energy ← agent2.energy − newbornEnergy/2
5:     newborn ← newAgent()
6:     newborn.energy ← newbornEnergy
7:     newborn.genotype ← crossover(agent1.genotype, agent2.genotype)
8:     mutate(newborn)
9:     evaluate(newborn)
10:    agent1.island.addAgent(newborn)
11: end function
```

Pseudocode 4 Pseudocode of EMAS meeting mechanism

```
1: function MEET(agent1, agent2)
2:     if agent1.fitness > agent2.fitness then
3:         energyToTransfer ← agent2.getEnergyToTransfer()
4:         agent1.energy ← agent1.energy + energyToTransfer
5:         agent2.energy ← agent2.energy − energyToTransfer
6:         if shouldDie(agent2) then
7:             die(agent2)
8:         end if
9:     else if agent1.fitness < agent2.fitness then
10:        energyToTransfer ← agent1.getEnergyToTransfer()
11:        agent1.energy ← agent1.energy − energyToTransfer
12:        agent2.energy ← agent2.energy + energyToTransfer
13:        if shouldDie(agent1) then
14:            die(agent1)
15:        end if
16:    end if
17: end function
```

EMAS is usually implemented as multi-deme algorithm; thus, agent migration from one evolutionary island to another is put forth in Pseudocode 5. To begin with, a target island is chosen. It may be randomly selected or chosen according to some other strategy. Then, the migrating agent is removed from its current island and introduced to the target one.

Pseudocode 5 Pseudocode of EMAS migration action

```
1: global evolutionaryIslands
2: function MIGRATE(agent)
3:     islandToMigrate ← evolutionaryIslands.get()
4:     agent.island.removeAgent(agent)
5:     islandToMigrate.addAgent(agent)
6: end function
```

Pseudocode 6 presents how the death of an EMAS agent may appear. It is a rather-straightforward action, as it consists only of removing the agent from the island where it is located.

Pseudocode 6 Pseudocode of EMAS death action

```
1: function DIE(agent)
2:     agent.island.removeAgent(agent)
3: end function
```

As previously noted, simple evolutionary algorithms fail to preserve population diversity. In the case of EMAS, it is quite simple to provide diverse solutions; i.a., by decomposing populations into different evolutionary islands and allowing agents to move from one island to another (the mechanism of allopatric speciation [72]).

EMAS has been formally proven to be able to solve optimization problems (this proof is based on the ergodicity of an appropriately constructed Markov chain, similar to the works of Vose [6]) [73, 74, 75].

Since the creation of EMAS, it has been applied to many problems (in each case, it has turned out to be very efficient and yield better results than classic evolutionary algorithms): classic continuous benchmark optimization [76], inverse problems [77], optimization of neural network architecture [78], multi-objective optimization [79], multi-modal optimization [51], financial optimization [80], etc.

It has also proven to be useful in research at different levels: formal modeling [81, 82], framework development [83], experimental research [84, 85, 86], etc.

EMAS is clearly a social metaheuristic (cf. Definition 1.2.1), as it is designed as a population of agents along with particular actions planned to be executed and communications to be realized. The metaheuristic has a visible cognitive potential,

as the agents perceive the solutions (or at least fitnesses) of other agents in the course of their meeting actions. The agency of EMAS individuals is, of course, inherent right from the inception of the algorithm; moreover, EMAS obviously belongs to the social metaheuristics class, as it processes a society of agents and each agent can undertake certain actions with others.

1.7. Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) are universal social metaheuristics stemming from evolutionary algorithms. As EAs, they use stochastic sampling of the solution space in order to produce new individuals (often called agents); however, in the case of EDAs, the probability distribution is usually derived from the information gathered in the current population and accordingly adapted [87]. EDAs are very simple (though very efficacious) metaheuristics. Later in this section, we will focus on one type of EDA: the COMpeting Mutating Agents (COMMA) and COMMA$_{op}$ algorithm, which can be treated as a social metaheuristic (cf. Definition 1.2.1).

EDA variants The general strategy realized in EDAs can be described as it is shown in Pseudocode 7.

Pseudocode 7 Pseudocode of EDA

```
1: population ← random_generation_of_agents()
2: while not stop_condition() do
3:     population ← select(population)
4:     probability_distribution ← compute(population)
5:     population ← sample(population, probability_distribution)
6: end while
```

In the beginning, the population of individuals is randomly generated, then the main loop of the algorithm is started. While the stopping condition is false, new population is constructed based on predefined selection, then new probability distribution is computed, based on the selected individuals and in the end, new population is constructed by sampling the solution space with the computed probability distribution.

The agent perspective of EDA is depicted in Figure 1.5. Each agent can be treated as an individual being, delivering solutions and being transferred to a new population based on newly computed probability distribution. Of course, as in the case of PSO, the sequential implementation of this agent-oriented algorithm will be the most natural. The agents processed in EDA utilize the common knowledge of other agents in order to build a next probability distribution, so a certain exchange of

information is present among them. Therefore, they (as well as their modifications; namely, $COMMA$ and $COMMA_{op}$ described later) can be treated as a society, fulfilling the requirements of the definition of a social metaheuristic.

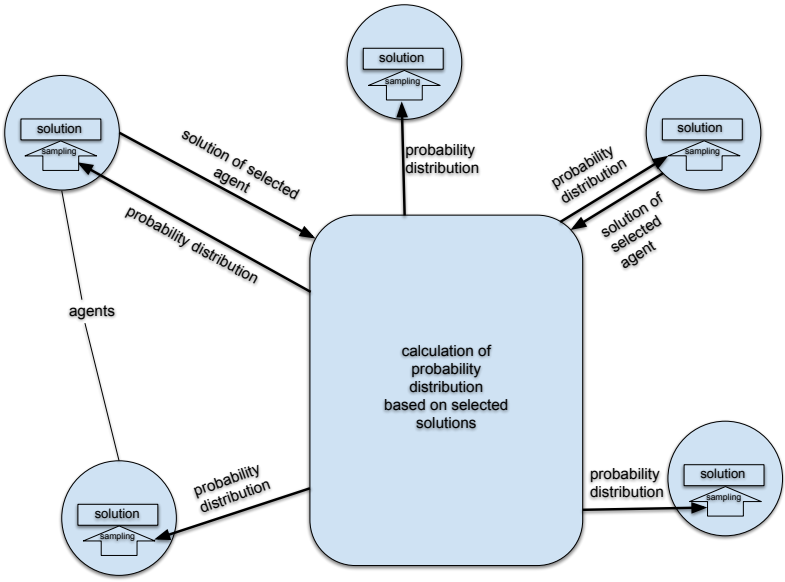

Figure 1.5. Classic EDA from point of view of agency

Thus, EDAs perceive the optimization process as a series of incremental updates of a certain probabilistic model, starting from a model of uniform distribution and finishing with a model solely generating global extrema (or their approximations). In an ideal case, the quality of the generated solutions will grow over time, and after a certain (hopefully reasonable) number of iterations, the algorithm will generate a global optimum (or its accurate-enough approximation). Different EDAs consist of different versions of the above-mentioned steps; however, the general idea remains the same – the strategy of generating new agents is iteratively adapted in order to increase the quality of the new solutions [88].

One interesting EDA-type algorithm is COMMA (COMpeting Mutating Agents), which was devised by Olivier Regnier-Coudert and John McCall [89, 90]. The COMMA algorithm performs the exploration and exploitation phases in parallel by using a population of agents and assigning different roles to each (in particular, using mostly mutation for generating new individuals, with the adaptation of its range coming from geometric inspirations). Moreover, it is quite easy to see that EDAs (and COMMA in particular) are quite closely related to agent-based computing systems, as they leverage some of their notions (e.g., autonomy in undertaking decisions about

individual mutations). Thus, these algorithms can be treated as a good starting points for introducing hybrid computing methods. In this algorithm, the sampling distribution is constructed by considering the population of agents sorted according to their fitness.

In this algorithm, for each position pos_j in population pop sorted in descending order for maximization, mutation distance d_j is set such that, for two agents at positions e and f, such that $d_e \leq d_f$ if $e < f$. As it may become beneficial to allow low-quality solutions to be accepted, probability p_j is also set for each pos_j. Each agent a_i is initially assigned a random solution (s_i). The population is then sorted by fitness. At each generation, each agent mutates s_i using distance $dist_i \in [1, d_r]$ defined according to its position r in the population. This step is equivalent to sampling from a distribution centered around s_i whose variance depends on r. If the mutated solution s_new has a better fitness than s_i, a_i replaces s_i with s_{new}. If s_{new} has a poorer fitness than s_i, s_{new} only replaces s_i with certain probability p_r. The algorithm of the original COMMA algorithm is shown in Pseudocode 8.

Pseudocode 8 Pseudocode of COMMA [90]

```
Initialize pop of σ agents with random solutions, distance vector d of size σ and
probability vector p of size σ
while not stop_condition() do
    sort pop by fitness
    for each agent a_i, i ∈ [0, σ − 1] do
        get position r of a_i in pop
        Sample new solution s_new with fitness fit_new by mutating s_i
        with distance dist_i selected with uniform probability from [1, d_r]
        if fit_new > fit_i then
            s_i ← s_new
        else
            s_i ← s_new with probability p_r
        end if
    end for
end while
```

The sampling range is inversely proportional to the fitness; thus, those agents with high fitness values (as related to the other members of the population) are mutated with a lower range than those individuals with lower fitness values. Thus, the whole algorithm exploits around "good" solutions while at the same time starting explorations around the "worse" ones. As the authors report, the above version of COMMA performed well when learning Bayesian network structures [89]. However,

the authors aimed at enhancing the proposed algorithm, by introducing multiple variation operators, thus proposing the COMMA$_{op}$ algorithm (see Pseudocode 14).

Pseudocode 9 Pseudocode of $COMMA_{op}$ [90]

Initialize pop of σ agents with random solutions
Initialize operator selection scale α with pairs $\{\phi_j, d_k\}$ for all selected operators ϕ_j and mutation distances d_k
Order α by $\rho_{\{}j, k\}$
Initialize $gen \leftarrow 0$ and maxGen; Initialize $fixedDistance$
repeat
    Sort *pop* by fitness in descending order
    $rate \leftarrow 1 - \frac{gen}{maxGen+1}$
    for each agent $a_i, i \in [0, \sigma - 1]$ **do**
        $r \leftarrow$ position of a_i in *pop*
        $r^- \leftarrow \frac{r \cdot |a|}{\sigma} - rate \cdot |\alpha|$
        if $r^- < 0$ **then** $r^- \leftarrow 0$
        end if
        $r^+ \leftarrow \frac{r \cdot |a|}{\sigma} + rate \cdot |\alpha|$
        if $r^+ > |\alpha| - 1$ **then** $r^- \leftarrow |\alpha| - 1$
        end if
        $r' \leftarrow random(r^-, r^+)$
        if $fixedDistance$ **then**
            Sample new solution s_{new} with fitness fit_{new} by mutating s_i with operator $\phi(r')$ and distance $d(r')$
        else
            Sample new solution s_{new} with fitness fit_{new} by mutating s_i with operator $\phi(r')$ and distance $dist_i$ selected with uniform probability from $(1, d(r'))$
        end if
        if $fit_{new} > fit_i$ **then** $s_i \leftarrow s_{new}$
        end if
    end for
    gen++
until $stopping\_condition()$

Apparently, COMMA can be treated as a social metaheuristic, as it governs agents that adapt their probability distribution to other individuals (a sorted set). Moreover, it can be classified as an agent-based metaheuristic.

COMMA$_{op}$ starts by the random generation of a population (*pop*). Moreover, the level of alteration $\rho_{j,k}$ associated with every possible pair $\{\phi_j, d_k\}$ of operator

and mutation distance is calculated and used to initialize operator selection scale α ($k \in \lfloor 1, n \rfloor$, and $j \in [1, n_{op}]$, where n_{op} is the number of single operators included in COMMA$_{op}$. This means that any value k can be considered for the mutation distance d_k for a chosen operator throughout the search. α is ordered with respect to $\rho_{j,k}$. For a non-deterministic operator whose associated $\rho j, k$ takes its value within a range, mean $\rho_{j,k}$ is used to order α. The maximum number of generations ($maxGen$) is calculated at this stage from the maximum number of fitness evaluations and population size σ. Finally, the $fixedDistance$ parameter needs to be set. It defines a given mutation distance and whether a mutation strictly uses the distance or can also use any value lower than the distance. This is another approach implemented in order to reduce the effect of some operators with a high associated level of alteration. Using a fixed distance can be geometrically interpreted as sampling solutions only from the edge of a distribution rather than from the whole distribution when $fixedDistance$ is set to false. Following this initialization step and at each generation, the population of agents is ordered by fitness, and the rate is updated (that is, the rate is decreased at each generation). Each agent then needs to sample a new solution. This is done according to its position (r) in the population. An operator and mutation distance pair is picked from α with respect to r but also by considering some variation represented by lower and upper bounds r^- and r^+, with $r^-, r^+ \in [0, |\alpha| - 1]$. Introducing such a variation ensures that all $\{\phi_j, d_k\}$ pairs can be used to generate solutions during the search. $d(r')$ and $\phi(r')$ stand for the mutation distance and the operator at the r'-th position in α, respectively. The COMMA$_{op}$ algorithm is described in Pseudocode 9 after the description given in [90].

1.8. Towards extension of social metaheuristics

The selected metaheuristics presented in this chapter cannot be treated as an exhaustive enumeration of the possible cases for extension towards introducing socio-cognitive inspirations; however, they were selected as a strong reference in order to build the base for the deliberations presented in the next chapter.

One should note that the most important feature shared by most of the presented algorithms is their conformance (on different levels) with agency standards (perhaps excluding the basic versions of evolutionary algorithms.

In fact, little effort is needed in order to enhance the autonomy of their individuals); in particular, the autonomy. All of the individuals utilized in these algorithms are (or can be with a little effort) enhanced with autonomy, even if it means only tossing a die in order to select a particular action to be executed.

The number of metaheuristics is immense, and it grows yearly (if not daily). Some researchers propose new algorithms for the sake of doing something novel

(cf. Sorensen [7]); according to common sense, such ways should be avoided. However, there are still social metaheuristics that may become the basis for new socio-cognitive methods, such as Differential Evolution [91], Steady State Genetic Algorithm [92], the or well-researched Stochastic Diffusion Search [93]. These possibilities will grow along with the extension of the state of the art in the field of social metaheuristics.

2. From social to cognitive inspirations in computing systems

Although perceived as being part of the computer science research field, metaheuristics are strongly connected to nature (in this case, usually biology). As previously mentioned, agent-systems take a lot of inspiration from sociology; however, there are areas of interest of computer science that borrow ideas from other natural phenomena (e.g., psychology). For example, the affective computing proposed by Rosalind Picard arose in 1995. According to her definition, "... is computing that relates to, arises from, or deliberately influences emotion or other affective phenomena" [94]. Considering another example, this research area encompasses methods for modeling and discovering the emotional states of computer users (e.g., for chat-bots or cognitive robots) or sentiment analysis (which is very popular nowadays) [95]. Another field of interest in IT is the study of user experience (with regards to the interface of a computer system) and its optimization (in order to increase its usability). However, one can argue that there are many more interesting aspects of psychology that may be applied in the case of computer science methods (in particular, computing with metaheuristics).

In this chapter, two phenomena that are studied in psychology are selected – perspective taking and social cognitive learning. The aim of the chapter is to form the idea of a socio-cognitive system and support the process of constructing novel socio-cognitive metaheuristics based on the previously presented social-ones. Therefore, agent-oriented systems are evaluated from the point of view of Social Cognitive Theory at the end of this chapter, as agency can be treated as one of its main features.

2.1. Perspective taking

In cognitive psychology, the character traits of egocentrism (taking one's own perspective) and altercentrism (taking another person's perspective into consideration) have long been recognized as playing a key role in interpersonal relationships

(see, for instance, [96, 97]). Moreover, brain-imaging studies have shown that altercentricity and the strategy of perspective taking develop in parallel with brain maturation and psychosocial development during adolescence [98, 99]. Perhaps mirroring this psychological development, artificial intelligence researchers have started to incorporate altercentricity into robots and autonomous systems in recent years [100]. We also continue to utilize the notions of egocentrism and altercentrism, adapting them appropriately for use in our computing system.

Typically, perspective taking is seen as a one-dimensional ability: the degree to which an agent can take another's perspective. However, recent research has explored a two-dimensional approach [101] where one distinguishes between the ability of an agent to handle conflict between its own perspective and those of another agent and the relative priority that an agent gives to his own perspective relative to that of the other. During social interactions, humans do not always share the same views. Being able to consider another person's point of view, therefore, requires us to put aside one's own perspective. This is particularly hard if one holds a strong view. Individuals endowed with good cognitive skills for managing conflicting information are, therefore, typically better perspective-takers [102]. However, humans differ in terms of how much they are interested in or willing to pay attention to others as compared to themselves. Sometimes, individuals focus only on their own perspective (egocentrism), while on other occasions, they can focus more on other people's perspectives (altercentrism) [96, 100].

The less a person focuses on his own perspective, the more motivated that person will be to engage in perspective taking [101]. Experimental research has suggested that these two dimensions (conflict handling and perspective priority) might be independent; factors such as guilt or shame affect each of these dimensions individually [103]. This two-dimensional approach to perspective taking inspired us to define four types of individuals:

- Egocentric individuals, focusing on their own perspective and becoming creative thanks to finding their own new solutions to a given task. These individuals do not pay attention to others nor are they inspired by the actions of others (or these inspirations do not become a main factor in their work).
- Altercentric individuals, focusing on the perspective of others and, thus, following the group of others. Such individuals become less creative, but they can still end up supporting good solutions by simply following them.
- Good-at-conflict-handling individuals, becoming inspired in a complex way by the actions of others. They consider different perspectives and choose the one considered best to them.
- Bad-at-conflict-handling individuals, acting purely randomly, sometimes following one perspective and sometimes another with no inner logic.

In recent work, it has been shown that the proportion of altercentric, egocentric, good, and bad perspective conflict handlers can fluctuate within humans depending on situational factors. In our first set of simulations, we chose three types of proportions found in humans as a starting point: one representing the proportion of perspective-taking profiles in a baseline condition (without the manipulation of situational factors) and two types of proportions corresponding to the effects of two situational factors (namely, guilt and anger). These two factors affect perspective taking in opposing ways: guilt heightens and anger lowers the proportion of altercentric individuals [103].

Summing up, it is quite natural that a certain population of individuals can comprise different individuals, possibly forming certain groups or clusters. Their behavior affecting other users and the realization of their goals might be different as well (based on their intrinsic characteristics). This view is going to be extended through the lens of the Social Cognitive Theory proposed by Bandura (which is described in the following section).

2.2. Social Cognitive Theory

The perspective-taking phenomenon sketched out in the previous section leads us towards a more general and interesting theory from the point of view of studying the social and cognitive characteristics of a population; namely, Social Cognitive Theory (SCT) (introduced by Bandura [104]). This theory is used in psychology, education, and communication and assumes that portions of an individual's acquisition of knowledge can be directly related to observing others in the course of their social interactions, their experiences, and outside media influences [10].

Thus, the individuals use this gathered information to guide their behaviors, not solely learning them by themselves (e.g., during the course of trials and errors). They can replicate others' deeds (trial and error) and predict the consequences based on observations, thus possibly reaching their goals sooner. However, one has to remember that certain ethic and moral control of the behavioral patterns should be maintained (c.f. Bandura's Bobo Doll experiment, when children mimicked the aggressive behavior of their colleagues without any further deliberation [105]).

Now, let us focus on the elements of SCT. Its basic concepts are explained by Bandura through a systematization of triadic reciprocal causation [106]. This schema shows how the reproduction of observed behavior (mimicking) is influenced by the interaction of the following three determinants:

- Personal: the level of an individual's self-efficacy toward behavior; i.e., does the learner believe in himself and in his personal abilities to correctly complete a behavior (cf. Section 2.1, egocentricity, and altercentricity)?

- Behavioral: the actual response an individual receives after he performs a behavior; i.e., can he receive some reward after performing the behavior correctly?
- Environmental: certain aspects of the environment or setting that influence an individual's ability to successfully complete a behavior; i.e., increasing self-efficacy by providing the appropriate support and materials.

The learners described by Bandura are not driven solely by the environment, inner forces, or interactions – instead, they are self-developing, self-regulating, self-reflecting, and proactive. Thus, the learners can be clearly and easily perceived from the point of view of agency (cf. the definition of an agent in Section 1.2 that, though aimed at describing the software agent, is clearly inspired by humans). In fact, human agency operates within the following three modes according to Bandura:

- Individual agency: a person works alone to achieve his own's goal (a classic example is James Bond striving to save the world under Her Majesty's orders).
- Proxy agency: a person takes advantage of another person to achieve his own's goal (a classic example would be the complex relationship between Francois du Tremblay and Cardinal Armand Richelieu in history).
- Collective agency: a group of people working together to achieve a common goal (the well-known War Room from *Doctor Strangelove...* film – an ensemble of experts).

Let us delve deeper into the description of human agency, being one of the cornerstones of SCT. The agency of humans has the following four core properties:

- Intentionality: individuals can actively decide whether they engage or not in certain activities.
- Forethought: individuals can anticipate the outcome of certain actions.
- Self-reactiveness: individuals can construct and regulate appropriate behaviors.
- Self-reflectiveness: individuals can reflect and evaluate the soundness of their cognitions and behaviors.

In order to recapitulate, SCT is closely connected with the process of knowledge acquisition or learning directly correlated to the observation of models that can consist of imitating other agents or information acquired from the media. The observations realized by the learners are depicted as follows:

- *Attention*: certain observers selectively pay attention to specific social behavior depending on accessibility, relevance, complexity, the functional value of the behavior, or some observer's personal attributes such as cognitive capabilities, value preferences, or preconceptions.

- *Retention*: the behavior and subsequent consequences are observed then converted to a symbol that can be accessed for future reenactments of the behavior. Note: a positive behavior should be followed with a positive reinforcement and vice-versa for negative behavior.
- *Production*: the symbolic representation of the original behavior is translated into action through reproduction of the observed behavior in seemingly appropriate contexts. During the reproduction of the behavior, a person receives feedback from others and can adjust his representation for future references.
- *Motivational process*: reenacts a behavior depending on the responses and consequences the observer receives when reenacting that behavior [10, 106].

Considering the actual applications of SCT, it has been successfully applied in the area of media analysis or media content and effect studies. For example, Bandura showed in 2011 [107] that heavily repeated images presented in the mass media can be processed and encoded by the viewers. Moreover, it was shown that people can learn how to perform certain behaviors through media modeling [106]. In the area of health-related communications, the spreading of information about quitting smoking, preventing HIV, engaging in safe sex, etc. was analyzed (see, e.g., [108]).

Summing up, the Social Cognitive Theory gives a firm background for understanding human behavior (in particular, that connected with learning, gathering new knowledge, mimicking others, etc). Simple yet effective assumptions allowing for the easy description of a learning process and a very close relationship to agency makes this theory a very attractive candidate for inspiration in software systems connected with the agency paradigm. Let us follow with grounding the relationships between SCT and possible references in agent-based systems.

2.3. Social cognitive agent systems

In such a way, the defined learning process conducted by individuals can be very naturally perceived from the point of view of agency. The autonomy of the human, his initiative, creativity, self-reliance and many other characteristics. fit well into the description of an agent (including software agents by Wooldridge, for example [47]) – see Section 1.2. Real-world individuals (human beings) and software agents (virtual beings inspired by humans) will thus take part in the process of learning. They are treated like autonomous beings, embedded in a certain environment, undertaking their own actions, and perceiving the actions of others (and the consequences of these actions) in order to learn something, for example (cf. [47, 69]). Moreover, their perspectives may be influenced by other individuals (cf. Section 2.1).

Seeking a reference between natural and artificial systems, one has to note that certain mechanisms introduced in agent-based computing or simulation systems can easily imitate the factors of triadic reciprocal causation identified by Bandura (at least partially):

- Behavioral: an agent can receive a certain reward for performing an action – one of the simplest examples would be to recall the Iterative Prisoner Dilemma and the payoff received by the players [109].
- Environmental: the agents can utilize certain resources and gather them in the environment. The number of resources gathered can influence their motivation and self-efficacy (depending on the appropriate definitions – cf. EMAS [70]).

The agency of the learners described by Bandura fits very well into the agency perceived in the world of software agents; e.g., agent-based modeling, agent-based simulation, agent-based computing, agency-related metaheuristic algorithms, etc.:

- Individual agency: an agent strives towards the realization of its own goal; e.g., becoming an IPD agent.
- Proxy agency: an agent leverages other agents in order to attain a goal; e.g., delegating specialized actions to dedicated agents (a computing agent delegates a local search to specialized agents – cf. memetic computing [110]).
- Collective agency: a group of expert individuals becomes a basis for calculating the global response of a system; e.g., an ensemble of expert algorithms for classification or prediction (cf. [111], for example).

Software agents can express similar characteristics like human agents can to a certain extent; however, their methods of modeling them rely heavily on the available computing methods, machine-learning models, or sometimes pure randomization.

- Intentionality: agents can actively decide whether or not they engage in certain activities; e.g., a computing agent can decide (even randomly) if it executes a certain action (for example, a mutation of its solution).
- Forethought: agents can anticipate the outcome of certain actions and build their own models (beliefs in the BDI architecture; e.g., estimating the payoff in an IPD simulation).
- Self-reactiveness: agents can build their own complex models of their behavior and follow this model in order to strive towards the realization of a goal (e.g., different agents can execute different sets of actions).
- Self-reflectiveness: agents can evaluate their models and adapt them appropriately based on the ongoing processes in the system being affected by the environment as well as other agents.

Such phenomena could be (and have already been) implemented in agent-oriented models such as Belief Desire Intention [112] or M-agent [113].

The agents can, thus, acquire certain knowledge, build their models, and follow their assumptions, choosing and modifying the actions based on the information gathered in the environment and observed among the other agents. The observations realized by the agents are depicted as follows:

- *Attention*: agents can selectively perceive the actions of other agents in order to help realize their own goals. They can take the perspective of others.
- *Retention*: the observations performed by agents can lead to the construction of cognitive models; even the simple statistics of certain actions performed (or not) by other agents can be taken into consideration before deciding on the agent's own action. Of course, the agents can be rewarded/punished for executing certain actions.
- *Production*: the actions observed and the model built become the reference for undertaking the agent's own actions. This process is related to the deliberation from the BDI model.
- *Motivational process*: this depends on the actual definition of motivation in the software agent world. It can of course be considered in constructing the cognitive model, putting the observations, deliberations, and actions of the agents into a feedback loop.

The software agents cannot be treated as equal to human agents, and their social and cognitive capabilities can only be inspired by actual humans. However, as it was shown, many apparent references exist between these two worlds; thus, the inspirations in agency that come from SCT seem to be well-situated and can be helpful in constructing new computing methods. This will be clearly shown later in this book.

2.4. Enhancing social metaheuristics with cognitive abilities

An ability to view a situation from another individual's perspective is thought to be a crucial socio-cognitive characteristic for successful social interactions. This allows people to understand and predict other individuals' behaviors and help them connect emotionally with others. People, however, are not all equally skilled at perspective-taking [114], and contextual factors (e.g., emotional state) also influence how efficiently they use these skills at a given moment [103, 115] and the extent to which they use these skills for a pro-social motive [116, 117]. Social interactions thus typically involve people with diverse levels of efficiency and motivation engaging in perspective taking.

If a model can be constructed showing how perspective taking influences individual behavior in a society and how macro-level social phenomena emerge from the interaction of people with different levels of perspective taking, it can help us understand why some societies seem harmonious whereas others are ridden with conflict; it would also be useful to devise strategies to reduce conflicts. Moreover, as our previous results suggest [118], these models may help us develop new optimization strategies for traditional computational problems; for example, in increasing the diversity of a search for a better exploration of the search space (similar to introducing islands into evolutionary computing methods). The current study addresses this issue, proposing a new metaheuristic algorithm based on socio-cognitive inspirations.

The basic idea of social systems (in particular, social metaheuristics – see Chapter 1) is to construct a system consisting of a number of individuals and make them work together as a society on reaching a common goal; in the case of metaheuristics, this goal is to search for the optimum solution of a certain quality function. Thus, many population-based metaheuristics can be treated as social ones with no exaggeration. After describing Social Cognitive Theory and its relationship to agency in this chapter, let us try to propose a second definition stemming from the one presented in Section 1.2 (Definition 1.2.1):

Definition 2.4.1. *A socio-cognitive metaheuristic is a computing algorithm belonging to a class of social metaheuristics whose way of conducting a search can be described using Social Cognitive Theory.*

Thus, introducing elements of social-learning, perceiving the behavior of other individuals, taking their perspectives, and planning the subsequent actions based on the model constructed after perceiving the actions of others can be treated as characteristics of a socio-cognitive metaheuristic. One has to remember the inherent agency of the learning individuals (according to SCT); therefore, socio-cognitive metaheuristics must express a certain level of agent-orientation. To sum up, a socio-cognitive metaheuristic will usually be constructed as a society of agents whose actions depend in a non-trivial way on the actions of other agents. These agents can perceive the actions of others, build cognitive models, deliberate over the outcomes of their actions, and construct their own strategies thanks to their cooperation with others and interaction with their environment.

The two main classes of social metaheuristics considered in this monograph that are enhanced with cognitive abilities can be divided into two groups based on the way they introduce the cognitive abilities depending on the nature of the algorithms (in particular, the way the individuals process the information). The notion of generation in nature-inspired metaheuristics is quite well-known and present from the beginning of EAs [34]; this particular feature leads us to consider when and how the cognitive abilities are introduced into a metaheuristic algorithm.

Thus, the two following definitions may be proposed:

Definition 2.4.2. *Inter-generation cognitive metaheuristics: generation-based metaheuristics process a population of individuals, constantly replacing them by constructing new generations based on the old ones using by predefined operators.*

Examples of such algorithms are, of course, evolutionary algorithms (processing a population based on selection and variation operators), genetic programming, immunological algorithms, differential evolution, estimation of distribution algorithms, and many more. Within a generation, the individuals can perceive others and act according to their observations (e.g., being divided into species or sexes). The gathered knowledge during each generation survives the individuals (they tend to explore and exploit new areas in a search space thanks to the application of variation operators). Note: algorithms such as EMAS utilize the notion of a desynchronized generation; however, the individuals are still replaced by new ones (although this does not happen for a whole society at the same time).

Definition 2.4.3. *Intra-generation cognitive metaheuristics: metaheuristics process the population of individuals without replacing those individuals. Knowledge is gathered following the actions of the individuals (agents) and is present in the population during the whole run of the algorithm (possibly also in the environment).*

Good examples of such algorithms are swarm metaheuristics (e.g., ACO and PSO) – there are no death- nor birth-like events in such computing algorithms.

Of course, the hybridization of these two approaches is possible and will be discussed in the last chapter of this monograph.

Constructing such algorithms can be easily realized by the hybridization of social-metaheuristics with dedicated techniques usually inspired by the phenomena occurring in real-world societies (like mimicking the behavior of others) that can be described by SCT. Let us refer to Talbi, who provides a concise way of classifying hybrid approaches based on selected design issues in [119]:

- In low-level hybrid techniques, a certain function of one algorithm is replaced with another optimization algorithm.
- At the same time, different optimization algorithms are combined in high-level hybrid techniques without changing their intrinsic behavior.
- In relay hybrid techniques, individual algorithms are applied in a line (one by one).
- In teamwork hybrid techniques, each algorithm performs an independent search.

The last two classes proposed by Talbi are mostly technically-oriented, as they consist of putting together the outcomes of completely independent algorithms (either

enhancing the outcome of one algorithm by another one – relay hybrid, or combining the outcomes of many independent ones – teamwork hybrid). However, the first two can become starting points for many social cognitive algorithms. For example, one can consider adding a local search to an evolutionary algorithm in the place of a mutation (creating one of the memetic algorithms) or changing the algorithm for computing the PSO individual movement vector in order to take the vectors of its neighbors into consideration, contrary to the original algorithm that considers only global and local ones. Both of these examples can be treated as social cognitive metaheuristics implemented as hybrids of existing computing methods.

In the remaining part of this book, let us focus on selected social cognitive metaheuristics built on top (or using parts) of actual social-metaheuristics, thus enhancing or introducing their cognitivity.

3. Socio-cognitive swarm metaheuristics

Human always wanted to mimic the nature in order to support the everyday tasks (e.g. observing of birds finally lead to devising of a way to soar in the air). Swarm metaheuristics are also an outcome of observation of nature. These are computing methods that are inspired by the movement and behavior of swarming animals; these are very efficient in optimization applications, although they were first designed for simulation purposes.

The first swarm algorithm was proposed by Reynolds [120] and consisted of a simulation of a flock movement based on three simple rules: alignment (the particle aligns its velocity to the velocities of its neighbors), separation (the particle avoids blocking its neighbors), and cohesion (the particle moves towards the mass center of its neighbors). This algorithm has been applied many times for preparing special effects in movies, for example (e.g., the stampede in the popular Disney movie *The Lion King*.

In computing, the most popular swarm intelligence algorithms are Ant Colony Optimization and Particle Swarm Optimization. Of course, there are some hybrids as well as some recent less popular computing methods like Stochastic Diffusion Search. Introducing cognitive inspirations to swarm metaheuristics is very natural and generally consists of the introduction of many species of individuals and the relationships among them. As swarm metaheuristics do not utilize the notion of generations, the socio-cognitive algorithms constructed on their basis can be classified as intra-generation ones (cf. Definition 2.4.3).

In the course of this chapter, the socio-cognitive versions of ACO and PSO are presented, along with a multi-type ACO algorithm proposed by Nowé [121, 122] (which can also be considered as socio-cognitive one). Besides the definitions of the algorithms, selected experimental results are referenced. At the end of the chapter, a very attractive possibility of the further development of the socio-cognitive computing paradigm is sketched out; that is a hybridization of the stochastic diffusion search algorithm. Finally, the relationship between the proposed metaheuristics to Social Cognitive Theory is discussed.

3.1. Socio-Cognitive Ant Colony Optimization

Ant Colony Optimization turned out to be a a very efficient metaheuristic in solving hard combinatorial problems like TSP [123] or its more complex versions connected with Vehicle Routing [124]. Also, the quadratic assignment problem [125] and different scheduling problems were successfully tackled using ACO. Without any doubts, ACO belongs to the group of social metaheuristics (see Definition 1.2.1). While realizing fruitful discussions with psychologists and computer scientists (Dana Samson, Henryk Bukowski, Tom Lenaerts, Bipin Indurkhya, and Ann Nowé), an idea arose to enhance ACO with cognitive possibilities, creating the first socio-cognitive metaheuristic algorithm (fully described in [11], for example). This idea is based on perspective-taking research results and is related to an interesting ACO-based metaheuristic proposed by Ann Nowé [121, 122].

As the proposed metaheuristic is a kind of hybrid, we start this chapter with a short overview of the existing ACO-related hybrid metaheuristics. Next, the idea of a Socio-Cognitive ACO is presented, and its relationship to the Social Cognitive Theory is described using the notions identified by Bandura and referenced in Section 2.2. Later, the experimental results of applying the novel metaheuristic to solving TSP are discussed.

The parameters of the proposed Socio-Cognitive ACO (in particular, the configuration of the ACO population) were initially set in an arbitrary way; therefore, the last part of this chapter is devoted to the automatic adaptation of these parameters, and the proposed ideas are summarized with the relevant experiments.

This section recalls the most important ideas and results presented in:

- Świderska et al. [11] in order to properly describe the first socio-cognitive metaheuristic algorithm, namely Socio-Cognitive ACO (SC-ACO),
- Byrski et al. [126] in order to properly describe the automatic adaptaion of the population structure of SC-ACO.

3.1.1. Selected hybrid metaheuristics based on ACO

The proposed ACO metaheuristic can be treated as a hybrid of ACO and social-cognitive theory-inspired mechanisms. Therefore, before proceeding further, let us take a look at the hybrid versions of ACO present in the state of the art.

Kabir et al. [127] developed their own new rules for the modification of pheromones, and the ants are guided in the correct directions while constructing graph (subset) paths using a bounded scheme in each and every step in the algorithm. The system was applied to the problem of feature selection. In [128], Wei and Yuren utilized the concept of a minimal spanning tree in order to guide the search realized by the ants solving Traveling Salesman Problem (TSP). Xing et al. [129] solved

the Capacitated Arc Routing Problem using an ACO version heavily dependent on problem-related information; i.e., clustering and priority of arcs. Myszkowski et al. [130] prioritized certain ants and allowed them to leave more pheromones while efficiently solving the resource-constrained project scheduling problem. In [131], Huang et al. presented an improved ACO algorithm characterized by adding a local search mechanism, a cross-removing strategy, and candidate lists, applying it to solve TSP.

ACO has been hybridized as a part of a chain of algorithms (HRH [high-level relay hybrid] according to Talbi's classification [33]); for example, Toksari [132] puts together ACO and Iterated Local Search in order to balance the exploitation and exploration of the whole search and applies the system to forecast electrical consumption in Turkey. Shang et al. uses ACO and PSO in a chain after statistically generating several good solutions [133] in order to solve TSP. In [134], Hertono et al. chain ACO and the 3-opt algorithm; in addition, they applied PSO in order to tune ACO's alpha and beta parameters, treating TSP as a benchmark.

ACO has been used in complex connections with other algorithms; e.g., Zhang and Tang use ACO as a greedy search heuristic as a part of a scatter-search metaheuristic [135] applied to solving a vehicle-routing problem. Wang and Guo [136] use ACO connected with Genetic Algorithm that is used to refine the ACO findings in solving TSP.

There are many more modifications of the classic ACO, such as hierarchical ACO, where additional means of control are used to manage the output of the ant species [137]. Another approach assumes that the ants are equipped with different skills (such as sight or speed) in order to realize global path-planning for a mobile robot [138]. Yet another very effective ant-based TSP solver is based on using two types of ants (classic and exploratory) and works by creating so-called "shortcuts" for the ants to move according to some predefined conditions (like staying close to some selected cities). [139]. In [140], the authors introduce different ant sensitivities to pheromones such that ants with higher sensitivities follow stronger pheromone trails while ants with lower sensitivities behave more randomly: together, they strive to maintain a desired balance between exploration and exploitation.

3.1.2. Multi-type ACO

A very interesting modification of ACO that is relevant to the metaheuristic presented in this chapter is the multi-type ACO [121, 122] proposed by Nowé et al., which defines many species of ants and makes complex stigmergic interactions among them possible (such as attraction and repulsion to/from the pheromones of different ant species). These algorithms have been successfully applied to such problems as edge disjoint path finding [121] and the protection of a light path [122].

The Multi-type ACO is based on Ant Colony System (ACS, [58]). ACS differs from basic ACO by three aspects: the extended state transition rule, the global updating rule (applied only to edges from the global best solution), and the additional local updating rule.

Ant k in iteration t chooses to move from components i to j by applying rule (3.1).

$$j = \begin{cases} max_{j \in allowed_k}\{[\tau_{ij}(t)] \cdot [\eta_{ij}(t)]^{\beta}\} & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases} \tag{3.1}$$

where q is a random number uniformly distributed in $[0, 1]$, q_0 is a parameter, and S is a random component selected with Equation (1.1). The state transition rule favors short edges and edges with a large amount of pheromone.

The classic ant algorithms utilize cooperation between ants according to natural inspirations. Socio-cognitive algorithms introduce more-complex interrelations between the ants (in fact, their types or species). This is the case with the multi-type ACO, which introduces direct competition into the algorithm. In this algorithm, the authors propose several different species (or types) of ants. Each ant of a single species cooperates with the ants belonging to the same species (exactly in the same way as in the original algorithm). However, ants that belong to different species compete. The species leave different pheromones; of course, communication between the different species (and inside the same species) relies on stigmergy. Thus, the pheromones left by different species will repel an ant looking for a path, while a pheromone left by the same species will attract the same ant (just like in the basic algorithm). This mechanism is aimed at finding disjoint solutions – each by different ant species (see Fig. 3.1).

The discussed algorithm is a modification of Ant Colony System:

$$j = \begin{cases} argmax_{u \in J_r^k}[\tau(r,u)] \cdot [\eta(r,u)]^{\beta} \cdot [1/\phi_s(r,u)]^{\gamma} & if q \leq q_0 \\ J & if q > q_0 \end{cases} \tag{3.2}$$

$$p_s(r,u) = \frac{[\tau_s(r,u)]^{\alpha} \cdot [\eta(r,u)]^{\beta} \cdot [1/\phi_s(r,u)]^{\gamma}}{\sum_{l \in J_i^k}[\tau_s(r,l)]^{\alpha} \cdot [\eta(r,l)]^{\beta} \cdot [1/\phi_s(r,l)]^{\gamma}} \tag{3.3}$$

In the given formulas above, $\phi_s(r,u)$ stands for the level of pheromones not belonging to type s on edge (r,u). This is a sum of all of the pheromone trails left by other ant species. The power γ stands for the sensitivity of the ant to the presence of a foreign trail. When this power is zero, the ant will behave like a classic Ant Colony System ant. A higher value of γ makes choosing a path with a foreign trail possible.

The authors have also modified the solution by making it sensitive to not only the cost but also to an increased number of foreign ants on this path.

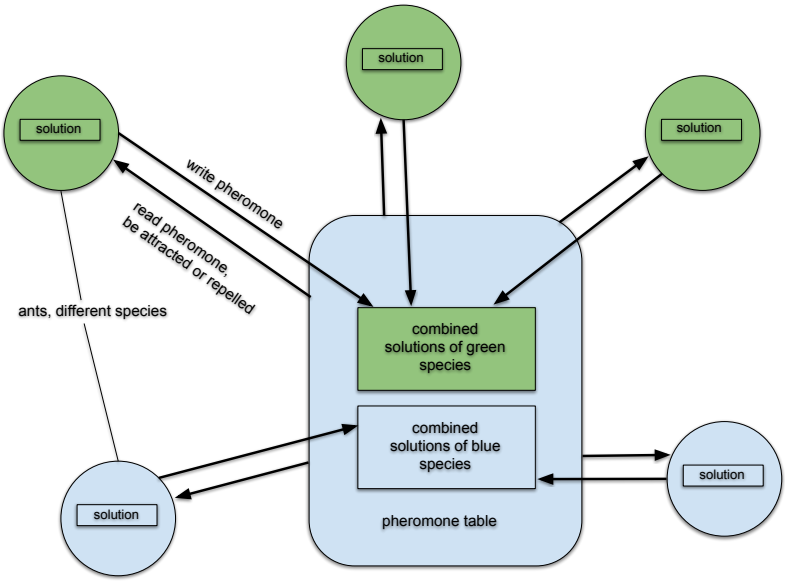

Figure 3.1. Multi-type Ant Colony System

The authors have thoroughly reviewed the parameters of the proposed algorithm [121] and applied the proposed model to find disjoint paths in optical network optimization [122]. The algorithm has also been applied for the optimal matching problem in weighted bipartite graphs [141].

3.1.3. Socio-Cognitive ACO

In this section, we present the ant system as a way of expressing the socio-cognitive behaviors of a population of ants by introducing various ant species that embody different behaviors from the point of view of their stigmergic interactions.

The work presented here is summarized after [118, 11]. A certain relationship between this idea and the concept of Ann Nowé presented in [121] and 3.1.2 can be perceived; however, our approach relies on much more complex relationships between the pheromones and the ants (not merely simple attraction or repellence).

The main idea of a Socio-Cognitive ACO is to introduce a number of different species of ants instead of only one and make them sensitive to the pheromones of other species in different ways. Different ant behavior will, of course, lead to different ways of exploring and exploiting the search space, increasing the diversity of the search itself (cf. the diversity measuring presented in [142]).

In the beginning, the particular behavior of species and their sensitivities were strongly inspired by the perspective-taking psychological research results concerning the behavior of human societies, (see Section 2.1). Getting inspired by the observation by Bukowski et al. [101] and following the two-dimensional approach to perspective taking led us to define four types of ants:

- Egocentric ants: focusing on their own knowledge, thus taking into consideration only the distance perceived in the graph and not paying attention to the knowledge of others expressed by the pheromone table.
- Altercentric ants: focusing on the perspectives of others, thus focusing only on the information contained in the pheromone table. If there are more than one ant species, they sum up the pheromones of all of them, not distinguishing individual traces.
- Good-at-conflict-handling ants: getting inspired in a complex way by the actions of other ants, considering different perspectives, and choosing the one considered the best for them (thus, assigning certain weights to the pheromones left by other ants).
- Bad-at-conflict-handling individuals: acting in a purely random fashion.

Based on the assumed profiles of the ant species and following the observations presented in [103] regarding the actual perspective taking profile of the human species, we first propose the following arbitrary structure of the populations:

- Control Sample (baseline proportions of different types of perspective takers found in a typical human population): where good conflict handlers form a major proportion with a roughly similar proportion of the other three types of perspective takers. It is to note that this is the sample with the highest proportion of egocentric individuals.
- Increased Good Conflict Handling Sample (proportions based on a population of humans that has been induced to feel anger): where proportion of good conflict handlers is further increased as compared to the control sample while reducing the fractions of the altercentric and egocentric individuals.
- Increased Altercentricity Sample (proportions based on a population of humans that has been induced to feel guilt); where the proportion of good conflict handlers and egocentric individuals is significantly decreased and is compensated by a higher proportion of altercentric individuals and (to a lesser extent) a higher proportion of bad conflict handlers.

Now, let us describe in detail the arbitrary ant species considered by us in the first set of our simulations. In a second set of simulations, we varied the proportions of the types of individuals in a more systematic way to examine their respective efficiency in finding a solution. Later in the chapter, we will show the adaptive way of stabilizing the population structure.

Classic ants These ants also play an important role in our populations. They consider both pheromones and distance while choosing their direction by computing *path attractiveness* in order to complete the cycle. In this case, an ant present at node i will choose the next edge according to the following attractiveness (equal to the one presented in Equation (1.1), repeated here for sake of clarity):

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}(t)]^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where $\tau_{ij}(t)$ is the level of a pheromone on edge (i, j) and $\eta_{ij}(t)$ is the visibility (or attractiveness) of the edge, which can be defined as an inverse distance between the cities in the case of TSP. $allowed_k$ is a set of possible transitions for ant k in its current state. Finally, α and β are parameters that express the relative priority of pheromones and attractiveness. The default factors are pheromone influence $\alpha = 2.0$ and distance influence $\beta = 3.0$.

Each type of ant described in Section 3.1.3 uses Equation (3.4) to calculate the probabilities for the subsequently chosen edges, differing only in **attractiveness** to various types of pheromones.

Multi-pheromone ACO In the proposed Socio-Cognitive ACO, the idea of having many pheromones instead of just one is implemented by introducing different "species" of ants and enabling their interactions (similar to the approach taken in [121]). The interaction is considered to be a partial inspiration (similar to perspective taking in the real world), realized by a particular ant reacting to the decisions taken by ants belonging to other species. This is made possible by having ants of different species leave different "smells" (see Fig. 3.2).

Different ants follow different rules (i.e., they consider the different properties of the paths) of computing the attractiveness factor. They utilize the smells of the pheromones left by other species in a predefined way. The proposed approach is a modification of the classic Ant Colony Optimization algorithm, but it can easily be adapted for other modifications and hybrids of this metaheuristic algorithm.

Different ant species leave pheromones that "smell" different, so the pheromone left at a particular edge is described in the following way:

$$\tau_{ij} = \tau_{ij}^{(EC)} + \tau_{ij}^{(AC)} + \tau_{ij}^{(GC)} + \tau_{ij}^{(BC)} \tag{3.5}$$

where a classic pheromone τ_{ij} for a particular edge is computed as a simple sum of the different pheromones (EC: egocentric, AC: altercentric, GC: good-at-conflict-handling, BC: bad-at-conflinct-handling) deposited by other species on this edge. The probability of choosing the next edge is, of course, given by Equation (1.1),

while τ_{ij} is given by the above-written formula. Of course, when using the attractiveness equation with the pheromone trace described in the above-given way, the actual value of τ_{ij} is a subject of normalization.

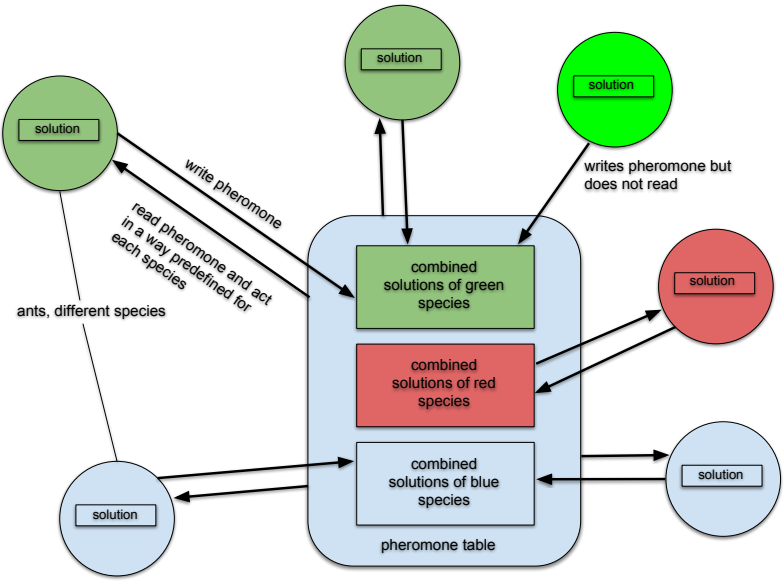

Figure 3.2. Socio-Cognitive ACO.

Other ants may react to different combinations of these pheromones. Of course, more species and more pheromones may be introduced into the system if necessary.

Based on this framework, details of the actions undertaken by various ant species are described below. It is to note that the chosen species (namely, Egocentric, Altercentric, Good-, and Bad-at conflict handling) were chosen based on the real-world features of the human population (based on the suggestions of one of the co-authors).

Egocentric ants (EC) These ants are supposed to be creative in trying to find a new solution. They care less about the other ants and their pheromone trails. Instead, they mostly focus on the distance of traveling on the path as a way of determining their next directions. An ant at node i will choose the next edge with an attractiveness computed as follows:

$$p_{ij}^{k}(t) = \begin{cases} \frac{\eta_{ij}(t)^{\beta}}{\sum_{k \in allowed_k} \eta_{ik}(t)^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where $\eta_{ij}(t)$ is the visibility (or attractiveness) of edge (i, j), which can be defined as an inverse distance between the cities in the case of TSP. $allowed_k$ is a set of possible transitions for ant k in its current state. Parameter β is set in a similar way as in the previous equations. The default distance influence is again $\beta = 3.0$.

Altercentric ants (AC) These ants follow the majority of the others, focusing on the pheromones without caring about the distance. So, an ant at node i will choose the next edge with the following attractiveness:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha}{\sum_{k \in allowed_k} \tau_{ik}(t)]^\alpha} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where $\tau_{ij}(t)$ is the level of a pheromone on edge (i, j). $allowed_k$ is a set of possible transitions for ant k in its current state. The α parameter is used as in previous equations; its default value is $\alpha = 2.0$.

Good-at-conflict-handling ants (GC) These ants observe the others and care about all existing pheromones (the particular weights are to be determined experimentally). So, an ant at node i will choose the next edge with its attractiveness computed using the following pheromone equation:

$$\tau_{ij} = \left(14 \cdot \tau_{ij}^{(EC)} + 2 \cdot \tau_{ij}^{(AC)} + 2.5 \cdot \tau_{ij}^{(GC)} + 0.5 \cdot \tau_{ij}^{(BC)}\right)^\alpha \tag{3.8}$$

The default pheromone influence is $\alpha = 2.0$. The values assumed for distance influence β, pheromone influence α, and the influences of the particular parts of the pheromones on the attractiveness perceived by the GC ants were obtained experimentally and confirmed after being used in two publications [118, 142]. Later in this chapter, the adaptation of these parameters will be discussed.

Bad-at-conflict-handling ants (BC) These ants behave randomly, irrespective of pheromones or distances. So, an ant at node i will choose the next edge with the following attractiveness:

$$p_{ij}^k(t) = \begin{cases} \frac{1}{\#allowed_k} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

The proposed metaheuristic algorithm has been applied to solving different instances of TSP; the first results showing the prevalence of the methods for selected problems were shown in [118]. Later, thorough research was realized in order to seek the optimal parameters (for a given problem) of the algorithm [11]. Next, the research consisted of checking the possibility of self-adapting the actual structure of the population [126]. Finally, dedicated measures of the diversity were proposed [142].

3.1.4. Selected experimental results

The experimental results were obtained from dedicated software developed in Python[1] and run on the Zeus supercomputer[2] We considered the Traveling Salesman Problem: find a Hamiltonian in a graph defined by a network of cities, with the goal being a cycle with the least cost (distance) [143]. The quality function of course returned the length of the Hamiltonian. The instances used in the experiments were taken from TSPLIB library[3].

Configuration and infrastructure The Zeus cluster is a supercomputer consisting of different kinds of two-processor servers with different processor frequencies, numbers of cores, numbers of cores per node, and RAM memory per node. Experiments were run on a machine with the following technical parameters: **Model**: HP BL2x220c G5, G6, G7, **Total number of cores**: 17516, **Processors**: 2x Intel Xeon L5420, L5640, X5650, E5645, **Number of cores per node**: 8-12, **Processor frequency**: 2.26-2.66 GHz, **RAM memory per node**: 16-24 GB. The experiments consisted of running in parallel many instances of a tested, sequential algorithm in order to be able to repeat the experiments within a reasonable time.

The following platform configuration was assumed for each experimental run:

- Number of ants: 100.
- Number of iterations (in order): 20, 50, 100.
- Number of trials for each experiment: 30. The final data is the average of these 30 trials.
- Tested data taken from TSPLIB: berlin52, eil51, kroB200, eil76, kroA100, kroE100, pr76, st70, lin105, rat195, ts225, pr264.

During the experiment, the following compositions of the simulated population were considered (with respect to the proportions of different ant species):

- Classic Ant Population **(ca)** : Only ants acting as in the classic ACO.
- Human-inspired sample populations:
  - Control Sample Population **(humControl)**: 22% egocentric, 15% altercentric, 45% good at conflict handling, 18% bad at conflict handling.

[1] `www.python.org`
[2] `http://plgrid.pl`
[3] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

  - Increased Altercentricity Sample Population **(humAlter)**: 3% egocentric, 46% altercentric, 23% good at conflict handling, 28% bad at conflict handling.
  - Increased Good Conflict Handling Sample Population **(humGood)**: 6% egocentric, 6% altercentric, 63% good at conflict handling, 25% bad at conflict handling.

- Modifications based on human-inspired sample populations:
  - Equal **(eq)** population: 25% egocentric, 25% altercentric, 25% good at conflict handling, 25% bad at conflict handling.
  - Equal without bad at conflict handling **(eqWithoutBad)** population: 34% egocentric, 33% altercentric, 33% good at conflict handling, 0% bad at conflict handling.
  - Egocentrical **(ego)** population: 55% egocentric, 15% altercentric, 15% good at conflict handling, 15% bad at conflict handling.
  - Egocentrical without bad at conflict handling **(egoWithoutBad)** population: 60% egocentric, 20% altercentric, 20% good at conflict handling, 0% bad at conflict handling.
  - Altercentrical **(alter)** population: 15% egocentric, 55% altercentric, 15% good at conflict handling, 15% bad at conflict handling.
  - Altercentrical without bad at conflict handling **(alterWithoutBad)** population: 20% egocentric, 60% altercentric, 20% good at conflict handling, 0% bad at conflict handling.
  - Good at conflict handling **(good)** population: 15% egocentric, 15% altercentric, 55% good at conflict handling, 15% bad at conflict handling.
  - Good at conflict handling without bad at conflict handling **(goodWithoutBad)** population: 20% egocentric, 20% altercentric, 60% good at conflict handling, 0% bad at conflict handling.

- Homogeneous populations (in order to check extent of species synergy):
  - 100% egocentric.
  - 100% altercentric.
  - 100% good at conflict handling.
  - 100% bad at conflict handling.

In the first step of the experiment, our goal was to find the most promising population configuration. We tested all of the above-mentioned populations (besides the classic ACO).

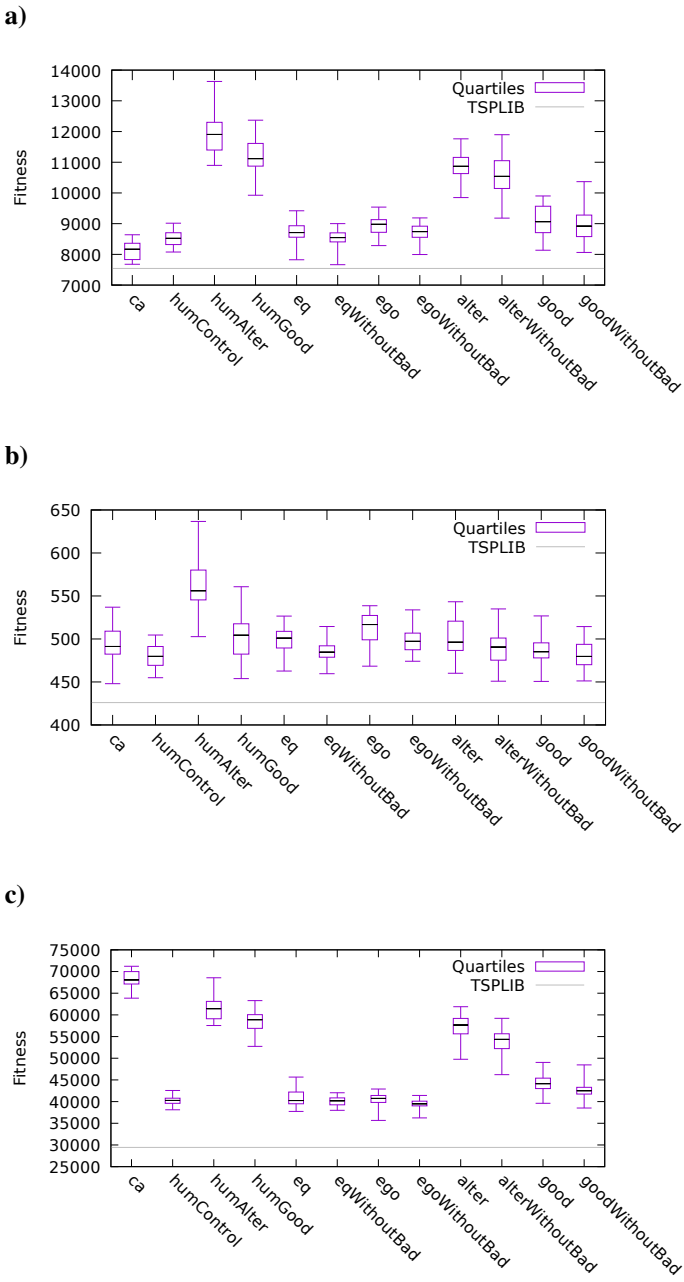

Figure 3.3. Fitness acquired in last iteration by each examined population for three tackled TSPLIB problems of different difficulties for the case of 100 ants and 100 iterations: a) problem **berlin52**, b) problem **eil51**, c) problem **kroB200**.

Table 3.1. Summaric results for each problem, comparing **egoWithoutBad** population and **classic ACO** population results (fitness, stdev, median, max, min) for 20, 50, and 100 iterations.

Iteration		**20 iterations**					**50 iterations**					**100 iterations**					**Best so far**
Instance	Algorithm	fitness	stdev	median	max	min	fitness	stdev	median	max	min	fitness	stdev	median	max	min	value
eil51	ca	478.99	15.34	474.03	507.18	448.42	491.93	15.13	489.02	525.50	463.08	494.15	19.41	491.28	536.96	448.00	426
	eWB	519.81	14.18	519.60	544.92	486.20	503.77	14.26	504.96	532.64	469.78	498.32	14.95	497.27	533.83	474.07	
berlin52	ca	11022.72	395.41	11039.33	11860.66	10048.46	9227.92	273.86	9294.71	9591.87	8336.74	8112.31	286.34	8167.93	8638.26	7677.66	7542
	eWB	**8757.05**	272.41	8768.28	9308.77	8130.56	**8763.54**	251.13	8751.44	9211.50	8063.69	8727.56	280.27	8740.42	9187.20	7994.74	
kroB200	ca	85493.27	2956.28	86323.39	89849.80	77795.71	80105.73	1650.93	80400.52	83895.09	74989.03	68184.50	1898.08	68066.80	71211.87	63857.67	29437
	eWB	**41154.82**	1048.77	41208.18	43309.33	38325.23	**40103.79**	1103.62	40124.60	41691.73	37220.13	**39455.89**	1074.52	39483.09	41424.30	36239.60	
eil76	ca	592.50	12.62	593.50	614.57	570.69	595.51	14.41	594.19	625.43	569.73	603.81	15.90	600.26	631.64	574.60	538
	eWB	678.96	22.02	682.98	707.50	616.87	657.52	17.56	661.72	689.51	619.89	645.41	13.93	644.36	673.51	618.29	
kroA100	ca	47085.66	2260.01	47603.09	51202.23	41132.88	42392.05	1474.85	42311.00	44470.66	38743.53	34173.37	1129.09	34235.51	36295.30	31460.71	21282
	eWB	**28090.20**	825.43	27962.27	29534.12	26268.45	**27806.59**	618.92	27790.66	28842.27	26687.32	**27483.31**	1031.67	27722.78	28712.37	24482.68	
kroE100	ca	47821.57	1861.18	48008.87	51651.63	42852.76	43036.32	1659.95	42967.44	46114.04	38046.19	34832.45	1366.99	34936.77	37365.10	31527.57	22068
	eWB	**28003.67**	1029.93	27858.90	29650.07	25300.71	**27575.23**	886.45	27617.61	29397.30	24976.15	**27356.84**	905.07	27433.92	29395.88	25732.59	
lin105	ca	29315.13	1286.47	29728.70	30780.70	25764.69	25132.15	1044.82	25262.40	27260.30	22900.89	20714.70	742.19	20793.44	22029.58	18956.58	14379
	eWB	**18754.85**	817.75	18956.95	19702.29	16295.73	**18441.99**	745.83	18316.22	19780.33	17097.03	**18262.87**	627.62	18159.40	19428.89	17044.86	
pr264	ca	134305.60	4576.44	134569.14	142567.66	122442.11	126263.89	5120.08	126053.25	134602.77	113578.57	112541.28	3786.08	113261.38	119005.07	100365.64	49135
	eWB	**69494.39**	2168.02	69514.77	76582.62	65531.74	**66513.92**	2290.60	67056.73	70811.54	62693.75	**66223.05**	2009.70	66303.07	70026.10	62333.61	
pr76	ca	222716.57	6737.87	223138.07	235414.64	203345.78	217027.41	8674.34	218211.88	233447.09	197292.25	200383.26	9507.43	200180.66	217390.34	176767.31	108159
	eWB	**133632.58**	4922.71	133640.96	144252.89	122087.23	**130152.10**	4622.92	129060.74	140814.70	121574.35	**130319.20**	4760.09	129943.33	142062.52	122624.22	
rat195	ca	4701.67	169.90	4738.26	4957.83	4310.48	2873.68	73.35	2878.64	3009.94	2689.53	2653.71	66.76	2649.12	2796.84	2481.44	2323
	eWB	**3169.40**	121.41	3180.48	3372.08	2803.98	2990.02	89.22	3005.12	3110.63	2810.15	2899.60	60.42	2905.48	2996.09	2784.33	
st70	ca	784.60	19.09	786.52	816.61	746.81	787.86	19.38	789.60	830.98	750.53	788.41	26.62	785.89	840.36	723.01	675
	eWB	872.87	22.98	880.16	910.50	817.82	842.41	25.01	841.32	892.36	789.38	841.68	23.56	842.31	888.89	783.94	
ts225	ca	470165.46	14501.94	472284.14	492594.79	429026.94	462478.86	15553.55	466419.57	482509.79	422507.27	444967.84	15368.94	447311.13	478072.14	411809.80	126643
	eWB	**165392.39**	4934.86	163219.77	174176.02	156692.07	**159771.84**	5287.88	159871.49	169159.16	146695.09	**158836.69**	4048.33	159712.68	165133.82	148808.70	

Three benchmarks were chosen in order to realize the experiments that would lead to assess each population's effectiveness:

- eil51 (best-known solution: 426)
- berlin52 (best-known solution: 7542)
- kroB200 (best-known solution: 29,437)
  The results of our test runs are shown in Figure 3.3.

It turned out that, in many tested cases, configurations without bad-at-conflict-handling ants (with the **"WithoutBad"** suffix in the population names) got our attention as quite effective ones; thus, we chose these configurations to opt for one for further examination.

Unlike small benchmarks (like eil51 – see Fig. 3.3b), where most of the tested populations gave almost the same final results, mid-sized and big benchmarks (respectively: berlin52 (see Fig. 3.3a) and kroB200 (see Fig. 3.3c) clearly demonstrated the low quality of certain configurations. As shown in Figure 3.3, the **egoWithoutBad** and **eqWithoutBad** configurations achieved the best fitness out of all of the tested socio-cognitive ones. Both these algorithms relied mostly on egocentric and altercentric ants and neglected the purely random component of the search (bad-at-conflict-handling ants). It seems that bad-at-conflict-handling ants introduce too much noise into the computation, instead of increase the diversity. Thus further experiments will be planned considering lower number of these ants, not necessarily equal to 0%.

In the general summary (considering each of the presented charts), the **egoWithoutBad** population appears to be a little better, especially when taking into consideration more-complex problems. Therefore, this configuration will be compared with the classic ACO.

Result summary Table 3.1 (on the interleaf) presents the results (fitness, stdev, median, max, and min) for each problem, comparing the **egoWithoutBad** population with the **classic ACO (ca)** population for the 20th, 50th, and 100th iteration steps. The bold font indicates the cases where the examined **egoWithoutBad** population turned out to be significantly better than the classic ACO. As one can see, the results clearly show that the socio-cognitive ants tested were better than those in the classic ACO in 6 cases out of 12. According to the well-known "no free lunch" theorem, this is an absolutely natural case; there is no "the best" optimization algorithm. This can be also perceived as a call for a further evaluation of the proposed metaheuristic algorithm and encourages the enhancement of other ant-like algorithms, not only the classic ACO with socio-cognitive inspirations.

3.1.5. Emergence of population structure in Socio-cognitive ACO

During our earlier experiments, we used an ad-hoc approach for setting the parameters of the socio-cognitive populations: either by looking for optimal configurations manually (e.g., checking the performances of the different compositions by trial and error) or by basing it on the data from the human population [103]. One of the best configurations found was a population of socio-cognitive ants containing mostly the so-called egocentric ants but omitting the totally random-working ants. This might have been a good choice; however, in order to verify this (or search for other good configurations of our metaheuristic), we had to either do some data-farming experiments (using the Scalarm data-farming system, for example[4]) or employing some kind of meta-algorithm (e.g., in order to evolve these parameters). There is also a third possibility: to allow the system to find these parameters by the means of emergence. This is the approach attempted here: to seek an optimal composition of a population for a given problem instead of following the trial-and-error approach.

Emergent behavior is defined in [144] as a phenomenon occurring suddenly in a complex system consisting of a sum of simpler entities. The behavior of the whole system can be more sophisticated that the behavior of the particular entities. Another definition can be found in [145], where the emergent behavior is described as a phenomenon that cannot be explained in a simple way based on an observation of the system as a sum of the simpler entities. Here, one should also refer to a classic cellular automata system (such as Conway's Game of Life) where the emergent behavior of the system can also be clearly observed by the complex structures emerging from the very simple rules [146].

Following these guidelines, we have proposed several strategies for the automatic adaptation of the population of the socio-cognitive entities (ants) to evolve towards an optimal composition of the computing population. Here, we describe these strategies and the outcomes of the evaluation experiments, which achieve a similar efficiency as those obtained in the previous research using an exhaustive search for the best configuration of the whole system. In the approach presented here, the optimal population composition arises emergently, so the search is not as tedious as in the earlier approach. Thus, the main contribution presented here does not consist of exceeding the up-to-date attainments in a search for the optimal parameters of a socio-cognitive system but rather in proposing ways to do this more easily and quickly. It must be noted that, similar to other metaheuristics, the Socio-Cognitive ACO is configured by a number of different parameters. Our approach relieves the user from the necessity to tuning them by trial and error.

Following the definition of emergence [144] as a phenomenon occurring suddenly in a complex system consisting of a sum of simpler entities and being inspired

[4] `www.scalarm.com`

by the behavior observed in cellular automata, for example (such as in Conway's Game of Life) and at the same time perceiving the whole computing system proposed here as a simulation of "living beings," we try to leverage the emergence phenomenon in order to properly configure the computing system without the need for doing data-farming-related experiments, checking thousands of different configurations of the system parameters, and looking for the only one that will still be impossible to be equally best for all of the considered problems (cf. no free lunch theorem [5]). Thus, we try to define the simple actions of individuals that affect the structure of the population, eventually hoping for stabilization and the attainment of good quality in the produced solution.

To sum up, the emergence in the multi-species Socio-Cognitive ACO is based on a dynamic exchange of the types of individuals when a certain condition is true (e.g., regarding the best solution observed in both species). Because the population consists of several species, all of the emergence strategies are based on the concept of the migrations of ants between them, changing the ants' species. The migration is realized when a certain condition (usually related to quality) is true, and the ants are chosen and moved to other species. Below, a summary of all of the proposed emergent migration settings is provided after [126].

Migration from worst to best species In this strategy, one ant belonging to the current worst species (including the ant with the worst solution from one iteration) changes its species to the one where the current best ant belongs. The condition for starting this strategy is decreasing the global result (the best solution of the ants belonging to all species) by a certain percentage.

Stepwise migration of one ant In this strategy, the migration is carried out in a more fluent way; because of this, the ants are "promoted" gradually. The species are sorted according to the best solutions belonging to them, and a certain ant is migrated from its current species to the species located one step above in the mentioned order. The condition for starting this migration is the same as in the case of the first migration strategy: the worsening of the best solution in the whole population by a certain percentage.

Stepwise migration of many ants In this strategy, many ants are migrated in a "stepwise" manner in a similar way to the previously described case of one ant. The number of migrated ants depends on the percentage of the worsening of the global result compared to the previous iteration (e.g., for a 3% worsening, 3 ants will be migrated, while for 0.5%, only one will be migrated). The condition for starting the migration is similar to the previous case.

Competition-based migration In this strategy, instead of considering the global result (and following its dynamical changes – either decreasing or increasing), the ants are migrated based on a predefined "competition" among them:

1. In an iteration when migration should arise, two species are randomly chosen from all of those available.
2. From these species, the better one (having the best current result) is selected.
3. Now, one ant from the worse species to the better one is migrated.

Migration in this method is run periodically, and the length of the period is one of its crucial parameters.

Stochastic migration In this method, certain probabilities are computed in each iteration:

- probability of leaving of one species by an ant,
- probability of joining a new species by an ant.

It is to note that the probability of leaving the best species (and joining the worst one) is equal to zero.

The probabilities of an ant leaving a certain species may be computed as follows:

1. For each i-th species (excluding the best one in the current iteration), a difference between the quality of its best individual $diff_i$ is computed according to the following equation:

$$diff_i = fitness_i - fitness_{best} \tag{3.10}$$

   where $fitness_{best}$ is the quality of the solution for the best of the species, while $fitness_i$ describes the quality of the solution in the current iteration for the species for which the difference is computed.

2. Next, for each i-th species (besides the best in the current iteration), the probability of an ant leaving this species is computed as:

$$p_i = \frac{diff_i}{\sum_{j \neq best} diff_j} \tag{3.11}$$

   being the fraction of the difference computed by Equation (3.10) and the sum of the differences for each of the j-th species (besides the best one).

The probabilities of an ant joining a certain species i (besides the worst one) are computed in an analogous way, while the equation for the difference of the quality is as follows:

$$diff_i = fitness_{worst} - fitness_i \tag{3.12}$$

where $fitness_{worst}$ is the quality of the solution in the current iteration for the worst of the species.

Having the probabilities computed, two species are chosen: the one that is about to be left by an ant, and the one that the ant will join.

Experiments on emergence in Socio-Cognitive ACO The experiments involved 100 iterations, the total number of the ants in the population was 100 (of course, the population consisted of dynamically-changing species), and the problem tackled was the Traveling Salesman Problem [143] using the selected classic TSPLIB[5] benchmarks.

The actual efficiency of the proposed emergent migration strategies tackling three problems of varying difficulties **eil51**, **berlin52**, and **kroB200** is described in [126], however, here we focus on the most difficult problem in this work (namely, **kroB200**). The emergent behavior of the ants is evaluated, and the efficacy of such a dynamically adapting metaheuristic is compared to the classic ACO and one of its best socio-cognitive versions (**egoWithoutBad** from [11] containing 60% EC, 20% AC, 20% GC, and 0% BC ants).

In all the graphs presented in this section, the labels of the ant species were used (AC, BC, EC, GC), CA stands for the classic ants, TSPLIB shows the best result found for this problem. Standard deviation showing the dispersion of the experiments was also shown.

Migration from worst to best species In this case, migration arises only when the global results worsen by a certain percentage while each iteration is observed. In Figure 3.4, the fitness as well as the number of particular ants in the species are presented, assuming a 2% worsening of the best quality between subsequent iterations.

In the case of the big problem (**kroB200**), the results of the emergent population are comparable to **egoWithoutBad**; however, they are significantly better than the classic ACO.

Observations of the percentage of particular species in this experiment reveals that the AC ants gradually dominated the other species.

In all of the experiments, the BC and EC ants are removed from the system by the 50th iteration. It seems that, in the case of the small problem, there is no need to employ complex techniques for perspective taking and inspiration on the other's solutions; simply following the others is good enough. It is quite self-explanatory: one does not need sophisticated methods to solve simpler problems. At the same time, for bigger problems, more-sophisticated methods prevail, especially in the case of the biggest problem tackled.

[5] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

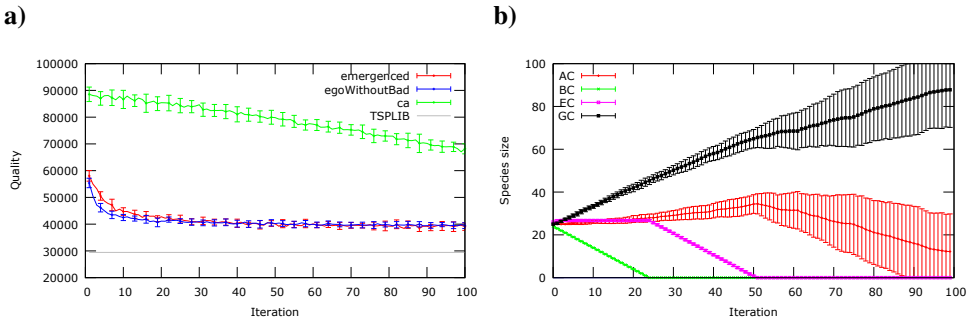

Figure 3.4. Quality (a) and structure (b) of the population for problem **kroB200** – migration from worst to best species with 2% decrease in quality.

Stepwise migration of one ant The emergent behavior in the stepwise migration starts when the global quality drops between the iterations by a predefined percentage. In the tested instances, a 2% worsening was assumed.

In Figure 3.5, the results showing the quality and percentage structure of the population are presented for the small problem. For this instance (similar to the previous experiment), the stepwise migration is a little better than with the competitive algorithms. In the medium-sized problem, the results are again similar to the previous setting, as the advantage is lost at the end of the computation (at around the 70th iteration). For the biggest instance tested, the results of the emergent and socio-cognitive populations are again very similar, and they are both much better than the classic ACO.

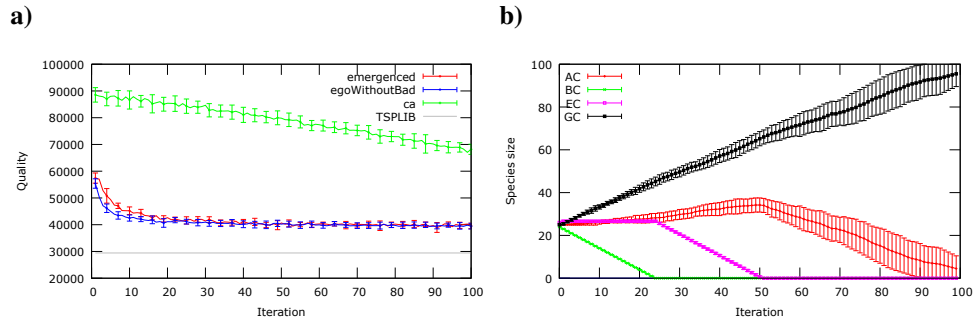

Figure 3.5. Quality (a) and structure (b) of the population for problem **kroB200** – stepwise migration of one ant assuming 2% decrease in quality.

The percentage of particular species is again quite similar to that of the previous setting; moreover, just like in the previous experiment, the AC ants prevail for the smallest problem, while for the bigger problem, the more sophisticated GC ants dominate the population. Therefore, the conclusions resulting from the previous experiment may be repeated here: complex problems need more-sophisticated solutions.

Stepwise migration of many ants Similar to the concept of the stepwise migration of one ant, emergent migration is possible in this case when the best quality decreases by a certain percentage value. However, the number of migrating ants currently depends on the extent of the quality decrease. In the tested cases, the decrease value was 2%.

In this case (see Fig. 3.6), the optimization of the smallest problem resulted in achieving a little better result than in the competitive algorithms (contrary to the migration of one ant); however, these results become visible starting from the 40th iteration. In the case of the medium-sized and big problems, emergence yields practically the same result as **egoWithoutBad**; however, it fares much better than the classic ACO.

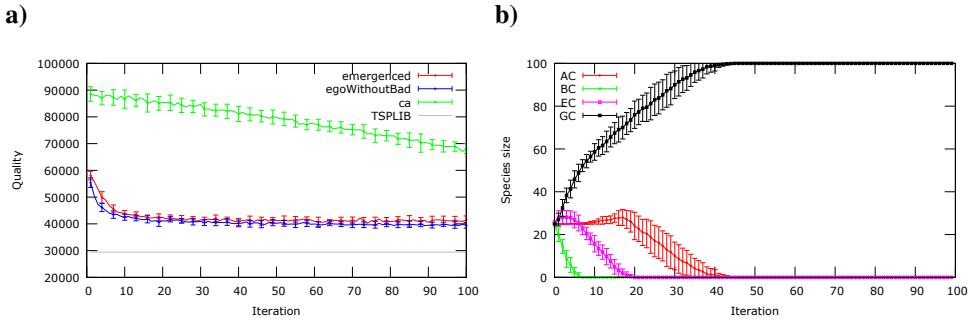

Figure 3.6. Quality (a) and structure (b) of the population for problem **kroB200** – stepwise migration of many ants assuming 2% decrease in quality.

It is to note that, in the case of the small and medium-sized problems, the configuration of the population becomes more or less stable (around 50% of the GC ants and AC) starting from the 30th iteration. At the same time, it seems that the GC ants prevail very quickly with the big problem and dominate the other species. In the first two experiments, the stability of the population structure is probably an effect of exchanging more ants (as compared to only one ant in the previous experiments). Moreover, the number of exchanged ants depends on the decrease of the quality. Therefore, it seems that achieving a certain stability is easier here. This observation is false for the

biggest problem, however – here, the GC ants prevail very quickly and yield much better results than in the case of the classic ants; however, they are the same as with **egoWithoutBad**.

Competition-based migration In this case, the emergence consists of running a tournament between the ants that are about to change species periodically. The frequency tackled in this experiment is two steps.

In the current case (see Fig. 3.7), the outcome is generally the same as in the previous cases; however, the situation in the population structure is quite different. It seems that nearly all of the species are present through the final iteration – only the random ants (BC) become extinct. The apparent diversity is caused by the type of emergence used, and the diversity of the population should assure the diversity of the search. Therefore, this emergence method should be one of the most preferred when tackling difficult problems.

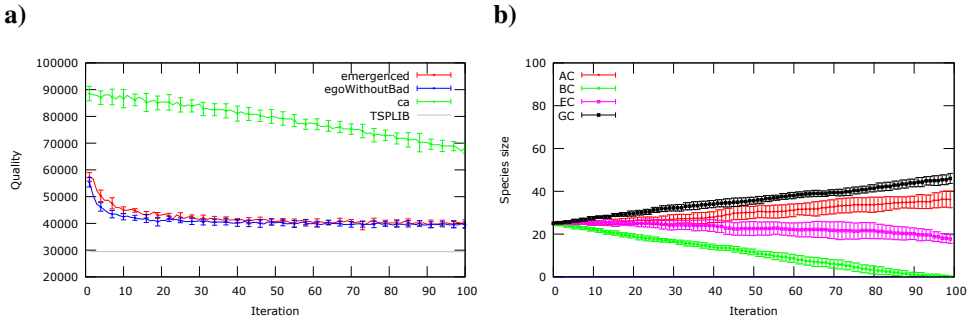

Figure 3.7. Quality (a) and structure (b) of the population for problem **kroB200** – competition-based migration with period of 2.

Stochastic migration According to the idea of stochastic migration, one of the ants leaves a randomly selected species in each iteration and moves to another (also randomly chosen).

In Figure 3.8, one can see the results of emergence with stochastic migration. It is clear that, in the case of first problem tackled, the emergent population prevailed in the end. For the next two problems, the results of the emergent population are the same as in the case of **egoWithoutBad**.

The population structure again shows that, in the simplest case, the AC ants are the most important species, while for the more complex problems, the GC ants prevail.

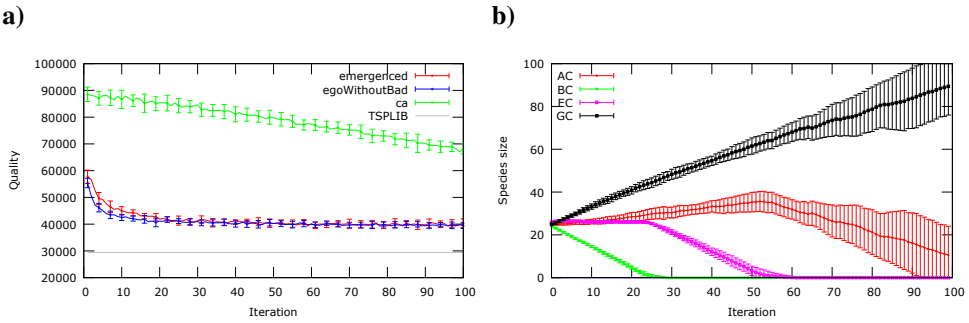

Figure 3.8. Quality (a) and structure (b) of the population for problem **kroB200** – stochastic migration

Summary For simpler problems, the results from all of the emergence strategies used were comparable with classic ants and **egoWithoutBad**; in several cases, the emergent population yielded better performances.

Observing the emergence strategies and compositions of the populations in the cases of the three problems tackled, it turns out that, for the simple problems, the AC ants following simple search algorithms performed better; however, for more difficult problems, more-sophisticated GC ants are required. One should compare this observation with the composition of the population for **egoWithoutBad**, where the most numerous species was EC (with AC and GC accounting for 20% each).

To sum up, there are many nearly equally optimal results of the search for the best composition of a population. However, it is not necessary to do such a search manually using the trial-and-error approach; instead, it is enough to define the emergence strategies and wait for the feasible configuration to stabilize.

3.1.6. Summary of Socio-Cognitive ACO research

In this section, we have presented the idea of a Socio-Cognitive ACO. Later, the social cognitive learning theory was referenced in order to build a proper perspective on the proposed metaheuristic. During the introductory experiments, we focused on arbitrarily chosen population configurations in the Socio-Cognitive ACO. First, they were inspired by real-world population configurations, then designed by hand based on the experiences gathered during researching this new metaheuristic. We have checked the efficacy of solving TSP with different instances of the Socio-Cognitive ACO, and many cases were identified when the proposed metaheuristic prevailed. In particular, the **egoWithoutBad** population turned out to be one of the best configurations tested.

One has to remember that metaheuristics are usually complex and require a large number of parameters to be adjusted before it can solve a certain problem efficiently. Seeking the optimal (or rather quasi-optimal) values of these parameters is usually realized by a trial-and-error approach, which can be tedious. Therefore, in the second part of the chapter, we have presented an enhancement of the Socio-Cognitive ACO by introducing emergence mechanisms for automatically configuring the population composition in terms of the ant species to optimize the performance. The emergence mechanisms are based on observing the quality value in the population and taking other actions that exchange ants among the species. The efficacy of these proposed mechanisms was tested again with several selected benchmark functions from the well-known TSPLIB library.

The results show that the emergence mechanisms yield an outcome very similar to that obtained from manual tuning. In particular, the **egoWithoutBad** population (which was found to be the best composition in our previous research) was easily reached by the emergent migration mechanism proposed here, thereby relieving the user of the tedious trial-and-error approach.

Besides ACO, other stigmergic or quasi-stigmergic systems such as Particle Swarm Optimization were considered as starting points for simulations (see the next section). One can study how incorporating socio-cognitive features might enhance their performance.

Browsing through the presented results, it is clear that, since the socio-cognitive ACO produced good results for many problems (when compared to its classic predecessor), future work should focus on enhancing other much better alternatives of ACO (e.g., Max Min Ant System or Ant Colony System) with socio-cognitive mechanisms.

3.2. Enhancing Particle Swarm Optimization with socio-cognitive inspirations

Many algorithmic techniques in computer science are inspired by natural and biological processes. For example, the decomposition of the populations in evolutionary and similar computing techniques popularized such solutions as the island model of evolution [72], which effectively enhanced the diversity of computing metaheuristics. In the same vein, hybrid algorithms incorporating memetic, immunological, and hierarchical processes have been implemented [33]. In our research project, we are following the same approach by bringing different inspirations together to yield interesting algorithms that are always needed (as stated by the famous No Free Lunch Theorem [5]).

In our earlier research (see Chapters 3, 3.1.5), we experimented with socio-cognitive Ant Colony Optimization (ACO) metaheuristics [118, 142] by introducing a number of sub-species into a classic ACO along with positing the inter-relations among them and by leveraging the stigmergic relationships among these artificial ants. These relationships incorporated the ants' abilities to change perspective, to get inspired by the solutions found by other species, and so on. [118]. This approach produced efficient results in discrete optimization (solving different instances of TSP).

Our goal here is to extend the proposed socio-cognitive inspirations by applying them (to the possible extent) to Particle Swarm Optimization (PSO). Our motivation for attempting this extension is as follows. In the case of ACO, our goal was to build a common knowledge base about the solutions found by different species and share them. This was relatively easy to implement because the "derivation" of this knowledge was present in the pheromone table. In the case of PSO, this knowledge about the global and local current optima is also quite natural, present, and easily accessible in the system. Therefore, we plan to introduce various species that will take different inspirations from the current optima of the other species and test the resulting algorithms on common benchmark functions.

In the next section, we provide a brief background, followed by a description of the socio-cognitive inspirations that are the basis for our approach. Then, we introduce the classic PSO and describe the socio-cognitive modification of this algorithm. Finally, we present the results of our experiments to evaluate the resulting algorithms, followed by our conclusions.

This section recalls the most important ideas and results presented in:

- Bugajski et al. [12] in order to properly describe the second socio-cognitive metaheuristic algorithm, namely Socio-Cognitive PSO (SC-PSO),
- Bugajski et al. [147] in order to properly describe the automatic adaptaion of the population structure of SC-PSO.

3.2.1. Selected hybrid metaheuristics based on PSO

A number of modifications and hybridizations of the PSO exist that were analyzed and treated as inspiration for the proposed algorithm. For example, the classic PSO was enhanced with a momentum factor [148], making the current velocity of the particle dependent on its previous values. In this way (similar to the momentum factor in neural network training), the exploration and exploitation capabilities are controlled. Increasing the momentum will make it possible to escape the local minima, and decreasing it – finding more accurate results (though, being prone to a lack of diversity). The proposed approach does not consider any historical data while guiding the search, but this is an interesting idea to explore in further research. Another

modification [149] considers the notion of a neighborhood, limiting the search for the particle with the best fitness. In the subsequent iterations, the velocity of the particle is computed based on its best fitness, the best fitness of the whole swarm, and the best fitness perceived in the neighborhood. This approach can enhance the diversity observed in the whole search. The neighborhood can be treated as a predecessor for the species used in the proposed approach. In [150], the authors propose the introduction of different species into PSO to tackle the multi-modal optimization problems. In this approach, there are many parallel-acting independent swarms that focus on the localization of different local minima. It is to note that the approach proposed here also divides the population into species, yet the interaction between them is planned and seems to be quite natural.

The evolutionary PSO attempts to hybridize the classic PSO with evolutionary algorithms [151]. The particles in the swarm replicate, and their parameters are treated as a genotype undergoing mutation. The dynamism of the population observed in this algorithm is somewhat similar to that observed in the proposed algorithm.

The adaptive PSO [152] is based on an analysis of the actual state of the population (there are rules for checking whether it is in the exploration phase, exploitation phase, or several other possibilities). The PSO parameters are changed to ones specific for a certain observed state. This is another approach that may be worth hybridizing in the future, though one must remember that testing the current state of a population against any rules is computationally intensive and may render the whole system inefficient if not realized appropriately.

Parallel Comprehensive Learning Particle Swarm Optimizer [153] is a novel approach extending the idea of multiple swarm optimization. In this modification, the search space is populated with dependent plural swarms based on the master-slave paradigm. The swarms work cooperatively and concurrently.

In the past, there have been attempts to decompose PSO into sub-populations and sub-species. Here, we would like to mention a few such notable approaches. In [154], the authors demonstrated a cooperative PSO: they employed many swarms and made them optimize different parts of the solution vector in cooperation. In another study ([155]), the authors modified PSO by introducing different swarms that shared information about their best solutions in order to escape the local extrema. Improved results were obtained in continuous optimization using this approach. In [150], the author extended the original PSO by dividing the particle swarm spatially into multiple clusters called species in a multi-dimensional search space. Each species explored a different area of the search space to find the global or local optima along that dimension. In [156], the number of particles in the sub-swarms were dynamically adjusted, yielding a competitive performance with respect to the examined benchmarks.

It is to note that the multi-species PSO makes a good starting point for the further hybridization of this potent metaheuristic. The approach presented here provides one such extension.

3.2.2. From perspective taking to enhancing PSO

Perspective taking is a core socio-cognitive ability that allows for smooth social interactions between humans. There is substantial variability in a human's perspective-taking performance. Part of the diversity can be explained by the relative weight that an individual gives to his/her own perspective relative to the perspective of other surrounding people [101]. Thus, some individuals give more weight to their own perspective (egocentric individuals), while other individuals give more weight to other people's perspectives (altercentric individuals).

In [118], we have explored the usefulness of human perspective-taking characteristics on the search capabilities of ACO, enhancing it by introducing different sub-populations; e.g., *Egocentric individuals* (focused on their own knowledge) or *Altercentric individuals* (perceiving the actions of other ants and using them as inspiration). Such complex populations achieved interesting results in discrete optimization (namely, solving TSP), increasing the efficiency in many of the studied instances.

As another swarm intelligence algorithm, PSO seems to be a natural candidate for introducing a socio-cognitive mechanism. In the case of ACO, the perspective was taken based on the different pheromone trails left by particular ant species, pushing the ants towards the sub-optimal solutions found by the pheromone-depositing ant species from the population. However, in the case of PSO, the mechanism of perspective taking may be implemented more easily. The act of following different current optima – best for the particular individual, best in the neighborhood, best globally – will be elaborated later, resulting in a metaheuristic system that is conceptually similar to the Socio-Cognitive ACO.

3.2.3. Socio-cognitively-inspired PSO

The classic implementation of the PSO presented above makes the algorithm very likely to become stuck in the local optima. Once each particle reaches its optimum state, the swarm stabilizes, making it impossible to leave this state. In such a situation, the best position of each particle, its neighborhood, and the whole swarm will be located at the same point, and the calculated velocity will always be 0. This situation is caused by the low number of parameters affecting a particle's behavior. Each particle is influenced by three factors – the best positions of the particle, the swarm, and the neighborhood – that affect the particle equally. This situation can be rectified by dividing the particles into different classes called species.

Each species has its own algorithm for calculating velocity. The differences between the algorithms for different species lies in the different weight settings for each particle species. Moreover, a neighborhood is introduced, being a subset of a swarm (in order to gradate the perspective taking, or inspiration of one particle by other ones into three levels, local, neighborhood and global). It is associated with n-dimensional vector $A_n = (x_1, x_2, \ldots, x_n)$, while $n \in \mathbb{N}$.

The execution of the SC-PSO algorithm begins by initializing the start values. Each particle P belonging to swarm S and in neighborhood N is initialized with the following values:

- Position X of particle P is initialized with a random vector belonging to search space D.
- The best-known position is initialized with current particle's position: $A_p := X$.
- Velocity V of particle P is initialized with a random vector belonging to search space D.
- The swarm's best position is updated by the following rule: $iff(A_p) < f(A_s)\ then\ A_s := A_p$
- the neighborhood's best position is updated by the following rule: $iff(A_p) < f(A_n)\ then\ A_n := A_p$.

Just as it is in the case of PSO, once all of the particles are initialized and uniformly distributed in the search space, the main part of the algorithm starts executing. During each iteration of the algorithm, the steps in the Pseudocode 10 are executed.

Pseudocode 10 Pseudocode of Socio-Cognitive Particle Swarm Optimization algorithm

for `each particle P in swarm S` **do**
  `update particle's position:`
    $X := X + V$
  `update particle's velocity:`
    $V := a(A_s - X) + b(A_n - X) + c(A_p - X) + \omega V; a, b, c, \omega \in [0, 1]$
    where ω is the inertia factor
  `update global best positions:`
    $iff(A_p) < f(A_s) then A_s := A_p$
    $iff(A_p) < f(A_n) then A_n := A_p$
end for

In SC-PSO, as described above, a particle's velocity is determined by the following equation:

$$V := a(A_s - X) + b(A_n - X) + c(A_p - X) + \omega V; a, b, c \in \{0, 1\}.$$

Changing the values of parameters a, b, and c modifies the impact of the three factors while, at the same time, forces the individuals to perceive the results achieved by the other PSO sub-species. A further step is to incorporate the perspective-taking inspiration: namely, the act of receiving inspiration from the results achieved by the other sub-species.

The species described here were created by choosing different configurations of parameters a, b, and c (see Tab. 3.2):

- *Normal*: the classic PSO, where the particle's decisions are affected by the particle's optimal solution, the neighborhood's optimal solution, and the global optimal solution, but each of these is given the same weight.
- *Global and local* This species is influenced only by its own best and the global best positions.
- *Global and neighborhood* This species calculates its velocity by combining the influences of the swarm's and neighborhood's best positions.
- *Local and neighborhood* This species ignores the global best solution and only takes into account the local bests – the neighborhood's and its own best positions.
- *Global only, Local only, Neighborhood only* These three species of particles are influenced by only a single factor: by it's own, the neighborhood's, or the swarm's best position, respectively.
- *Random* The last species of particles introduces randomness into the algorithm. It takes into consideration all three parameters, but the relative weights of the parameters are randomly reassigned during each iteration (the sum of the weights is always equal to 3).

The weight calculation is described by the Pseudocode 11.

Table 3.2. Configurations of socio-cognitively-inspired sub-species for different decision factors

No.	Species	Swarm [a]	Neighbor [b]	Particle [c]
1	Normal	1	1	1
2	Global and local	1	0	1
3	Global and neighborhood	1	1	0
4	Local and neighborhood	0	1	1
5	Global only	1	0	0
6	Local only	0	0	1
7	Neighborhood only	0	1	0
8	Random	random	random	random

Pseudocode 11 Calculation of weights

$a = random(0, 3)$
$b = random(0, 3 - a)$
$c = 3 - a - b$

While conducting experiments with different PSO configurations, we consider populations that contain 'pure' species (only one kind of species) as well as those containing various proportions of different species. This allows us to study the optimization potential of the proposed metaheuristics.

The Socio-Cognitive PSO algorithms were applied to solving classic multidimensional optimization problems. The results showing their prevalence for the selected instances were presented in [12], while the auto-adaptation of the population structure of the particles was later presented in [147].

3.2.4. Experiments on Socio-Cognitive PSO

The experimental results presented here were obtained using one node of the Zeus supercomputer (Intel Xeon HP BL2x220c, 23 TB RAM, 169 TFlops total computing power). During the experiments, the well-known benchmarks (namely, the Rastrigin, Griewank, Rosenbrock, and Ackley [157] functions) in 100 dimensions were executed. The search space was a 100-dimensional hypercube limited in each dimension to a range of [-100,100]. The experiments were repeated 30 times. Each experiment for the Rastrigin and Rosenbrock benchmarks consisted of conducting $3 \cdot 10^6$ iterations of the algorithm. The experiments for Griewank and Ackley consisted of $3 \cdot 10^6$ iterations. The supercomputing power was used to run many sequential experiments in parallel, in order to obtain repeatability-related results within a reasonable time.

All of the experiments were conducted for 25-particle swarms consisting of different species. The particles were grouped in one-dimensional neighborhoods of a size of 5, so each swarm was split into 5 neighborhoods each containing 5 consecutive particles. In each graph presented in Figures 3.9 and 3.10 (on the interleaf), the proportion of the examined species in the tested population is shown at the bottom, with all of the other particles being divided equally among the remaining species. Each species was examined by running simulations for five different swarms. Each swarm contained 0, 4, 11, 18, or 25 particles of the examined species and an equal share of the remaining species. For example, if the *normal* species is being examined, 100% means that the population contains only classic PSO particles, while 44% means that the population consists of 11 classic PSO particles, with the remaining 14 particles assigned equally to the 7 remaining species.

In this way, we searched for the efficient configurations of the whole system, checking them against the executed benchmarks, being aware that (as suggested by the well-known *No Free Lunch Theorem* [5]) no one single configuration can prevail for all possible optimization problems.

Examining the graphs that display the best qualities observed for the different configurations of species (Figs. 3.9 and 3.10) for optimizing the Rastrigin function, we can see that, in almost all of the experiments, the examined proportions of the species (44%, 16%, and 72%) yielded a performance close to the optimum. The best-achieved results for the Rastrigin function were around 4700. Of course, this was still far from the global optimum; however, it was much closer to the global optimum than the classic PSO (which got stuck at $35\,129.23$).

3.2.5. Adaptation of Population Structure in Socio-cognitive PSO

The diversity is often enhanced in population-based metaheuristics by the decomposition of the main population into sub-populations (cf. parallel evolutionary algorithm). Another popular approach is introducing niching and speciation techniques such as crowding or fitness sharing [158]. It is to note that the decomposition of the population may be realized without geographical separation; moreover, the sub-populations (in this case, sub-species) can affect each other when striving to reach a common goal.

In our earlier research, we proposed a socio-cognitive computing approach [118, 142, 12] based on introducing different species into the population of a metaheuristic algorithm (in particular, Ant Colony and Particle Swarm) and allow these species to take inspiration from the work of the other ones. Thus, the socio-cognitive research leads to the definition of the concept of "perspective taking," and thanks to the observation of different interactions about the individuals (and even populations of individuals), novel optimization algorithms can be abstracted and applied based on the well-known classic one. This approach was aimed at reaching a better diversity in the population during the search, and has proven beneficial for many tested benchmarks [118, 12].

In the beginning of our research, the structure of the computing population (both in the ACO and PSO cases) was arbitrarily selected. This was an apparent conflict with real-life populations, where a certain adaptation of the number of individuals is imminent because of the existence of the phenomena of death and reproduction. Another important matter is that the structure of the population gravely affects the capabilities of exploration and exploitation, which are crucial for optimization ability. Therefore, we propose several methods of the self-adaptation of the number of individuals in Socio-Cognitive PSO computing and evaluate their efficiency.

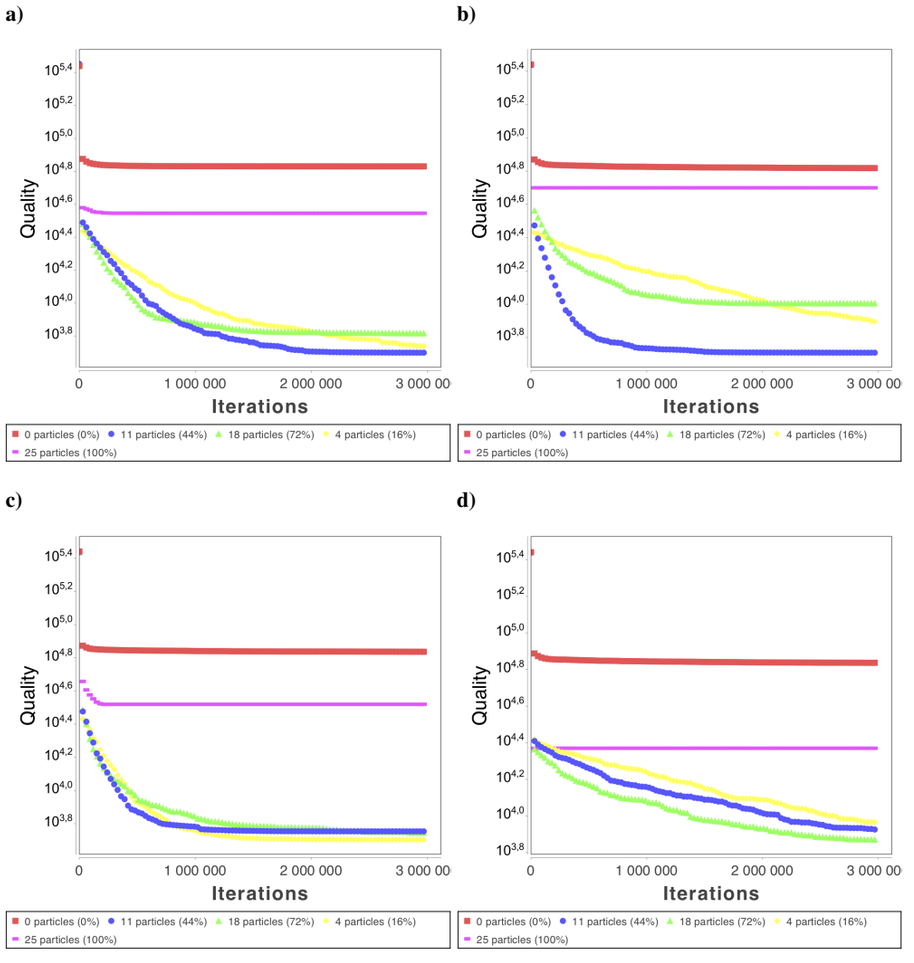

Figure 3.9. Different fractions of socio-cognitive subpopulations optimizing Rastrigin function: a) normal, b) global and local, c) global and neighborhood, d) local and neighborhood.

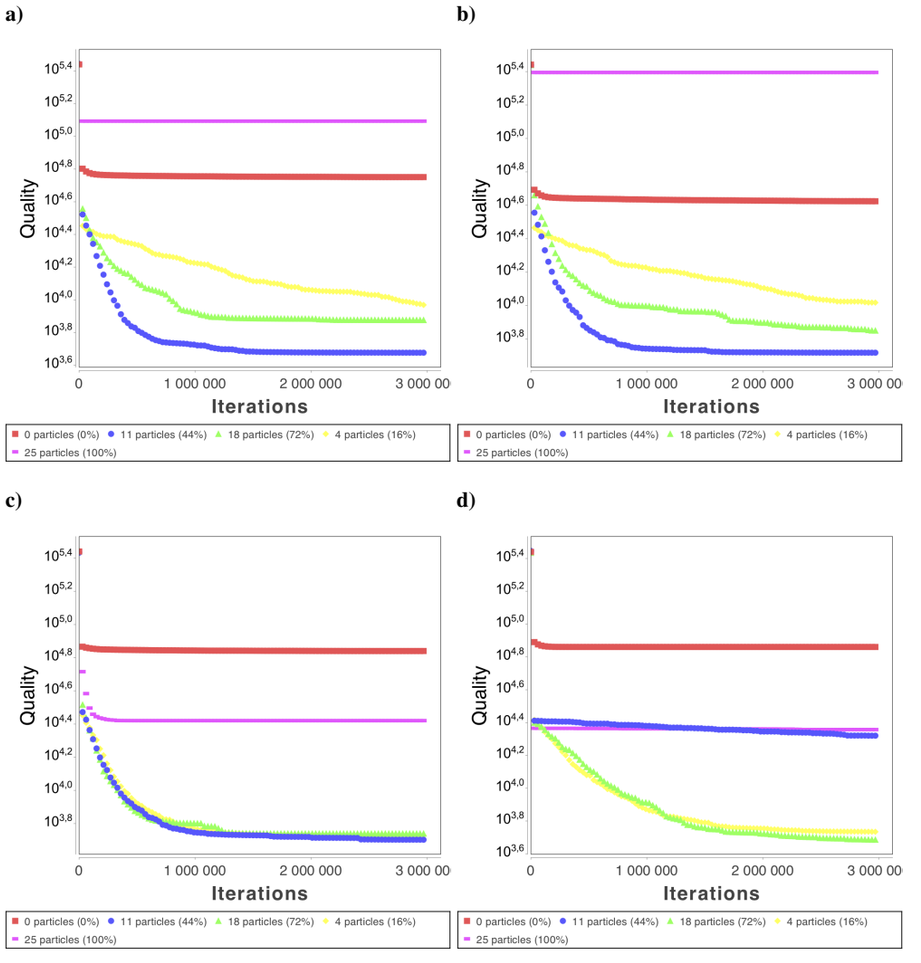

Figure 3.10. Different fractions of socio-cognitive subpopulations optimizing Rastrigin function: a) global only, b) local only, c) neighborhood only, d) random

Adaptation of Population Structure in Socio-cognitive PSO The results shown in [12] tackled static swarms. Finding an optimal configuration of such a complex system with so many degrees of freedom requires a high number of tedious tests. However, we instead tried to define dynamic rules of the adaptation of the population structure considering the efficiency of solving the current problem, thus constituting a self-adaptive system. This self-adaptation is designed in two phases: in the first one, a ranking of the species is constructed, and in the second, the chosen species (based on the previously selected ranking) exchange some of their individuals.

Ranking of species The ranking of species that is constructed before the actual exchange of individuals may be perceived as a certain permutation of the eight defined species based on criteria such as the number of particles or the achieved results. The ranking is constructed periodically (after a certain number of algorithm iterations) according to the one of the following strategies:

- **DefaultOrderFunction**: the same as the initial, arbitrary order, presented in Table 3.2, treated as a starting point and a reference for the adaptation.
- **RandomOrder**: random permutation of all eight species.
- **NumberOrder**: this ranking considers the number of particles in each species, with those having the lowest number being preferred in order to retain balance in the whole population.
- **BestLocalOrder**: this ranking considers the best fitness located in each species; the species are ordered from the best ones to the worst.
- **BestWorstLocalOrder**: this ranking is constructed similar to the previous one; this time, it considers the worst fitnesses, making a possible transfer of the worse particles to a better species in order to seek better solutions.
- **BestAvgLocalOrder**: this ranking is similar to the **BestLocalOrder**; however, it considers the average value of the fitness for the species, making it possible to assess the whole species, not only its individual particles.

Choosing particles to be moved Based on one of the chosen orders, the choice of particles to be moved between species is to be made. The strategy considers the current ranking (see Section 3.2.5) of species and the fitnesses of the particular individuals; it is realized as follows:

1. Choosing the source species.
2. Choosing the target species.
3. Choosing the particular particle to be moved from the source to the target species.
4. Moving the particle between the species.

The source species is chosen randomly from the non-empty species (omitting the first one in the ranking). The target species is chosen based on its ranking; it must be higher than the source species in the ranking. There may be more than one instance of moving particles during one change (N parameter). The particle may be moved to a species that is further than one step in the ranking (R parameter). The actual strategy may be described using Pseudocode 12.

Pseudocode 12 Execution of particle-choosing function

for `i in 1..N` **do**
  Select species with at least one representative:
    $source \leftarrow notEmpty(species)$
  Remove species ranked at the first place from the list:
    $remove(source,\ ranking[1])$
  Select a random source species:
    $I_S \leftarrow random(size(source))$
    $S \leftarrow selectRandom(I_S)$
  Select a random destination species from a higher ranking place than the source species:
    $I_{max} \leftarrow index(ranking,\ S)$
    $I_{min} \leftarrow max(1,\ I_{max} - R)$
    $I_D \leftarrow random(I_{min},\ I_{max})$
    $D \leftarrow ranking[I_D]$
  Select a particle from the source species:
    $P \leftarrow select(S)$
  Perform the shift:
    $shift(P, S, D)$
end for

The possible functions that actually choose the particles are described below:

- **DefaultShiftFunction**: this function does not change the species at all; instead, it is used as a reference point for testing other functions.
- **BestLocalShift**: this function chooses the particle having the best fitness in the species.
- **WorstLocalShift**: this function chooses the particle having the worst fitness in the species.
- **RandomShift**: this function randomly chooses a particle from the species.

The mechanism of changing the species is in several ways similar to the one used in Evolutionary PSO (EPSO) [151]. In both systems, the weights used for determining the velocity of the particles are adapted during the simulation. An obvious difference may be observed during the setting of the new weights (one of eight species in

the case of the proposed algorithm), while the mutation is applied in the EPSO based on randomizing the weights that are constant during the whole life of the particle. Besides this, EPSO, contrary to the proposed system, requires an additional evaluation of the accuracy functions that may influence its efficiency for the complex problems.

Experiments on adaptation of population structure in Socio-Cognitive PSO The examined system was implemented in Java and run on one of the nodes of the AGH Cyfronet Zeus supercomputer (HP BL2x220c, Intel Xeon, 23 TB RAM, 169 TFlops). Each experiment was repeated 30 times. The tested problems were Rastrigin, Rosenbrock, Ackley, Styblinski-Tang, and Schwefel [157] defined in $[-20, 20]^{100}$ hypercube. The maximum number of iterations was 5000 for each experiment, and the adaptations of the population structure were realized every 100 iterations.

Results obtained for dynamic changes of population structure In each of the tested cases, the starting configuration of the swarm consisted of 25 particles; i.e., four particles of the normal species and three particles for each of the other species. During each adaptation, four changes reaching no farther than two positions in the species ranking were made. The average outputs of the experiment are presented in Table 3.3.

Of course, as it was seen in [12] for the static configurations, there is no optimal configuration (cf. [5]). For different optimization problems, different species rankings and particle-choosing strategies turned out to be the best. However, a clear and simple observation done for all of the tested cases is that lack of changes in the population structure shows that it is worse than the adaptation (cf. the results produced while using **DefaultShiftFunction**).

In the case of the Rastrigin function, the best configuration was applying **BestLocalOrder** and **WorstLocalShift**. Thanks to this configuration, the particles with worse fitnesses got a second chance by moving to a better species.

The optimization of Rosenbrock required to also choose **BestLocalOrder** along with **RandomShift**. Here, the obtained results were twice as good as those obtained in the case of the static population. However, the results obtained for this benchmark appear to be very large. This is caused by the considerable size of the search space, which is significantly larger than commonly used sizes.

Tackling Ackley and Schwefel required us to use **BestAvgLocalOrder** and **RandomShift**. Similar results were obtained when using **RandomOrder** with **RandomShift**. This configuration guarantees high diversity in all stages of the simulation.

Table 3.3. Comparison of final quality obtained for different species shift strategies

Rastrigin				
	Random Shift	**Default Shift**	**WorstLocal Shift**	**BestLocal Shift**
BestWorstLocalOrder	1377.52	1554.03	1369.91	1301.65
BestLocalOrder	1323.13	1529.74	**1270.73**	1367.3
BestAvgLocalOrder	1332.67	1501.72	1372.2	1335.81
NumberOrder	1400.13	1562.09	1394.5	1373.37
DefaultOrderFunction	1303.98	1507.78	1326.13	1285.67
RandomOrder	1398.1	1562.08	1376.07	1319.65
Rosenbrock				
	Random Shift	**Default Shift**	**WorstLocal Shift**	**BestLocal Shift**
BestWorstLocalOrder	2239729.98	3865859.54	2095901.25	2256950.39
BestLocalOrder	**1806336.23**	3711206.89	2211100.19	1959016.92
BestAvgLocalOrder	2167858.42	3585491.72	1918816.53	1885024.62
NumberOrder	2771812.88	3684251.84	2715837.91	2507387.31
DefaultOrderFunction	2027358.76	3739261.87	1976868.6	2099273.05
RandomOrder	1854098.48	3695230.64	2140310.31	2172197.33
Ackley				
	Random Shift	**Default Shift**	**WorstLocal Shift**	**BestLocal Shift**
BestWorstLocalOrder	10.62	11.44	10.58	10.63
BestLocalOrder	10.57	11.31	10.45	10.49
BestAvgLocalOrder	**10.28**	11.4	10.59	10.68
NumberOrder	10.83	11.3	11.02	10.92
DefaultOrderFunction	10.52	11.43	10.54	10.47
RandomOrder	10.3	11.44	10.54	10.38
Styblinski-Tang				
	Random Shift	**Default Shift**	**WorstLocal Shift**	**BestLocal Shift**
BestWorstLocalOrder	6506.97	19180.38	**4986.73**	8574.16
BestLocalOrder	6079.08	18710.99	5627.31	6277.09
BestAvgLocalOrder	7111.71	18449.19	6131.63	5085.29
NumberOrder	16808.14	19803.32	14411.84	16649.83
DefaultOrderFunction	8314.22	18935.85	5290.14	7190.57
RandomOrder	6348.9	20551.64	7170.19	6155.77
Schwefel				
	Random Shift	**Default Shift**	**WorstLocal Shift**	**BestLocal Shift**
BestWorstLocalOrder	40550.47	40570.08	40555.87	40561.78
BestLocalOrder	40435.58	40600.25	40454.22	40436.46
BestAvgLocalOrder	**40396.07**	40541.43	40617.07	40460.58
NumberOrder	40453.62	40569.09	40454.54	40403.49
DefaultOrderFunction	40547.99	40541.05	40519.08	40467.93
RandomOrder	40437.49	40580.44	40431.44	40475.01

Optimizing the Styblinski-Tang benchmark showed the best results when compared relatively to all of the benchmarks. The best configuration for this case was the use of **BestWorstLocalOrder** with **WorstLocalShift**, which turned out to be more than three times better than in the case of the static population.

Closer examination of a particular case In order to take a closer insight into a particular run of the simulation, an experiment with the Rastrigin function was selected in a 100-dimensional space limited to [–5, 5]. Eight hundred iterations of the algorithms were conducted, and **BestLocalOrder** was selected as the strategy of choosing the species and **WorstLocalShift** for choosing the particles – an optimal configuration observed for Rastrigin in the previous section. Ranking and moving the particles were realized every 100 iterations. In each adaptation, four particles were moved to a species not further than two positions in the ranking from the source species. The starting population consisted of four particles belonging to the first species (cf. Tab. 3.2) and three particles from the subsequent ones.

In Figure 3.11, a comparison of the results obtained by a swarm with the adaptation of the species to the static multi-species swarm and classic PSO implementation is presented. The adaptive swarm reached a significantly better result than the static one; i.e., 21.04 for the adaptive one and 167.14 for the static one. The classic PSO implementation rapidly became stuck in the local optima; i.e., obtaining a result equal to 539.48.

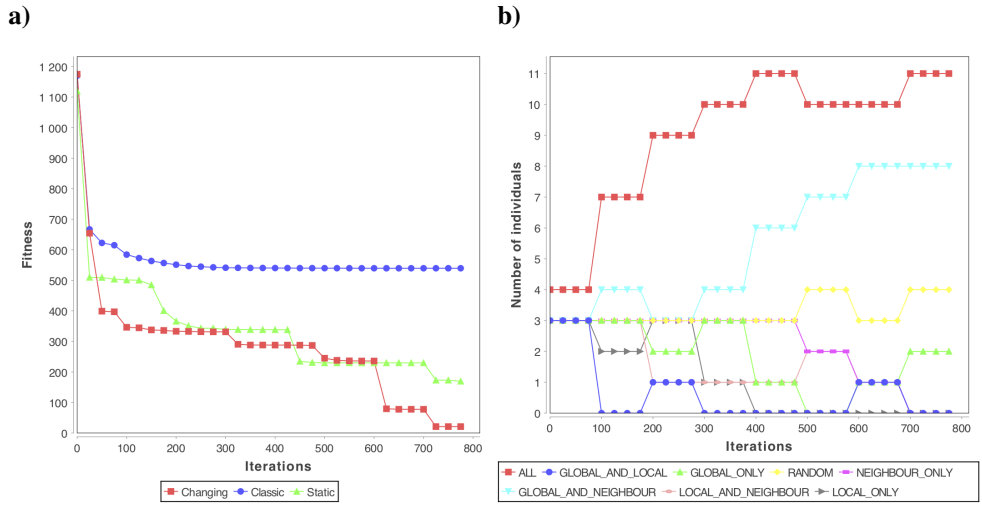

Figure 3.11. Fitness for static and adaptive PSO (a) and number of particles in adaptive PSO (b), optimization of 100-dimensional Rastrigin problem.

Moreover in Figure 3.11, one can see that, after each adaptation (every 100 steps), a more or less significant fall in the value of the best fitness is observed in the case of the adaptive algorithm. An analysis of the structure of the population showed that, at the end of the simulation, the species with the highest number was the one that was in the highest position in the ranking.

What is more, an observation of the share of the species in the population shows that a particularly large number of Global and Local species turned out to be efficient (retaining the classic properties of PSO); in addition, the Random species evolved to a significant share (apparently increasing the diversity in the search). Of course, one can easily see that certain species seem to be useless (at least using the proposed parameters); e.g., *Neighbor_Only* or *Global_Only*, signalizing that the use of such particles may lead to a quick stagnation.

3.2.6. Summary of socio-cognitive PSO research

We presented a new vision of PSO consisting of different cooperating species. The species constituting the swarm leverage the socio-cognitive ideas by getting inspired by the results presented by the other species present at the same time in the population. Our results indicate that the proposed metaheuristic performs better than the classic PSO. We have examined a number of different swarm configurations and have gathered data for further development of the algorithm to enhance their performance and to compare them with other state-of-the-art algorithms.

Our experiments were, of course, burdened with the curse of dimensionality, which makes searching for the optimum quite difficult. However, we would like to point out that Sugimoto et al. [155] achieved tangibly better results by getting close to 258 for the 100-dimensional Rastrigin function. Nevertheless, we feel that our results show the efficacy and the potential of the proposed metaheuristics, and improving their overall efficiency will require introducing detailed modifications focused on the initial algorithm.

The next part of this chapter tackles the building of a strategy for adapting the complex population structure in the Socio-Cognitive PSO. The research showed that the proposed algorithm is better than its static counterpart. A closer insight into the particular simulation allowed us to support this thesis with additional observations.

In this way, we have extended two important social metaheuristics (PSO and ACO), showing the gain from making them cognitive. Remembering the inherent agency of such metaheuristics, we will now try to focus on agent-oriented metaheuristics in order to seek good candidates for introducing the cognitive inspirations.

3.3. Socio-cognitive Stochastic Diffusion Search

A successful swarm intelligence metaheuristic algorithm with a very strong mathematical background (see, e.g., [159, 160]) and appealing metaphor is Stochastic Diffusion Search (SDS), which was introduced by Bishop in 1989 [93].

The idea of SDS is based on the so-called mining game [161] (see Pseudocode 13). In this game the miners try to dig out some gold from the nearby mines, meet after the day of hard work and pass the stories about their successes and defeats to other miners. On the next day, the less successful miners can get inspired by the better ones, changing their place of work to mimic their colleagues.

Pseudocode 13 Mining game

1: Allocate each miner (agent) to a random hill (hypothesis) to pick a region randomly
2: **repeat**
3: Miners evaluate their amount of gold (hypothesis evaluation),
4: Miners are classified into happy (active) and unhappy (inactive) groups.
5: Unhappy miners communicate with other miners. If they meet a happy miner, they consider his hill as a hint for the next search; otherwise, they select a new random hill.
6: **until** miners have congregated around the largest amount of gold

Thus, a search and optimization metaheuristic can be easily formed that treat miners (agents) as a means of randomly sampling the solution space and their meetings as information flow (in order to control the stochastic search) (see Fig. 3.12).

In the case of optimization, the evaluation of the hypotheses will, of course, cover the evaluation of a fitness function of a particular solution assigned by the agent. The diffusion of the hypotheses will consist of changing the solution assigned to the agent based on another solution (cf. perspective taking (see Section 2.1, mutation in differential evolution algorithm (see Section 4.6, changing of behavior in particle swarm optimization (see Section 1.4)).

Researchers have developed a number of modifications of the basic SDS metaheuristic. For example, the diffusion phase has been extended in [162]. Three recruitment modes were introduced: a passive one (the same as in the original algorithm), an active one (consisting of communication between the solutions by active agents to randomly chosen passive ones), and a dual one that brings these two together: active agents seeking passive ones and passive seeking active.

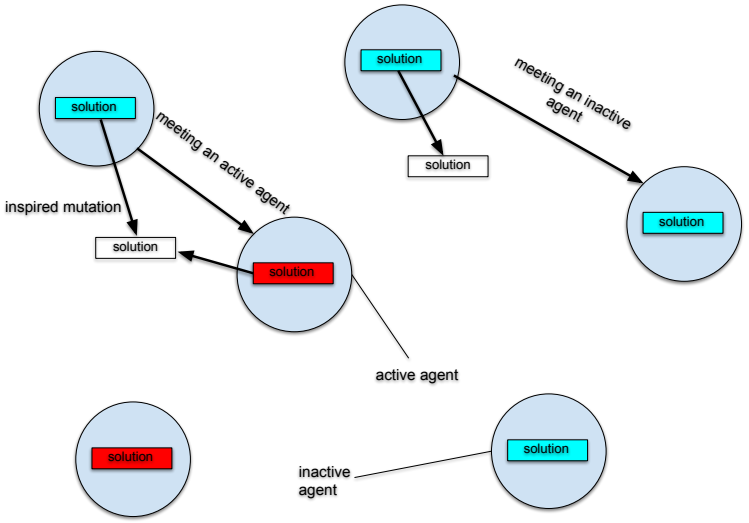

Figure 3.12. Stochastic diffusion search

The authors of [163] introduce asynchronous diffusion, implementing a fully parallel SDS algorithm, as the mechanisms used can be easily updated to asynchronous ones. This version can also be compared with EMAS, where the communication mechanisms were also very successfully transferred into massively parallel ones, and such implementations were worked out (using very robust functional programming approaches: Erlang [164] and Scala [165]).

Other modifications of the base algorithm consider the problems to be solved; e.g., introducing composite hypotheses [166] or decomposing populations into two subpopulations in order to construct appropriate solutions based on two independently sought parts [167].

The relationship between the agents (namely, the diffusion of the hypotheses) are convincing enough to assign this metaheuristic to the class of socio-cognitive ones (cf. Definition 2.4.1). Moreover, a number of possible modifications are further possible from the point of view of Social Cognitive Theory; e.g., similar to SC-ACO and SC-PSO, additional species can be introduced, and complex relationship among them can be easily implemented. It is easy to imagine egocentric or altercentric (or hybrid) individuals affecting the information-diffusion process by their behavior.

To sum up, SDS remains a very good starting point for the further development of socio-cognitive algorithms and becomes one of our main future research goals.

3.4. Socio-cognitive swarm intelligence algorithms in light of Social Cognitive Theory

Let us now reference the features of the metaheuristics mentioned in this chapter to the elements of Social Cognitive Theory. Of course, some of them are clearly visible and were incorporated in the metaheuristics by design, and some others are identified as possible development areas in future research.

Discussing the relevance of the presented algorithms to Social Cognitive Theory, we will use the four marks to confirm the presence of the feature in the algorithm or to assess the difficulty of introducing such a feature.

The mentioned marks are as follows:

- P: feature is present.
- E: easy implementation, requiring minor changes to the algorithm (e.g., simple changes of behavior of individuals, like introducing a local search).
- M: moderate implementation effort, requiring moderate changes to the algorithm (e.g., changes of several algorithm parts, like introducing the perception of history and considering this information during reproduction).
- C: complex implementation, requiring major changes to the algorithm (e.g., significant changes in the algorithm structure and behavior, like introducing different species and modifying the operators processing the population of individuals accordingly).

In the descriptions below, either the presence of a particular behavior or aspect of SCT is pointed out, or an exemplary mechanism is sketched-out along with an estimation of the effort needed to adapt the algorithm.

In the beginning, let us focus on the factors of triadic reciprocal causation identified by Bandura [10]:

- Personal: the agent can rely on itself or on other agents (cf. perspective taking (see Section 2.1). Agents with higher self-efficacy will rely on themselves, while those with lower self-efficacy will rely on other agents.
  - SC-ACO: the level of self-efficacy of the ant influences its way of perceiving the world and its actions. In the case of Socio-Cognitive ACO algorithms, the egocentric agents can be perceived as more self-reliant, while altercentric agents are more dependent on the knowledge of others. Good-at-conflict-handling agents can have balanced characteristics, not to mention those having arbitrarily chosen configurations of pheromone-perceiving weights (P).
  - MT-ACO: introducing a similar mechanism of species and relationships among miners like in SC-ACO and SC-PSO is possible, but the effort required is, of course, apparently high (C).

  - SC-PSO: In the case of a Socio-Cognitive PSO, there are agents taking into consideration all of the possible optima (the swarm's, the neighbor's, and the particle's own). At the same time, other agents can take into consideration only the neighborhood optimum or only it's own. So, the self-reliance depends on the type or species of a particle (similar to SC-ACO (P)).
  - SDS: the active agent does not rely on the knowledge of other agents (at least in the basic version of the algorithm), while the inactive agent must get inspired by the solutions carried by others (P).

- Behavioral: this feature consists of existing forms of affecting the self-efficacy of the agent (by rewarding or punishing it).
  - SC-ACO, MT-ACO, SC-PSO: the change of the quality of the solution can be perceived as an inherent reward or punishment (P); other additional mechanisms rewarding the better particles and punishing the worse can be introduced.
  - SDS: the change of the status of the miner can be perceived as an inherent reward (P); other additional mechanisms rewarding the better particles and punishing the worse can be introduced.

- Environmental: certain events or resources found in the agents' environment can affect the ability of the agents to realize a certain task (e.g., making it easier or more difficult, even impossible).
  - SC-ACO, MT-ACO: the ants can be allowed to leave higher levels of pheromones if they find good solutions (e.g., as compared with others) (M).
  - SC-PSO: a very nice example would be to modify certain fitness function/leave markers in the search space, asking the agents to avoid them (E).
  - SDS: the solutions found by the agents can be limited (treated as resources); the main goal of the optimization would be to control the agents-robots excavating the mine (M).

As was shown earlier, the agency of the learners described by Bandura fits very well into the agency perceived in the world of software agents; in particular, particles/agents can be easily described by these notions:

- Individual agency: autonomy is a prerequisite for individuality, and all of the individuals considered in these algorithms have agent-oriented features, as they can undertake decisions on their own.
  - SC-ACO: besides the inherent agency, the species designed in SC-ACO affect the individuality of the agent. Altercentric agents are more dependent on other agents, while the egocentric ones depend on their own knowledge and neglect others (P).

  - MT-ACO: the species in this algorithm are introduced because of the problem, and the agents consider both their own knowledge and the knowledge of others (just as in the classic ACO (P)).
  - SC-PSO: the particles decide on their own about their movement; moreover, depending on their species, their degrees of relying on others may change (P).
  - SDS: the agents meet other ones and get inspired by their solutions in order to improve their own; however, they undertake decisions about their actions on their own (P).

- Proxy agency: this feature can be realized by the delegation of certain actions to other agents. A really simple means of implementation would be to use a local search to support the agents finding new solutions in a memetic manner.
  - SC-ACO, MT-ACO: implementation of a local search would involve introducing additional ants, which will slightly modify (realize a kind of mutation) the base solution produced by the ant that initiates the local search (M).
  - SC-PSO: the implementation of local particles flying around the particle delegating the local search should be not difficult to realize (M).
  - SDS: introducing hierarchical meetings (sending a number of agents to meet other agents and then process their outcomes) might be a good method for introducing this feature into SDS (C).

- Collective agency: all of the considered algorithms can be treated as an ensemble of experts; they all solve a problem on their own, and the outcome is a collective answer (either encoded in a pheromone table or available as the solution of "the best of agents"). Thus, this is for all of the considered algorithms (SC-ACO, MT-ACO, SC-PSO, SDS (P)).

The relationship between PSO agents/particles and human agency can be found by considering the following properties:

- Intentionality: the agency present in all of the considered algorithms results in undertaking certain actions intentionally by the agents; i.e., ants choose the next edge, particles modify their velocity, etc. Even purely random actions are intentionally realized (e.g., bad-at-conflict-handling ants). Therefore, all of the considered algorithms SC-ACO, MT-ACO, SC-PSO, SDS (P).

- Forethought: this feature is connected with the ability to predict certain effects introduced into the environment after the execution of certain actions. This requires working out a model of the environment and build a predictor (based on one of the machine-learning techniques, or even some simple statistical methods, for example).

  - SC-ACO, MT-ACO, SDS: monitoring certain parameters (e.g., diversity of the search, using a method defined particularly for those algorithms, like lambda-branching for ACO [168]), the ants could predict the coming value of this parameter and undertake actions in order to counteract it; e.g., the coming loss of diversity (M).
  - SC-PSO: particles could extrapolate their future paths and, based on knowing their trail, try to avoid the areas visited before that are to be visited again soon (M).

- Self-reactiveness: an ability to adapt the actions depends, of course, on the definition of an action. In the area of computing, an agent can monitor its own quality (for example) and appropriately adapt the length of the step realized in the solution space (for example), so this feature can be easily implemented with the help of auto-adaptation mechanisms.
  - SC-ACO, SC-PSO: auto-adaptation of the behavior of ants/particles was implemented based on observing the features of the whole population (see Sections 3.1.5 and 3.2.5) (P).
  - MT-ACO: auto-adaptation of the ant's behavior can be easily realized (e.g., the parameters of attractiveness can be adapted based on the observation of certain parameters of the whole ant colony) (E).
  - SDS: the ways of inspiring inactive agents by the solutions presented by active agents can be adapted based on observing the diversity of the search, for example (E).

- Self-reflectiveness: this feature may be treated as an upper-level of self-reactiveness; the agent should observe its behaviors and perceptions (in particular, their history (adaptation)), and based on some machine-learning model (e.g., a neural network) or even simple statistical methods (like observing trends) be able to reflect on the parameters of auto-adaptation and regulate those.
  - SC-ACO, MT-ACO, SC-PSO, SDS: in the case of all of the considered algorithms, introducing such a model needs a certain amount of effort but is not overly complex, as such a procedure would affect the particular agents even though the implementation of such a model can be complex (M). At the same time, one should remember that introducing a very complex model on the level of agents would dramatically increase the computational complexity of the whole algorithm.

Introducing such a mechanism might bring the Socio-Cognitive PSO or ACO significantly closer to the standards of agent-oriented models such as BDI [112] or M-agent [113].

Just as it was proposed above (see the descriptions of self-reactiveness, self-reflectiveness, or forethought features), the agents can acquire certain knowledge, build and adapt their models, and follow their assumptions, executing their actions based on the information gathered in the environment and observed among other agents. These observations realized by the agents can be described as follows:

- *Attention*: the perception of certain individuals is limited by the means of accessing the environment and other agents states. This depends, of course, on the actual algorithm, and such access is usually somehow limited (e.g., by population decomposition) in order to increase the diversity of the search.
  - SC-ACO, SC-PSO: depending on the features of a particular species, the ants or particles perceive more (e.g., altercentric) or fewer (e.g., egocentric) deeds of others and count these observations in the collection of perceptions needed in order to undertake subsequent actions (P).
  - MT-ACO: the ants perceive the pheromones of other species and react accordingly to the type of the perceived pheromone (whether they are attracted or repelled) (P).
  - SDS: the information made available by the active agents may be regulated. In the applications to computing, the inactive agent can perceive a full solution of the active one or only some derivative (E).
- *Retention*: observations performed by the agents can lead to the construction of (even simple) cognitive models, that will help in undertaking further decisions.
  - SC-ACO, SC-PSO, MT-ACO: even though the behavior of other agents is observed in both algorithms, the outcome of the observation is volatile – it is used only for undertaking a current decision and then it is forgotten. Retention of such information may lead to construct even a simple statistical model and consider it during the computing of attractiveness or changing the direction of flight (E).
  - SDS: an agent can construct a simple statistical model describing the solutions of the active agents that were encountered and use it as a tabu list in order to search elsewhere for a new solution (E).
- *Production*: perception of certain behaviors can lead to (at least partial) reproduction in order to seek a solution near the observed one (so the search process is controlled in a concise way, not totally random like in monte-carlo methods [21]).
  - SC-ACO, SC-PSO: the configuration of particular species allow them to reproduce the behavior of other agents (e.g., following the global or local best solution, or following certain pheromones) (P).

  - MT-ACO: the reproduction of other ants' behavior is intrinsic because of the existence of a pheromone table and stigmergic communication/perception (P).
  - SDS: the reproduction of other agents' actions (or information) is inherent to the SDS algorithm and takes place in the course of the diffusion process (meetings of inactive and active agents) (P).
- *Motivational process*: this feature may be implemented based on the reenactment (perhaps with adaptation) of the behaviors that were copied from (or inspired by) the other agents.
  - SC-ACO, MT-ACO: the ant can observe the changes of its quality based on the actions inspired by other agents (or rather, the parameters of the search) during a certain period of time (E).
  - SC-PSO: the particles observe the actions of others, but dedicated models can be built in order to assess the quality of the outcomes of their own movements (E).
  - SDS: the agent can observe the influence of the inspirations (taken from the active agents) on its quality and react accordingly (E).

As one can see, the referenced Socio-Cognitive ACO and PSO as well as MT-ACO and SDS have many features that are compatible with Social Cognitive Theory, although there is still room for introducing more in order to further increase the cognitivity of the particular agents, increase the learning capability, and enhance the quality of the collective intelligence that consists of particular agents. Let us now focus on the classic and agent-based metaheuristics that are also strongly related to Social Cognitive Theory.

4. Socio-cognitive classic and EMAS-related hybrid metaheuristics

In the computing domain, there are a lot of evolutionary-inspired algorithms, starting from the genetic algorithm, evolution strategies, and genetic programming as well as the many hybrids. Seeking socio-cognitive inspirations leads to co-evolutionary algorithms (which consist of introducing more than one species [169]) as well as to other hybrids of evolution (e.g., EMAS). It is to note that different species cooperating or competing in these algorithms are subjected to the generation exchange procedure – the individuals do not prevail throughout the whole run of the algorithm (contrary to ACO or PSO metaheuristics). Besides being exchanged between the species, the knowledge is passed to subsequent generations by means of variation operators. Thus, the metaheuristics described in this chapter can be classified as inter-generational ones (cf. Definition 2.4.2).

The most popular way of applying agency to computing problems is to use agents for management purposes at a technical level of application. An agent community may take care of the the management of the distributed system, utilizing their autonomy and auto-adaptation to implement scheduling or load balancing, for example. The distributed management of the system deployed in a cluster or grid requires a definition of the node topology, neighborhood, and migration policies that are affected by the current load of the nodes. The well-known standards for constructing multi-agent environments (such as FIPA [170]) do not provide such capabilities. Another important functionality missing in many platforms is the notion of distance between agents, measured as a factor of communication throughput. Grochowski and Schaefer [171, 172, 45] proposed Smart Solid Architecture that supports these requirements.

EMAS is an agent-oriented metaheuristic described in Section 1.6 that is constructed in a completely different way. In this approach, agents become a part of the algorithm itself, allowing it to use their autonomy, situatedness, communication means, and many other features integrated into the algorithm. The individuality of

the agents calls for extending their capabilities towards cognitive features, making the construction of socio-cognitive systems such as EMAS hybrids possible.

In this chapter, several such metaheuristics are described. We start with delving into the evolutionary and co-evolutionary algorithms that seek methods that can be classified as socio-cognitive ones. Later, we discuss four EMAS hybrids. Two of the systems discussed, (namely, Co-evolutionary EMAS and Elitist EMAS) were proposed by Rafał Dreżewski and Leszek Siwik, respectively.

An interesting hybrid of the EDA algorithm (namely, $COMMA_{op}$) is also described, along with its hybrid that utilizes several EMAS-based methods. This is followed by taking a look at the Differential Evolution algorithm and its hybrid with EMAS, followed by another EMAS hybrid (this time with PSO).

Finally, memetic and cultural algorithms are presented along with the relevant EMAS hybrids. The chapter is finalized by a discussion of the matching of the presented metaheuristics to the Social Cognitive Theory characteristics.

It is to note that all the algorithms presented in this chapter fall into category of social metaheuristics (cf. Definition 1.2.1) and are potentially good material for extending towards social-cognitivity. Moreover, some of socio-cognitive features (cf. Definition 2.4.1 are already present in many of them).

4.1. Parallel and co-evolutionary algorithms

Solving difficult search and optimization problems (e.g., black-box ones) introduces additional requirements to evolutionary algorithms concerning the ability to avoid or escape from the local minima, as their well-known problem is so-called premature-convergence. Maintaining diversity of the population is crucial to achieve a balance among the most important features of search techniques, namely *exploration* and *exploitation* [34, 67].

As defined by March "..., exploration includes those things captured by terms such as search, variation, risk taking, experimentation, play, flexibility, discovery, and innovation" [173], while exploitation "...includes such things as refinement, choice, production, efficiency, selection, implementation, and execution" [173].

In terms of metaheuristics, exploration is the ability to conduct a broad search in every part of the admissible search space in order to provide a reliable estimate of the global optimum, whereas exploitation consists of refining a search to produce a better solution [119].

In order to maintain a balance between exploration and exploitation, methods for preserving the diversity of the evolutionary population are needed. Delving into more biological inspirations, following certain speciation models well-known in evolutionary biology can lead to obtaining more-diverse populations.

These inspirations can be divided into three groups [174]:

- Allopatric models (connected with the geographical separation of subpopulations with some migration mechanism applied; one well-known example is the island-model of evolution [50]).
- Parapatric models (partial overlapping of subpopulations that can be connected with the cellular model of evolution [68].
- Sympatric models (with more than one population influencing each other by complex relationships); i.e., co-evolutionary algorithms [175].

Parallel evolution Thinking of classic diversity enhancement techniques, several decomposition and co-evolutionary techniques come to mind. *Niching* (or *speciation*) techniques [176] are aimed at introducing useful population diversity by forming subpopulations (also called "species"). *Allopatric* (or *geographic*) *speciation* may be considered when individuals of the same species become isolated due to geographical or social changes. *Decomposition* approaches of so-called *parallel evolutionary algorithms* (PEA) model such phenomena by introducing non-global selection (mating) and some spatial structure of the population [72] (see Fig. 4.1).

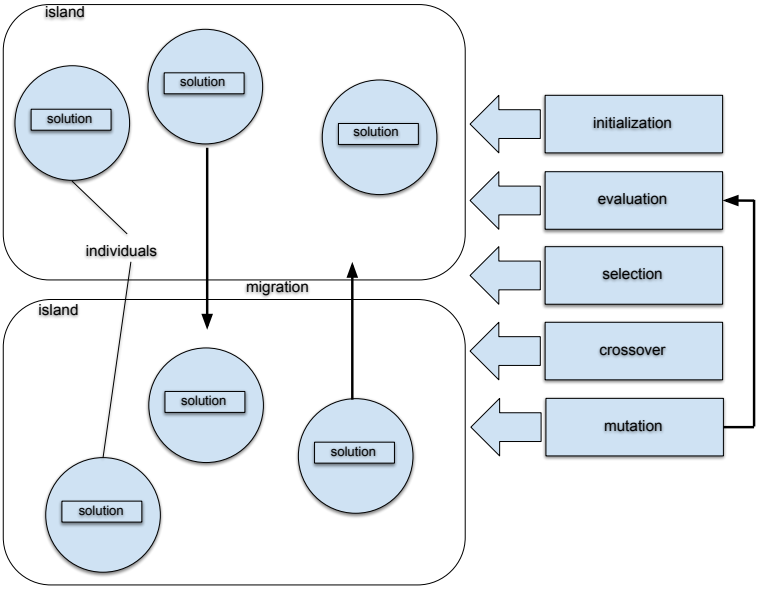

Figure 4.1. Parallel evolutionary algorithm

In a *coarse-grained* PEA (also known as a *regional* or *multiple deme* model), the population is divided into several subpopulations (regions, demes); selection is limited to individuals inhabiting one region, and a migration operator is used to move (copy) selected individuals from one region to another. In a *fine-grained* PEA (also called a *cellular* model), individuals are located in some spatial structure (e.g., lattice), and selection is performed in the local neighborhood.

The decomposed populations can be treated as species, and migration (the relationship between the species) allows them to pass on a certain part of their knowledge, incorporate it, and use it in a future search. Thus, parallel evolutionary algorithms can be classified as very primitive socio-cognitive metaheuristics and can become a very simple basis for constructing more-sophisticated socio-cognitive algorithms.

Co-evolution A consequence of the following sympatric and allopatric models is the proposal of co-evolutionary algorithms by Hillis [175]. In the state of the art, two general approaches can be spotted; namely, [177]:

- approaches using competition (i.e., competitive fitness functions) making the individual compete by following a certain game in order to compute the fitness function (see, e.g., [178]),
- approaches using multiple population (see, e.g., [179]).

In *cooperative* co-evolutionary algorithms, the fitness of each individual is not computed directly based on the definition of the problem to be solved; it is based on the results from the interactions with other individuals residing in the population. In *cooperative* co-evolutionary algorithms, a problem to be solved is decomposed into several subproblems to be solved by different algorithms in separate subpopulations [180]. Cooperation between individuals from different subpopulations may only take place during a phase of computing the fitness for the complete solution. The fitness value is computed only for the group of individuals from different subpopulations that form a complete solution to the problem (see Fig. 4.2).

Although all of the species are processed by the same (evolutionary-like) algorithm, the actual evaluation of the fitness of particular individual not only depends on the quality of its solution but also on individuals from other species (e.g., the best ones, or the most diverse ones). Thus, the search is driven not only by a local exploration, but an interaction arises between different parts of the algorithm (just like in the case of the island model of evolution). Previously, it was migration – now, it is a kind of perspective-taking or inspiration. Thus, the cooperative co-evolutionary algorithm can be treated as a simple socio-cognitive metaheuristic.

It is to note that this metaheuristic has great potential to be further modified and hybridized towards introducing more-sophisticated dependencies among the species, further enhancing the cognitivity of the base algorithm.

In *competitive* co-evolutionary algorithms, two individuals usually compete with each other in a tournament, and their "competitive fitness" corresponds to the outcome of this competition [181]. In each algorithm step, a given individual from one subpopulation competes with opponents taken from other sub-populations. The results of this competition have an impact on the current fitness of the individual that mates partners coming from the same subpopulation.

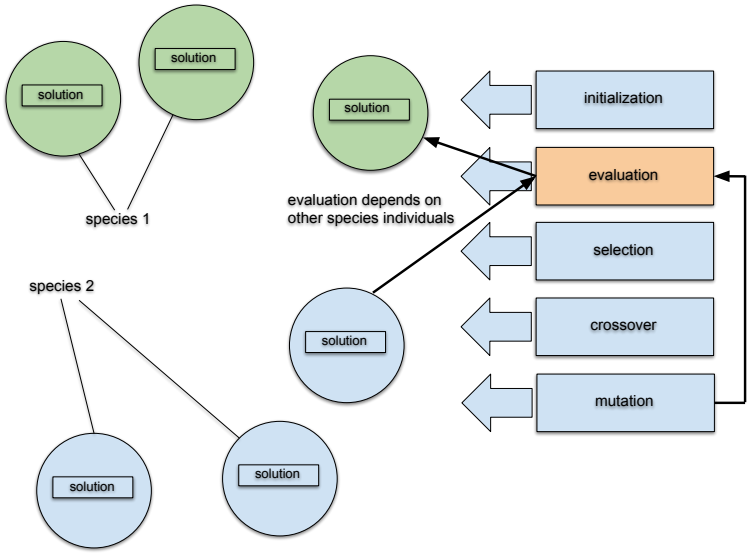

Figure 4.2. Cooperative co-evolutionary algorithm

This mechanism can be applied irrespective of the number of subpopulations used in the algorithm – it can be used even if there is only a single population. In this case, the opponents are chosen from the same population (see Fig. 4.3).

A competitive evaluation may be found in the Iterated Prisoner Dilemma simulation problem [109] (which is popular in the literature) that is coupled, in fact, with the optimization problem (the optimization of prisoner strategy). Moreover, certain selection methods in evolutionary algorithms have competitive characteristics; e.g., tournament selection [68, 34]. The tournament is also used for evaluation in EMAS, see Section 1.6.

Generally speaking, competitive evaluation consists of introducing inter-individual relationships; e.g., those created during the selection (or evaluation) mechanism (similar to perspective-taking). This mechanism allows us to treat the competitive co-evolutionary algorithms as simple socio-cognitive metaheuristics.

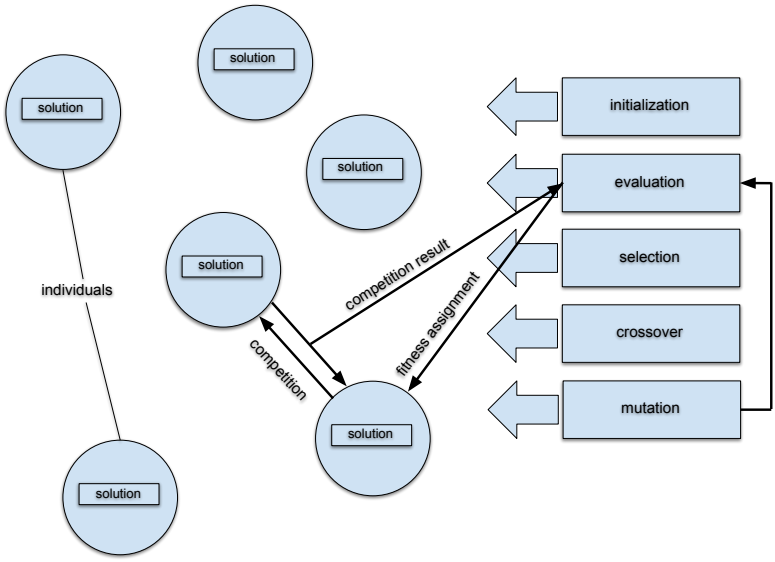

Figure 4.3. Competitive co-evolutionary algorithm

As in the case of the previous one, this metaheuristic has great potential to be further modified and hybridized towards introducing more-sophisticated dependencies among the species, further enhancing the cognitivity of the base algorithm.

4.2. Co-evolutionary EMAS metaheuristics

Co-evolution mechanisms were introduced into EMAS in order to maintain population diversity. This approach advocated by Rafał Dreżewski was successfully applied to solving multi-modal optimization problems consisting of finding all (or most) of the function extremes [176].

Dreżewski focuses on sympatric models of speciation – where new species evolve while inhabiting the same geographic region. It is modeled in evolutionary algorithms by techniques based on the modification of the parent-selection mechanism, like *fitness sharing* or techniques based on the modification of the mechanism of selecting individuals for a new generation such as *deterministic crowding* [182]. Ideally, each of the species should be placed within one of the basins of attraction of the local maxima (which might be difficult to achieve in practice).

Although co-evolutionary techniques and *sexual selection* mechanisms [183] promote population diversity, they were not widely used as a basis for constructing

niching techniques. These techniques were a point of reference for the model of the co-evolutionary multi-agent system (CoEMAS) [184].

As in the case of classical evolutionary algorithms, basic EMAS-based systems without any special mechanisms promoting useful population diversity are unable to solve multi-modal optimization problems because they are not able to form species located within the basins of attraction of many different local optima. They would rather locate only one solution; however, population diversity would be much higher than in the case of a classical evolutionary algorithm.

The basic idea behind the CoEMAS model is to introduce the possibility of the co-evolution of species and co-evolution of sexes (sexual selection) within EMAS. In the exemplary systems presented in the next sections, co-evolutionary interactions between species and sexual selection are used in order to construct niching techniques in a evolutionary multi-agent system and promote useful population diversity.

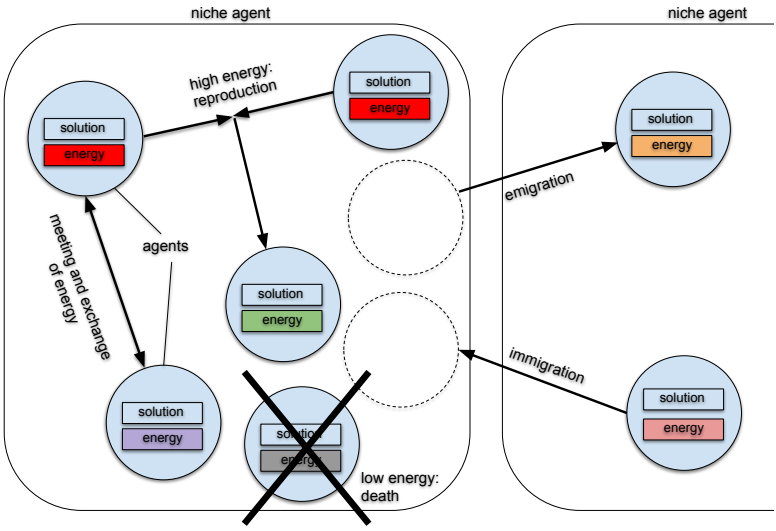

Figure 4.4. CoEMAS with co-evolving species

In the **co-evolutionary multi-agent system with co-evolving species** (nCoEMAS), two different types of computational agents exist. They will be called 'niches' and 'individuals' (*agent n* and *agent i* in Figure 4.4). Individuals 'live' within niches, where they are assigned in the beginning of the run of the algorithm; they can search for a suitable agent-niche and migrate into it. They can even 'create' a new agent-niche and migrate into it when they are not able to find a suitable niche.

The main goal of the mechanism of migrating between and creating new niches is to split the population of agents-individuals into such subpopulations (species) so that each would be located within a distinct basin of attraction of the local maxima.

Mating is restricted to only agents-individuals located within the same agent-niche. As in the case of basic EMAS model, reproduction is initiated by an agent that has enough resources to reproduce. The agent initiating a reproduction process searches for other ready-for-reproduction partners from the same niche and chooses one of them. Two children are generated; each offspring receives some number of resources from each of their parents.

Agents-niches can migrate within an environment, but only when they have enough resources (because each migration costs some number of resources that are returned to the environment). Two agents-niches can merge with each other: merging takes place when the centers of gravity of sub-populations that belong to two different niches are located within the basin of attraction of the same local optima. The system is applied to multi-modal optimization problems; therefore, the goal is to localize most of the basins of attraction of the local maxima.

Resources circulate between an environment and the agents. The whole number of resources within the system is constant, so the number of agents cannot grow infinitely. The environment gives an equal number of resources to each agent-niche, then they redistribute the resources among the agents-individuals proportionally to their fitness.

The **multi-agent system with sexual selection** (sCoEMAS) utilizes the co-evolution of sexes and sexual selection in order to obtain the effect of speciation. In sCoEMAS, two sexes exist: female and male (*agent X* and *agent Y* in Figure 4.5).

The reproduction process is initiated by a female agent when it has enough resources to reproduce. Generally, female agents use more resources than male agents during the reproduction process; this is why they reproduce less frequently than male agents. Therefore, there are always more male agents ready for reproduction than female agents, and the female agents choose partners from several ready-for-reproduction male agents. A female agent located within an evolutionary island searches for a partner in such a way that it chooses one ready-for-reproduction male agent from the same island. The partner is chosen on the basis of genotype similarity; the more similar the two agents from the opposite sex are, the more probable it is that the female agent will choose the male agent. Reproduction takes place when a pair composed of agents of the two opposite sexes is formed. Such a pair of agents is formed permanently for several steps of the algorithm because of the difficulties and costs involved in searching for a partner. The offspring (two children) are generated with the use of mutation and recombination operators and receive some of the resources from their parents: more from the female agent and less from the male one.

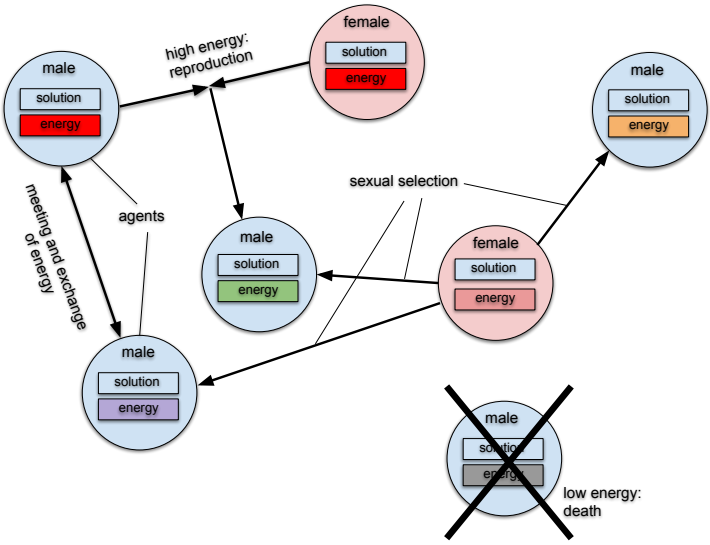

Figure 4.5. SCoEMAS with sexual selection

Agents can migrate within an environment, but it costs some number of resources. Similar to nCoEMAS, the number of resources present within sCoEMAS remains constant; the environment distributes these resources between agents proportionally to their fitness.

Dreżewski widely applies the devised metaheuristic algorithm to multi-modal optimization solving; e.g., different difficult benchmark problems [185], multi-criteria optimization [186], real-life problems like portfolio optimization [187], or generating investment strategies [188, 189]. The authors maintains an up-to-date webpage with complete information on their achievements[1].

4.3. Clonal Selection Algorithm and immunological EMAS

The human immune system plays a key role in maintaining the stable functioning of one's body. It allows for the detection and elimination of dysfunctional endogenous cells (termed infectious cells) and exogenous microorganisms (infectious non-self cells, such as bacteria and viruses) that enter the body through various routes, including the respiratory and digestive system as well as via damaged dermal

[1] `http://drezewski.eu5.net`

tissue. Lymphocytes play a key role in humoral immunity (a cellular immune layer). T-lymphocytes mature in the thymus into two distinct subpopulations: T-helper and T-killer cells (the latter acting as removing agents for dysfunctional cells of the body). T-cells are subjected to a process called negative selection in the thymus, where they are exposed to a wide variety of self proteins that are destroyed if they recognize them [190].

Artificial immune systems inspired by the human immunity have been the subject of increased researcher interest for about 20 years. Different immune-inspired approaches have been applied to many problems, such as classification or optimization [191].

The most often used algorithm of negative selection corresponds to its origin and consists of the following steps:

1. Lymphocytes are created; as yet, they are considered immature.
2. The binding of these cells (affinity) to present self-cells (e.g., good solutions to some problems) is evaluated.
3. Lymphocytes that bind themselves to ”good” cells are eliminated.
4. Lymphocytes that survive are considered mature.

Mature lymphocytes are presented with the cells that have unknown origins (they may be self or non-self cells), and they are believed to have the possibility of being classified [192].

Immune-based algorithms may be used in optimization problems – one such approach is known as the Artificial Immune Iterated Algorithm (AIIA), which was originally presented in [193] and modified in [194]. The algorithm consists of the following steps:

1. A population of individuals (called *antibodies*) is randomly generated. Each individual represents a single solution in the search space.
2. The best antibody is chosen (*antigen*).
3. The group of antibodies with the highest affinity (similarity) to the antigen is selected (*clonal selection*).
4. Each individual is cloned and mutated; if the best clone is better than the original, the original is replaced (*somatic hyper mutation*).
5. Individuals with low fitnesses are replaced by randomly generated new ones (*apoptosis*).

In this way, good solutions of the problem are retained in the population, and the whole population is attracted by the currently chosen antigen.

Another popular immunologically-inspired optimization algorithm is the Clonal Selection Algorithm (CSA) This algorithm treats the extrema sought as antigens and evolves the repertoire of the memory cells in order to get closer to the extrema.

The pseudocode of this algorithm is as follows [195]:

1. Create a set of P solutions containing a subset of memory cells (M) and the remaining antibodies (Pτ).
2. Select the n-best antibodies based on their similarity to the antigens (i.e., value of the fitness function).
3. Clone the n-best antibodies, creating a clone population (C). The number of clones of a certain antibody is proportional to its affinity (the value of the fitness function).
4. Mutate the population of clones; the mutation rate is proportional to the affinity, creating population C*.
5. Select the best clones from C* and replace them with a part of the memory cells M (it is also possible to replace a part of the remaining antibodies (Pτ))
6. Replace d randomly picked antibodies with new ones (randomly generated). The antibodies with lower affinities will be replaced with a higher probability.

The main idea of applying immunological inspirations to speeding up the process of selection in EMAS is based on the assumption that 'bad' phenotypes come from 'bad' genotypes. Immune-inspired approaches have been applied to many problems such as classification or optimization (e.g., [196]). The most frequently used algorithms of clonal and negative selection correspond to their origin and are used in many applications [191].

After an analysis of the existing algorithms and the metaphor itself, the immunological EMAS was proposed in [197]. The general structure of immunological EMAS (iEMAS) is shown in Figure 4.6. A new group of agents (acting as lymphocyte T-cells) is introduced [84]; they are responsible for recognizing and removing those agents with genotypes similar to the genotype patterns of these lymphocytes. Another approach may to introduce a specific penalty applied by the T-cells for the recognized agents (a certain amount of the agent's energy is removed) instead of removing them from the system. Of course, some predefined affinity (lymphocyte-agent matching) function must exist that may be based on the percentage difference between corresponding genes, for example.

Agents-lymphocytes are created in the system after the action of death. The late-agent genotype is transformed into lymphocyte patterns by means of a mutation operator, and the newly created lymphocyte (or group of lymphocytes) is introduced into the system. In both cases, the new lymphocytes must undergo the process of negative selection. In a specific period of time, the affinity of immature lymphocyte patterns with 'good' agents (possessing relatively high amounts of energy) is tested. If it is high (the lymphocytes recognize 'good' agents as 'non-self'), they are removed from

the system. If the affinity is low, it is assumed that they will be able to recognize 'non-self' individuals ('bad' agents), leaving those agents with high energy intact. The life span of the lymphocytes is controlled by a specific renewable resource (strength) used as a counter by the lymphocyte agent.

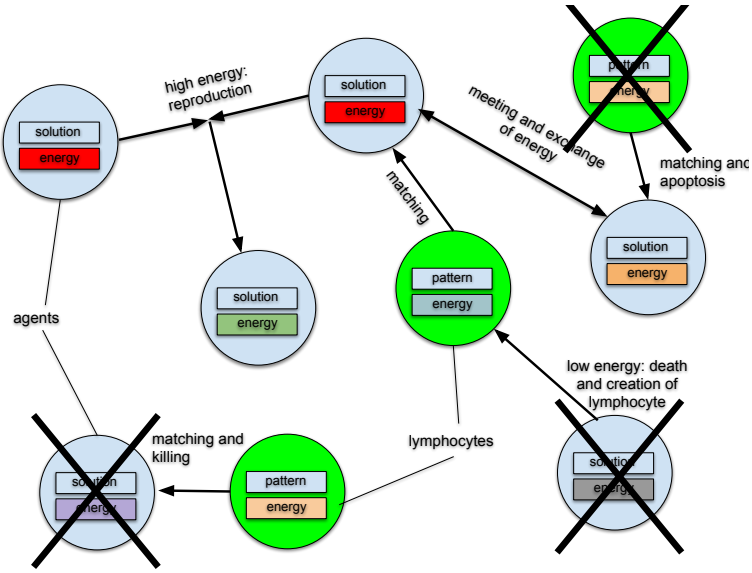

Figure 4.6. Immunological EMAS

Therefore, EMAS is enhanced by adding lymphocyte agents, altering the action of the agent's death, and adding three lymphocyte-related actions:

- *Death* – the EMAS action of death is redefined: during this action, the agent produces one or more lymphocyte agents, passing its mutated genotype to them and setting their strength to the maximum value.
- *Killing* – mature lymphocyte (with energy below a certain level) removes (or weakens) one of its neighboring agents if it finds that the genotype of this agent matches its own using a predefined affinity function. Immature lymphocytes (with strengths in excess of a certain level) are checked as to whether they match an agent with high energy; in this case, the lymphocyte is removed from the system.
- *Apoptosis* – a lymphocyte with a zero level of strength is removed from the system.
- *Give* – this action controls the negative selection process and overall lymphocyte acting time by simply decreasing the level of a lymphocyte's strength, thus allowing it to perform other actions (e.g., killing and apoptosis).

The concept of iEMAS is especially advantageous in applications requiring time-consuming fitness evaluations like the evolution of a neural network architecture [84].

The iEMAS metaheuristic clearly divided the population into two species being able to perceive themselves, interact, and affect one another (the lymphocytes can remove the computing agents, as the lymphocytes' genotype is constructed based on that of the agent). Thus, iEMAS can be perceived as belonging to the socio-cognitive class of metaheuristics.

A number of research experiments have been realized using iEMAS, focusing on single-criteria optimization, utilizing well-known multi-dimensional benchmark functions [84], the immunological paradigm was used for intrusion detection and [198], optimization of neural network architecture [199]. Moreover, it is to note that the correctness of iEMAS, as an algorithm has been formally proven (a dedicated formal model was constructed based on Markov Chains following the idea of Michael Vose's model [6]) [200].

4.4. Elitist EMAS for multi-objective optimization

In multi-criteria optimization problems, several (usually contrary) goals must be fulfilled in order to provide the user with appropriate solutions. Moreover, there is no one global solution; instead, the proper result is a set of solutions that are equally good from the point of view of all of the considered criteria (belonging to the so-called Pareto front). For the last 20 years, a variety of evolutionary multi-criteria optimization techniques have been proposed. In Deb's typology of evolutionary multi-objective algorithms (EMOAs), the elitist and nonelitist ones are first distinguished [201]. Each of these groups include many practically used algorithms, such as elitist EMOAs (Rudolph's algorithm [202], distance-based Pareto GA [203], strength Pareto EA [204]), non-elitist EMOAs (vector-optimized evolution strategy [205]), and random-weighted (GA [206]). It is to note that elitism (i.e., differing the population of individuals based on some measure for example, a derivative of the domination function and restricting the variation operators to an elite or regular group) is an efficient tool in dealing with multi-criteria optimization problems. Such a mechanism was devised by Leszek Siwik and introduced into EMAS (see, e.g., [207]).

Elitism can be introduced into EMAS in many different ways. In the case that is considered, it is based on a slightly modified structure of the environment (see Figure 4.7). Compared to the structure of the environment shown in Figure 1.4, so-called elitist island is introduced and special actions that can be performed by selected agents are added, allowing them to migrate (one-way only) to this very island. Thus, the agents that have decided to migrate to this island are not able to go back from an elitist island to ordinary islands and cannot take part in the process of evolution.

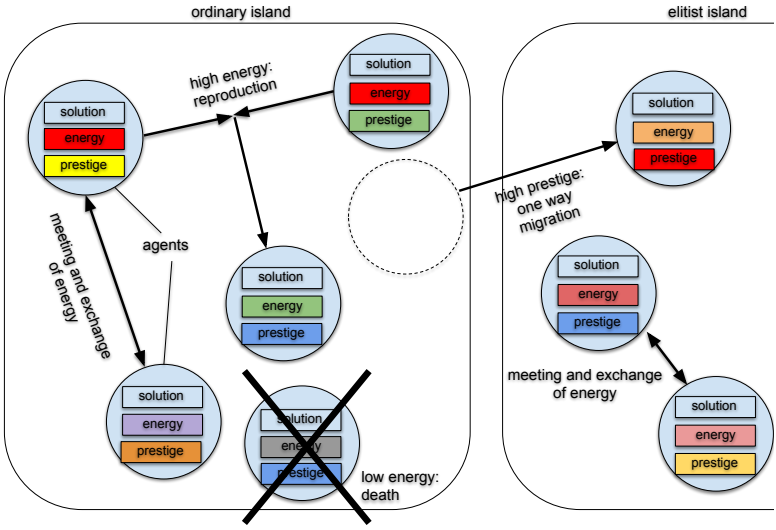

Figure 4.7. Elitist evolutionary multi-agent system

A mechanism that allows for the identification of elitist agents is based on the level of an additional resource called *prestige*, which is gathered (and is not lost) by agents during their lifetimes. At the beginning of their lives, the level of *prestige* equals *zero*. Then, every time an agent dominates (in the sense of a *domination relationship*) any other agent, its level of *prestige* increases; therefore, it may be assumed that agents with a high level of this specific resource belong to the elite. This mechanism allows for the realization of the above-mentioned idea of *non-dominating ranking* in a really elegant, easy to understand, and implemented way that does not require any additional complicated operations nor computations.

It turns out that, despite the other presented advantages, applying elitist operator(s) to an evolutionary multi-agent system gives us the opportunity to introduce additional mechanism(s) responsible for improving the spreading of individuals over the Pareto frontier. One such mechanism can be realized as follows: during meetings, an agent is able to check whether a solution represented by its opponent is located in its surroundings at the frontier (i.e., if the distance between the solutions represented by the agents is less than the given value of ϵ). If so, the agent increases its personal counter, storing the number of individuals located in the same fragment of the Pareto frontier. Additionally, asking its opponents about the number of individuals located in their surroundings, the agent is able to gather (partial) knowledge of the average number of individuals located in some areas of the Pareto frontier over time (i.e., in

surrounding areas represented by agents it met). Consequently, if the agent becomes an elitist one, it can make a decision about migrating to the elitist island only when the number of individuals located in its surroundings is greater than the average number of individuals located in other regions of the Pareto frontier.

The mechanism described allows for a deeper exploration of those areas of the Pareto frontier that are less sampled than in its other regions. A similar mechanism can be applied to the space of decision variables, thus concerning the Pareto set. Of course, both mechanisms can be used simultaneously, and this can be of great importance in real-life problems where both the Pareto set and Pareto frontier are often disconnected.

Additionally, the ϵ parameter can be managed adaptively by the agents. In its simplest form, it can be linearly smaller and smaller over an agent's lifetime. In more-advanced implementations, it can be decreased by agents on the basis of their interactions with the environment and other agents.

Siwik divided the agent population into elite and regular ones and introduced dedicated parameters like prestige, making the agents perceive one another, and decide about their actions based on their parameters. Thus, his work can be perceived as belonging to the socio-cognitive class of metaheuristics.

Siwik also applied the elitist EMAS to the optimization of noisy multi-objective problems [208], constrained multi-criteria optimization problems [209], constructed a semi-elitist version of the algorithm [210], and put effort into optimizing different components of the proposed computing method (see, e.g., [207]).

4.5. Socio-cognitive COMMA$_{op}$

COMMA (see Section 1.7) can be hybridized with EMAS [211], becoming something closer to a cultural algorithm, with the knowledge shared at a certain point by the whole agent (sub)population transferred and changed by particular agents. Such a system can be also clearly classified as a socio-cognitive one as the (sub)populations of agents will form societies where information will be perceived, exchanged, and transferred in order to reach a common goal; i.e., searching for optimal value of the fitness function.

As elements of socio-cognitive-related research have already proven to be effective in other computing-related applications (namely, discrete optimization using PSO [12] and ACO [11]), the authors have decided to try to enhance the EDA algorithm (as this class is very successful in solving discrete problems) with a selected socio-cognitive mechanism, striving towards the full hybridization of EMAS and EDA. This approach seems to be right, especially because the individual adaptation of the mutation of each of the agents makes the COMMA algorithms share a similar paradigm

like in EMAS (as this can be perceived as a certain autonomy of the agents processed in COMMA$_{op}$). Therefore, the full hybridization of COMMA$_{op}$ and EMAS seems to be a promising idea.

In order to precisely describe the modifications of the original algorithm, COMMA$_{op}$ is simplified by the aggregating steps visible in Pseudocode 14, thus constituting a classic EDA algorithm where the exploration and exploitation is realized by sampling the search space with the calculated random distribution that is being adapted during the subsequent steps of the algorithm.

Pseudocode 14 Simplified pseudocode of $COMMA_{op}$ [90]

```
1: Initialization of agents and other parameters
2: repeat
3:     Sort population of agents
4:     Calculate mutation rates
5:     Sampling of new solutions by adaptive mutation
6: until stopping_condition()
```

COMMA$_{\mathbf{op}}$ with population decomposition A quite natural evolution of COMMA$_{op}$ that may be considered following the idea of EMAS and generally the parallel evolutionary algorithms concept [72] is the decomposition of the population, which usually brings new quality with regards to the diversity of the search. Therefore, the notion of evolutionary islands is introduced into the original algorithm along with a simple migration strategy. Thus, the mutation ranges are computed inside each of the islands.

The modified algorithm can be described as shown in Pseudocode 15 (note that, in this and the subsequent pseudocodes, the changes introduced with regards to the previous version of the algorithm are displayed in bold).

Pseudocode 15 Pseudocode of COMMA$_{op}$ with population decomposition

```
1: Initialization of agents and other parameters
2: repeat
3:     Migrate agents among islands with low probability
4:     Sort population of agents
5:     Calculate mutation rates
6:     Sampling of new solutions by adaptive mutation
7: until stopping_condition()
```

It is easy to see that, in the beginning, all of the populations of agents are initialized on each of the islands. Then, the migration is realized (with a small probability), and the following step of the original COMMA$_{op}$ algorithm (cf. Pseudocode 14) is realized on each of the islands subsequently.

COMMA$_{\mathbf{op}}$ with cloning and death of agents Another natural extension of COMMA$_{op}$ inspired by EMAS is the introduction of the resource-based selection-like mechanism reminiscent of the distributed selection in EMAS. The notion of energy is introduced; this value is computed for all of the agents. The action of energy exchange (similar to the one in EMAS) is used – the better agent takes a part of the worse agent's energy. Finally, the notion of cloning is used for those agents that exceed a certain amount of energy, and the notion of death – for those whose energy falls below a certain level. This algorithm is shown in Pseudocode 16 (the changes in the algorithm compared to the previous are displayed in bold). The steps concerning cloning and mutation are realized on each island along with the exchanges of energy between the agents.

Pseudocode 16 Pseudocode of COMMA$_{op}$ with cloning and death

```
1: Initialization of agents and other parameters
2: repeat
3:     Migrate agents among islands with low probability
4:     If agent's energy is higher than certain level: clone agent in population
5:     If agent's energy is lower than certain level: remove agent from population
6:     Sort population of agents
7:     Calculate mutation rates
8:     Sampling of new solutions by adaptive mutation
9: until stopping_condition()
```

COMMA$_{\mathbf{op}}$ with crossover The final extension of COMMA$_{op}$ inspired by EMAS is the introduction of a crossover with mutation (other than the original EDA-style mutation) instead of the cloning process is shown in Pseudocode 17. Thus, the full hybrid of the original COMMA$_{op}$ with EMAS-related notions is attained. Now, one should turn to checking the point of the whole endeavor.

Introducing EMAS mechanisms into the EDA algorithm (namely, COMMA$_{op}$) create an algorithm where different agents contribute to the global (to some extent) knowledge (the probability distribution) and then use the EMAS operators in order to perceive themselves and affect one another. Thus, the devised algorithm can be perceived as belonging to the socio-cognitive class of metaheuristics.

The proposed COMMA$_{op}$ hybrids were tested against the original algorithm using three QAP benchmark instances proposed by Éric Taillard (freely available on his website). The obtained results show that such modifications can bring new quality into the EDA research, making this already agent-oriented algorithm much closer to the idea of a socio-cognitive system. In the conducted experiment case, the visible improvement in the results produced by COMMA$_{op}$ was presented. Moreover, searches conducted by the modified COMMA$_{op}$ algorithms tend to be more focused than with the original algorithm.

Pseudocode 17 Pseudocode of $COMMA_{op}$ with crossover

```
Initialization of agents and other parameters
repeat
    Migrate agents among islands with low probability
    If agent's energy is higher than certain level: crossover of agent with another one on this island and mutate offspring
    If agent's energy is lower than certain level: remove agent from population
    Choose two agents and based on their fitness: exchange part of their energy
    Sort population of agents
    Calculate mutation rates
    Sampling of new solutions by adaptive mutation
until stopping_condition()
```

4.6. Differential Evolution and hybrid EMAS/DE

Differential Evolution (DE) is a very easy-to-implement yet successful metaheuristic algorithm belonging to the class of population-based optimization methods. The algorithm assumes a cooperation between the individuals (agents) right from the start, introducing a complex mutation mechanism (based on several neighbors) as the one and only means for developing a set of solutions. This algorithm can be treated as a sophisticated extension of the evolutionary algorithm (though not directly nature-inspired).

DE was originally developed as a real-valued parameter optimization algorithm [91]; however, multiple researchers have successfully proven its usefulness in the case of hybridization with both global optimization and local search algorithms [212]. This metaheuristic algorithm is used for solving continuous optimization problems, working in a similar way to evolutionary ones. Its pseudocode is shown in Pseudocode 18, and it is presented in Figure 4.8.

Pseudocode 18 Pseudocode of Differential Evolution

1: Initialize randomly all agents
2: **repeat**
3: **for** each agent x in population **do**
4: choose randomly three agents (individuals), a, b and c (distinct from each other and x)
5: Compute agent x's next position y as a result of the following vector equation: $y = a + F \cdot (b - c)$, where $F \in [0, 2]$ if y is better than x, replace x in the population with y.
6: **end for**
7: **until** the stopping condition is met

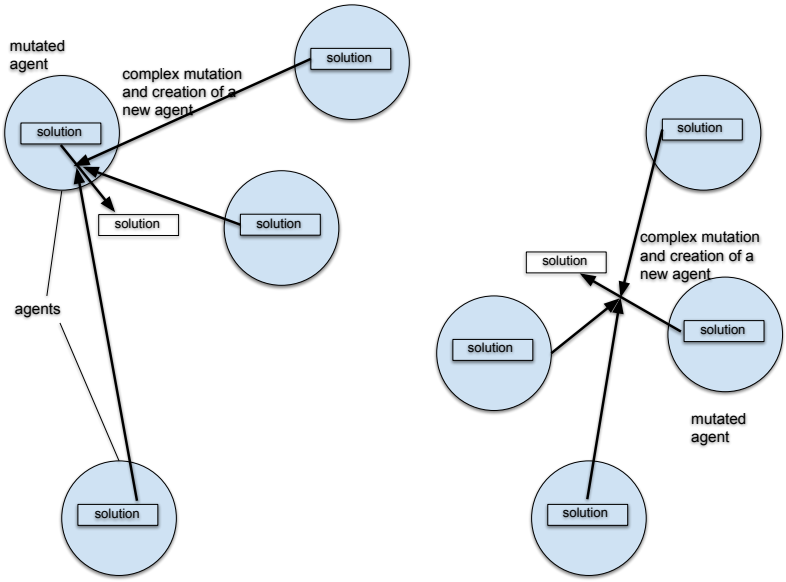

Figure 4.8. Differential Evolution algorithm

Because of the cooperation and relationships similar to perspective taking (inspiration during mutation by several agents), the algorithm can be classified as a socio-cognitive one. The research considering the further development of this metaheuristic (e.g., by introducing different species and relationships among them) is one of the interesting starting points for further additions to the socio-cognitive computing paradigm.

The DE algorithm has a proven efficiency in its hybridization with both global optimization and local search methods; however, there are also applications of DE

for discrete problems [136]. The "mutation" mechanism involved in DE is a very attractive step that can be hybridized in other methods; e.g., it has already been tried in PSO [136].

There are also numerous research works treating the hybridization of DE with other (mainly bio-inspired) global optimization algorithms such as Ant Colony Optimization [213], Simulated Annealing [214], Artificial Immune Systems [35], and many more [212].

A possible hybridization strategy of EMAS and DE (the research by Byrski et al. is going on; the first publication on this hybrid method is about to be completed) incorporates single DE steps for each member of a population with specific energy (e.g., lower than a certain level – assuming that these agents should be improved more than others). Therefore, a single step of the hybrid algorithm is either equivalent to the EMAS step or consists of the EMAS step enriched by a single DE step, possibly improving the results (see Figure 4.9).

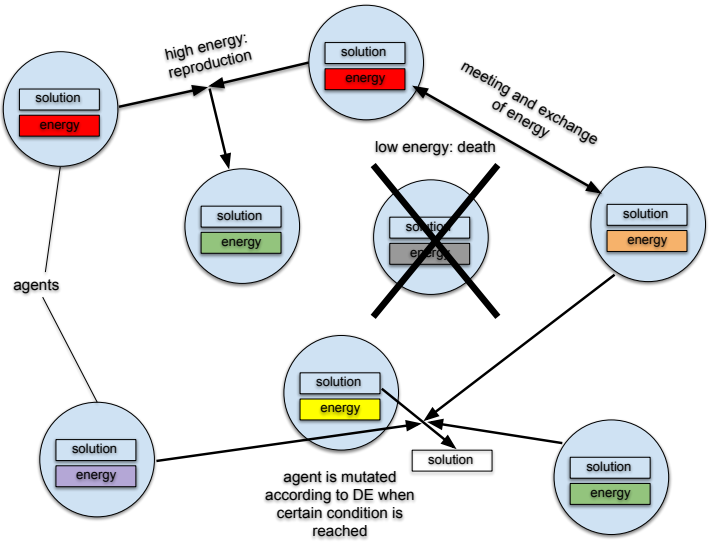

Figure 4.9. Hybrid of EMAS and Differential evolution algorithm

In this way, the obtained hybrid version of EMAS utilizes DE as a complex mutation operator. Thus, an interaction between the different agents present in the system arises, and the agent being mutated gets inspired by the solutions of the other agents; therefore, this hybrid will meet the requirements of the socio-cognitive computing paradigm. Moreover, the socio-cognitive features of the algorithm may be further extended and adapted; e.g., more interrelationships between the agents can

be employed, and the mutation can consider more or fewer agents or compute the resulting vector using a similar approach to a local search in memetic algorithms, for example. Of course, more species can be introduced into the system, and the mutation can consider them in an arbitrarily planned way.

4.7. EMAS and Particle Swarm Optimization

The idea of the hybridization of EMAS with PSO follows cultural and memetic inspirations by utilizing the PSO-defined movements of the solutions (agents' genotypes) as a kind of additional "local-search" algorithm for making the "worse" agents better by updating their solutions (see Fig. 4.10). This is not entirely a local-search algorithm, as PSO (of course) is a well-known global optimization technique; however, the planned synergy seems to be attractive and potentially not resistant to early-convergence problems.

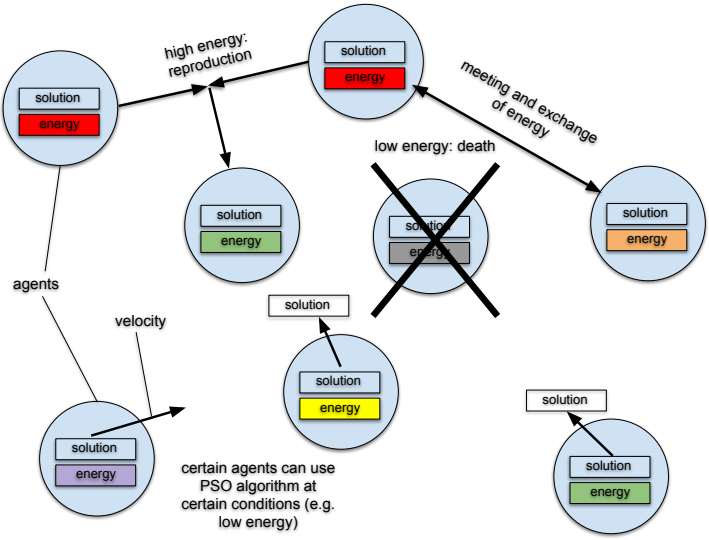

Figure 4.10. Hybrid of EMAS and PSO

In the proposed hybrid algorithm [215], the agent may be treated as either a regular EMAS agent – when its energy is higher than certain fixed level – and as a PSO particle – when its energy is lower (a dedicated energy threshold, the so-called "move" energy, is considered a parameter of the algorithm). Thus, better agents are evolved using well-known evolutionary methods, while worse agents update their solutions based on PSO rules.

Both EMAS and PSO have been counted into social metaheuristics; while fulfilling Definition 1.2.1, there is no doubt that their hybrid can also be classified as social. Moreover, this hybrid seems to be much closer to the socio-cognitive paradigm. The easy addition of different species and their relationships that further enhance and update the methods of a local search can lead to obtaining many novel successful metaheuristics.

4.8. Cultural algorithm, memetic algorithm, and memetic EMAS

Evolutionary metaheuristics have been continually developed throughout the years. Enhancement by hybridization with other search techniques seems particularly interesting, as this makes it possible to combine the advantages of respective methods. One example of such a hybridization that has been implemented with success is the introduction of memetic algorithms as a hybrid of evolutionary computing with a dedicated local search in order to improve exploitation in exploration-oriented methods.

Memetic algorithms originate from Richard Dawkins's theory of memes [216]. A meme is understood as a "unit of culture" that carries ideas, behaviors, and styles. This unit spreads among people by being passed from person to person within a culture by speech, writing, and other means of direct and indirect communication.

According to the theory elaborated by Dawkins (also known as *memetics*), memes undergo phenomena analogous to evolution. They compete, and those that are more prolific and seem to be useful for individuals are more likely to spread and be inherited. They can also be changed in the process of mutation. Unfit memes (which influence individuals in a harmful manner) become extinct and disappear. Noteworthy is the fact that memes can spread both vertically (in the course of inheritance) and horizontally (by various means of communication and knowledge sharing) [217, 218].

Memetic algorithms take advantage of population-based metaheuristics and local search methods and blend them together. The first researcher who proposed and applied a memetic metaheuristic with success was Pablo Moscato, who managed to combine an evolutionary algorithm and a simulated annealing method with the aim of solving the traveling salesman problem [4]. Memetic algorithms (initially popularized by Radcliffe and Surry [219], for example) have been proven to provide remarkable success [220]. The hybridization of evolutionary algorithms and local search methods was formalized by Krasnogor and Smith in [221].

Memetic algorithms may be classified as cultural algorithms, which were introduced by Robert G. Reynolds in 1994 [222]. They take into consideration both the evolutionary process and cultural relationships between individuals in the search

process. In systems utilizing the cultural algorithms, a culture represents knowledge about a search space (environment). Such knowledge constitutes a belief space (knowledge base). Individuals can share this information and communicate it to each other in order to notify others about promising or valueless regions of a search space. Thus, the culture affects the evolutionary process.

The cultural algorithm is shown in Pseudocode 19. In respect to the classic evolutionary search method, two additional operations have been introduced: influencing the population by cultural information (cf. line 10) and updating the belief space with knowledge acquired by individuals (cf. line 11).

Pseudocode 19 Pseudocode of Cultural Algorithm

```
 1: function SEARCH
 2:     P_0 ← initializePopulation()
 3:     knowledgeBase_0 ← initializeKnowledgeBase(P_0)
 4:     t ← 0
 5:     while ¬stopConditionIsMet do
 6:         evaluate(P_t)
 7:         P'_t ← select(P_t)
 8:         P_{t+1} ← crossover(P'_t)
 9:         mutate(P_{t+1})
10:         influence(P_{t+1}, knowledgeBase_t)
11:         knowledgeBase_{t+1} ← update(P_{t+1})
12:         t ← t + 1
13:     end while
14:     return best(P)
15: end function
```

Usually, a local search is applied in the course of evaluation (Baldwininan local search) or mutation (Lamarckian local search).

Memetic algorithms have been applied to numerous real–world problems; e.g., recognizing alphabetic characters and geometric figures [223], classification of image objects [224], designing frequency sampling filters [225], designing digital circuits [226], and many more.

Baldwinian local search According to the Baldwinian theory, an individual's predispositions and learning capabilities are inherited during reproduction [227]. The Baldwin effect follows the Darwinian theory of natural selection, as reproductive success is affected by an individual's learning capabilities passed on to its offspring by inheritance (while the genetic code itself remains unchanged).

Regarding evolutionary metaheuristics, a local search algorithm based on the Baldwinian theory is usually applied in the course of the evaluation process (see, e.g., [228]). Numerous potential descendants of the evaluated individual are generated, their fitness values are calculated, and the highest one is assigned to the individual (while its characteristics encoded in the genotype do not change). That is, fitness – which, in this case, represents learning capabilities – implies how good the solution will potentially be in future generations.

Lamarckian local search Eighteenth- and 19th-century biologist Jean-Baptiste Lamarck proposed a theory according to which an individual's characteristics acquired during its lifetime may be inherited by its offspring [229]. Each individual may improve and change its genetic material, which is then inherited by its descendants. Nowadays, Lamarckism has been entirely discredited as totally inconsistent with Darwin's Theory of Evolution [230].

In respect to the implementation of a Lamarckian local search in evolutionary algorithms, it is usually applied in the course of the mutation process. As the result of a local search starting at the point represented by an individual's genes, numerous solutions are sampled, and the most satisfactory one replaces the individual's genotype. Further selection is based on the fitness calculated for the new genotype. Lamarckian evolution may be applied to crossover as well – different combinations of parental genotypes might be analyzed, and the best would be chosen.

As with the Baldwin effect, a Lamarckian local search has been proven to be an effective solution for search and optimization problems [231, 232, 233].

Figure 4.11 depicts an evolutionary algorithm enhanced with a local search. The memetic method has been realized as an additional local search operator that processes a population in the process of evolution.

One of the main drawbacks of evolutionary metaheuristics is the computational overhead that results from their intrinsic feature – incessant modifications of population and, consequently, an immense number of evaluations performed towards obtaining a satisfactory solution [34]. This issue becomes even more demanding in the case of memetic algorithms that perform a significantly increased number of evaluations. In addition, noteworthy is the fact that fitness functions are often complex, computationally expensive, and time-consuming; therefore, they should not be overused.

Cultural and Memetic algorithms can, of course, be perceived as modifications of a classic evolutionary algorithm. A cultural algorithm introduces a base of knowledge that contains information shared by the individuals and used during the execution of variation operators. Of course, CA can become a basis for the introduction of different species or sexes and appropriately adapting the variation operators and selection.

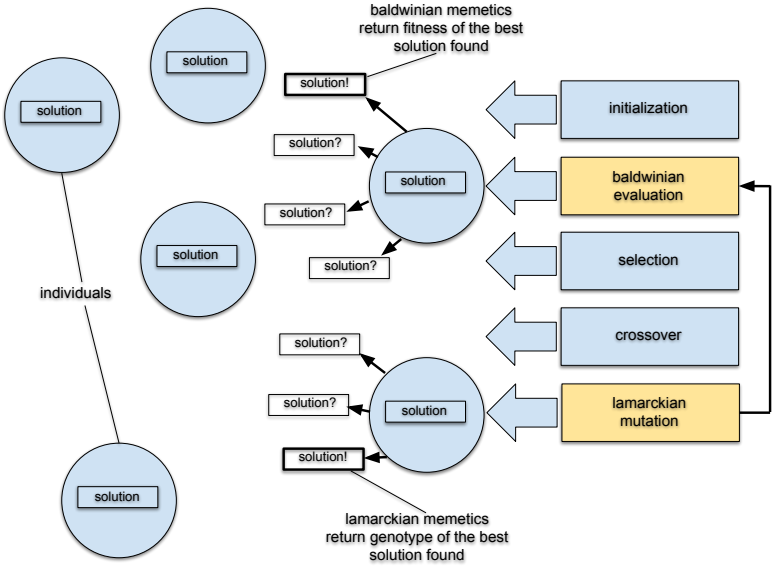

Figure 4.11. Memetic Evolutionary Algorithm (Baldwinian and Lamarckian)

Therefore, CA is a good starting point for developing new socio-cognitive algorithms. Note that the base of knowledge and sharing of the information makes CA quite close to the idea present in ACO, for example (pheromone table, cf. Section 1.3).

A memetic algorithm is a hybrid of an evolutionary algorithm and a local search method. Therefore, different species and different ways of perception can be introduced into MA when thinking about social-cognitivity (as in the case of a classic evolutionary algorithm – cf. Section 4.1). Therefore, MA will serve as the basis for a new socio-cognitive algorithm as well as a classic EA or CA.

Memetic EMAS EMAS can be enhanced with memetic algorithms in a very straightforward manner (see, e.g. [234]). Implementation of a local search may be realized by a modifying evaluation operator (the Baldwinian local search model) or mutation operator (the Lamarckian local search model).

Baldwinian memetics may be implemented in EMAS by returning the best fitness of its potential descendants found in the process of a local search, not the actual fitness value of the agent being evaluated. The genotype of the evaluated agent remains unchanged.

Lamarckian memetics are implemented in EMAS by running a local search procedure during the process of reproduction or at any moment during an agent's lifetime. An agent's genotype is mutated numerous times, and the best-encountered genotype is returned. Therefore, contrary to the Baldwinian model, both the agent's genotypes and fitness values are changed.

When handled with care, local search algorithms can enhance an individual's genotypes and bring it closer to the local or global extrema.

An outline of EMAS with memetization during the course of reproduction is illustrated in Figure 4.12. An agent applies a local search algorithm in order to create different solutions (represented with small circles, each containing a new genotype). These are evaluated, and the best one (marked with a bold border) replaces the agent's genotype. The local search may be applied only by the agent that has just been created during the process of reproduction.

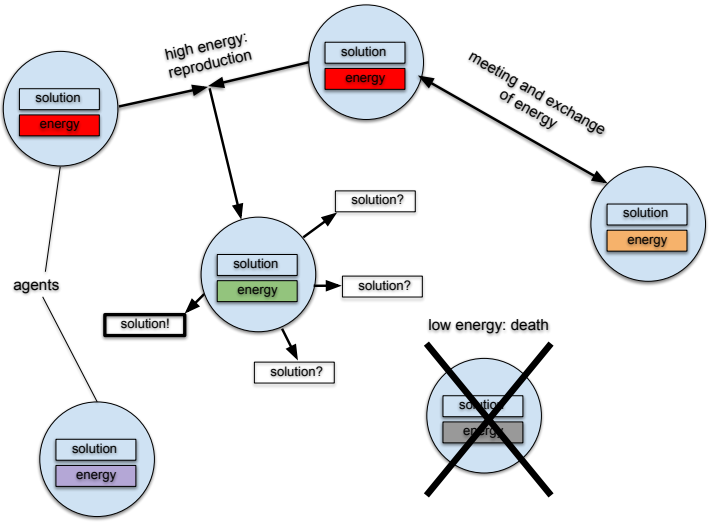

Figure 4.12. Memetic EMAS with local search realized in course of reproduction

The first successful experiments concerning the hybridization of EMAS with memetization were presented in [235], tackling real-value problems. In [236], a memetic variant of EMAS was employed to deal with the combinatorial optimization. Further research on this topic was developed in [237], where a mechanism of efficient fitness evaluations was additionally introduced in order to tackle real-value, multi-modal benchmark problems in up to 5000 dimensions. These algorithms are currently topic used in novel hybrid EMAS versions.

Lifelong Memetization in Memetic Multi-Agent System In the most common case, memetic algorithms are applied once at a precisely defined moment of an agent's lifetime (such as mutation or evaluation). However, bearing in mind the agent's autonomy and parallel ontogenesis, it is possible to run a local search from time to time at any arbitrary moment of an agent's lifetime based on the conditions of the environment or other factors. Thus, an agent can autonomously decide at any point in time whether it should apply a local search; what is more, one agent can run a search several times. It is noteworthy that such a mechanism can lead to the gradual improvement of the whole population, even between reproductions.

Algorithm depicted in Figure 4.13 introduces the realization of lifelong memetization in EMAS. Following its assumptions, a local search may be repeatedly applied during an agent's lifetime. First of all, an agent verifies whether it should run a memetic algorithm (e.g., this decision might be made by chance). If memetization is to be run, the appropriate action is performed, and the agent is provided with a genotype found by a local search and fitness that corresponds with this genotype.

Figure 4.13 depicts a schema of the lifelong memetization realized in EMAS. Contrary to Figure 4.12, all agents are able to memetize at any arbitrary moment of their lives.

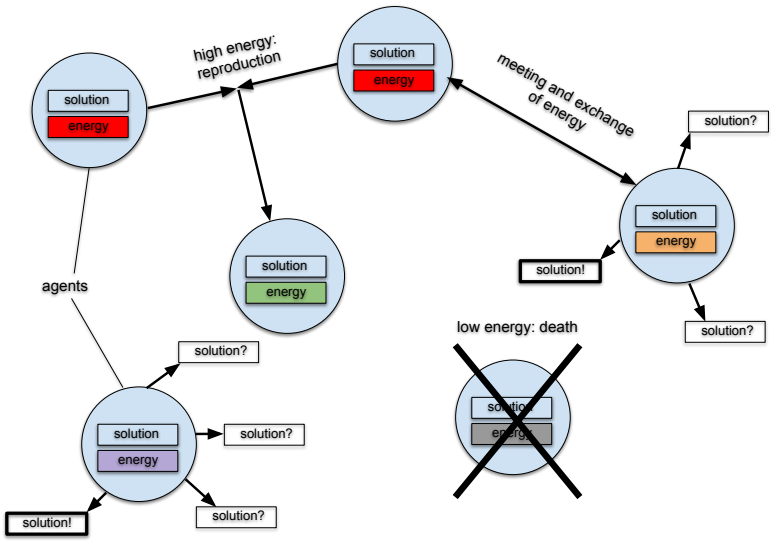

Figure 4.13. Memetic EMAS with local search realized during agent's lifetime

In [234], the first research on the subject of the hybridization of EMAS with lifelong memetization was discussed. This topic was then continued in [238].

As previously mentioned, memetic algorithms are believed to improve the results yielded by classic methods. However, the hybridization of a local search with EMAS does not differ remarkably from a similar approach applied to classic evolutionary algorithms – efficiency still remains the main issue. One has to handle memetization with care, not to hamper the computations by the increased number of evaluation events. Therefore, mechanisms that improve the efficiency of memetics are essentially needed.

Memetic versions of EMAS were defined in [85] and applied to solving high-dimensional benchmark problems (using a dedicated fitness buffering mechanism [237]). Another very practical application was applying a memetic EMAS to solving an inverse problem; namely, the optimization of a rotating disc [239].

Both of these hybrids belong to the class of agent-based metaheuristics (actually EMAS-like algorithms); therefore, they can be very easily enhanced with socio-cognitive mechanisms such as introducing additional species or sexes and complex relationships among them. They can be perceived as a good starting point for defining new metaheuristics.

4.9. Classic metaheuristics in light of Social Cognitive Theory

Let us now reference the features of the classic metaheuristics referenced in this chapter to the elements of Social cognitive Theory. Some of these already express some features of SCT, and many of them can become a good basis for extending towards SCT metaheuristics by the means of different modifications and hybridizations. Just as it was proposed in Section 3.4, all of these features will be annotated with a letter corresponding to the estimated difficulty of introducing these features (P: present, E: easy, M: moderate, C: complex).

In the beginning, let us focus on the factors of triadic reciprocal causation identified by Bandura:

- Personal: the algorithms considered here utilize none (PEA) or only selected (DE) features of agency. Individuals are not aware of their existence, and they do not perceive others. However, in the case of DE or MA, individuals interact with others in order to realize their tasks. In order to implement the first two factors of triadic reciprocal causation, the introduction of agency is unavoidable (it may be done in a similar way as in EMAS; i.e., the introduction of distributed selection and parallel ontogenesis). Then, observation and consequences of the observation can be introduced in order to modify the parameters of the variation

operators used during reproduction, for example. It is to note that a very similar mechanism was used in Evolution Strategies [65], where individuals inherited not only their genotypes but also the parameters of mutation.

  - PEA, CA, CSA: these algorithms need to be enhanced with agency first (cf. Section 1.6), then they can observe the history of their successes and defeats and use this information in order to adapt the variation operators (M). The history may also be prepared and inherited in the Lamarckian way; i.e., the success and defeat count can be passed on to the offspring.
  - CEA, DE: in these algorithms, a kind of cooperation is included as either inspiration (in DE) or competition (in CEA). If this process is made longer than only one-step, certain observations can be made, and the next steps of the algorithm (planned for particular individuals) can be parametrized based on these observations (E).
  - MA: this setting is similar to PEA; but in this case, the observations of the history of the individual can lead to adapting the parameters of the local search involved. Lamarckian inheritance can also be involved (M).

- Behavioral: reward or punishment may be one-step (but this is nothing new when compared to a classic selection mechanism), agency and resource-based selection (like in EMAS) can be considered, or a Lamarckian inheritance of the history of an individual may be used in order to make the individual perceive its deeds and build a simple cognitive model on top of them, using the outcome to parametrize its actions (e.g., changing the parameters of the variation operators).

  - PEA, CA, CSA: the introduction of any agency features is first needed in the case of these algorithms; even Lamarckian inheritance may be a first step towards reaching this goal. The individual can gradually observe how its assessment (based on changes in the quality function, for example) varies and, using this information, can affect the variation operator parameters (M).
  - CEA, DE: if the cooperation/competition involved is enhanced, a particular individual can consider its history as a basis for getting rewards or punishment, and this information can be considered in planning their next actions (e.g., affecting the parameters of the variation operators) (E).
  - MA: this setting is similar to PEA; but in this case, observations of the history of the individual can lead to gathering punishment or reward (points) and, based on this information, the individual can affect the parameters of the local search involved. Lamarckian inheritance can also be involved (M).

- Environmental: different notions of environment may be considered; e.g., the search space can be annotated (however, this approach would be very inefficient when high-dimensional problems are considered because of the “curse of

dimensionality" [240]), a virtual environment may be introduced (like evolutionary islands in coarse-grained PEA or the grid in fine-grained PEA), and certain information (e.g., in the form of resources) may be scattered so the individuals can consider the information in the course of undertaking their actions. Of course, certain aspects of agency would also be necessary.

  - PEA: a virtual environment is present in this algorithm either in the form of evolutionary islands or a grid. Certain information can be scattered (e.g., pointing out the promising or unpromising areas in the search space). Awards can also be implemented similar to the EMAS finite resources needed for reproduction (E).
  - CEA, DE, CSA, MA: all of these algorithms can be easily implemented following the coarse-grained decomposition of a population; in such settings, the same approach that is described in the case of PEA can be used (C).
  - CA: this algorithm utilizes the notion of a global knowledge base; the information present in this base can be annotated with additional hints for individuals, showing the reliability of a particular element, for example (E).

As shown before, the agency of the learners described by Bandura fits very well into the agency perceived in the world of software agents, so part of the algorithms considered here would be easily adaptable towards these features. The other algorithms should be enhanced from the point of agency first:

- Individual agency: any individual having certain features of agency (like DE) can already be classified as having this feature. The others must first be enhanced with agent features (at least partial autonomy).
  - PEA, CSA: if the individuals are equipped with any agent features (e.g., parameters affecting the application of variation operators), this feature will be present (E).
  - CEA: if the competing individuals are enhanced with any individual parameters affecting the competition (thus, selection), this feature may be easily reached (E).
  - DE: reproduction based on several genotypes can be influenced by the particular parameters of the individual, fulfilling the requirements for this feature (E).
  - CA: if the individuals are equipped with any agent features (e.g., parameters affecting the consideration of particular information present in the knowledge base), this feature will be present (E).
  - MA: the setting is similar to PEA; but in this case, the local search may be influenced by the individual parameters (E).

- Proxy agency: this may, of course, be realized as the employment of another individual and delegating certain tasks to be fulfilled. Thus, hierarchical methods of computing may be considered. MA becomes an especially good example here because of the local search run for each individual.
  - PEA, CEA, DE, CA, CSA: a local search can be introduced, making the algorithm similar to a memetic algorithm. Other means can also be followed; e.g., the construction of a decentralized tabu list (cf. Section 4.3) (M).
  - MA: a local search is conducted here for all of the individuals (P).
- Collective agency: practically all of the metaheuristics can be treated as collectively-intelligent; however, this feature may be reached very simply (choosing the best solution of all present) or in a more complex way (choosing a certain weighted average) following the mixture-of-experts approach [241].
  - PEA, CEA, DE, CA, CSA, MA: in all of these algorithms, the final solution is usually selected as the best of those currently present. It is to note that these solutions are produced using a complex intelligent search (metaheuristics can be treated as much more complex and intelligent algorithms than simpler heuristics: Monte Carlo methods [21]).

The relationship between the referenced metaheuristics and human agency can be found by considering the following properties (again, as in the case of the previous features, at least the basic aspects of agency are required to be implemented first):

- Intentionality: assuming the individuals are enhanced with even simple agency features (e.g., certain parameters of the search are encoded just like in the solution or perhaps also passed on to the offspring [Lamarckian inheritance]), these parameters can be used for undertaking certain actions. In the simplest cases, they can affect the operators or cause some of them to be neglected (e.g., the individual can decide to avoid mutation).
  - PEA, CEA, DE, CA, CSA, MA: all of these algorithms can consider the introduction of certain parameters (like an additional genotype describing the mutation ranges) that will affect the variation operators that are applied. In this way, the intentionality may be easily introduced (E). As an alternative, of course, upgrading to an EMAS-like hybrid may be considered.
- Forethought: A dedicated cognitive model must be implemented in order to follow the changes of a selected parameter (like the diversity of a population, for example). Such models can be inherited in a Lamarckian way because of the generation mechanism present in all of these algorithms.
  - PEA, CEA, DE, CA, CSA, MA: even a simple statistical model can follow the events (meetings, inspirations, reproduction partners, etc.) and learn

from them; later, they can be used for predicting the diversity of the search in the system and the appropriate control of the parameters of the individuals, for example (M).

- Self-reactiveness: the individual should participate in control over the course of an algorithm (e.g., its preferences and decisions should affect the use of certain variation operators). This may be based on persisting certain parameters and an inference of the information based on an observation of its changes. The actual model can be very simple or complex, but the individual needs to have at least a little agency introduced.
  - PEA, CEA, DE, CA, CSA, MA: a simple or complex model can be trained based on an observation of the events happening to an individual; the inferred information can then be used to select the next actions or to skip some of them (M).
- Self-reflectiveness: a higher-level mechanism to the one considered in fulfilling self-reactiveness. The individuals can observe their efficiency and efficacy based on their decisions affecting the selection of actions (e.g., variation operators applied during the search) and adapt the strategy of undertaking those decisions.
  - PEA, CEA, DE, CA, CSA, MA: a self-reactiveness feature is required; based on top of this, another cognitive model would serve to adapt the strategy of particular individuals in the context of adapting the actions realized by them in the system (C).

Introducing such a mechanism might bring the referenced algorithm significantly closer to the standards of agent-oriented models such as BDI [112] or M-agent [113].

Individuals (agents) can acquire certain knowledge, build and adapt their models, and follow their assumptions, executing their actions based on the information gathered in the environment and observed among other agents. These observations realized by the individuals/agents are described as follows:

- *Attention*: this type of observation requires a perception of the environment and other individuals.
  - PEA: attention is not present in the basic version of the algorithm, but after introducing a local search or sophisticated selection mechanism (e.g. resource-based), for example, it may be easily introduced (E).
  - CEA, DE, CSA: the introduction of even a simple perception and autonomy would lead to fulfilling this feature (as a competition/comparison of individuals exist in these algorithms), so certain means for communication and exchange of information are easy to add (E).

  - CA: individuals perceive the knowledge base and interpret the information contained there (P).
  - MA: individuals perceive and store the information gathered during a local search (P).

- *Retention*: observing the behavior of other individuals is generally not present in these algorithms; however, it can be implemented right after enhancing the agency of the individuals. In a simpler case, only the parameters of other algorithms can be retained and used in the decision process by other individuals.
  - PEA, CEA, DE, CA, CSA, MA: storage of information regarding the development of the particular solution may be implemented in the individual; this information may be passed on to the offspring in Lamarckian inheritance (E).

- *Production*: the stored information about the behavior of other individuals/agents can be used for reenacting the same action. Thus, the perspective-taking comes into fruition; in the simplest case, different inspirations of the solutions presented by other individuals may be used.
  - PEA, CA, CSA, MA: observation of the solutions presented by other individuals; (e.g., residing on the same island) can be used to plan the reproduction parameters or avoid certain areas in the solution space (M).
  - CEA, DE: the interaction between individuals is already considered in the course of the selection/diffusion process (P).

- *Motivational process*: this feature may be implemented as a reenactment (with some changes possibly being introduced) of the behaviors copied from other individuals, providing that the outcome was positive (evaluated by the agent itself or, better, by the other agents). In the simplest case other than behavior, the information contained by other agents can be perceived and become the basis for getting inspired. This may be considered as observing the production feature and adapting their outcomes.
  - PEA, CEA, DE, CA, CSA, MA: the strategy of reenacting certain behaviors or, rather, getting inspired by certain information presented by other individuals cay be observed and adapted based on the outcomes (M).

As one can see, the referenced metaheuristics have many features that are compatible with Social Cognitive Theory (although there is still room for introducing more) in order to further increase the cognitivity of the particular agents, increase the learning capability, and enhance the quality of the collective intelligence that consists of particular agents.

4.10. Hybrid EMAS-related metaheuristics in light of Social Cognitive Theory

Let us now reference the features of the EMAS hybrids referenced in this chapter to the elements of Social Cognitive Theory. Many of them should have a lot in common with SCT because of the intrinsic agency that is present in these systems by design. Just as it was proposed in Section 3.4, all of these features will be annotated with a letter corresponding to the estimated difficulty of introducing these features (P: present, E: easy, M: moderate, C: complex).

Again, let us focus in the beginning on the factors of triadic reciprocal causation identified by Bandura:

- Personal: the agency feature is inherent to EMAS and its hybrids, certain means for assessing the quality of self is needed by the agent, and (in EMAS-like algorithms) it may be very simply based on gathered energy.
  - COEMAS, iEMAS, elEMAS, EMAS/PSO, EMAS/DE, memEMAS: all of these algorithms utilize the finite resource – energy – in order to assess the quality of the solutions (and the agent) (P). Based on the level of energy, certain actions can be done that cannot be done when the energy level falls.
  - COMMA$_{op}$: a similar energy mechanism was also introduced into this algorithm, so there is no doubt that self-efficacy assessment can also be based on this resource (P).
- Behavioral: the response of the system in the EMAS algorithms is based on the energy-exchange mechanism; i.e., better agents receive more energy and can ultimately realize more actions, while worse agents will be removed from the system.
  - COEMAS, iEMAS, elEMAS, EMAS/PSO, EMAS/DE, memEMAS, COMMA$_{op}$: the resource present in the system affects the ability of particular agents to realize certain actions (P).
- Environmental: after assuming a certain real or virtual environment (either a search space (though very complex) or virtual structure (an effect of the decomposition of the population)), information can be scattered, or parts of the environment can be annotated (and other agents can be made aware of this).
  - COEMAS, elEMAS, EMAS/PSO, EMAS/DE, memEMAS, COMMA$_{op}$: all of these algorithms may be implemented in a virtual environment (like the island-model of evolution); in these islands, certain information may be scattered (e.g., excessive energy, information about attractive search space zones, etc.) (M).

– iEMAS: one of its versions [84] introduced a dedicated container for the energy left by early-removed agents. Other agents could pick up this energy and work further (P).

As shown before, the agency of the learners described by Bandura fits very well into the agency perceived in the world of software agents. In particular, agents can be easily described by these notions:

- Individual agency: this feature is intrinsic to all of the agent-oriented algorithms, including EMAS hybrids (of course).
  - COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature is present in all of the considered algorithms (P).
- Proxy agency: one of the actions of the agents might be the introduction of a new agent into a system, delegate a certain task to it (or to a group of such agents), and gather and process the results. One can imagine easily an agent delegating several more specific agents that will be responsible for a thorough search of a part of its vicinity (another way of implementing memetization); this applies to all of the referenced algorithms; e.g., an evolutionary algorithm could be implemented in an agent way, delegating the search of particular parts of a search space to specialized agents (those having appropriately tuned variation operators, for example). The HGS algorithm was proposed by Schaefer and Kołodziej [242] in such a way.
  - COEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE: introducing dedicated agents, realizing that a local search could fulfill the requirements of this feature, for example (M).
  - iEMAS: this feature is present as the immunological agents are dispatched during the death of an agent to constitute a decentralized and distributed tabu list (P).
  - memEMAS: this feature is present as the local search is heavily utilized in both options of the memetic EMAS discussed in this book (P).
- Collective agency: All of these algorithms are population-based, as they produce a lot of solutions. However, the actual outcome is chosen out of all individuals; this might be a very simple implementation of the described feature. Actually, this solution should be an outcome of the work of individuals; e.g., an evolutionary island, or a certain species of individuals. Thus, this feature may be easily introduced into all of the agent-based algorithms and with moderate effort into the classic ones.
  - COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature is present in all of the considered algorithms (P).

The relationship between the referenced metaheuristics and human agency can be found by considering the following properties:

- Intentionality: each agent in EMAS hybrids decides on its own about undertaking of certain decisions; e.g., performing the action of reproduction or memetization. Making the individual in evolutionary and co-evolutionary algorithms autonomous will easily fulfill this feature.
  - COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature is present in all of the considered algorithms (P).
- Forethought: an agent can build its own model in order to predict the outcomes of its actions (this is not implemented in the basic EMAS nor its hybrids). Such a model can be used to make sure that too many agents are not removed if one of them uses certain levels of energy transfers during the meetings, for example (e.g., too high parts of energy are transferred, and the population becomes extinct). Making the individuals in evolutionary and co-evolutionary algorithms autonomous will prepare the ground for introducing such a feature, and it is connected with moderate effort; the same applies for $COMMA_{op}$.
  - COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature can be introduced with limited effort into all of the considered algorithms (M).
- Self-reactiveness: an agent can have its own model built in order to adapt its parameters and choose the appropriate actions based on the successes in the localization of the sub-optimal solutions of the problem, for example. A similar model can be introduced into the $COMMA_{op}$ algorithm.
  - COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature can be introduced into all of the considered algorithms (M).
- Self-reflectiveness: assuming that self-reactiveness is present, the strategy of undertaking certain actions may be monitored and, based on the outcomes of the search, may be further adapted.
  - COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature can be introduced into all of the considered algorithms (M).

Introducing such a mechanism might bring the referenced algorithm significantly closer to the standards of agent-oriented models such as BDI [112] or M-agent [113].

The agents can acquire certain knowledge, build and adapt their models, and follow their assumptions, executing their actions based on the information gathered

in the environment and observed among other agents. These observations realized by the agents are described as follows:

- *Attention*: agents perceive the vicinity in the solution space and the solutions of other agents (utilizing the neighborhood notion, either in the search or virtual space, and modifying their actions accordingly). Such a mechanism can be introduced into all of the referenced methods; e.g., providing that the population of agents is embedded in a certain environment and a dedicated neighborhood notion is introduced.

  – COEMAS, iEMAS, elEMAS, COMMA$_{op}$, memEMAS: this feature can be introduced into all of these algorithms (E).

  – EMAS/PSO, EMAS/DE: both of these algorithms perceive their neighbors and use them for inspiration when realizing the reproduction or changing of position of an agent (P).

  Retention: the observations performed by the agent can lead to the construction of cognitive models based on the actual and historical features perceived in the population. The agent can accumulate this knowledge, deliberate, and infer new information and estimate further actions; however, this is not implemented in the proposed metaheuristics. The introduction of such a mechanism should be easy in the case of all EMAS hybrids.

  – COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature can be introduced into all of the considered algorithms (E).

- *Production*: the actions realized by the agents stem in a straightforward way from their observations and retained information. This mechanism is inherently present in all of the EMAS hybrids.

  – COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature can be introduced into all of the considered algorithms (P).

- *Motivational process*: a dedicated mechanism for monitoring the strategy of particular agents can be introduced; this strategy can be adapted based on the efficacy of an agent. The introduction of such a mechanism into EMAS hybrids should not pose any problems.

  – COEMAS, iEMAS, elEMAS, COMMA$_{op}$, EMAS/PSO, EMAS/DE, memEMAS: this feature can be introduced into all of the considered algorithms (E).

As one can see, the referenced metaheuristics have many features that are compatible with Social Cognitive Theory (although there is still room for introducing more) in order to further increase the cognitivity of the particular agents, increase the learning capability, and enhance the quality of the collective intelligence that consists of particular agents.

5. Summary

This monograph is aimed at proposing a novel class of metaheuristics known as socio-cognitive computing. These algorithms stem from psychological and sociological inspirations and are well-rooted in the agent-based computing paradigm. In this monograph, a number of such algorithms were identified, referenced, and described in the light of Social Cognitive Theory-related characteristics. Namely, a number of EMAS-related algorithms and their hybrids were referenced, an existing multi-type ACO algorithm and (last but not least) two novel socio-cognitive PSO and ACO metaheuristics were described. All of these algorithms have already been published, so this monograph can be treated as a means for providing a complete guide to the reader for the proposed class of algorithms. The presentation of social-metaheuristic algorithms allowed us to discuss introducing cognitive abilities and features into these computing methods.

In Table 5.1, the effort needed for introducing the social cognitive aspects discussed in this book are shown. The basic socio-cognitive algorithms (namely, SC-ACO and SC-PSO) appear to be well-fitted into the socio-cognitive computing paradigm, being relevant to many aspects of Social Cognitive Theory (or the implementation seems to be easy). SDS also seems very relevant to SCT, so this may be considered one of the next steps for the further enhancement of socio-cognitive metaheuristics.

The algorithms can be divided into several groups considering their relevance to particular aspects of SCT; e.g., classic metaheuristics share the difficulty of introducing the human agency mode (individuality (E)), human agency properties (intentionality (E), forethought and self-reactiveness [M), and self-reflectiveness [C]), and type of observation (retention (E) and motivational process (M]).

Considering EMAS hybrids, these algorithms share the difficulty of introducing triadic reciprocal causation (personal and behavioral (P)), human agency modes (individual and collective (P)), human agency properties (intentionality (P)), and other types of observations (retention (E), production (P), and motivational process (E)).

Table 5.1. Estimation of effort needed for introducing aspects of Social Cognitive Theory into discussed metaheuristics (P: Present, E: Easy, M: Moderate, C: Complex).

	Swarm intelligence				Classic metaheuristics						Hybrid EMAS-related metaheuristics						
Algorithm	Socio-Cognitive ACO	Multi-type ACO	Socio-cogntive PSO	Stochastic diffusion search	Parallel evolutionary algorithm	Co-evolutionary Algorithm	Differential evolution	Cultural algorithm	Clonal selection algorithm	Memetic algorithm	Co-evolutionary EMAS	Immunological EMAS	Elitist EMAS	COMMAop	EMAS/PSO	EMAS/DE	Memetic EMAS
Triadic reciprocal causation																	
Personal	P	C	P	P	M	E	E	M	M	M	P	P	P	P	P	P	P
Behavioral	P	P	P	P	M	E	E	E	M	M	P	P	P	P	P	P	P
Environmental	M	M	E	M	E	C	C	E	C	C	M	P	M	M	M	M	M
Human agency modes																	
Individual	P	P	P	P	E	E	E	E	E	E	P	P	P	P	P	P	P
Proxy	M	M	M	C	M	M	M	M	M	P	M	P	M	M	M	M	P
Collective	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
Human agency properties																	
Intentionality	P	P	P	P	E	E	E	E	E	E	P	P	P	P	P	P	P
Forethought	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Self-reactiveness	P	E	P	E	M	M	M	M	M	M	M	M	M	M	M	M	M
Self-reflectiveness	M	M	M	M	C	C	C	C	C	C	M	M	M	M	M	M	M
Types of observations																	
Attention	P	P	P	E	E	E	E	P	E	P	E	E	E	E	P	P	E
Retention	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
Production	P	P	P	P	M	P	P	M	M	M	P	P	P	P	P	P	P
Motivational process	E	E	E	E	M	M	M	M	M	M	E	E	E	E	E	E	E

In order to summarize the observations presented in Table 5.1, two additional tables were prepared. The first of them (Tab. 5.2) shows the difficulty of adopting the SCT features into the particular groups of the discussed algorithms. Besides the obvious observation that swarm intelligence algorithms are well-fit into SCT (as there are very many features of SCT present in these algorithms), it is somewhat surprising that the best-suited group of algorithms are EMAS hybrids. The reason for this is probably the inherent agency of these computing methods. Classic metaheuristics have the highest number of C and M codes, so it seems that they need the most work in order to be fully adopted to SCT. The classic algorithms have the highest number of C and M codes; this is easy to explain: in order to be enhanced with SCT features, these algorithms need to have the agency features introduced first, so a significant effort is needed.

Table 5.2. Difficulty of adopting SCT features in particular groups of considered algorithms presented as number of occurrences of difficulty codes (P: present, E: easy, M: moderate, C: complex) in Table 5.1.

	P	E	M	C
Swarm intelligence	28	12	14	2
Classic metaheuristics	11	29	34	10
Hybrid EMAS-related metaheuristics	47	19	32	0

The second table (Tab. 5.3) shows the difficulty of adopting particular groups of SCT features in all of the discussed algorithms. P and M difficulty codes are of the highest number; however, the observations are also very easy to implement (38 E codes). The C codes are very low in number; therefore, the conclusion can be drawn that all of the selected and discussed algorithms can adopt SCT features without extended effort.

Table 5.3. Difficulty of adopting different SCT groups of features in all considered algorithms presented as number of occurrences of difficulty codes (P: present, E: easy, M: moderate, C: complex) in Table 5.1.

	P	E	M	C
Triadic reciprocal causation	22	8	16	5
Human agency modes	20	0	13	1
Human agency properties	13	8	41	6
Types of observations	20	38	10	0

The selection of the algorithms discussed in this book is, of course, arbitrary; however, the context presented goes beyond these algorithms, and the elements of Social Cognitive Theory may be further discussed in their application to other metaheuristics. Therefore, this monograph is not a closed one that focuses only on selec-

ted aspects of the metaheuristics, but it may be treated as a starting point for future deliberations on extending such algorithms in the way covered by the proposed socio-cognitive computing paradigm.

Putting together the information given in this book, the description of the algorithms and deliberations on introducing aspects relevant to social cognitive learning (like autonomy/agency, perspective taking, or diversity of population) gives the reader a firm starting point for extending this paradigm, allowing us to introduce new algorithms based on existing ones.

Thus, the ideas coined by Albert Bandura several decades ago are still vital, not only in a psychological context but also in the area of metaheuristic computing.

List of acronyms

- ACO – Ant Colony Optimization
- CA – Cultural Algorithm
- CEA – Co-evolutionary Algorithm
- COEMAS – Co-evolutionary EMAS
- CSA – Clonal Selection Algorithm
- DE – Differential Evolution
- EA – Evolutionary Algorithm
- elEMAS – elitist EMAS
- EMAS – Evolutionary Multi-Agent System
- EMAS/DE – Evolutionary Multi-Agent System hybridized with Differential Evolution
- EMAS/PSO – Evolutionary Multi-Agent System hybridized with Particle Swarm Optimization
- iEMAS – immunological EMAS
- MA – Memetic Algorithm
- MAS – Multi-Agent System
- memEMAS – Memetic EMAS
- MT-ACO – Multi-type Ant Colony Optimization
- PEA – Parallel Evolutionary Algorithm
- PSO – Particle Swarm Optimization
- SC-ACO – Socio Cognitive Ant Colony Optimization
- SC-PSO – Socio Cognitive Particle Swarm Optimization
- SCT – Social Cognitive Theory
- SDS – Stochastic Diffusion Search
- TSP – Traveling Salesman Problem

Bibliography

[1] J. Holland, *Adaptation in natural and artificial systems*. MIT Press, 1975.

[2] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15, Frommann-Holzboog, 1973.

[3] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[4] P. Moscato, "Memetic Algorithms: A Short Introduction," in *New Ideas in Optimization* (D. Corne and et al., eds.), pp. 219–234, McGraw-Hill, Maidenhead, 1999.

[5] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[6] M. Vose, *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, 1998.

[7] K. Sörensen, "Metaheuristics the metaphor exposed," *International Transactions in Operational Research*, vol. 22, no. 1, pp. 3–18, 2015.

[8] M. Dorigo, *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Milano, Italy, 1992.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE International Conference on Neural Networks, 27 November – 1 December, Perth*, vol. 4, pp. 1942–1948, IEEE, 1995.

[10] A. Bandura, *Social foundations of thought and action: a social cognitive theory*. Prentice-Hall, Englewood Cliffs, N.J., 1986.

[11] A. Byrski, E. Świderska, J. Łasisz, M. Kisiel-Dorohinicki, T. Lenaerts, D. Samson, B. Indurkhya, and A. Nowé, "Socio-cognitively inspired ant colony optimization," *Journal of Computational Science*, vol. 21, pp. 397–406, 2017.

[12] I. Bugajski, P. Listkiewicz, A. Byrski, M. Kisiel-Dorohinicki, W. Korczynski, T. Lenaerts, D. Samson, B. Indurkhya, and A. Nowé, "Enhancing particle swarm optimization with socio-cognitive inspirations," *Procedia Computer Science*, vol. 80, pp. 804–813, 2016.

[13] A. Byrski, R. Drezewski, L. Siwik, and M. Kisiel-Dorohinicki, "Evolutionary multi-agent systems," *The Knowledge Engineering Review*, vol. 30, pp. 171–186, 3 2015.

[14] R. M. Ragnarsson, H. Stefánsson, and E. I. Ásgeirsson, "Meta-heuristics in multi-core environments," *Systems Engineering Procedia*, vol. 1, pp. 457–464, 2011.

[15] T. Luong, N. Melab, and E. Talbi, "Gpu computing for parallel local search metaheuristic algorithms," *IEEE Transactions on Computers*, vol. 62, pp. 173–185.

[16] S. Pimminger, S. Wagner, W. Kurschl, and J. Heinzelreiter, "Optimization as a service: On the use of cloud computing for metaheuristic optimization," in *Computer Aided Systems Theory - EUROCAST 2013* (R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, eds.), pp. 348–355, Springer, Berlin, 2013.

[17] Z. Pooranian, M. Shojafar, J. H. Abawajy, and A. Abraham, "An efficient meta-heuristic algorithm for grid computing," *Journal of Combinatorial Optimization*, vol. 30, pp. 413–434, Oct 2015.

[18] A. Byrski, *Agent-Based Metaheuristics in Search and Optimisation*. AGH University of Science and Technology Press, 2013.

[19] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*. Springer, Berlin, Heidelberg, 2004.

[20] Z. Michalewicz, "Ubiquity symposium: Evolutionary computation and the processes of life: the emperor is naked: evolutionary algorithms for real-world applications," *Ubiquity*, vol. 2012, pp. 1–13, 2012.

[21] M. Kalos and P. Whitlock, *Monte Carlo Methods*. Wiley, Hoboken, NJ, 2008.

[22] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2003.

[23] R. Horst and P. Pardalos, *Handbook of Global Optimization*. Kluwer, 1995.

[24] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, New York, NY, 1998.

[25] S. Droste, T. Jansen, and I. Wegener, "Upper and lower bounds for randomized search heuristics in black-box optimization," *Theory of Computing Systems*, vol. 39, pp. 525–544, 2006.

[26] M. Ali, C. Storey, and A. Törn, "Application of stochastic global optimization algorithms to practical problems," *Journal of Optimization Theory and Applications*, vol. 95, pp. 545–563, 1997.

[27] D. Wolpert and W. Macready, "No free lunch theorems for search," Tech. Rep. SFI-TR-02-010, Santa Fe Institute, 1995.

[28] F. Glover and G. Kochenberger, *Handbook of Metaheuristics*. Springer, Berlin, Heidelberg, 2003.

[29] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[30] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Metaheuristics* (F. Glover and G. Kochenberger, eds.), vol. 57 of *International Series in Operations Research & Management Science*, pp. 457–474, Springer US, 2003.

[31] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, and A. Chatterjee, *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer, Berlin, Heidelberg, 2005.

[32] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu Enterprises, 2012.

[33] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Wiley, Hoboken, NJ, 2009.

[34] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1989.

[35] D. Dasgupta, *Artificial Immune Systems and Their Applications*. Springer, Berlin, Heidelberg, 1999.

[36] M. Jackson, "Social systems theory and practice: The need for a critical approach," *International Journal of General Systems*, vol. 10, no. 2–3, pp. 135–151, 1985.

[37] W. Mayrhofer, "Social systems theory as theoretical framework for human resource management – benediction or curse?," *Management Revue*, vol. 15, no. 2, pp. 178–191, 2004.

[38] E. H. Durfee and J. Rosenschein, "Distributed problem solving and multiagent systems: Comparisons and examples," in *Proceedings of the 13th International Workshop on DAI* (M. Klein, ed.), (Lake Quinalt, WA, USA), pp. 94–104, 1994.

[39] S. McArthur, V. Catterson, and N. Hatziargyriou, "Multi-agent systems for power engineering applications? part i: Concepts, approaches, and technical challenges," *IEEE Transactions on Power Systems*, vol. 22, pp. 1743–1752, November 2007.

[40] J. George, M. Gleizes, P. Glize, and C. Regis, "Real-time simulation for flood forecast: an adaptive multi-agent system staff," in *Proceedings of the AISB'03 Symposium on Adaptive Agents and Multi-Agent Systems, 7-11 April, Aberystwyth*, pp. 109–114, University of Wales, 2003.

[41] N. Jennings, P. Faratin, M. Johnson, T. Norman, P. OBrien, and M. Wiegand, "Agent-based business process management," *International Journal of Cooperative Information Systems*, vol. 5, no. 2–3, pp. 105–130, 1996.

[42] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, 2008.

[43] B. Lobel, A. Ozdaglar, and D. Feijer, "Distributed multi-agent optimization with state-dependent communication," *Mathematical Programming*, vol. 129, no. 2, pp. 255–284, 2011.

[44] M. Wooldridge, *An Introduction to Multiagent Systems*. John Wiley & Sons, Hoboken, NJ, 2004.

[45] P. Uhruski, M. Grochowski, and R. Schaefer, "A two-layer agent-based system for large-scale distributed computation," *Computational Intelligence*, vol. 24, pp. 191–212, July 2008.

[46] P. Bouvry, H. González-Vélez, and J. Kołodziej, eds., *Intelligent Decision Systems in Large-Scale Distributed Environments*. Springer, Berlin Heidelberg, 2011.

[47] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[48] N. R. Jennings and M. J. Wooldridge, "Applications of intelligent agents," in *Agent Technology: Foundations, Applications, and Markets* (N. R. Jennings and M. J. Wooldridge, eds.), pp. 3–28, Springer-Verlag, Heidelberg, Germany, 1998.

[49] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.

[50] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[51] R. Dreżewski, "Co-evolutionary multi-agent system with speciation and resource sharing mechanisms," *Computing and Informatics*, vol. 25, no. 4, pp. 305–331, 2006.

[52] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Bradford Books, Cambridge, MA, 2004.

[53] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," in *New Ideas in Optimization* (D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price, eds.), pp. 11–32, McGraw-Hill Ltd., Maidenhead, 1999.

[54] M. Dorigo, G. Di Caro, and L. Gambardella, "Ant algorithms for discrete optimization," tech. rep., IRIDIA/98-10, Université Libre de Bruxelles, Belgium, 1999.

[55] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[56] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the ant system. a computational study.," in *Working papers SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, April 1997*, pp. 1–14, 1997.

[57] T. Stützle and H. H. Hoos, "Max–min ant system," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914, 2000.

[58] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[59] C. Darwin, *On the Origin of Species by Means of Natural Selection*. London: Murray, 1859.

[60] J. Holland, "Outline for a logical theory of adaptive systems," *Journal of the ACM*, vol. 3, pp. 297–314, 1962.

[61] H.-P. Schwefel, "Kybernetische evolution als strategie der experimentellen forschung inder strömungstechnik," tech. rep., Technische Universität, Berlin, 1965.

[62] H. Schwefel, *Evolution and optimum seeking*. Wiley, Chichester, 1995.

[63] L. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, New York, 1967.

[64] L. Fogel, "Autonomous automata," *Industrial Research*, vol. 4, pp. 14–19, 1962.

[65] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, 1997.

[66] J. Arabas, *Wykłady z algorytmów ewolucyjnych*. WNT Warszawa, 2001.

[67] Z. Michalewicz, *Genetic Algorithms Plus Data Structures Equals Evolution Programs*. Springer-Verlag, Secaucus, NJ, 1994.

[68] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford UK, 1996.

[69] K. P. Sycara, "Multiagent systems," *AI Magazine*, vol. 19, pp. 79–92, 1998.

[70] K. Cetnarowicz, M. Kisiel-Dorohinicki, and E. Nawarecki, "The application of evolution process in multi-agent world (MAW) to the prediction system," in *Proceeding of the 2nd International Conference on Multi-Agent Systems (ICMAS'96), December 9th-13th, Kyoto* (M. Tokoro, ed.), AAAI Press, 1996.

[71] M. Kisiel-Dorohinicki, "Agent-oriented model of simulated evolution," in *Proceedings of the 29th Conference on Current Trends in Theory and Practice of Informatics: Theory and Practice of Informatics, November 22–29, Milovy*, pp. 253–261, Springer-Verlag, Berlin, Heidelberg, 2002.

[72] E. Cantú-Paz, "A summary of research on parallel genetic algorithms," *IlliGAL Report No. 95007. University of Illinois, Chicago, IL*, 1995.

[73] A. Byrski and R. Schaefer, "Formal model for agent-based asynchronous evolutionary computation," in *2009 IEEE Congress on Evolutionary Computation, 18-21 May, Trondheim*, pp. 78–85, IEEE Xplore, 2009.

[74] A. Byrski, R. Schaefer, M. Smołka, and C. Cotta, "Asymptotic guarantee of success for multi-agent memetic systems," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 61, no. 1, pp. 257–278, 2013.

[75] R. Schaefer, A. Byrski, J. Kołodziej, and M. Smołka, "An agent-based model of hierarchic genetic search," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3763–3776, 2012.

[76] A. Byrski, "Tuning of agent-based computing," *Computer Science (AGH)*, vol. 14, no. 3, pp. 491–512, 2013.

[77] K. Wróbel, P. Torba, M. Paszyński, and A. Byrski, "Evolutionary multi-agent computing in inverse problems," *Computer Science*, vol. 14, no. 3, pp. 367–384, 2013.

[78] A. Byrski, M. Kisiel-Dorohinicki, and E. Nawarecki, "Agent-based evolution of neural network architecture," in *Proc. of the IASTED Int. Symp. on Applied Informatics, Innsbruck* (M. Hamza, ed.), pp. 242–247, IASTED/ACTA Press, 2002.

[79] L. Siwik and R. Dreżewski, "Evolutionary Multi-modal Optimization with the Use of Multi-objective Techniques," in *Artificial Intelligence and Soft Computing* (L. Rutkowski and et al., eds.), vol. 8467 of *Lecture Notes in Computer Science*, pp. 428–439, Springer, Berlin, Heidelberg, 2014.

[80] R. Dreżewski, J. Sepielak, and L. Siwik, "Classical and Agent-Based Evolutionary Algorithms for Investment Strategies Generation," in *Natural Computing in Computational Finance* (A. Brabazon and M. O'Neill, eds.), vol. 185 of *Studies in Computational Intelligence*, pp. 181–205, Springer-Verlag, Berlin, Heidelberg, 2009.

[81] A. Byrski and R. Schaefer, "Stochastic Model of Evolutionary and Immunological Multi-Agent Systems: Mutually Exclusive Actions," *Fundamenta Informaticae*, vol. 95, no. 2–3, pp. 263–285, 2009.

[82] R. Schaefer, A. Byrski, and M. Smołka, "Stochastic Model of Evolutionary and Immunological Multi-Agent Systems: Parallel Execution of Local Actions," *Fundamenta Informaticae*, vol. 95, pp. 325–348, Apr. 2009.

[83] Ł. Faber, K. Pietak, A. Byrski, and M. Kisiel-Dorohinicki, "Agent-Based Simulation in AgE Framework," in *Advances in Intelligent Modelling and Simulation: Simulation Tools and Applications* (A. Byrski, Z. Oplatková, M. Carvalho, and M. Kisiel-Dorohinicki, eds.), pp. 55–83, Springer, Berlin, Heidelberg, 2012.

[84] A. Byrski and M. Kisiel-Dorohinicki, "Agent-based evolutionary and immunological optimization," in *Computational Science – ICCS 2007: 7th International Conference, Beijing, China, May 27 - 30, 2007, Proceedings, Part II* (Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, eds.), pp. 928–935, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[85] A. Byrski, W. Korczynski, and M. Kisiel-Dorohinicki, "Memetic multi-agent computing in difficult continuous optimisation," in *Advanced Methods and Technologies for Agent and Multi-Agent Systems, Proceedings of the 7th KES Conference on Agent and Multi-Agent Systems - Technologies and Applications (KES-AMSTA 2013), May 27-29, 2013, Hue City, Vietnam*, pp. 181–190, IOS Press, Amsterdam, 2013.

[86] S. Pisarski, A. Rugała, A. Byrski, and M. Kisiel-Dorohinicki, "Evolutionary Multi-Agent System in Hard Benchmark Continuous Optimisation," in *Applications of Evolutionary Computation: 16th European Conference, EvoApplications 2013, Vienna, Austria, April 3–5, 2013. Proceedings* (A. I. Esparcia-Alcázar, ed.), pp. 132–141, Springer, Berlin, Heidelberg, 2013.

[87] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 103–117, 2012.

[88] M. Pelikan, M. Hauschild, and F. Lobo, "Introduction to estimation of distribution algorithms," Tech. Rep. 2012003, Missouri Estimation of Distribution Algorithms Laboratory, 2012.

[89] O. Regnier-Coudert and J. McCall, "Competing mutating agents for bayesian network structure learning," in *Parallel Problem Solving from Nature - PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I* (C. Coello and et al., eds.), pp. 216–225, Springer, Berlin, Heidelberg, 2012.

[90] O. Regnier-Coudert, J. McCall, and M. Ayodele, "Geometric-based sampling for permutation optimization," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pp. 399–406, ACM, New York, NY, 2013.

[91] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[92] D. Whitley and J. Kauth, "GENITOR: A different genetic algorithm," in *Proceedings of the 1988 Rocky Mountain Conference on Artificial Intelligence*, pp. 118–130, Computer Science Department, Colorado State University, 1988.

[93] J. Bishop, "Stochastic searching networks," in *Proc. 1st IEE Conf. on Artificial Neural Networks, 16–18 October 1989, London, UK*, pp. 329–331, IET, 1989.

[94] R. Picard, *Affective computing*. MIT Press, Boston, MA, 2000.

[95] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, Cambridge, 2015.

[96] H. Feldman and M. E. Rand, "Egocentrism-altercentrism in the husband-wife relationship," *Journal of Marriage and Family*, vol. 27, no. 3, pp. 386–391, 1965.

[97] J. Nadel, "Some reasons to link imitation and imitation recognition to theory of mind," in *Simulation and Knowledge of Action* (J. Doric and J. Proust, eds.), pp. 119–135, John Benjamins, New York, 2002.

[98] H. Bukowski, A. Curtain, and D. Samson, "Can you resist the influence of others? altercentrism, egocentrism and interpersonal personality traits," in *Proc. Of the Annual Meeting of the Belgian Association for Psychological Sciences (BAPS), May 28th*, Universite catholique de Louvain, 2013.

[99] S. Choudhury, S.-J. Blakemore, and T. Charman, "Social cognitive development during adolescence," *Social Cognitive and Affective Neuroscience*, vol. 1, no. 3, pp. 165–174, 2006.

[100] M. Johnson and Y. Demiris, "Perceptual perspective taking and action recognition," *International Journal of Advanced Robotic Systems*, vol. 2, no. 4, pp. 301–308, 2005.

[101] H. Bukowski, *What Influences Perspective Taking A dynamic and multidimensional approach*. PhD thesis, Université catholique de Louvain, 2014.

[102] E. Fizke, D. Barthel, T. Peters, and H. Rakoczy, "Executive function plays a role in coordinating different perspectives, particularly when one's own perspective is involved," *Cognition*, vol. 130, no. 3, pp. 315–334.

[103] H. Bukowski and D. Samson, "Can emotions influence level-1 visual perspective taking?," *Cognitive Neuroscience*, vol. 7, no. 1–4, pp. 182–191, 2016.

[104] A. Bandura, "Self-efficacy: Toward a unifying theory of behavioral change," *Psychological Review*, vol. 84, no. 2, pp. 191–215, 1977.

[105] A. Bandura, D. Ross, and S. Ross, "Transmission of aggression through imitation of aggressive models," *Journal of Abnormal and Social Psychology*, vol. 63, pp. 575–582, 1961.

[106] A. Bandura, "Social cognitive theory of mass communication," in *Media Effects: Advances in Theory and Research* (J. Bryant and M. Oliver, eds.), pp. 94–124, Routledge, New York, NY, 2002.

[107] A. Bandura, "The social and policy impact of social cognitive theory," in *Social Psychology and Evaluation* (M. Mark, S. Donaldson, and B. Campbell, eds.), pp. 33–70, Guilford Press, New York, NY, 2011.

[108] S. Martino, R. Collins, D. Kanouse, M. Elliott, and S. Berry, "Social cognitive processes mediating the relationship between exposure to television's sexual content and adolescents' sexual behavior," *Journal of Personality and Social Psychology*, vol. 89, no. 6, pp. 914–924, 2005.

[109] R. Axelrod, *The Evolution of Cooperation*. Basic Books, 1984.

[110] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[111] U. Naftaly, I. N., and D. Horn, "Optimal ensemble averaging of neural networks," *Network: Computation in Neural Systems*, vol. 8, no. 3, pp. 283–296, 1997.

[112] M. Rao and P. Georgeff, "Bdi-agents: From theory to practice," in *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95), 12-14 June, San Francisco, CA*, ACM, 1995.

[113] K. Cetnarowicz, *A Perspective on Agent Systems: Paradigm, Formalism, Examples*. Springer, Berlin, Heidelberg, 2014.

[114] T. T. Brunyé, T. Ditman, G. E. Giles, C. R. Mahoney, K. Kessler, and H. A. Taylor, "Gender and autistic personality traits predict perspective-taking ability in typical adults," *Personality and Individual Differences*, vol. 52, no. 1, pp. 84–88, 2012.

[115] M.-L. Yang, C.-C. Yang, and W.-B. Chiou, "When guilt leads to other orientation and shame leads to egocentric self-focus: Effects of differential priming of negative affects on perspective taking," *Social Behavior and Personality: An International Journal*, vol. 38, no. 5, pp. 605–614, 2010.

[116] A. D. Galinsky, W. W. Maddux, D. Gilin, and J. B. White, "Why it pays to get inside the head of your opponent in negotiations," *Psychological Science*, vol. 19, no. 4, pp. 378–384.

[117] C. Keysers and V. Gazzola, "Dissociating the ability and propensity for empathy," *Trends in Cognitive Sciences*, vol. 18, no. 4, pp. 163–166, 2014.

[118] M. Sekara, Michal-Kowalski, A. Byrski, B. Indurkhya, M. Kisiel-Dorohinicki, D. Samson, and T. Lenaerts, "Multi-pheromone ant colony optimization for socio-cognitive simulation purposes," *Procedia Computer Science*, vol. 51, pp. 954 – 963, 2015.

[119] E.-G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of Heuristics*, vol. 8, pp. 541–564, 2002.

[120] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 25–34, ACM, New York, NY, 1987.

[121] A. Nowé, K. Verbeeck, and P. Vrancx, "Multi-type ant colony: The edge disjoint paths problem," in *Ant Colony Optimization and Swarm Intelligence* (M. e. a. Dorigo, ed.), pp. 202–213, Springer, Berlin, Heidelberg, 2004.

[122] P. Vrancx, A. Nowé, and K. Steenhaut, "Multi-type aco for light path protection," in *Learning and Adaption in Multi-Agent Systems* (K. Tuyls, P. Hoen, K. Verbeeck, and S. Sen, eds.), vol. 3898 of *Lecture Notes in Computer Science*, pp. 207–215, Springer, Berlin, Heidelberg, 2006.

[123] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proc. of Congress on Evolutionary Computation, 6th-9th July 1999, Washington DC*, vol. 2, pp. 1470–1477, IEEE, 1999.

[124] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.

[125] A. R. Montero and A. S. López, "Ant colony optimization for solving the quadratic assignment problem," in *2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI), 25-31 October, Cuernavaca*, pp. 182–187, IEEE Xplore, 2015.

[126] A. Byrski, E. Świderska, J. Lasisz, M. Kisiel-Dorohinicki, T. Lenaerts, D. Samson, and B. Indurkhya, "Emergence of population structure in socio-cognitively inspired ant colony optimization," *Computer Science*, vol. 19, no. 1, pp. 81–98, 2018.

[127] M. M. Kabir, M. Shahjahan, and K. Murase, "A new hybrid ant colony optimization algorithm for feature selection," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3747–3763, 2012.

[128] L. Wei and Z. Yuren, "An effective hybrid ant colony algorithm for solving the traveling salesman problem," in *2010 International Conference on Intelligent Computation Technology and Automation*, vol. 1, pp. 497–500, May 2010.

[129] L. N. Xing, P. Rohlfshagen, Y. W. Chen, and X. Yao, "A hybrid ant colony optimization algorithm for the extended capacitated arc routing problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, pp. 1110–1123, Aug 2011.

[130] P. B. Myszkowski, M. E. Skowroński, Ł. P. Olech, and K. Oślizło, "Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem," *Soft Computing*, vol. 19, pp. 3599–3619, Dec 2015.

[131] L. Huang, C. Zhou, and K. Wang, "Hybrid ant colony algorithm for traveling salesman problem," *Progress in Natural Science*, vol. 13, no. 4, pp. 295–299, 2003.

[132] M. D. Toksari, "A hybrid algorithm of ant colony optimization (aco) and iterated local search (ils) for estimating electricity domestic consumption: Case of turkey," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 776–782, 2016.

[133] G. Shang, J. Xin-zi, T. Kezong, and Y. Jingyu, "Hybrid algorithm combining ant colony optimization algorithm with particle swarm optimization," in *Proc. of Chinese Control Conference, 7–11 August, Harbin*, pp. 1428–1432, IEEE Xplore, 2006.

[134] G. Hertono, Ubadah, and B. Handari, "The modification of hybrid method of ant colony optimization, particle swarm optimization and 3-opt algorithm in traveling salesman problem," *Journal of Physics: Conference Series*, vol. 974, no. 1, pp. 12032–12039, 2018.

[135] X. Zhang and L. Tang, "A new hybrid ant colony optimization algorithm for the vehicle routing problem," *Pattern Recognition Letters*, vol. 30, no. 9, pp. 848–855, 2009.

[136] C. Wang and X. Guo, "A hybrid algorithm based on genetic algorithm and ant colony optimization for traveling salesman problems," in *Proc. of 2nd International Conference on Information Science and Engineering, 24–26 April, Shanghai*, pp. 4257–4260, IEEE, 2010.

[137] M. Rusin and E. Zaitseva, "Hierarchical heterogeneous ant colony optimization," in *Proc. of Federated Conference on Computer Science and Information Systems, 9-12 September, Wroclaw*, pp. 197–203, IEEE Xplore, 2012.

[138] J.-W. Lee and J.-J. Lee, "Novel ant colony optimization algorithm with path crossover and heterogeneous ants for path planning, 14-17 march, vina del mar," in *Proc. of 2010 IEEE Conference on Industrial Technology (ICIT)*, pp. 559–564, IEEE, 2010.

[139] A. Hara, S. Matsushima, T. Ichimura, and T. Takahama, "Ant colony optimization using exploratory ants for constructing partial solutions," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–7, July 2010.

[140] C. Chira, D. Dumitrescu, and C. Pintea, "Heterogeneous sensitive ant model for combinatorial optimization," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, Atlanta, GA*, GECCO '08, pp. 163–164, ACM, New York, NY, 2008.

[141] K. Schiff, "Algorytm wielu kolonii mrówek dla optymalnego dopasowania w ważonych grafach dwudzielnych," *Elektrotechnika i Elektronika*, vol. 27, no. 2, pp. 115–119, 2008.

[142] E. Swiderska, J. Lasisz, A. Byrski, T. Lenaerts, D. Samson, B. Indurkhya, A. Nowé, and M. Kisiel-Dorohinicki, "Measuring diversity of socio-cognitively inspired ACO search," in *Applications of Evolutionary Computation – 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I* (G. Squillero and P. Burelli, eds.), pp. 393–408, Springer, Berlin, Heidelberg, 2016.

[143] G. Gutin, "Traveling salesman problem," in *Encyclopedia of Optimization* (C. A. Floudas and P. M. Pardalos, eds.), pp. 3935–3944, Springer, Boston, 2009.

[144] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities*. Scribner, New York, NY, 2001.

[145] D. Pais, *Emergent Collective Behavior in Multi-Agent Systems: An Evolutionary Perspective*. PhD thesis, Princeton University, 2012.

[146] M. Gardner, "Mathematical games – the fantastic combinations of john conway's new solitaire game „life"," *Scientific American*, vol. 223, pp. 120–123, 1970.

[147] I. Bugajski, A. Byrski, M. Kisiel-Dorohinicki, T. Lenaerts, D. Samson, and B. Indurkhya, "Adaptation of population structure in socio-cognitive particle swarm optimization," *Procedia Computer Science*, vol. 101, pp. 177–186, 2016. 5th International Young Scientist Conference on Computational Science, YSC 2016, 26-28 October 2016, Krakow, Poland.

[148] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. of IEEE World Congress on Computational Intelligence, 4–9 May, Anchorage*, pp. 69–73, IEEE, 1998.

[149] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, 6–9 July, Washington, DC*, vol. 3, pp. 1958–1962, IEEE, 1999.

[150] M. Iwamatsu, "Multi-species particle swarm optimizer for multimodal function optimization," *IEICE Transactions on Information and Systems*, vol. 89, no. 3, pp. 1181–1187, 2006.

[151] V. Miranda and N. Fonseca, "Epso-evolutionary particle swarm optimization, a new algorithm with applications in power systems," in *Proc. of Transmission and Distribution Conference and Exhibition 2002: Asia Pacific, 6-10 October, Yokohama*, vol. 2, pp. 745–750, IEEE/PES, 2002.

[152] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[153] Ş. Gülcü and H. Kodaz, "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 33–45, 2015.

[154] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[155] M. Sugimoto, T. Haraguchi, H. Matsushita, and Y. Nishio, "Particle swarm optimization containing plural swarms," in *In Proc. of 2009 International Workshop on Nonlinear Circuits and Signal Processing NCSP'09, 1–3 March, Waikiki, Hawaii*, pp. 584–587, 2009.

[156] G. Yen and W. F. Leong, "Dynamic multiple swarms in multiobjective particle swarm optimization," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 39, pp. 890–911, July 2009.

[157] J. Digalakis and K. Margaritis, "An experimental study of benchmarking functions for evolutionary algorithms," *International Journal of Computer Mathemathics*, vol. 79, pp. 403–416, April 2002.

[158] T. Bäck, D. Fogel, and Z. Michalewicz, eds., *Handbook of Evolutionary Computation*. IOP Publishing and Oxford University Press, Bristol, 1997.

[159] S. Nasuto and J. Bishop, "Steady state resource allocation analysis of the stochastic diffusion search," *Biologically Inspired Cognitive Architectures*, vol. 12, pp. 65–76, 2015.

[160] D. Myatt, J. Bishop, and S. Nasuto, "Minimum stable convergence criteria for stochastic diffusion search," *Electronics Letters*, vol. 40, pp. 112–113, Jan 2004.

[161] M. M. al Rifaie and J. M. Bishop, "Stochastic diffusion search review," *Paladyn, Journal of Behavioral Robotics*, vol. 4, no. 3, pp. 155–173, 2013.

[162] D. Myatt, S. Nasuto, and J. Bishop, "Alternative recruitment strategies for stochastic diffusion search," in *Proc. of AISB06: Symposium on Exploration vs. Exploitation in Naturally Inspired Search, Bristol*, pp. 181–187, 2006.

[163] K. de Meyer, *Foundations of stochastic diffusion search*. PhD thesis, University of Reading, Reading, UK, 2003.

[164] W. Turek, J. Stypka, D. Krzywicki, P. Anielski, K. Pietak, A. Byrski, and M. Kisiel-Dorohinicki, "Highly scalable erlang framework for agent-based metaheuristic computing," *Journal of Computational Science*, vol. 17, pp. 234–248, 2016.

[165] D. Krzywicki, W. Turek, A. Byrski, and M. Kisiel-Dorohinicki, "Massively concurrent agent-based evolutionary computing," *Journal of Computational Science*, vol. 11, pp. 153–162, 2015.

[166] D. Myatt and J. Bishop, "Data driven stochastic diffusion networks for robust high-dimensionality manifold estimation – more fun than you can shake a hyperplane at," in *Proc. of SCARP, Reading, UK*, University of Reading, 2003.

[167] J. Bishop, "Coupled stochastic diffusion processes," in *Proc. School Conference for Annual Research Projects (SCARP), 24th May 2003, Reading, UK*, pp. 185–187, University of Reading, 2003.

[168] L. M. Gambardella and M. Dorigo, "Ant-q: A reinforcement learning approach to the traveling salesman problem," in *Machine Learning Proceedings 1995* (A. Prieditis and S. Russell, eds.), pp. 252–260, Morgan Kaufmann, San Francisco, CA, 1995.

[169] D. J. Cavicchio, *Adaptive search using simulated evolution*. PhD thesis, University of Michigan, Ann Arbor, Michigan, USA, 1970.

[170] "FIPA: The foundation for intelligent physical agents." `http://fipa.org/` accessed 9.10.2018.

[171] P. Uhruski, M. Grochowski, and R. Schaefer, "Multi-Agent Computing System in a Heterogeneous Network," in *Proceedings of the International Conference on Parallel Computing in Electrical Engineering, Warsaw*, pp. 233–238, IEEE Computer Society Press, 2002.

[172] P. Uhruski, M. Grochowski, and R. Schaefer, "Octopus – computation agents environment," *Inteligencia Artificial, Revista Iberoamericana de IA*, vol. 9, no. 28, pp. 55–62, 2005.

[173] J. March, "Exploration and exploitation in organizational learning," *Organization Science*, vol. 2, pp. 71–87, 1991.

[174] R. Dreżewski, "The agent-based model and simulation of sexual selection and pair formation mechanisms," *Entropy*, vol. 20, no. 5, pp. 342–372, 2018.

[175] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, pp. 228–234, June 1990.

[176] S. W. Mahfoud, "A comparison of parallel and sequential niching methods," in *In Proceedings of the Sixth International Conference on Genetic Algorithms, 15-19 July, San Francisco, CA*, pp. 136–143, Morgan Kaufmann, 1995.

[177] J. Morrison and F. Oppacher, "A general model of co-evolution for genetic algorithms," in *Artificial Neural Nets and Genetic Algorithms*, pp. 262–268, Springer, Vienna, 1999.

[178] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proceedings of the 5th International Conference on Genetic Algorithms*, (San Francisco, CA, USA), pp. 264–270, Morgan Kaufmann Publishers Inc., 1993.

[179] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature — PPSN III* (Y. Davidor, H.-P. Schwefel, and R. Männer, eds.), pp. 249–257, Springer, Berlin, Heidelberg, 1994.

[180] M. Potter and K. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[181] J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355–375, 1995.

[182] S. W. Mahfoud, "Crowding and preselection revisited," in *Parallel Problem Solving from Nature — PPSN-II* (R. Männer and B. Manderick, eds.), (Amsterdam), pp. 27–36, Elsevier, Amsterdam, 1992.

[183] J. Sánchez-Velazco and J. A. Bullinaria, "Gendered selection strategies in genetic algorithms for optimization," in *Proceedings of the UK Workshop on Computational Intelligence (UKCI 2003)* (J. M. Rossiter and T. P. Martin, eds.), pp. 217–223, University of Bristol, Bristol, 2003.

[184] R. Dreżewski, "A model of co-evolution in multi-agent system," in *Multi-Agent Systems and Applications III* (V. Mařík, J. Müller, and M. Pěchouček, eds.), pp. 314–323, Springer-Verlag, Berlin, Heidelberg, 2003.

[185] R. Dreżewski, "Agent-based simulation model of sexual selection mechanism," in *Agent and Multi-Agent Systems: Technologies and Applications. 9th KES International Conference, KES-AMSTA 2015 Sorrento, Italy, June 2015, Proceedings* (G. Jezic, R. J. Howlett, and L. C. Jain, eds.), vol. 38 of *Smart Innovation, Systems and Technologies*, pp. 155–166, Springer International Publishing, Berlin, Heidelberg, 2015.

[186] R. Dreżewski, "Coevolutionary multiagent systems in multimodal optimization," in *Proceedings of the 2nd International Conference on Philosophy and Computer Science Processes of Evolution in Real and Virtual Systems (PERVS 2001)* (R. Dreżewski, M. Kisiel-Dorohinicki, J. Werszowiec-Płazowski, and M. Suwara, eds.), pp. 87–94, Department of Computer Science, AGH University of Science and Technology, Kraków, 2002.

[187] R. Drezewski and L. Siwik, "Co-evolutionary multi-agent system for portfolio optimization," in *Natural Computing in Computational Finance* (A. Brabazon and M. O'Neill, eds.), pp. 271–299, Springer-Verlag, Berlin, Heidelberg, 2008.

[188] R. Dreżewski and J. Sepielak, "Evolutionary system for generating investment strategies," in *Applications of Evolutionary Computing, EvoWorkshops 2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and EvoTransLog, Naples, Italy, March 26-28, 2008. Proceedings* (Giacobini, M. et al., ed.), pp. 83–92, Springer, Berlin, Heidelberg, 2008.

[189] R. Dreżewski, J. Sepielak, and L. Siwik, "Generating robust investment strategies with agent-based co-evolutionary system," in *Proc. of 8th International Conference on Computational Science, Kraków, Poland, June 23–25, 2008, Proceedings, Part III* (M. Bubak, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, eds.), pp. 664–673, Springer, Berlin, Heidelberg, 2008.

[190] W. Johnson, L. DeLanney, and T. Cole, *Essentials of Biology*. Holt, Rinehart and Winston, New York, NY, 1969.

[191] S. Wierzchoń, "Function optimization by the immune metaphor," *Task Quaterly*, vol. 6, no. 3, pp. 1–16, 2002.

[192] S. Wierzchoń, *Sztuczne systemy immunologiczne*. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2001.

[193] A. Gaspar and P. Collard, "From GAs to artificial immune systems: Improving adaptation in time dependent optimisation," in *Proc. of the 1999 Congress on Evolutionary Computation – CEC'99, 6-9 July, Washington DC*, pp. 1859–1866, IEEE Publishing, 1999.

[194] K. Trojanowski and S. Wierzchoń, "Studying properties of multipopulation heuristic approach to non-stationary optimisation tasks," in *Proc. of the International Conference on Intelligent Information System, Intelligent Information Processing and Web Mining, June 2–5, Zakopane*, Springer, Berlin, Heildelberg, 2003.

[195] L. N. de Castro and F. J. V. Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 239–251, June 2002.

[196] D. Dasgupta and L. F. Nino, *Immunological Computation Theory and Applications*. CRC Press, Boca Raton, FL, 2008.

[197] A. Byrski, "Immunological selection mechanism in evolutionary multi agent systems," in *Materiały Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Kazimierz Dolny 24–26 Maja 2004*, pp. 19–26, Politechnika Warszawska, 2004.

[198] A. Byrski and M. Carvalho, "Agent-based immunological intrusion detection system for mobile ad-hoc networks," in *Computational Science - ICCS 2008, 8th International Conference, Kraków, Poland, June 23-25, 2008, Proceedings, Part III*, pp. 584–593, Springer, Berlin, Heidelberg, 2008.

[199] A. Byrski and M. Kisiel-Dorohinicki, "Immune-based optimization of predicting neural networks," in *Computational Science - ICCS 2005, 5th International Conference, Atlanta, GA, USA, May 22-25, 2005, Proceedings, Part III*, pp. 703–710, Springer, Berlin, Heidelberg, 2005.

[200] A. Byrski, R. Schaefer, and M. Smolka, "Markov chain based analysis of agent-based immunological system," *Transactions on Computational Collective Intelligence*, vol. 10, pp. 1–15, 2013.

[201] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Hoboken, NJ, 2001.

[202] G. Rudolph, "Evolutionary search under partially ordered finite sets," in *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)* (M. F. Sebaaly, ed.), pp. 818–822, ICSC Academic Press, Dubai, 2001.

[203] A. Osyczka and S. Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural and Multidisciplinary Optimization*, vol. 10, pp. 94–99, October 1995.

[204] E. Zitzler and L. Thiele, "An evolutionary algorithm for multiobjective optimization: The strength pareto approach," Tech. Rep. 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 1998.

[205] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Parallel Problem Solving from Nature. 1st Workshop, PPSN I* (H. Schwefel and R. Manner, eds.), vol. 496, pp. 193–197, Springer-Verlag, Berlin, 1991.

[206] T. Murata and H. Ishibuchi, "Moga: multi-objective genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation, 29 November–1 December, Perth*, vol. 1, pp. 289–294, IEEE, 1995.

[207] L. Siwik and M. Kisiel-Dorohinicki, "Improving the quality of the pareto frontier approximation obtained by semi-elitist evolutionary multi-agent system using distributed and decentralized frontier crowding mechanism," in *Adaptive and Natural Computing Algorithms, 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11-14, 2007, Proceedings, Part I* (B. Beliczynski, A. Dzielinski, M. Iwanowski, and B. Ribeiro, eds.), pp. 138–147, Springer, Berlin, Heidelberg, 2007.

[208] L. Siwik and S. Natanek, "Elitist evolutionary multi-agent system in solving noisy multi-objective optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China*, pp. 3319–3326, IEEE, 2008.

[209] L. Siwik and S. Natanek, "Solving constrained multi-criteria optimization tasks using elitist evolutionary multi-agent system," in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China*, pp. 3358–3365, IEEE, 2008.

[210] L. Siwik and M. Kisiel-Dorohinicki, "Semi-elitist evolutionary multi-agent system for multiobjective optimization," in *Computational Science - ICCS 2006, 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part III* (V. N. Alexandrov and et al., eds.), pp. 831–838, Springer, Berlin, Heidelberg, 2006.

[211] A. Byrski, M. Kisiel-Dorohinicki, and N. Tusiński, "Extending estimation of distribution algorithms with agent-based computing inspirations," in *Transactions on Computational Collective Intelligence XXVII* (J. Mercik, ed.), pp. 191–207, Springer International Publishing, Cham, 2017.

[212] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 4–31, Feb 2011.

[213] J.-P. Chiou, C.-F. Chang, and C.-T. Su, "Ant direction hybrid differential evolution for solving large capacitor placement problems," *IEEE Transactions on Power Systems*, vol. 19, pp. 1794–1800, Nov 2004.

[214] B. Liu, X. Zhang, and H. Ma, "Hybrid differential evolution for noisy optimization," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 1–6 June, Hong Kong*, pp. 587–592, IEEE, 2008.

[215] L. Płaczkiewicz, M. Sendera, A. Szlachta, M. Paciorek, A. Byrski, M. Kisiel-Dorohinicki, and M. Godzik, "Hybrid swarm and agent-based evolutionary optimization," in *Proc. of International Conference on Computational Science ICCS 2018, 11-13 June, Wuxi*, pp. 89–102, Springer, Cham, 2018.

[216] R. Dawkins, *The Selfish Gene*. Oxford Paperbacks, Oxford University Press, 1989.

[217] G. Graham, *Genes: A Philosophical Inquiry*. Taylor & Francis, Abingdon, 2002.

[218] F. Heylighen, "Evolution, Selfishness and Cooperation; Selfish Memes and the Evolution of Cooperation," *Journal of Ideas*, vol. 2, no. 4, pp. 70–84, 1992.

[219] N. Radcliffe and P. Surry, "Formal Memetic Algorithms," in *Evolutionary Computing: AISB Workshop* (T. Fogarty, ed.), vol. 865 of *Lecture Notes in Computer Science*, pp. 1–16, Springer-Verlag, Berlin, 1994.

[220] W. Hart, N. Krasnogor, and J. Smith, "Memetic evolutionary algorithms," in *Recent advances in memetic algorithms*, vol. 166 of *Studies in Fuzziness and Soft Computing*, pp. 3–27, Springer-Verlag, Berlin, Heidelberg, 2005.

[221] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[222] R. G. Reynolds, "An introduction to cultural algorithms," in *Proceedings of the Third Annual Conference on Evolutionary Programming, 24–26 February, San Diego, CA*, pp. 131–139, World Scientific, River Folge, 1994.

[223] J. Aguilar and A. Colmenares, "Resolution of pattern recognition problems using a hybrid genetic/random neural network learning algorithm," *Pattern Analysis and Applications*, vol. 1, no. 1, pp. 52–61, 1998.

[224] M. Mignotte, C. Collet, P. Pérez, and P. Bouthemy, "Hybrid genetic optimization and statistical model based approach for the classification of shadow shapes in sonar imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, pp. 129–141, 2000.

[225] S. P. Harris and E. C. Ifeachor, "Automatic design of frequency sampling filters by hybrid genetic algorithm techniques," *IEEE Transactions on Signal Processing*, vol. 46, no. 12, pp. 3304–3314, 1998.

[226] C. Reis, J. A. T. Machado, and J. B. Cunha, "A memetic algorithm for logic circuit design," in *CONTROL'05 Proceedings of the 2005 WSEAS International conference on Dynamical systems and control, Italy November 02–04, Venice*, pp. 598–603, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, 2005.

[227] J. M. Baldwin, "A New Factor In Evolution," *American Naturalist*, vol. 30, pp. 441–451, 536–553, 1896.

[228] G. Hinton and S. Nowlan, "How learning can guide evolution," *Complex Systems*, vol. 1, pp. 495–502, 1987.

[229] N. Eldridge and S. Gould, "Punctuated equilibria: An alternative to phyletic gradualism," in *Models in Paleobiology* (T. Schopf, ed.), Freeman, Cooper and Co., San Francisco, CA, 1972.

[230] G. Simpson and W. Beck, *Life: An Introduction to Biology*. Brace & World, Harcourt, 1965.

[231] C. R. Houck, J. A. Joines, and M. G. Kay, "Utilizing lamarckian evolution and the baldwin effect in hybrid genetic algorithms," tech. rep., NCSU-IE, 1996.

[232] K. Ku, M. Mak, and W. Siu, "A study of the Lamarckian evolution of recurrent neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 31–42, Apr 2000.

[233] B. Ross, "A Lamarckian Evolution Strategy for Genetic Algorithms," in *The Practical Handbook of Genetic Algorithms* (L. Chambers, ed.), pp. 1–16, CRC Press, Boca Raton, FL, 1999.

[234] W. Korczynski, A. Byrski, and M. Kisiel-Dorohinicki, "Buffered local search for efficient memetic agent-based continuous optimization," *Journal of Computational Science*, vol. 20, pp. 112–117, 2017.

[235] A. Byrski and M. Kisiel-Dorohinicki, "Memetic computing in selected agent-based evolutionary systems," in *28th European Conference on Modelling and Simulation, ECMS 2014, Brescia, Italy, May 27-30, 2014*, pp. 495–500, European Council on Modelling and Simulation, 2014.

[236] M. Kolybacz, M. Kowol, L. Lesniak, A. Byrski, and M. Kisiel-Dorohinicki, "Efficiency of memetic and evolutionary computing in combinatorial optimisation," in *Proceedings of the 27th European Conference on Modelling and Simulation, ECMS 2013, Ålesund, Norway, May 27-30, 2013* (W. Rekdalsbakken, R. T. Bye, and H. Zhang, eds.), pp. 525–531, European Council for Modeling and Simulation, Wilhelmshaven, 2013.

[237] W. Korczynski, A. Byrski, and M. Kisiel-Dorohinicki, "Efficient memetic continuous optimization in agent-based computing," in *International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA*, pp. 845–854, Elsevier, Amsterdam, 2016.

[238] W. Korczynski, M. Kisiel-Dorohinicki, and A. Byrski, "Lamarckian and lifelong memetic search in agent-based computing," in *Applications of Evolutionary Computation* (G. Squillero and K. Sim, eds.), pp. 253–265, Springer International Publishing, Cham, 2017.

[239] W. Korczynski, A. Byrski, R. Debski, and M. Kisiel-Dorohinicki, "Classic and agent-based evolutionary heuristics for shape optimization of rotating discs," *Computing and Informatics*, vol. 36, no. 2, pp. 331–352, 2017.

[240] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 2016.

[241] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, Mar. 1991.

[242] R. Schaefer and J. Kolodziej, "Genetic search reinforced by the population hierarchy," in *Proc. of Foundations of Genetic Algorithms VII*, pp. 383–401, Morgan Kaufmann, Burlington, MA, 2003.