ROZPRAWY
MONOGRAFIE

268

ALEKSANDER BYRSKI

Agent-based Metaheuristics in Search and Optimisation

AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY PRESS KRAKOW 2013

Published by AGH University of Science and Technology Press

Reviewers:
Grzegorz Dobrowolski, AGH University of Science and Technology, Poland,
Juan Carlos Burguillo Rial, University of Vigo, Spain.

Author of the monograph is an employee of
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

ISSN 0867-6631
ISBN 978-83-7464-587-4

Wydawnictwa AGH
al. A. Mickiewicza 30, 30-059 Kraków
tel. 12 617 32 28, tel./fax 12 636 40 38
e-mail: redakcja@wydawnictwoagh.pl
www.wydawnictwa.agh.edu.pl

Contents

ALEKSANDER BYRSKI
Agent-based Metaheuristics in Search and Optimisation

Abstract

In the domain of computing, an everlasting requirement for developing new metaheuristics for particular problems, coming right from the well-known *no free lunch theorem*, may be observed. The need for new search and optimisation methods, hybrid ones in particular, paves the way for the development of different metaheuristics, going beyond classical methods (such as population-based ones). Evolutionary multiagent systems (EMAS), which brings together interesting features of agency (such as autonomy) and inspirations coming from population-based techniques, is a good example of such promising methods. However, constructing complex metaheuristics without a detailed description of their structure and behaviour may become pointless, and novel methods, though yielding promising results in particular cases, may be underestimated, because they have not been fully understood and analysed. This dissertation focuses on the issues concerning the justification of using agent-based metaheuristics (in particular EMAS and its variants), preparing of dedicated formal model, conducting an analysis aimed at proving so-called asymptotic guarantee of success and performing experimental analysis of the considered methods. These issues may be treated as the most important and novel aspects of this dissertation. In the beginning of the monograph, a systematic state-of-the-art review is given, then the concepts of EMAS and its modifications are discussed, later the formal model of structure and dynamics of the system using Markov-chains is described. Finally, the outcomes of a broad series of experiments on selected benchmark and real-world problems are discussed. The results presented in this dissertation are useful for practitioners who would to use agent-based metaheuristics and to obtain a deeper insight into the details of their design, experimental and formal features.

ALEKSANDER BYRSKI
Agentowe metaheurystyki w poszukiwaniach i optymalizacji

Streszczenie

Rozwiązywanie trudnych problemów poszukiwawczych i optymalizacyjnych zawsze będzie wymagać tworzenia złożonych, często przybliżonych metod. Sankcjonuje to sformułowanie twierdzenia znanego jako *no free lunch theorem*, wskazującego na konieczność wynajdywania coraz to nowych, w szczególności hybrydowych metod, wychodzących poza ramy określone przez tradycyjne już uniwersalne algorytmy optymalizacji (takie jak np. metody populacyjne). Typowym przykładem tej klasy metod są ewolucyjne systemy wieloagentowe (*ang. evolutionary multi-agent systems*, EMAS), łączące cechy agentowości (takie jak autonomia) oraz inspiracje pochodzące z technik populacyjnych. Niniejsza monografia koncentruje się na opracowaniu agentowych metaheurystyk (w szczególności EMAS i jego wariantów), konstrukcji modelu formalnego, przeprowadzeniu analizy ukierunkowanej na dowiedzenie tzw. asymptotycznej gwarancji sukcesu oraz wykonaniu eksperymentalnej weryfikacji badanych metod. Wspomniane tematy są najbardziej nowatorskimi aspektami prezentowanej monografii, szczególnie przeprowadzenie pełnego dowodu ergodyczności EMAS znacznie wykracza poza do tej pory spotykane efekty analizy metaheurystyk (koncentrujące się na bardzo szczególnych przypadkach, czy to algorytmów, czy też rozwiązywanych problemów). Na początku pracy przedstawiono przegląd stanu wiedzy, następnie zaprezentowano koncepcję EMAS i jego wariantów, wreszcie przedstawiono model formalny stanowiący bazę do analizy dynamiki EMAS na podstawie badaĹ„, odpowiednio skonstruowanego łańcucha Markowa, a także jego cech takich jak ergodyczność. Monografia kończy się prezentacją wyników badań eksperymentalnych dotyczących rozwiązywania zarówno problemów benchmarkowych, jak i rzeczywistych. Prezentowane rezultaty mogą być przydatne do pogłębienia wiedzy w zakresie agentowych systemów obliczeniowych i ich własności formalnych, wreszcie stanowić mogą podbudowę dla adaptacji prezentowanych metaheurystyk agentowych do partykularnych zadań stawianych przez zainteresowanych badaczy.

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

Preface

Many interesting search and optimisation problems are too difficult to solve using analytical methods. Michalewicz and Fogel have provided a number of features identifying such difficult problems, e.g. when the admissible search space is so big that performing an exhaustive search is impossible; the problem is so complex that it requires a simplified model for even trying to produce any solution, or the goal function describing the quality of the solutions is noisy or varies in time.

Problems, in which candidate solutions may only be sampled randomly, but there is practically no means of deriving them on the basis of the existing knowledge of the search space, are called "black-box problems". They constitute an additional challenge to the search and optimisation systems.

Such problems may only be solved using general-purpose methods (i.e., *heuristics*), taking into consideration little if any information from a problem domain. Heuristics provide "good-enough" solutions without concern as to whether they may be proved to be correct or optimal. It may be said that these methods trade-off precision, quality, accuracy and execution time in favour of computational effort. Such methods which are usually referred to as the *methods of last resort* are necessary for dealing with difficult problems

A general definition of a heuristic method, without giving details of a particular problem, accurate definition of search space or operators is called a *metaheuristic*, and is usually defined as a general purpose, nature-inspired search method.

A particular metaheuristic method (however excellent it may be for solving a certain problem, or a certain class of problems) may never become the ultimate answer to solving all possible optimisation problem (cf. *no free lunch theorem*). Therefore, looking for novel metaheuristics will always be necessary. These methods may be developed when seeking for inspiration in different domains of life such as biology, sociology or culture, but they also may be put together (hybridised) using the methods already developed.

For the last forty years, a growing interest has been observed in the systems where a task to solve is decomposed into smaller parts (subtasks), which are dealt

with separately and later the are synthesized into an overall solution. Such an approach may be described as *distributed problem solving*, and is usually easily implemented in parallel environments such as multi-core machines, clusters or grids. It should be noted that multi-agent systems belonging to a popular class of methods in artificial intelligence are an effective implementation of distributed problem solving. Agents are perceived as autonomous beings, which are able to interact with their environment and other agents and bear the features of intelligence. In these systems, a task to solve is usually decomposed into subtasks, which are entrusted to agents. Each agent's goal is to solve its part, and different features of agency affect this process, e.g. autonomy allows for self-adaptation of the agent's strategy.

In 1996, Krzysztof Cetnarowicz proposed an evolutionary multi-agent system (EMAS) dedicated to solving computing problems, with interesting features like distributed selection and lack of global control. Since then the idea of EMAS has been applied to different problems (e.g. single, multimodal and multicriteria optimisation). This approach still retains high potential possibilities of extension and hybridisation (e.g. with cultural or memetic mechanisms).

It is noteworthy that since the inception, EMAS-related research has yielded different modification of this system (utilising elitist, coevolutionary or immunological inspirations). Based on these modifications, effective solutions to many difficult problems have been provided such as evolution of neural-network architecture, multimodal optimisation and financial optimisation to name but a few. EMAS has thus proved to be a versatile optimisation mechanism in practical situations.

Multi-agent systems provide a good basis for the development of hybrid search and optimisation systems, however it should be noted that in this way, more and more complex computing systems are created. Also, using common sense and remembering *Ockham's razor* rule, one should apply complex search techniques solely to difficult problems. Therefore metaheuristics, in particular agent-oriented ones, should be treated as the *methods of last resort*, and should not be applied to simple problems.

On the other hand, the need to build complex (hybrid) systems, calls for performing a more in-depth analysis of features of their work. A detailed description of its structure and behaviour is required for a full understanding of them, moreover, providing means for stochastic analysis may yield additional, important results such as confirmation whether the system works at all (meaning, whether or even when it is able to localise the result).

Several formal models aimed at proving different features of evolutionary metaheuristics have been constructed. One of the first and important models of metaheuristic methods was Michael Vose's model, which proves that for a fixed size population, simple genetic algorithm (SGA) can be modelled by a Markov chain, and after further assumption that the mutation rate is positive, this chain is ergodic. This result

formally justifies SGA as a well-defined global optimisation algorithm. Other approaches to model evolutionary algorithms to be mentioned are different models for single-population evolutionary algorithms, proposed by Davis, Mahfoud or Rudolph. In particular, Rudolph's model was used to prove the first hitting time for a $1+1$ evolution strategy optimising a convex function. Unfortunately, there is lack of general models, as all those mentioned above are oriented on analysis of particular methods.

To sum up, the following needs in the domain of computing metaheuristics may be identified:

1. **Apparent difficulty** with many search and optimisation problems calls for **developing novel metaheuristics**, and agent-based hybrids seem to provide the appropriate answer, due to their intrinsic features such as autonomy and flexibility.
2. **Developing a complex metaheuristic** requires a careful construction of a **detailed formal model**, both in order to ease understanding of the structure and behaviour of the system, and to justify its existence by providing formally correct observations about the particular abilities of the method.
3. Apart from the **formal verification of metaheuristics**, an **accurately planned testing** of the method, based on popular difficult benchmark and real-world problems is necessary. Moreover, research on the influence of changing certain parameters on the efficiency of the method is necessary due to the fact that metaheuristics usually offer a vast number of parameters to tune up. Such observations may be further used to verify the formal model and finally, to enhance the method proposed in the first place.

In an effort to deliver appropriate tools for satisfying the complex needs, conducting dedicated research focused on

> **working out novel, effective metaheuristics, utilising agent-based approach and hybridising different metaheuristics, along with providing appropriate means for their formal and experimental verification**

seems to be necessary.

In order to fulfil these goals, the following research tasks are reported in this monograph:

- State-of-the-art review of current popular metaheuristics, grounding a base for further considerations on the construction of hybrid methods.
- Description of classical and modified EMAS-related metaheuristics.
- Formal description of EMAS and related systems structure and behaviour.

- Construction of Markov-chain based model for EMAS and related systems used as a base for stochastic analysis of EMAS features.
- Outcomes of stochastic analysis of the systems discussed in the dissertation.
- Conducting a series of experiments aimed at testing efficiency of the proposed agent-based metaheuristics for difficult benchmark and real-world problems.
- Testing the influence of selected parameters of the proposed agent-based metaheuristics on their overall efficiency.

The structure of this monograph is as follows. **Chapter 1** presents a systematic state-of-the-art review. It begins with identification of difficult problems (so-called "black-box") search problems and justification for the use of complex metaheuristics to solve them. Later, selected single-solution metaheuristics are discussed as methods providing base for constructing hybrid (e.g. memetic) systems described later in this work. Then, evolutionary metaheuristics are briefly described putting emphasis on the features that are common to all population-based metaheuristics, such as measuring and enhancing the diversity, or the stopping criteria. In the next section, problems of hybridisation of metaheuristics are outlined, and additional hybrid metaheuristics are discussed (memetic, cultural and immunological). Again, this selection is imposed by the systems discussed later in this monograph, which are extensively inspired by these techniques. Later, the possibilities of enhancing computing systems with agency are presented, and finally, existing deficits in the theory and practice of metaheuristics are identified.

Chapter 2 begins with a description of agent-based architectures of computing systems. Then, a concept of Evolutionary Multi-agent Systems (EMAS) is discussed, followed by a formal definition of the system state and transition functions of particular agents. Later, detailed definitions of agents' actions are given. The presentation of the agent-based system management follows. A similar section structure is repeated for the description of immunological EMAS (iEMAS), i.e. description of the concept, the system state and the management structure. Finally, the need for providing accurate formal analysis of the presented systems is justified.

Chapter 3 starts with the construction of a Markov-chain modelling dynamics of EMAS. Then, after necessary assumptions, the ergodic theorem is defined and a full formal proof is given. Selected technical details of the proof (i.e., estimates of lower bounds of probabilities and upper bounds of steps required to perform the proof stages) were transferred to Appendix B, in order to retain clarity of the text. Later, the same structure of presentation is retained for iEMAS, however, in this case no full formal proof has been constructed, and only necessary conjecture is formulated and the proof outline is described. The chapter is concluded with a short description of actual goals reached in the formal analysis of agent-based metaheuristics.

Chapter 4 presents the results of a wide-ranging series of experiments conducted in order to evaluate the efficiency of agent-based metaheuristics, as compared to classical search methods. In the first section, EMAS is evaluated using selected high-dimensional benchmark functions. After presenting the benchmarks, comparison between EMAS and PEA (parallel evolutionary algorithm) is made, using classical (evolutionary) and memetic versions of these methods. Later, immunological version of EMAS is tested versus the classical EMAS. In the next section, tuning of selected EMAS parameters is considered. After observing an impact of changing certain EMAS parameters (energy-related and probabilistic), and iEMAS (lymphocyte parameters), the results are summed up, which provides a base for further use in order to adapt these metaheuristics to particular problems. In the last section, efficiency of EMAS is tested in two selected real-world problems (step and flash imprint lithography inverse problem and optimisation of Sudoku solving strategy).

The last part of the monograph is a summary, followed by Appendix A (containing a description of the configuration for which the experimental results were obtained) and Appendix B (giving technical details of the EMAS ergodicity proof). The final part of the monograph is bibliography.

⋆ ⋆ ⋆

Hereby I would like to thank all the people without whom this work may not have been written. First of all I wish to express my deep gratitude to my mentor, Professor Edward Nawarecki, and my main collaborator, Marek Kisiel-Dorohinicki. My thanks are also extended to Professor Robert Schaefer who works with me on stochastic models of computing systems, Professor Grzegorz Dobrowolski and Professor Krzysztof Cetnarowicz, who eagerly support me in my university work. It is to note that seminal works of Professor Krzysztof Cetnarowicz on Evolutionary Multiagent Systems became a starting point for my research.

I would also like to thank my colleagues from Intelligent Information Systems Group and Department of Computer Science for a friendly atmosphere which encouraged me to conduct my research. My thanks go to my prominent students: Wojciech Korczyński, Krzysztof Wróbel, Paweł Torba, Mateusz Krzysztoń, Zofia Flasińska, Mateusz Polnik and Mateusz Kumięga for their help in preparing experimental results presented in this monograph.

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

Aleksander Byrski
Kraków, July 2012–April 2013

1. Contemporary search metaheuristics

Tackling difficult search problems calls for applying unconventional methods. This necessity is imposed by having little or no knowledge of the intrinsic features of the problem, topology of 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) seem to be the best choice, both at algorithmic and implementation level, and as they process more than one solution at a time, they can evade local extrema easier than single-solution approaches. Moreover, it is easy to implement them efficiently using ubiquitous parallel systems, such as multi-core processors, graphical processing units, clusters and grids.

In this state-of-the-art review chapter, difficult (so-called "black-box") search problems are identified, and metaheuristic methods are presented as one of the reasonable ways to deal with them. Next, evolutionary metaheuristics are briefly described, providing the basis for further considerations. After discussing possible ways of using hybrid metaheuristics, the issue of agency in computing systems is discussed. Finally, existing deficits in the theory and practice of metaheuristics are identified, which prepares for the description of original results concerning Evolutionary Multiagent Systems that are presented in the next chapters.

1.1. Search problems and heuristic techniques

Popular real-world search problems usually consist in finding a set of parameters of a certain model, according to specific criteria. These criteria are usually expressed as certain functions of the mentioned parameters. The goal of search is to optimise the criteria function, therefore these problems belong to the class of optimisation, consisting in finding all global minimisers $\arg\min\{\Phi(x)\}, x \in \mathcal{D}$ of the objective function: $\Phi : \mathcal{D} \to [0, M]$, 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 proved to be correct or optimal [132]. It may be said, that these methods trade-off precision, quality, accuracy and execution time in favour of computational effort. Such methods are necessary for dealing with difficult problems, usually being referred to as the methods of last resort (see, e.g. [133]).

One of the simplest heuristics is a greedy search algorithm, randomly generating solutions and accepting only these which fulfil the predefined criteria. More sophisticated examples of heuristics are the Monte Carlo [102] search methods. Heuristics may also be hybridised (see Section 1.3.1), in order to improve effectiveness of other search methods (as A* algorithm or alpha-beta pruning in tree-search [157]).

Using common sense and remembering Ockham's razor rule, one should apply complex search techniques solely to difficult problems. Therefore, this study does not include a number of popular tasks, such as optimisation of convex functions or linear programming, with their reliable techniques [92].

1.1.1. Difficult search problems

Michalewicz and Fogel in [132] propose several reasons why the problem may be considered *difficult*, e.g. the number of possible solutions is too large 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 and therefore many solutions are sought.

Certain search problems, which 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 optimisation problems [140]). The setting of such problems is sometimes called "black-box scenario" [56].

According to a definition given in the previous section, let us assume that there exists a meta-algorithm covering all randomised search heuristics working on the finite search space $\mathcal{D}$. Functions to be optimised are all functions that may be described as $f : \mathcal{D} \to [0, M]$. Now the "black-box" scenario is defined as follows [56, 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 the 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 time), it can be called a "black-box problem". In other words, this notion encompasses all the problems, whose candidate solutions may be sampled randomly, but there is no means of deriving them on the basis of the existing knowledge of the 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

Randomised algorithms (those using random, or pseudo-random choices) are usually classified as Monte Carlo (because they provide an approximate solution) or Las Vegas (because they finally provide a correct solution, if enough time is given) algorithms [6].

Unfortunately, there is no guarantee that heuristics will find satisfactory solutions, therefore, their features observed for particular problems must be verified empirically. This is simply because theoretical analyses take a number of assumptions of the algorithm (that is inevitable when constructing a simplified model of reality). On the other hand, this simplification may hamper the applicability of the model in real-world scenarios, therefore an experimental verification is required to make sure that both heuristic and its model are valid.

Yet a heuristic may often give an approximate solution with controllable adequacy, which means that a process solving can be stopped by a decision maker once he is satisfied.

Although these algorithms process certain points from the problem domain, having at the same time complete information about the value of the criteria function of this particular solution, the global features of the search, such as e.g. the information about closeness to the optimum, remain hidden. The whole search process consists in more or less complex iterative sampling of the problem domain.

Complex approaches that may be used to solve such difficult problems (see, e.g. [2]) somehow relieve the user of a deep understanding of intrinsic relations 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 well-known "no free lunch theorem" must be kept in mind. Wolpert and Macready prove that all search and optimisation techniques are statistically identical when compared for all problems described in certain domain [193, 192]. So there is still much to be done to adapt parameters of these techniques to solve certain problems.

1.1.2. Metaheuristic and heuristic search methods

A general definition of a heuristic algorithm, without giving details such as particular problem, accurate definition of search space or operators is called a *metaheuristic*. In this way, a simple heuristic algorithm, such as greedy search may be defined

as, e.g. “iterative, local improving of a solution based on random sampling”, without going into details of the nature of random sampling or the explored space. Therefore, metaheuristics are usually defined as general purpose, nature-inspired search algorithms [74].

Blum and Roli [13] provide a summary of the metaheuristic properties:

- they are approximate and usually non-deterministic,
- their goal is to efficiently explore the search space seeking for (sub-)optimal solutions,
- they “guide” the search process,
- they may incorporate mechanisms dedicated to avoiding being trapped in local extrema,
- they are not problem-specific,
- they can utilise search experience (usually implemented as some kind of memory mechanism) to guide the search.

Another type of heuristic algorithms are so-called hyper-heuristics, which utilise more advanced mechanisms (e.g. from the domain of machine learning) to optimise the parameters of the search, or even select an appropriate lower-level search method [16].

A simple, but effective classification of metaheuristics (cf. [52, 15]), which gives a sufficient insight into the 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) [178].
- Population-based metaheuristics explicitly work with a population of solutions and put them together in order to generate new solutions. The examples are evolutionary algorithms [78], immunological algorithms [39], particle swarm optimisation [104], ant colony optimisation [51], memetic algorithms [137] and other similar techniques.

These techniques are usually nature-inspired and follow different phenomena observed in e.g. biology, sociology, culture or physics.

1.1.3. Selected single-solution metaheuristics

This monograph is focused on population-based metaheuristics, however, in this section selected and popular single-solution metaheuristics will be presented in short, forming a base for further considerations, in particular with regard to the creation of hybrid systems.

Stochastic hill climbing One of the simplest and most popular single-solution metaheuristics is the stochastic hill-climbing algorithm (see Pseudocode 1.1.1). Starting with a random solution, the algorithm selects subsequent ones (also randomly) and accepts them only if they improve the value of a certain predefined goal function. The algorithm was primarily designed to solve combinatorial optimisation problems (for classical implementations refer to, e.g. [70, 135, 100]), however, it can be easily used in continuous optimisation by appropriately defining the search step. This algorithm may be easily incorporated into hybrid systems (e.g. fulfilling the role of local-search algorithm in memetic computing) [189].

Pseudocode 1.1.1: PSEUDOCODE OF HILL CLIMBING ALGORITHM

$current \leftarrow RandomSolution()$
for $i \in [1, max]$
do $\begin{cases} candidate \leftarrow randomNeighbour(current) \\ \textbf{if } goalFunction(candidate) \geq goalFunction(current) \\ \quad \textbf{then } current \leftarrow candidate \end{cases}$

Simulated annealing A more sophisticated approach to random local search is simulated annealing. Inspiration for this algorithm comes from an annealing process in metallurgy. In this process, a material is heated and slowly cooled under specific conditions in order to increase the size of the crystals in the material and to reduce possible defects that may arise in the cast. Using this metaphor, each solution in the search space is treated as a different value of internal system energy.

The system may be heated (in this case the acceptance criteria of the new samples are relaxed) or cooled (in this case the acceptance criteria are narrowed). Once the system is cooled down, the final suboptimal solution is obtained. To sum up, in this algorithm, the search space is probabilistically resampled based on Metropolis-Hastings algorithm [82] for simulating samples from a thermodynamic system [107] (see Pseudocode 1.1.2). This algorithm was also designed for combinatorial optimisation, but may be easily adapted to continuous problems [121].

Tabu search Tabu search is a global optimisation metaheuristic, however, it may be easily used for controlling an embedded heuristic technique (creating a hybrid search method). It is a predecessor of a large family of derivative approaches which introduce memory structures into metaheuristics.

The main goal of the algorithm is to help the search process avoid returning to recently visited areas of the search space (cycling). The method is based on maintaining

a short-time memory of the recent solutions visited in the course of search, refusing to accept the new solutions which are the same (or close) as the ones contained in the memory (see Pseudocode 1.1.3).

The algorithm was introduced by Glover and applied to optimisation of employees duty roster [76] and Travelling Salesman Problem [75]. Nowadays it is one of the most popular algorithms hybridised with other search techniques (see, e.g. [101, 127]).

Pseudocode 1.1.2: PSEUDOCODE OF SIMULATED ANNEALING ALGORITHM

$current \leftarrow randomSolution()$
$best \leftarrow current$
for $i \in [1, max]$
do
- $S_i \leftarrow randomNeighbour(current)$
- $temp \leftarrow rcalculateTemperature(i, tempMax)$
- **if** $goalFunction(S_i) \leq goalFunction(current)$
  - **then**
    - $current \leftarrow S_i$
    - **if** $goalFunction(S_i) \leq goalFunction(best)$
      - **then** $best \leftarrow S_i$
  - **else if** $(exp(\frac{goalFunction(current) - goalFunction(S_i)}{temp}) > random())$
  - **then** $current \leftarrow S_i$

Pseudocode 1.1.3: PSEUDOCODE OF TABU SEARCH ALGORITHM

$best \leftarrow RandomSolution()$
$tabuList \leftarrow \emptyset$
while not $stoppingCondition()$
do
- $candidateList \leftarrow \emptyset$
- **for** $(candidate \in neighborhood(best))$
- **if** (**not** $contains(tabuList, best)$)
  - **then** $candidateList \leftarrow candidateList \cup \{candidate\}$
- $candidate \leftarrow locateBestCandidate(candidateList)$
- **if** $(goalFunction(candidate) \leq goalFunction(best))$
  - **then** $tabuList \leftarrow tabuList \cup \{candidate\}$
- $best \leftarrow candidate$
- $trimTabuList()$

1.2. Evolutionary metaheuristic techniques

The origins of the evolutionary algorithms may be found in the 19th-century works of Gregor Mendel—the first 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 the theory of evolution [38]. 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 [89, 90] in 1975 modelled the process of evolution of the individuals constructed with the use of binary code. He was the first researcher to utilise predefined operators used to change genotypes, which were similar to crossover and mutation. He found out that the average fitness of this population tends to increase. A similar algorithm under the name of **genetic algorithm** was later popularised by David Goldberg [78].
- Ingo Rechenberg [149] and Hans-Paul Schwefel [165] researched optimisation of mechanical devices by permuting randomly-generated solutions. Having observed certain similarities to the biological evolution process in their approach, they invented methods known under the name of **evolution strategies** [167].
- Lawrence Fogel [66] tried to model the process of inception of artificial intelligence upon an approach based on self-organisation. He evolved finite automata aimed at understanding a predefined language [65]. This approach was called **evolutionary programming**, and after further adaptation became a popular technique in optimisation [66].
- John Koza tried to work on 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 [112].

  A detailed survey of evolutionary techniques may be found in [8].

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

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

The stopping criterion is an important feature of the evolutionary approach is (cf. Section 1.2.3).

In evolutionary algorithms, the population is processed in steps called "generations". One generation consists of several phases which introduce changes into population. These phases, which are executed in the presented order, are as follows [4, 78, 131]:

1. Initialisation: random generation of individuals fulfilling predefined constraints.
2. Evaluation: computing the value of the fitness function for all the individuals.
3. Selection: determination of so-called mating pool, comprising the individuals that will become parents of the next population.
4. Crossover: producing offspring of parents belonging to a mating pool.
5. Mutation: introducing additional random changes into the newly generated individuals.

   The detailed algorithm is presented in Pseudocode 1.2.1.

Pseudocode 1.2.1: GENERAL PSEUDOCODE OF EVOLUTIONARY ALGORITHMS

$$
\begin{array}{l}
population \leftarrow initialisePopulation() \\
\textbf{while not } stoppingCondition() \\
\left\{ \begin{array}{l}
evaluations \leftarrow evaluate(population) \\
matingPool \leftarrow selection(population, evaluations) \\
population \leftarrow crossover(matingPool) \\
population \leftarrow mutation(population)
\end{array} \right.
\end{array}
$$

1.2.1. Avoiding the local extrema

Solving difficult search and optimisation problems (e.g. black-box ones) introduces additional requirements for the evolutionary algorithms concerning the ability to avoid or escape from the local minima. This feature is crucial to achieve a balance between the most important features of search techniques, namely *exploration* and *exploitation* [78, 131].

Exploration, as defined by March "... includes things captured by terms such as search, variation, risk taking, experimentation, play, flexibility, discovery, innovation" [126] while exploitation "... includes such things as refinement, choice, production, efficiency, selection, implementation, execution" [126]. In terms of metaheuristics, exploration is the ability to conduct broad search in every part of the admissible search space, in order to provide a reliable estimate of the global optimum, whrereas exploitation consists in refining the search to produce a better solution [176].

A number of modifications of classical evolutionary algorithms were proposed in order to improve their capability of avoiding the local extrema of the fitness function:

- Modifying the lifetime of individuals, as a precaution against the domination of the good individuals (possibly residing in a local extremum) with high reproduction probability. The lifetime of the individual may depend straightforwardly on its fitness value [166].
- Introducing additional random features, e.g. randomly generated immigrants to the population (independent of those created during reproduction), periodic renewal of the population (restarting the search based on the individuals generated in the previous run), random distortions of the fitness function [4].
- Weakening competitiveness during selection, implemented as, e.g. fitness sharing (based on lowering the value of fitness function for the individuals residing in the same local extremum, leading to decrease their reproduction probability) [78].
- Constraining the selection range by dividing the base population into subpopulations. The most important methods are parallel evolutionary algorithms built according to the island and diffusion models [30].
- Preselection (implemented as a modification of the succession scheme) based on removing a certain individual from the population during reproduction, e.g. removing one of the parents (e.g. worse one in terms of the fitness function value). A more complex technique is crowding based on removing from the population an individual that is the most "similar" to the newly-created one [33].

Avoiding the local extrema, and at the same time achieving a balance between exploration and exploitation is closely connected with the notion of population diversity.

1.2.2. Diversity in evolutionary algorithms

In evolutionary search methods, when taking into consideration that many solutions are processed at the same time with variation operators (e.g. crossover and mutation in evolutionary algorithms), maintaining population diversity is crucial. Lack of diversity leads to stagnation and the system may focus on locally optimal solutions (in other words—trapped in a local optimum), lacking the diversity to escape [129]. Therefore, providing appropriate means of measuring and retaining population diversity is a very important task, which in this monograph is measured according to the following two methods:

- Morrison–De Jong (MOI) measure based on the concept of moment of inertia for centroid (centre of gravity computed for points distributed in multi-dimensional

space) [136]. This measure is closely dependent on the distribution of the individuals across the search space.

- Minimal standard deviation (MSD) of each gene computed for all individuals in the population, similar to the column-based method proposed by De Jong [44]. This simple measure focuses on dispersion of the average values computed for individual genes.

It is easy to see that MOI measure tends to be more general than MSD, therefore, observing both measures may yield interesting results that show different aspects of population diversity.

In population-based methods, retaining diversity is an outcome of a balance between exploration and exploitation abilities (cf. classical evolutionary algorithms discussed in a textbook by Michalewicz [131]). An extensive survey of exploration and exploitation balance retaining methods for evolutionary algorithms is given in [36], however, this point of view may be easily extended to all population-based metaheuristics.

Referring to selected classical diversity enhancement techniques, several decomposition and coevolutionary techniques come in mind. *Niching* (or *speciation*) techniques [125] 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 population [30].

In a *coarse-grained* PEA (also known as *regional* or *multiple deme* model), the population is divided into several subpopulations (regions, demes) and 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 neighbourhood.

In *coevolutionary* algorithms, the fitness of each individual is not computed directly based on the definition of the problem to be solved, but results from interactions with other individuals residing in the population. In *cooperative* coevolutionary algorithms, a problem to be solved is decomposed into several subproblems solved by different algorithms in separate subpopulations [147]. Cooperation between individuals from different subpopulations may take place only during a phase of computing fitness for the complete solution. The fitness value is computed only for the group of the individuals from different subpopulations, which form a complete solution to the problem.

In *competitive* coevolutionary algorithms usually two individuals compete with each other in a tournament and their "competitive fitness" corresponds to the outcome of this competition [141]. In each algorithm step, a given individual from one subpopulation competes with its opponents taken from other sub-populations. The results of this competition have an impact on the current fitness of the individual which mates with partners coming from the same subpopulation. This mechanism can be applied irrespectively of the number of subpopulations used in the algorithm—it can be used even if there is only a single population. In this case, opponents are chosen from the same population.

1.2.3. Stopping criteria for the evolutionary algorithms

The two main features of evolutionary algorithms are their random nature and the fact that they produce suboptimal results. Both these features lead to an important conclusion: the user is unsure whether the solution found recently is close to the optimal. He also does not know, when (and if) the optimal solution will be generated. Therefore, an appropriate definition of stopping criteria is crucial for these search techniques.

One of the immanent features of evolutionary algorithms is asymptotic convergence (the probability of reaching the optimal solution converges to 1 as time tends to infinity). There is a lack of mathematical proofs of this feature for popular metaheuristics, however, Michael Vose presented a detailed proof of asymptotic convergence for a simple genetic algorithm modelled with the use of Markov chains [188].

Practical stopping criteria seldom utilise detailed the analysis of the search process, instead they consider the basic features of the observed generations [131, 4, 78]:

1. Monitoring the solutions generated by the algorithm based on the phenomenon that at the beginning of the search better individuals (compared to the current population) are created more frequently than later, e.g.:

   - Criterion of the *maximal cost* based on the assumption that if a certain cost reaches a predefined value, the algorithm is stopped (e.g. after checking that the maximal number of generations has been reached).
   - Criterion of *satisfactory fitness function level*: the algorithm is stopped, when the best individual crosses a certain, predefined value of the fitness function. The application of this criterion is somewhat risky, because the user must set arbitrary level of the fitness function when its properties are unknown. Moreover, none of the individuals may cross the predefined fitness value, so in the worst case the algorithm can run infinitely.
   - Criterion of a the *minimal improvement speed*, based on the observation of the results produced by the algorithm which is stopped when there is no

further improvement in the solution in the predefined period of time. This criterion is also risky as the algorithm may get stuck in a local extremum of the fitness function for a longer time; so again, the accurate choice of the mentioned period of time should require the knowledge of the properties of the fitness function in order not to stop the search prematurely.

2. Monitoring exploration features of the algorithm. These criteria are based on experimental observation of the loss of diversity (caused, e.g. by using of the crossover operator), e.g.:

   - Criterion of *loss of population diversity* based on computing a certain diversity measure and stopping the search when its value falls below a certain level. This approach is based on the assumption that the algorithm, having passed the exploration phase, starts exploitation when new individuals are generated near a certain extremum of the fitness function (not necessarily the global one).
   - Criterion of the *deterioration of the self-adaptive mutation operator* based on the experimentally proved hypothesis that the mutation range adaptation tends to decrease in the evolutionary algorithms (this is again caused by passing into the exploitation phase).

The easiest of these criteria are of course the cost ones (based on generation count, or on the overall time passed from the beginning of computation).

1.3. Hybrid search methods

This monograph focuses on population-based metaheuristics and, moreover, on their specific hybridisations with agent-based computing paradigm, as it will be dealt with in Chapter 2. Therefore, localisation of the methods of interest in the domain of hybrid techniques is also desired. Here, a classification of hybrid methods proposed by Talbi [177] is given, followed by presentation of several popular hybrid techniques. These techniques became an inspiration for different aspects of the systems presented in this monograph.

1.3.1. Classification of hybrid methods

Talbi in [177] provides a concise way of classification of hybrid approaches based on selected design issues:

- In low-level hybrid techniques, a certain function of one algorithm is replaced with another optimisation algorithm.

- At the same time, in high-level hybrid techniques, different optimisation algorithms are combined without changing their intrinsic behaviour.
- In relay hybrid techniques, the individual algorithms are applied in a line, one by one.
- In teamwork hybrid techniques, each algorithm performs an independent search.

Based on the above-mentioned issues, Talbi [178, 177] identifies four main groups of hybrid metaheuristics:

- LRH (low-level relay hybrid): an optimisation method is embedded in a single-solution algorithm (e.g. hybridisation of local-search method inside a simulated annealing).
- LTH (low-level teamwork hybrid): an optimisation method is embedded in a population-based algorithm (e.g. memetic systems hybridizing local-search with evolutionary computation).
- HRH (high-level relay hybrid): several optimisation algorithms are executed in a sequence (e.g. local-search yields an initial population for evolutionary algorithm, then the results of evolution are processed by tabu search).
- HTH (high-level teamwork hybrid): several optimisation algorithms executed parallelly cooperate to find a solution (e.g. parallel evolutionary algorithm).

The computing systems discussed in this monograph cross two classes: LTH and HTH (see Chapter 2).

Again, following Talbi [177], other important attributes of hybrid algorithms may be identified:

- Homogeneous and heterogeneous hybrids combining the same or different algorithms (e.g. parallel evolutionary algorithm consisting of identically configured islands vs. differently parametrised or completely different algorithms running on the islands).
- Global and partial hybrids combining the algorithms searching the whole space or its part (e.g. evolutionary algorithm vs. coevolutionary algorithm).
- General and specialist hybrids solving the same or different target optimisation problems (e.g. memetic algorithm vs. meta-evolution [80]).

Following this classification, the systems discussed in this monograph may be perceived as homogeneous, global and general hybrids.

Both single-solution and population-based approaches may be used to find the final solution, however, the latter seem to suit the difficult problems better (i.e. black-box optimisation). Population-based approaches produce subsequent approximations of the solution in parallel, utilising more information about the solutions already found (by the means of variation operators such as crossover and mutation) than single-solution oriented approaches (such as simulated annealing).

To sum up, a combination of two or more metaheuristics can be perceived as a hybrid metaheuristic. This class of algorithms is very simple from a practitioner's point of view (their implementation based on connecting different "components" is easy).

1.3.2. Cultural and memetic computing

The cultural algorithms extend the evolutionary computing field by adding new capabilities of influencing the search process: the cultural space and multi-level interactions (not only cooperative and competitive relations imposed by the selection procedure and new population generation, but also the belief or knowledge space, along with the possibilities of its genetic modification is considered). Cultural and memetic algorithms are inspired by Richard Dawkins' theory of cultural evolution [41], which assumes that culture may also be decomposed into self-replicating parts (so-called memes) that will compete to prevail in the environment.

Cultural algorithms Culture includes habits, knowledge, beliefs, customs, and morals of members of society. Culture and environment where the society lives are interrelated, and the former interacts with the latter, via positive or negative feedback cycles. Culture also influences the individual evolution, by constructing different rules affecting, e.g. the possibilities of reproducing of certain individuals (compare, e.g. castes in India).

Thus, the cultural algorithm may be defined as a multi-level system that takes advantage of both evolutionary and cultural relations between the individuals in order to influence the ontogeny and manage the search process. For example, during the development of evolutionary search individuals accumulate information about the environment and its different features. This information constitutes the belief space also known as knowledge base and it (or its parts) can be communicated to other individuals. Then, feedback is given, e.g. by communicating interesting areas of the search space (positive feedback) or issuing a warning about the pitfalls (negative feedbacks).

Individuals become units who are capable of learning, being at the same time subjects of some embedded search technique (typically an evolutionary algorithm) and utilising a higher-order mechanism to process the cultural information (e.g. some inference engine) to affect the evolution search process. The knowledge base may contain:

- Normative knowledge: e.g. a collection of value ranges for individuals, acceptable behaviour etc.
- Domain specific knowledge: intrinsic information about the domain of the problem.

- Situational knowledge: information about the ongoing search events (e.g. the best solution so far).
- Temporal knowledge: history of the search space, e.g. a tabu list.
- Spatial knowledge: information about the topography of the search space.

The general algorithm of a cultural metaheuristic is presented in Pseudocode 1.3.1. One of an interesting uses of this approach is Learnable Evolution Model proposed by Ryszard Michalski [134], utilising an inference engine to guide the evolutionary search process. Cultural algorithms may be perceived as a hybridisation of evolutionary, or generally, population-based metaheuristic, with a predefined knowledge base used as a reference point by certain operators, though it does not fit into the Talbi's classification (see Section 1.3.1).

Pseudocode 1.3.1: PSEUDOCODE OF CULTURAL ALGORITHM

$population \leftarrow initialisePopulation()$
$knowledgeBase \leftarrow initialiseKnowledgeBase(population)$
while not $stoppingCondition()$
$\begin{cases} evaluations \leftarrow evaluate(population) \\ matingPool \leftarrow selection(population, evaluations) \\ population \leftarrow crossover(matingPool) \\ population \leftarrow mutation(population) \\ population \leftarrow influence(population, knowledgeBase) \\ knowledgeBase \leftarrow update(population) \end{cases}$

Memetic algorithms Memetic algorithms belong to a class of cultural algorithms and historically are evolutionary algorithms enhanced by hybridisation with local-search methods (the first successful approach was made by Pablo Moscato [137], who hybridised the evolutionary search with a local improvement, using of simulated annealing to solve Traveling Salesman Problem). The evolutionary algorithm utilises the local-search method (in the simplest case, the greedy local search or more sophisticated local search techniques, such as simulated annealing or tabu search) within its evolutionary cycle (in the course of evaluation or mutation). A scheme of the memetic algorithm is shown in Pseudocode 1.3.2. Note that usually only one memetic local search procedure (Baldwinian or Lamarckian) is used:

- Baldwinian local search according to Baldwin theory predispositions may be inherited during reproduction [9]. This method is usually implemented as local search procedure called in the course of the evaluation process. The evaluated

individual is assigned the fitness function value computed for one of its possible descendants (effects of the local-search starting from this individual). First approaches that may be classified as Baldwinian memetic were oriented on so-called Baldwin-effect [4, 85]. The attained improvement does not change the genetic structure (genotype) of the individual that is transferred to the next generation. The individual is kept the same as before local search, but the selection is based on the improved fitness after local search. Baldwin effect follows natural evolution (Darwinian), i.e., learning improves the fitness and selection is based on fitness. The improvement, in this case, is passed indirectly to the next generation through fitness.

Pseudocode 1.3.2: GENERAL PSEUDOCODE OF MEMETIC ALGORITHM

$population \leftarrow initialisePopulation()$
while not $stoppingCondition()$
$\begin{cases} evaluations \leftarrow BALDWINIANEVALUATION(population) \\ matingPool \leftarrow selection(population, evaluations) \\ population \leftarrow crossover(matingPool) \\ population \leftarrow LAMARCKIANMUTATION(population) \end{cases}$

function *BALDWINIANEVALUATION*$(population)$
return $localSearchFitnesses(population)$

function *LAMARCKIANMUTATION*$(population)$
return $localSearchGenotypes(population)$

- Lamarckian local search accorgind to Lamarck's theory characteristics of individuals acquired in the course of life may be inherited by their descendants [58]. This method is usually implemented as a local search procedure called in the course of execution of mutation or crossover operator. The search for a mutated individual is based not only on a stochastic one-time sampling from the solution space, it may be a much more complex process, being an outcome of the local search starting from this individual. In the same way the memetic crossover may be implemented, by trying different combinations of parents' genotypes, until a satisfactory is found. In Lamarckian evolution, individuals improve during their lifetime through local search and the improvement is passed to the next generation. The individuals are selected based on improved fitness and are transferred to the next generation with the improvement incorporated in the genotype.

Although both theories have not been fully verified, metaheuristics based on them are effective in many problems (see, e.g. [117, 190]). Memetic algorithms retain exploration properties of evolutionary algorithms with enhanced exploitation based on the local-search. This makes memetic algorithms faster, but the problem of diversity loss arises and must be duly dealt with [138].

In a comparison of Baldwin and Lamarckian learning, Whitley, et al. [190] showed that utilising either form of learning would be more effective than the classical genetic algorithm without any local improvement procedure. Though Lamarckian learning is faster, it may be susceptible to premature convergence to a local optimum as compared to Baldwin learning. Yao [199] examined both Lamarckian evolution and Baldwin effect in combination with an evolutionary algorithm and local search. The obtained results show that there is no significant difference between Lamarckian-evolution-style combination and Baldwin-effect-style combination.

When implementing a memetic algorithm, it must be decided where (in which phase of evolutionary computation) and when (in which evolutionary cycle) the local search should be applied. Moreover, it must also be decided which individuals need to be improved, how long the local search should run and which memetic model should be used (Baldwinian or Lamarckian)? It is noteworthy that this hybridisation introduces additional complexity to the evolutionary algorithms, emphasising their applicability as the last resort methods (cf. [133]).

Krasnogor and Smith [115] provide a formalisation of the means of hybridisation of evolutionary algorithms with local search methods by introducing a concept of schedulers. These schedulers are used to decide when and where to introduce the local search during the evolution process:

- Fine-grained mutation and crossover schedulers can replace the mutation and crossover operators running the local-search (in this way, e.g. Lamarckian memetics are implemented).
- Coarse-grained scheduler can replace the selection (in this way, e.g. Baldwinian memetics are implemented) and may decide which local search methods are used, which individuals are subject to this process etc.
- Meta-scheduler retains and provides historical information to the local search method, making possible implementation of tabu-search like techniques.

The memetic algorithm may consist of all schedulers, but whether it becomes better, or simply more complex, can only may be discovered in the course of experiments.

Memetic algorithms may be perceived as LTH hybrids, according to Talbi's classification (see Section 1.3.1), merging population-based (evolutionary) system along with a local-search technique.

1.3.3. Immunological metaheuristic techniques

Immunology, which a branch of biomedical science, is situated between medicine and biology, is over 100 years old. Louis Pasteur and Robert Koch are said to have been its founders, as they were the first to carry-out research on disease-causing micro-organisms.

The notion of "artificial immune systems" refers to a class of systems based on ideas, theories and components inspired by the structure and functioning of immune system of vertebrates [43].

The beginning of artificial immune systems dates back to the 1980s, when a proposal was made to apply theoretical immunological models to machine learning and automated problem solving [171, 88, 61]. These algorithms received more attention at the beginning of 1990s (see, e.g. [10, 95]).

First research studies in the field were inspired by theoretical models (e.g. immune network theory) and applied to machine learning, control and optimisation problems. Computer Immunity by Forrest et al. [69, 67] and Immune Anti-Virus by Kephart [105, 106] are classical examples.

These works were formative for the field as they provided an intuitive application domain captivating a broader audience and assisting in differentiating the work as an independent sub-field.

Modern Artificial Immune systems are inspired by one of three sub-fields: clonal selection, negative selection and immune network algorithms. The techniques are used for clustering, pattern recognition, classification, optimisation, and other similar machine learning problem domains [42]. Below, negative selection and clonal selection algorithms will be discussed in detail, as they provided inspiration for part of the research presented later in this monograph, see Chapter 2.

Generally, artificial immune system functions as a classifier, identifying certain patterns (similar to detecting intruders—antigens in the immune system of vertebrates). Therefore, dedicated heuristics to solve classification problems (based on creating antibodies identifying certain patterns) can be easily implemented. However, search and optimisation problems may also be solved by an immune approach (after undertaking certain assumptions, e.g. treating the optima of the goal function as antigens that should be identified) [39].

Immune systems may be perceived as a hybridisation of evolutionary, or generally, population-based metaheuristic, with an affinity checking mechanism, used in certain operations, though they do not fit into the Talbi's classification (see Section 1.3.1).

Clonal selection The clonal selection process serves as an inspiration for a group of algorithms that may be applied to classification or optimisation problems. The clonal

selection theory was proposed by Burnet to describe the behaviour and capabilities of antibodies in acquired immunity [17]. According to this theory, the presence of antigens causes a selection of the most similar lymphocytes. When the lymphocyte is selected, it binds with the antigen and proliferates, creating thousands of copies of itself (affected by small copying errors so-called somatic hypermutation, broadening the possibilities of detecting the antigens). The lymphocyte creates two main types of cells: short living plasma cells that help to remove the antigens by producing antibody molecules, and long-lived memory cells active in a secondary immune response when the antigen is found again in the organism.

Thus this strategy may be seen as a general learning method, including a population of adaptive information units (representing sample solutions to the problem) subject to a competitive process based on selection. This selection consists in proliferation with mutation and produces a generation of individuals better that become fitter to solving the problem.

The general clonal selection algorithm (see Pseudocode 1.3.3) involves a selection of antibodies (candidate solutions) based on affinity, computed either by matching against an antigen pattern (in classification problem solving) or via evaluation of the pattern by the cost function (in optimisation problems). Selected antibodies are subject to cloning, and the number of cloned antibodies is proportional to affinity. At the same time, the hypermutation of clones is inversely proportional to clone affinity. The resulting set of clones competes with the existing antibody population for membership in the next generation. In the end, several low-affinity members of the population are replaced with randomly generated antibodies. When the classification problem is considered, the memory solution set must be maintained in order to represent the solution patterns.

Pseudocode 1.3.3: PSEUDOCODE OF CLONAL SELECTION ALGORITHM

$population \leftarrow initialisePopulation()$
while not $stoppingCondition()$
$affinities \leftarrow computeAffinities(population)$
$selected \leftarrow selection(population, affinities)$
$clones \leftarrow clone(selected, affinities)$
$clones \leftarrow hypermutate(clones)$
$clonesAffinities \leftarrow computeAffinities(clones)$
$population \leftarrow population \cup chooseBest(clones, clonesAffinities)$
$population \leftarrow population \setminus chooseWorst(population, affinities)$
$population \leftarrow partialRandomReplace(population)$

Other algorithms based on clonal selection that are worth mentioning are: CLIGA algorithm [37], the B cell algorithm [103], and Forrest's algorithm [68].

Negative selection The negative selection algorithm was inspired by "self-nonself discrimination" observed in the acquired immune system of vertebrates. The proliferation of lymphocytes during the clonal selection makes possible to generate a wide range of detectors to cleanse the organism of harmful antigens. What is in this process, is that self-cells belonging to the organism are avoided and should not be detected by the lymphocytes in the case of a healthy immune system. In other words, during the process, no self-reactive immune cells are created. Such a set of antibodies is achieved during proliferation, due to a negative selection process that selects and removes the autoimmune cells (that bind with self-cells). This process is observed in nature during the generation of T-lymphocytes in thymus.

The self-nonself discrimination process, which uses a negative selection, consists in creating the anomaly and change detection system modelling anticipation and variation based on a certain set of well-known patterns. Thus, an appropriate model is built by generating patterns that do not match an existing set of available (self or normal) patterns. The resulting non-normal model is then used to detect the unknown by matching the newly-received data to the non-normal patterns.

The principles of this algorithm are quite simple (see Pseudocode 1.3.4). After generating random detectors, they are matched against the set of self-patterns. Those matching ones are removed, while non-matching are kept in the detector repertoire. When the repertoire is ready (a certain number of detectors has been reached), then, based on the set of nonself immune cells, a classification of foreign patterns may be performed (again, according to the same affinity measure that was used to generate this set of antibodies).

The first negative selection algorithm was proposed by Forrest [69] and applied to the monitoring of changes in the filesystem (corruptions and infections by computer viruses), and later formalised as a change detection algorithm [48, 47].

1.4. Agent-based computing

In 1970s, there was a growing interest in the systems, where a task to be solved was decomposed into smaller parts (subtasks), in order to solve them separately and later integrate the solution. This approach may be described as distributed problem solving, and it is usually easy to implement in parallel environment such as multicore machines, clusters or grids. A typical example of such a system is the master-slave evolution model [30], where the master delegates computation of the fitness function

Pseudocode 1.3.4: PSEUDOCODE OF NEGATIVE SELECTION ALGORITHM

$self \leftarrow set\ of\ self\ patterns$
$repertoire \leftarrow \emptyset$
while not $isComplete(repertoire)$
do $detectors \leftarrow generateRandomDetectors()$

$selfClass \leftarrow \emptyset$
$nonSelfClass \leftarrow \emptyset$
for all $d \in detectors$
if not $match(d, repertoire)$
then $selfClass \leftarrow selfClass \cup \{d\}$
else $nonSelfClass \leftarrow nonSelfClass \cup \{d\}$

value to its slaves and waits for the completion of subtasks in order to start another generation [57].

In the field of multi-agent systems bearing a significant legacy from the distributed problem solving, distributed individuals (agents) received great attention, mainly because they were perceived as autonomous beings, being capable of interacting with their environment and other agents, bearing the features of intelligence.

In fact, during the last decades intelligent and autonomous software agents have been widely applied in various domains, such as power systems management [128], flood forecasting [73], business process management [99], intersection management [53], or solving difficult optimisation problems [120], just to mention a few. The key to understand the concept of a multi-agent system (MAS) is 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 [196]. That is why, one of major application areas of multi-agent systems is large-scale computing [181, 14].

Agents play an important role in the integration of artificial intelligence subdisciplines, which is often related to a hybrid design of modern intelligent systems [157]. In most similar applications reported in the literature (see, e.g. [158], [35] for a review), evolutionary algorithm is used by an agent to aid realisation of some of its tasks, often connected with learning or reasoning, or to support coordination of some group (team) activity. In other approaches, agents constitute a management infrastructure for a distributed realisation of an evolutionary algorithm [182, 183, 184].

Agent and agent-based system definitions According to one of the most popular definitions proposed by Wooldridge, an agent is a computer system situated in an

environment, and capable of undertaking independent, autonomous actions in this environment in order to fulfil tasks on behalf of its user [195]. Autonomy is perceived as one of the most crucial features of the agent.

Any computer program that manages a certain apparatus (e.g. a thermostat) or affects the state of the computer system (e.g. Unix system daemon) may be perceived as an agent.

It seems that a definition of the agent introduces a new name for some existing, well-known programming techniques. At the same time, intelligent agents, which are part of complex systems, bring new quality crossing the borders of already existing computer systems, enhancing the notion of an object or process with additional, important features, e.g. [195, 45] helping other agents to fulfil 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 initiative,
- social ability: agents are able to interact with other agents (also with users).

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

The notion of agent system is based directly on the notion of 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 which interact with one another [98, 62].

Agents act in their environment, and different groups of agents may perform their tasks in different parts of the environment. In particular, their activities may overlap. As an example, the possibility of communicating between agents that are "close" in the environment may be given (of course, their closeness depends strongly on the notion of neighbourhood, if such a notion was implemented), or direct interaction with the environment (e.g. only one agent-robot may pass through the door at a time) [194].

Main features of the multi-agent systems are, e.g. [45]: distribution, decentralisation, interaction, organisation, situatedness, openness, emergency and adaptation.

Agents have also been used in computing systems to enhance the search and optimisation capabilities with the above-mentioned agency features.

Agent-based management of computing systems One of the possible applications of agency in computing systems is situated at a technical level. An agent community may take care of the the management of the distributed system, utilising their autonomy and auto-adaptation to implement, e.g. scheduling or load balancing.

Distributed management of the system deployed in a cluster or grid requires a definition of node topology, neighbourhood and migration policies that are affected by a current load of nodes. Well-known standards for constructing multi-agent environments (such as FIPA [64]) 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 [182, 183, 184] proposed Smart Solid Architecture (see Fig. 1.1) that supports these requirements. This architecture is similar to EMAS architecture (cf. Section 2.2.1), where agents are also homogeneous.

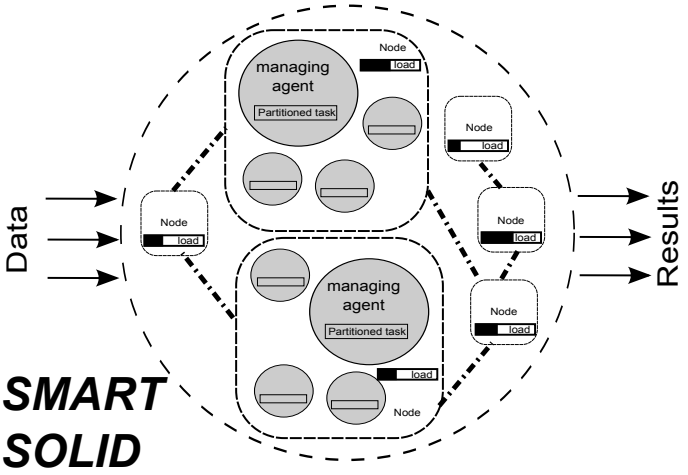


Figure 1.1. Smart solid architecture

However, in this approach the task is divided into subtasks and these are delegated to agents which are to solve them. Agents in this model accomplish two goals: perform computation of the task and try to find a better execution environment (computing node) for the task, based on the load and throughput information. It is noteworthy that agents do interact with one another, in order to be able to solve the assigned task. Agents may undertake the following actions:

- execute the task in order to solve it and communicate the results to other agents,
- denominate the load requirements,
- based on the requirements compute the possible migration capabilities of the agent,
- further divide the task, create the child agents,
- migrate to another execution environment carrying the task.

An effective scheduling algorithm was also introduced in the discussed environment and tested by Schaefer, Grochowski and Uhruski, utilising the natural inspiration of the diffusion mechanism [81]. This system was successfully applied e.g. in the problem of mesh generation for computed-aided engineering applications [164].

Agent-based computing system architecture Another means of connecting together agency and computing systems merges these two notions. In this approach, not only solving of the part of the problem is delegated to the agent, but the agent becomes a part of computing system.

Recalling the most important features of multi-agent systems (MAS) that are adopted by the researchers [62] and can be regarded as definitions of the term:

- a multi agent system is constituted by a set of agents and a sub-system called environment;
- agents perform certain actions that are characteristic to them, resulting in changes of themselves or the environment;
- some of elementary agents' actions form mechanisms perceived as specific in multi-agent systems in general, e.g. negotiation, communication, exchange of resources etc.;
- providing that there is a structure embedded in the environment (a graph) reflecting spatial phenomena of the system, it forms a base for migration of agents;
- an agent that is treated as a black box (observed from the outside) may have some human attributes like autonomy, intelligence etc.

Adding the following assumptions that:

- there is a relatively large number of agents in the system,
- there are some mechanisms incorporated into the system that are used for generating and removing agents, causing dynamic changes in the number of agents,

an interesting type of system called a mass multi-agent system may be obtained [50]. Such system may be used for simulation purposes due to their granularity as there exist of a large number of similar objects manifesting some kind of autonomy. It can also be used as a platform facilitating the realisation of problem-solving techniques, in which case it is a called computing multi-agent system (CMAS) and will constitute a base for further considerations (see Fig. 1.2 and refer to Section 2.1).

From these two approaches, where one is focused on agent-based management of the computation process, and the other is based on decomposition of the problem, entrusting its parts to agents, the latter has become the main inspiration for the computing systems discussed in this monograph (see Chapter 2). However, the former one will also be observed (though not discussed in detail) in the formal definition of the system, constituting an agent-based management infrastructure (see Sections 2.2.4, 2.3.3).

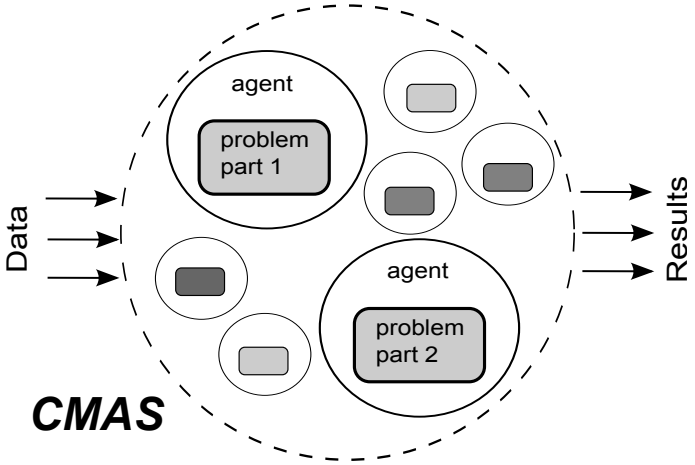


Figure 1.2. Computing multi-agent systems (CMAS)

1.5. Vacant niches in theory and practice

Selected metaheuristics that were presented in this chapter do not exhaust, however, a plethora of such methods that are available nowadays (cf., e.g. [178]). The important thing is that these methods are often hybridised, and the resulting systems become a valuable weapons of choice when dealing with complex "black-box" problems. Among popular approaches, hybrid systems utilising tabu search with other metaheuristics (see, e.g. [179, 186, 185]), or simulated annealing (see, e.g. [201, 118, 119]) yielding successful results may be considered. Therefore, further search for hybrid systems is worth undertaking because looking for synergy of different approaches may lead to obtaining efficient methods in solving particular difficult problems.

In particular, agent-oriented hybrids seem to have much potential, as the feature of autonomy, makes it possible to employ various techniques not exploited before. Hybridisation with different local search methods is of course possible (including tabu search, simulated annealing, or different memetic-oriented approaches). Moreover, particular agents may decide whether to use or not a certain local search operator or any other technique, to adapt parameters of their search operators in order to maintain a balance between exploration and exploitation of the search space.

Designing complex search methods needs not only employing sufficient methods for validation of their outcome, but also verification of their design and analytical features.

Several formal models aimed at proving different features of evolutionary metaheuristics have been constructed. One of the first and important models of metaheuristic methods was Michael Vose's SGA model [187], which proves that simple genetic algorithm (SGA) with the population of a fixed size [78] can be modelled by a Markov chain, and after further assumption that the mutation rate is positive, this chain is ergodic. This result, which formally confirms that SGA is a well-defined global optimisation algorithm (belonging to Las Vegas class [91]), become an inspiration for preparing a Markov-chain based models for agent-based metaheuristics (see Section 3) and proving their ergodicity feature.

Other approaches to model evolutionary algorithms, which are worth mentioning, are different models for single-population, e.g. [40, 175, 79, 124, 153, 154, 155] and parallel evolutionary algorithms, e.g. [1, 31, 63, 180]). Using some of these models, the researchers (e.g. [156]) have proved selected practical features such as first hitting time (see, e.g. [84, 11]) for simple settings (e.g. evolution strategy $1 + 1$ solving the problem of optimisation of a convex function).

There also exist other extensions of Vose's model dealing with hybrid metaheuristics (see, e.g. [111, 29, 163]). Though the formal proofs of asymptotic features are unavailable with one extension, the ergodicity feature has been proved by Schaefer, et al., for parallel Simple Genetic Algorithm in [161], as an extension of the works by Vose.

Apart from these systems, more advanced search and optimisation techniques like memetic or agent-based computational systems lack such models, with some notable exceptions. For example, [198] considers an abstract model of memetic algorithms based on the application of gradient-based local search to the whole population on a generation-basis and provides a sufficient condition for quasi-convergence (i.e., asymptotically finding one of the best k solutions in the search space, where k is the population size). Also, [139] considers adaptive MAs and indicate that only static, greedy and global adaptation strategies (i.e, strategies that use no feedback, check all possible memes and pick the best one, or use a complete historical knowledge to decide on the choice of meme respectively) are globally convergent using elitist selection mechanisms. This stems from [152, 32] and cannot be proved in general for local adaptive strategies.

In general, there is still a lack of a comprehensive stochastic model of the wide class of population-based, agent-based, cultural or memetic computing systems. It may result from the fact that such systems are still poorly defined and formalised, which of course has to be done before making any attempt to analyse their features analytically. Moreover, these systems are complex, so thorough preparations for detailed formal model leading to significant theoretical results are complicated and very time-consuming.

It is easy to see that careful preparation of general models for such systems will surely lead to a better understanding of them and it is necessary to make sure that one does not only propose a complex metaheuristic, but provides a potential user with a firm background, proving that it makes sense to use such a complex computing method. At the same time, above-mentioned models of metaheuristics (cf. Rudolph [156]), focused on proving certain features (e.g. first hitting time), considering even simple, convex functions, should not be underestimated. Again, referring to the well-known "no free lunch theorem", it is hardly probable that one model can fulfil all the researchers' needs. Therefore various approaches to modelling of metaheuristics, offering results of different types will always be valuable.

2. Evolutionary multi-agent systems

Practice proves that an evolutionary algorithm works properly (e.g. in terms of searching for a globally optimal solution) if the population consists of very different individuals, then so-called diversity in the population is preserved [7]. Yet many algorithms tend to prematurely lose this crucial feature and therefore there is a risk that the population may get stuck in some part of the search space (e.g. in the basin of attraction of a certain local extremum instead of searching for a global one).

The situation described above may be explained by the fact that the model of evolution employed by simple evolutionary algorithms lacks many important features observed in organic evolution [8]. This includes dynamically changing environmental conditions, lack of global knowledge, no generational synchronisation, coevolution of species, evolving genotype-phenotype mapping, etc. That is why many variants of classical evolutionary algorithms were proposed, introducing additional mechanisms following the most important phenomena in evolutionary biology. Among these, population decomposition and coevolution had the most inspiring effect on the genesis of evolutionary multi-agent systems.

In 1996, Krzysztof Cetnarowicz [34] proposed an evolutionary multi-agent system (EMAS), which is an agent-oriented computing metaheuristic with interesting features like distributed selection and lack of global control. Since then the idea of EMAS has been applied to different problems (e.g. single, multimodal and multicriteria optimisation). EMAS turned out to be a very good base for introducing different extensions and hybridisation (e.g. using memetic or immunological mechanisms), and encouraged to conduct research at different levels (e.g. formal modelling [29, 163], framework development [60], experimental research [21, 145, 24]). Therefore, EMAS is the main topic of this monograph, aimed at exploiting its most promising features (i.e. in-depth background, formal definition, formal proofs of selected features and various experimental results).

This chapter is dedicated to the EMAS concept and its variations. First, possible architectures of agent-based computing systems are identified. Next, an idea of evolutionary multi-agent computing system (and its memetic version), along with detailed

formal modal of its structure and behaviour is presented. Later, the immunological variant of EMAS is presented, retaining the same presentation structure. Finally, the need for formal verification of the concept is stated.

2.1. Agent-based architectures of computing systems

Starting with a multi-agent perspective (see Section 1.4), three types of computing agent system architectures, which form consecutive levels of increasing complexity, can be distinguished. The main purpose of these architectures is to serve as a means of incorporation of computing methods into cooperating, autonomous entities—agents.

Figure 2.1 presents an example of a classical approach to hybrid computing system—an evolutionary system using a stochastic hill-climbing algorithm constituting in fact a memetic system. Each individual represents a solution, and the evolving population explores and exploits the feasible search space. The whole system is embodied in an agent utilising this idea to adapt to the specific environment [109].

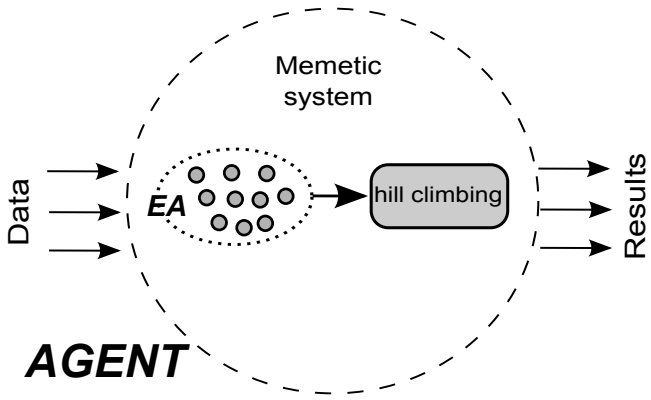


Figure 2.1. A hybrid computing system located in a single agent

Evolutionary multi-agent system shown in Figure 2.2 can be seen as next step in possible specialisation [34]. In this case, evolutionary processes work at a population level—agents searching for a solution to a certain problem are able to generate new agents and may be eliminated from the system on the basis of adequately combined evolutionary operators. The predefined distributed selection mechanism increases the possibility of reproduction for the best agents (in terms of fitness function value).

The result of the search is formed as a set of results obtained by single agents. The architecture of EMAS is homogeneous, which means that the agents are identical as far as an algorithm and built-in actions are considered [109].

EMAS may be easily hybridised with local search techniques forming a memetic variant of the base system. In this case, each agent performs local search in the course of its life in the system. This may be carried out during reproduction or at arbitrarily chosen moments.

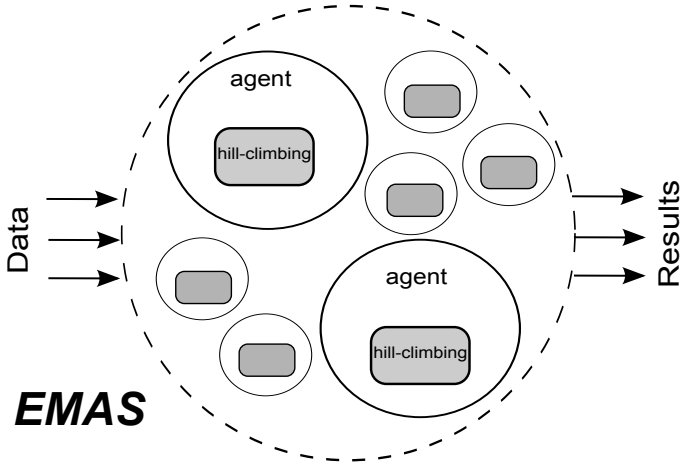


Figure 2.2. A population of homogeneous agents

The highest specialisation level is reached when the agent population is heterogeneous (see Figure 2.3). Some of them may use stochastic hill-climbing, the others—tabu-search etc. The result is stipulated as a consequence of the outcome of the negotiation process among the agents. Many different techniques and protocols can be applied to this purpose. It may be said that these systems closely approach typical multi-agent ones operating in the computer network environment [109].

EMAS (including its memetic variant, see Section 2.2.1) and iEMAS (see Section 2.3) described in this monograph, belong to the second and third class, respectively.

2.2. Evolutionary multi-agent system

Evolutionary processes are by nature decentralised and therefore evolutionary processes in a multi-agent system at a population level may be easily introduced. It means that agents are able to *reproduce* (generate new agents), which is a kind of

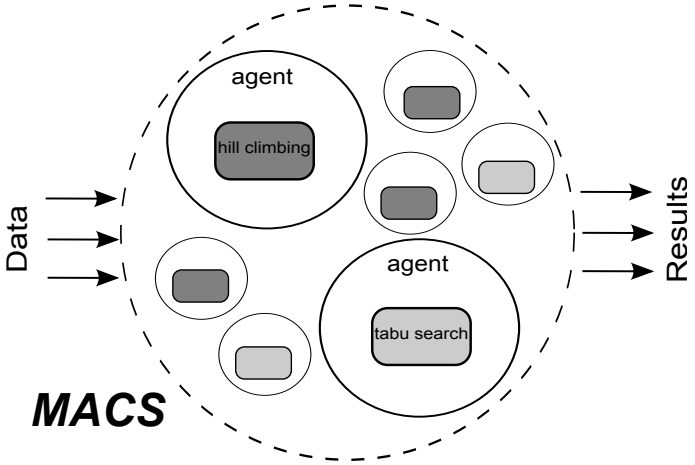


Figure 2.3. A population of heterogeneous agents

cooperative interaction, and may *die* (be eliminated from the system), which is the result of competition (selection). A similar idea with limited autonomy of agents located in fixed positions on some lattice (like in a cellular model of parallel evolutionary algorithms) was developed by Zhong et al. [200]. The key idea of the decentralised model of evolution in EMAS [108] was to ensure full autonomy of agents.

Such a system consists of a relatively large number of rather simple (reactive), homogeneous agents, which have or work out solutions to the same problem (a common goal). Due to computational simplicity and the ability to form independent subsystems (sub-populations), these systems may be efficiently realised in distributed, large-scale environments (see, e.g. [28]).

The formal model presented in this section is recalled after Byrski, Schaefer, Smołka, Cotta [25].

2.2.1. EMAS concept

Leaving aside for a moment the formal definition of EMAS, the system can be described as follows.

Agents in EMAS represent solutions to a given optimisation problem. They are located on islands representing distributed structure of computation. The islands constitute local environments, where direct interactions among agents may take place. In addition, agents are able to change their location, which makes it possible to exchange information and resources all over the system [108].

In EMAS, phenomena of inheritance and selection—the main components of evolutionary processes—are modelled via agent actions of *death* and *reproduction* (see Fig. 2.4). As in the case of classical evolutionary algorithms, inheritance is accomplished by an appropriate definition of reproduction. Core properties of the agent are encoded in its genotype and inherited from its parent(s) with the use of variation operators (mutation and recombination). Moreover, an agent may possess some knowledge acquired during its life, which is not inherited. Both inherited and acquired information (phenotype) determines the behaviour of an agent. It is noteworthy that it is easy to add mechanisms of diversity enhancement, such as allotropic speciation (cf. [30]) to EMAS. It consists in introducing population decomposition and a new action of the agent based on moving from one evolutionary island to another (migration) (see Fig. 2.4).

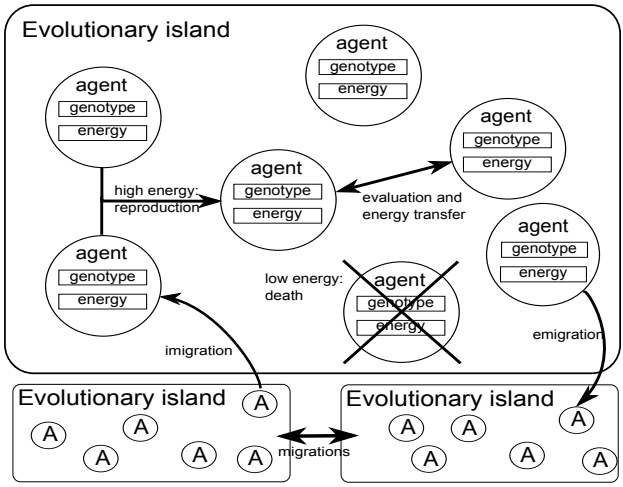


Figure 2.4. Evolutionary multi-agent system (EMAS)

Assuming that no global knowledge is available, and the agents being autonomous, selection mechanism based on acquiring and exchanging non-renewable resources [34] is introduced. It means that a decisive factor of the agent's fitness is still the quality of solution it represents, but expressed by the amount of non-renewable resource it possesses. In general, the agent gains resources as a reward for "good" behaviour, and looses resources as a consequence of "bad" behaviour (behaviour here may be understood as, e.g. acquiring sufficiently good solution). Selection is then realised in such a way that agents with a lot of resources are more likely to reproduce, while a low level of resources increases the possibility of death. So according to classical Franklin's and Graesser's taxonomy—agents of EMAS can be classified as Artificial Life Agents (a kind of Computational Agents) [71].

To recapitulate, the main advantage of the approach in this discussion is that it covers various specialised evolutionary techniques in one coherent model, which enables the following:

- local selection allows intensive exploration of the search space, and explicitly defined living space facilitates implementation in a distributed computational environment, which is similar to parallel evolutionary algorithms,
- evaluation of agents, or more generally, the way a phenotype (behaviour of the agent) is developed from a genotype (inherited information) depends on its interaction with the environment, similarly to coevolutionary algorithms.

Many optimisation tasks, which have already been solved with EMAS and its modifications, have yielded better results than certain classical approaches. They include, among others, optimisation of neural network architecture [22], multiobjective optimisation [169], multimodal optimisation [54] and financial optimisation [55]. EMAS has thus been proved to be a versatile optimisation mechanism in practical situations. A summary of EMAS-related review has is given in [19].

EMAS may be held up as an example of a cultural algorithms, where evolution is performed at the level of relations among agents, and cultural knowledge is acquired from the energy-related information. This knowledge makes it possible to state which agent is better and which is worse, justifying the decision about reproduction. Therefore, the energy-related knowledge serves as situational knowledge (see Section 1.3.2). Memetic variants of EMAS may be easily introduced by modifying evaluation or variation operators (by adding an appropriate local-search method).

The idea of memetic EMAS consist in putting together a local search technique and the evaluation or variation operators utilised in EMAS. Therefore, implementation of Baldwinian and Lamarckian memetics in EMAS may be easily carried out in the following way:

- Baldwinian memetics: this implementation is done in much the same way as in classical evolutionary computing where in the course of evaluation of a a certain individual, the actual returned fitness is computed for one of its potential descendants (after running the local search procedure in the genotype domain, starting from the evaluated individual). Usually this is an iterative process that involves many mutations, or using any other dedicated local search technique. The result returned is the fitness of the best encountered genotype, while the genotype of the evaluated individual remains unchanged.
- Lamarckian memetics: agent may improve its genotype (by running a local search procedure in the genotype domain, starting from the evaluated individual, similarly as in the case of Baldwinian memetics). Usually this is an iterative process, involving many mutations, or using any other dedicated local search technique. The result returned is the best encountered genotype. In the case of

Lamarckian memetics, the genotype of the evaluated individual is changed. This procedure may be performed either during the reproduction or at the arbitrarily chosen moments of agent's life, depending solely on agent's own decision.
Detailed formal model of EMAS is given in the next section.

2.2.2. Formal definition of EMAS

The structure and behaviour of EMAS is described in this section, according to a model presented in [163] that was refined and extended in [25].

Preliminaries As stated in Section 2.2.1, computing agents in EMAS may be seen as autonomous individuals. Each agent is capable of observing its environment by gathering important information, making decisions which affect its activity, and performing actions leading to introducing changes in the system state see, e.g. [96, 97].

Each agent is assigned a genotype that belongs to the finite universum U, $\#U = r < +\infty$ which can be the a set of binary strings, real numbers or other codes providing basis for solving particular global optimisation problem. Agents are assigned to locations (islands) and can migrate between them.

Genetic operations performed on agents' genotypes, such as crossover and mutation, are practically similar to those used in classical evolutionary algorithms and lead to creating a new agent (see Section 2.2.3). The EMAS agent can also create its offspring, using the predefined action of cloning with mutation.

As described in the previous section, the selection mechanism is based on the existence of a non-renewable resource called *life energy*, which is gained and lost when agents perform actions (see [108]).

EMAS has the following characteristic features:

- *Quasi-signature of an agent*: it is composed of its (invariant) genotype gen and the numerical identifier of the copy n that is changed during the migration.
- *Fitness function*: the function $\psi : U \to [0, M]$ related in some way to the objective Φ where again $\mathbb{R}_+ \ni M < +\infty$. In the simplest case $\psi(gen) = \Phi(\eta(gen))$, where $\eta : U \to \mathcal{D}$ is the decoding function.
- *Variable location of agents*: active EMAS agents are present in locations described by a set of immutable integer labels $Loc = \{1, \ldots, s\}$. The locations are linked together by channels along which agents may migrate from one location to another. The symmetric relation $Top \subset Loc^2$ determines the topology of locations. It is assumed that the connection graph $\langle Loc, Top \rangle$ is coherent and does not change during the system evolution.
- *Dynamic collection of agents*: agents belong to the predefined finite set Ag, which can at every moment be one-to-one mapped into the set $U \times P$, where

$P = \{1, \ldots, p\}$ and p is assumed to be the maximal number of agents containing the same genotype. In other words, each agent $ag_{gen,n} \in Ag$ contains one potential solution to a given problem encoded as $gen \in U$. More than one agent may be present in the system containing this solution and the index $n \in P$ is used to distinguish them. Furthermore, it is assumed that each location has its own separate subset of admissible genotype copy numbers P_i, i.e. $P = \bigcup_{i \in Loc} P_i$, $P_i \cap P_j = \emptyset$ for $i \neq j$ and $n \in P_i$ as long as the agent with the temporary copy number n resides at i.

- *Variable energy of agents*: each agent may possess only one of the following quantised energy values: $0, \Delta e, 2 \cdot \Delta e, 3 \cdot \Delta e, \ldots, m \cdot \Delta e$.

Although a pair (gen, n) is not a true identifier because of the variability of its second component n, it properly distinguishes different agents at any time-moment. Reference to a certain agent will be given as: *agent* $ag_{gen,n}$ remembering that this notation is time-dependent. Note that in the context of finding the objective function minimisers, the identity of agents (i.e. solution holders) is less important. Thus the crucial agent attributes are the genotype (and the fitness as its derivative) and the life energy (see the sequel), whereas the copy number plays only an auxiliary role. On the other hand, if this approach was to be aligned one with the Belief-Desire-Intention (BDI) model [148], a true agent identifier could be constructed by means of a quasi-signature as the composition of the agent's genotype with the sequence of the agent's copy numbers at subsequent moments, putting 0 at the moments when the agent is active. However, it is worth noting that such an identifier would not be very useful unless it was stored in a globally synchronized repository, and the need of the global synchronisation would, in turn, prevent the concurrent performance of some crucial EMAS actions.

EMAS state The following three-dimensional incidence and energy matrices $x \in X$ with s layers (corresponding to all locations) $x(i) = \{x(i, gen, n),\ gen \in U,\ n \in P\},\ i \in Loc$ will provide a basis for the description of the system state. The layer $x(i)$ will contain energies of the agents in the i-th location. In other words, the condition $x(i, gen, k) > 0$ means that the k-th clone of the agent carrying the gene $gen \in U$ is active, its energy equals $x(i, gen, k)$ and it may be found in i-th location.

Now, the following coherency conditions are introduced:

- $(\cdot, j, k)$-th column contains at most one value greater than zero, which means that the agent with the k-th copy of the j-th genotype may be found in only one location at a time, whereas other agents containing copies of the j-th genotype may be present in other locations;

- entries of incidence and energy matrices are non-negative $x(i,j,k) \geqslant 0$ for $1 \leqslant i \leqslant s, 1 \leqslant j \leqslant r, 1 \leqslant k \leqslant p$ and $\sum_{i=1}^{s}\sum_{j=1}^{r}\sum_{k=1}^{p} x(i,j,k) = 1$, which means that the total energy of the whole system is constant and equal to 1;
- $x(i, gen, n)$ can be positive only for n acceptable in the location i, i.e. $n \in P_i$;
- each layer $x(i)$ has at most q_i values greater than zero, denoting the maximal capacity of the i-th location and, moreover, the quantum of energy Δe is less than or equal to the total energy divided by the maximal number of individuals that may be present in the system,

$$\Delta e \leqslant \frac{1}{\sum_{i=1}^{s} q_i}, \tag{2.1}$$

  which makes it possible to achieve the maximal population of agents in the system;
- reasonable values of p should be greater than or equal to 1 and less than or equal to $\sum_{i=1}^{s} q_i$; it is assumed that $p = \sum_{i=1}^{s} q_i$, which assures that each configuration of agents in locations is available, in respect of the total number of active agents $\sum_{i=1}^{s} q_i$; increasing p over this value does not enhance the descriptive power of the presented model;
- the maximal number of copies for each location $\#P_i$ should not be less than q_i in order to make possible for the system to reach a state in which particular location is filled with clones of one agent; on the other hand due to the previous assumption $\#P_i$ cannot be greater than q_i; therefore it is finally assumed that $\#P_i = q_i$.

Gathering all these conditions, the considered set of three-dimensional incidence and energy matrices constituting the EMAS space of states may be described:

$$\begin{aligned} X = \Bigg\{ & x \in \{0, \Delta e, 2 \cdot \Delta e, 3 \cdot \Delta e, \ldots, m \cdot \Delta e\}^{s \cdot r \cdot p}, \\ & \text{subject to: } \Delta e \cdot m = 1, \sum_{i=1}^{s}\sum_{j=1}^{r}\sum_{k=1}^{p} x(i,j,k) = 1, \\ & x(i,j,k) = 0 \text{ for } 1 \leqslant i \leqslant s, 1 \leqslant j \leqslant r, k \notin P_i, \\ & \sum_{j=1}^{r}\sum_{k=1}^{p} [x(i,j,k) > 0] \leqslant q_i \text{ for } 1 \leqslant i \leqslant s, \\ & \sum_{i=1}^{s} [x(i,j,k) > 0] \leqslant 1 \text{ for } 1 \leqslant j \leqslant r,\ 1 \leqslant k \leqslant p \Bigg\} \end{aligned} \tag{2.2}$$

where $[\cdot]$ is the indicator function, ie. $[\texttt{true}] = 1$ and $[\texttt{false}] = 0$.

Structure and behaviour of EMAS EMAS may be modelled as the following tuple:

$$\langle U, Loc, Top, Ag, \{agsel_i\}_{i\in Loc}, locsel, \{LA_i\}_{i\in Loc}, MA, \omega, Act\rangle, \tag{2.3}$$

where:

MA (Master Agent) is used to synchronise the work of locations; it allows to perform actions in particular locations; this agent is also used to introduce the necessary synchronisation into the system;

$locsel : X \to \mathcal{M}(Loc)$ is a function used by MA to determine which location should be permitted to perform the next action;

LA_i (Local Agent) is assigned to each location; it is used to synchronise the work of Computing Agents (CA) present in the location; LA_i chooses the CA and allows it to evaluate a decision and perform the action, at the same time requesting permission from MA to perform this action;

$agsel_i : X \to \mathcal{M}(U \times P)$ is a family of functions used by LA_i to select the CA that may perform the action, such that each location $i \in Loc$ has its own function $agsel_i$; probability $agsel_i(x)(gen, n)$ vanishes when the agent $ag_{gen,n}$ is inactive in the state $x \in X$ or is present in location different from i-th one ($n \notin P_i$);

$\omega : X \times U \to \mathcal{M}(Act)$ is a function used by agents to select actions from the set Act; both symbols will be explained later.

Act is a predefined, finite set of actions.

Hereafter $\mathcal{M}(\cdot)$ stands for the space of probabilistic measures. Moreover, the following convention will be used to describe the discrete probability distributions $a : X \times Par \to \mathcal{M}(A)$ on the set A, depending on the state $x \in X$ and an optional parameter $p \in Par$.

- $a(x, p)(d)$ is the probability of $d \in A$ (lower-case letter), assuming the current state x and the parameter p. It simplifies a more rigorous notation $a(x, p)(\{d\})$.
- $a(x, p)(W), W \subset A$ is the probability of set W (upper-case letter).

The names a, A, Par are of course generic. The variables d and p may also be defined as tuples. In the case of probability distribution on the finite set A, this notation is unambiguous.

The population of agents is initialized by means of introductory sampling. This may be regarded as a one-time sampling from X, according to a predefined probability distribution (possibly the uniform one) from $\mathcal{M}(X)$. Each agent starts its work immediately after being activated. At every observable moment only one agent present

in each location gains the possibility of changing the state of the system by executing its action.

The function $agsel_i$ is used by LA_i to determine which agent present in the i-th location will be the next one to interact with the system. After being chosen, the agent $ag_{gen,n}$ chooses one of the possible actions according to the probability distribution $\omega(x, gen)$. Note that there is a relationship of this probability distribution and the concept of fine-grain schedulers introduced into a syntactic model of memetic algorithms in [114]. It must be noted that the selection of action by all agents, which carry the same genotype gen in the same state x, is performed according to the same probability distribution $\omega(x, gen)$ and does not depend on the genotype copy number n. In the simplest case, ω returns the uniform probability distribution over Act for all $(x, gen) \in X \times U$.

Next, the agent applies to LA_i for the permission to perform this action. When the necessary permission is granted, the agent $ag_{gen,n}$ performs the action after checking that a condition defined by formula (2.5) has been fulfilled. If during the action the agent's energy is brought to 0, this agent suspends its work in the system (it becomes inactive).

MA manages the activities of LA_i and allows them to grant their agents permission to carry out requested tasks. The detailed managing algorithm based on the rendezvous mechanism [86] is described in Section 2.2.4.

A subset of states in which the agent $ag_{gen,n}$ is active is denoted as:

$$X_{gen,n} = \{x \in X \mid \exists\, l \in Loc : x(l, gen, n) > 0\}\,,\ (gen, n) \in U \times P \tag{2.4}$$

Each action $\alpha \in Act$ will be represented as a pair of function families $(\{\delta_\alpha^{gen,n}\}_{(gen,n)\in U\times P}, \{\vartheta_\alpha^{gen,n}\}_{(gen,n)\in U\times P})$. The functions

$$\delta_\alpha^{gen,n} : X \to \mathcal{M}(\{0, 1\}) \tag{2.5}$$

make it possible to take a decision, to perform the action. The action α is performed with probability $\delta_\alpha^{gen,n}(x)(1)$ by the agent $ag_{gen,n}$ at state $x \in X$ and rejected with probability $\delta_\alpha^{gen,n}(x)(0)$. Because the action may be invoked only by the active agent, the function $\delta_\alpha^{gen,n}$ always has to return a negative decision for all $x \in X \setminus X_{gen,n}$ and only the restriction $\delta_\alpha^{gen,n}|X_{gen,n}$ constitutes the crucial part of this function, so

$$\delta_\alpha^{gen,n}(x) = \begin{cases} \delta_\alpha^{gen,n}|X_{gen,n} & x \in X_{gen,n} \\ (1, 0) & x \in X \setminus X_{gen,n}. \end{cases} \tag{2.6}$$

Next, the formula

$$\vartheta_\alpha^{gen,n} : X \to \mathcal{M}(X) \tag{2.7}$$

defines non-deterministic state transition functions, therefore $\vartheta_\alpha^{gen,n}$ is caused by executing action α by the agent $ag_{gen,n}$. The value of $\vartheta_\alpha^{gen,n}(x)(x')$ denotes the probability of passing from the state x to x' resulting from the execution of the action α by the agent $ag_{gen,n}$. The function is only invoked if the agent is active, therefore it is enough to define a restriction $\vartheta_\alpha^{gen,n}|X_{gen,n}$ for each action α, and let it take an arbitrary value on $X \setminus X_{gen,n}$.

If any action is rejected, the trivial state transition

$$\vartheta_{null} : X \to \mathcal{M}(X) \tag{2.8}$$

such that for all $x \in X$

$$\vartheta_{null}(x)(x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

is performed.

The probability transition function for the action α performed by the agent with the n-th copy of the genotype gen

$$\varrho_\alpha^{gen,n} : X \to \mathcal{M}(X) \tag{2.10}$$

is defined by the formula

$$\varrho_\alpha^{gen,n}(x)(x') = \delta_\alpha^{gen,n}(x)(0) \cdot \vartheta_{null}(x)(x') + \delta_\alpha^{gen,n}(x)(1) \cdot \vartheta_\alpha^{gen,n}(x)(x'), \tag{2.11}$$

where $x \in X$ denotes the current state, and $x' \in X$ is the consecutive state resulting from the conditional execution of α.

Note that it is formally possible to consider a very large (yet finite) set Act, comprising all actions up to a certain description length (using a Gödel numbering [77] or any appropriate encoding). This implies that this set may be implicitly defined by such an encoding, allowing much flexibility in the set of actions available (a connection can be drawn with multi-meme algorithms [113]).

Observation 2.2.1 ([25, Observation 2.1])**.** *Given the agent $ag_{gen,n} \in Ag$ it is enough to define two restrictions $\delta_\alpha^{gen,n}|X_{gen,n}$ and $\vartheta_\alpha^{gen,n}|X_{gen,n}$ in order to establish the probability transition function $\varrho_\alpha^{gen,n}$ associated with the execution of the action α—see Eq. (2.10) and Eq. (2.11).*

The agents' actions may be divided into two distinct types:

- global—changing the state of the system in two or more locations, so only one global action may be performed at a time,
- local—changing the state of the system in one location considering only the state of locally present agents; only one local action for one location may be performed at a time.

Therefore, the Act set is divided in the following way:

$$Act = Act_{gl} \cup Act_{loc} \tag{2.12}$$

Informally speaking, if the location $i \in Loc$ contains the agent performing a certain local action $\alpha \in Act_{loc}$, only the entries of layer $x(i)$ of the incidence and energy matrix are changed (other changes have zero probability). Moreover, these actions do not depend on other layers of x. The action $null$ is obviously "the most local one", because it does not change anything at all.

The above description can be formalised as follows:

Definition 2.2.1 ([25, Definition 3.1])**.** *The action $\alpha \in Act$ is* local *($\alpha \in Act_{loc}$) if, and only if, for any agent which can execute the action (i.e., $\forall\, (gen, n) \in U \times P$) we have:*

1. *α does not change anything except for part of the state that describes the location l in which $ag_{gen,n}$ is performing the action α (i.e., $x(l)$ being the $l-$th layer in matrix x), so*

$$\forall\, x \in X : \varrho_{\alpha}^{gen,n}(x)(x_{next}) = 0, \tag{2.13}$$

   for $x_{next} \in X$ such that $\exists\ i \neq l :\ x_{next}(i) \neq x(i)$ and x_{next} denotes one of the states which is supposed to be reached at the step immediately after state x appears;

2. *α is independent upon any other layers of x which means that*

$$\begin{gathered}
\forall\, x_1, x_2 \in X, x_1(l) = x_2(l), \\
\forall\, x_{1,next}, x_{2,next} \in X, x_{1,next}(l) = x_{2,next}(l) \\
\text{and for each } i \neq l \\
x_1(i) = x_{1,next}(i),\ x_2(i) = x_{2,next}(i) \\
\varrho_{\alpha}^{gen,n}(x_1)(x_{1,next}) = \varrho_{\alpha}^{gen,n}(x_2)(x_{2,next}).
\end{gathered} \tag{2.14}$$

All other actions are considered global (elements of Act_{gl}).

What makes local actions important is the fact that if they are executed by agents present in different locations, they commute. Detailed formal proof of the local actions commutativity is provided in [25].

Local actions must be mutually exclusive within a single location, and global actions are mutually exclusive in the whole system, so only one global action may be performed at a time in the system, but many local actions (one in each location at most) may be performed at the same time.

2.2.3. EMAS actions

Consider the following set of actions [25]:

$$Act \quad = \quad \{get, repr, migr, clo, lse\}, \tag{2.15}$$

where get lets a better agent take a part of life energy from a worse agent and may make the agent with low energy inactive, $repr$ activates the agent as an offspring agent in the system, $migr$ denotes migration of agents between two locations, clo activates the agent as a mutated clone agent in the system, whereas lse allows the local search methods to incorporate into EMAS. Such sample set of actions cover almost all search activities appearing in GA and MA as selection, mutation, crossover and local optimisation.

Denote by l the location of the current active agent containing the n-th copy of the genotype gen performing the action (i.e., $x(l, gen, n) > 0$). Notice that if the particular state x is established, the location of each active agent is unambiguously determined. Lower index "next" will be used to denote the state which may appear in the next step assuming some current state, e.g. x_{next} is the subsequent possible value of x.

Action performing distributed selection The energy transfer action get is based on the idea of agent rendezvous. Agents meet one of their neighbours (it chooses randomly one of its neighbours—agents from the same location or "island") and during this meeting a quantum of energy Δe flows in direction described by a certain stochastic function cmp. The most probable direction is from the worse evaluated agent to the better one, which may be considered a kind of a tournament (see e.g. [131]).

By Observation 2.2.1 the following may be obtained:

Observation 2.2.2 ([25, Observation 6.1])**.** *The probability transition function* $\varrho_{get}^{gen,n} : X \to \mathcal{M}(X)$ *associated with action get is determined by:*

$$\delta_{get}^{gen,n} | X_{gen,n}\,(x)(1) = \begin{cases} 1 & \text{if } NBAG_{l,gen,n} \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

$$NBAG_{l,gen,n} = \\ \big\{(j,k) : x(l,j,k) > 0 \text{ and } (j \neq gen \text{ or } k \neq n)\big\} \tag{2.17}$$

$$\vartheta_{get}^{gen,n}|X_{gen,n}\,(x)(x') = \frac{1}{\#NBAG_{l,gen,n}} \sum_{(gen,n)\in NBAG_{l,gen,n}}$$

$$\Bigg(cmp(gen,gen')(0)\cdot[x' = next(x,gen,n,gen',n')]+$$

$$cmp(gen,gen')(1)\cdot[x' = next(x,gen',n',gen,n)]\Bigg) \tag{2.18}$$

$$cmp : U\times U \to \mathcal{M}(\{0,1\}) \tag{2.19}$$

$$next(x,a,b,a',b') = x_{next}:$$

$$x_{next}(i,j,k) = \begin{cases} x(i,j,k) - \Delta e & \text{if } j = a \text{ and } k = b \\ & \text{and } i = l \\ x(i,j,k) + \Delta e & \text{if } j = a' \text{ and } k = b' \\ & \text{and } i = l \\ x(i,j,k) & \text{otherwise.} \end{cases} \tag{2.20}$$

Explanation. The decision of the action get, $\delta_{get}^{gen,n}$, defined by Eq. (2.16), depends upon the existence of at least one neighboring agent in the same location and is performed by checking the contents of the $NBAG_{l,gen,n}$ set defined by Eq. (2.17). The arbitrary state of the system when the decision is evaluated by the agent $ag_{gen,n}$ is denoted by $x \in X$.

The transition function uses the function cmp to compare the meeting agents. This is a probabilistic function that takes advantage of the fitness function ψ in order to compare the agents. The better fitness the genotype has, the greater probability that it will get the quantum of energy from its neighbour. A lower probability is assigned to the reverse flow.

Technically, if $cmp(gen, gen')$ is sampled as 0 then agent with the genotype gen increases its energy and the second agent with genotype gen' looses its energy. If $cmp(gen, gen')$ takes the sampled value 1, the energy is passed in the opposite direction.

In the case of the positive evaluation of this decision, the state transition described by Eq. (2.18) is performed. This formula comes from Bayes' theorem, which makes us check all possible agents from $NBAG_{l,gen,n}$. For each agent contained in this set a different state transition is performed as described by the function $next(\cdot,\cdot,\cdot,\cdot,\cdot)$. The direction of the energy transfer is determined using the function cmp.

The state transition function is constructed according to Eq. (2.20). The incidence matrix $x_{next} \in X$ is obtained from x by changing two entries related to a pair of agents $ag_{gen,n}, ag_{gen',n'}$ that exchanged energy.

Observation 2.2.3 ([25, Observation 6.2])**.** *The value of the probability transition function imposed by Eqs. (2.10), (2.11), (2.16)–(2.20) performed by the agent $ag_{gen,n}$ present in the location l, depends only on the elements of the system state contained in its location. The action get may only introduce changes in the state entries associated with the location l. In other words, assume that x is a current state, all states that differ from x outside the l-th layer have a null probability in the next step.*

Explanation. Observation 2.2.3 stems from Eqs. (2.10), (2.11), (2.16)–(2.20). Part of them that introduce changes in the state entries depends on and refers only to the entries in the current l-th location. All other entries are simply rewritten to the next state.

Actions inspired by the genetic operations A decision on the reproduction action $repr$ is based on the idea of the agent rendezvous (similarly to get). An agent with sufficient energy (above a certain predefined threshold e_{repr}) meets one of its neighbours and creates offspring agent based on their solutions. The genotype of the offspring agent is selected according to the predefined family of probability distributions $mix : U \times U \to \mathcal{M}(U)$ associated with the sequence of genetic operations (e.g. crossover followed by mutation, see [187]). In particular, $mix(gen, gen')(gen'')$ denotes the probability that gen'' is born of the parents gen and gen'. A part of the parents' energy ($e_0 = n_0 \cdot \Delta e$, n_0 is even) is passed onto the offspring agent.

By Observation 2.2.1 the following may be obtained:

Observation 2.2.4 ([25, Observation 6.3])**.** *The probability transition function $\varrho_{repr}^{gen,n} : X \to \mathcal{M}(X)$ associated with the action $repr$ is determined by:*

$$\delta_{repr}^{gen,n}|X_{gen,n}\,(x)(1) = \begin{cases} 1 & \text{if } x(l,gen,n) > e_{repr} \text{ and } RPAG_{l,gen,n} \neq \emptyset \\ & \text{and } \sum_{j=1}^{r} \sum_{k \in P_l} [x(l,j,k) > 0] < q_l \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

$$RPAG_{l,gen,n} = \left\{ (gen', n') \in NBAG_{l,gen,n}; x(l, gen', n') > e_{repr} \right\} \tag{2.22}$$

$$\vartheta_{repr}^{gen,n}|X_{gen,n}\,(x)(x') =$$
$$\frac{1}{\#RPAG_{l,gen,n}} \sum_{(gen',n')\in RPAG_{l,gen,n}} \sum_{gen''\in U}$$
$$mix(gen,gen')(gen'') \cdot cpchoose(x,x',gen,n,gen',n',gen'') \tag{2.23}$$

$$cpchoose(x,x',gen,n,gen',n',gen'') =$$
$$\begin{cases} [x'=x] \text{ if } FC_{l,gen''} = \emptyset \\ \dfrac{1}{\#FC_{l,gen''}} \displaystyle\sum_{m\in FC_{l,gen''}} [x' = next(x,gen,n,gen',n', \\ gen'',m)] \text{otherwise,} \end{cases} \tag{2.24}$$

where:

$$FC_{l,gen''} = \left\{ o \in P_l \mid x(l,gen'',o) = 0 \right\} \tag{2.25}$$

$$next(x,a,b,a',b',a'',m) = x_{next}:$$
$$x_{next}(i,j,k) = \begin{cases} x(i,j,k) - \frac{e_0}{2} & \text{if } j \in \{a,a'\} \text{ and } k \in \{b,b'\} \\ & \text{and } i = l \\ e_0 & \text{if } j = a'' \text{ and } k = m \text{ and } i = l \\ x(i,j,k) & \text{otherwise.} \end{cases} \tag{2.26}$$

Remark 2.2.1 ([25, Remark 6.1])**.** *Note that if*

$$\sum_{j=1}^{r} \sum_{k\in P_l} [x(l,j,k) > 0] < q_l,$$

which means that the location l is not full, then our assumptions guarantee that for every genotype gen

$$FC_{l,gen} \neq \emptyset,$$

i.e. there is a copy number to take.

Explanation. The decision that the agent $ag_{gen,n}$ performs the action $repr$, i.e. $\delta_{repr}^{gen,n}$ (defined by Eq. (2.21)) is based on the condition that there is at least one neighbouring agent in the same location and both agents have sufficient energy (higher than e_{repr})

to produce an offspring. This condition is verified by checking the contents of the set $RPAG_{l,gen,n}$ defined by Eq. (2.22). Therein $x \in X$ denotes the arbitrary state of the system when the decision is evaluated by the agent $ag_{gen,n}$.

In the case of positive evaluation of this decision (i.e., there exists $ag_{gen',n'}$ with sufficient energy) and enough space in the location (i.e., the number of agents does not exceed q_l), the state transition described by Eq. (2.23) is performed. This formula stems from Bayes' theorem which makes us check all possible agents from $RPAG_{l,gen,n}$. For each agent contained in this set a different state transition is performed, as described by function $next(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$. The probability of choosing an agent is equal to $(\#RPAG_{l,gen,n})^{-1}$.

The agent $ag_{gen,n}$ that initiated the action with its neighbour $ag_{gen',n'}$ becomes a parent and selects the offspring agent genotype gen'' using the probability distribution $mix(gen, gen') \in \mathcal{M}(U)$. The offspring agent ($ag_{gen'',n''}$) is created if there is enough room in the parental location (in this case there is always a free copy number, see Remark 2.2.1). If there is more than 1 inactive copy, the copy-number n'' of the offspring agent is selected with uniform probability distribution—see Eqs. (2.24), (2.25)).

The state transition function is constructed in the way described in Eq. (2.26). The incidence matrix $x_{next} \in X$ is obtained from x by changing entries related to agents $ag_{gen,n}$, $ag_{gen',n'}$, $ag_{gen'',n''}$. Part of the parents' energy is passed to the offspring agent with the genotype gen'', which is activated in the location l (and whose energy is set to e_0).

Observation 2.2.5 ([25, Observation 6.4])**.** *The value of the probability transition function imposed by Eqs.* (2.10)*,* (2.11)*,* (2.21)–(2.26)*, performed by the agent $ag_{gen,n}$ present in the location l depends only on the elements of the system state contained in its location. This function does not introduce any changes in other locations.*

Explanation. Eqs. (2.21)–(2.26) involve only those entries of the incidence matrix that are related to the location l, so both assumptions fully satisfy the conditions of Definition 2.2.1.

A decision on the migration action $migr$ may be undertaken by the agent with enough energy to migrate if there exists a location that is able to accept it (its number of agents does not exceed the maximum). When these conditions are met the agent is moved from one location to another one.

Observation 2.2.6 ([25, Observation 6.5])**.** *The probability transition function $\varrho_{migr}^{gen,n} : X \to \mathcal{M}(X)$ associated with the action $migr$ is determined by:*

$$\delta^{gen,n}_{migr}|X_{gen,n}\,(x)(1) = \begin{cases} 1 & \text{if } (x(l,gen,n) > e_{migr} \text{ and } \#ACCLOC_l > 0) \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

$$ACCLOC_l = \Bigg\{ Loc \setminus \{l\} \ni l' : \big((l,l') \in Top\big) \text{ and } \left(\sum_{j=1}^{r} \sum_{k \in P_{l'}} [x(l',j,k) > 0] < q_{l'} \right) \Bigg\} \tag{2.28}$$

$$\vartheta^{gen,n}_{migr}|X_{gen,n}\,(x)(x') = \frac{1}{\#ACCLOC_l} \sum_{loc' \in ACCLOC_l} \frac{1}{\#FC_{loc',gen}} \sum_{m \in FC_{loc',gen}} [x' = next(x,gen,n,loc',m)], \tag{2.29}$$

where $FC_{loc',gen}$ is given by Eq. (2.25) and

$$next(x,a,b,c,m) = x_{next} : x_{next}(i,j,k) = \begin{cases} 0 & \text{if } i = l \text{ and } j = a \\ & \textit{and } k = b \\ x(l,a,b) & \text{if } i = c \text{ and } j = a \\ & \textit{and } k = m \\ x(i,j,k) & \text{otherwise.} \end{cases} \tag{2.30}$$

Explanation. The decision on the action $migr$, $\delta^{gen,n}_{migr}$ (defined by Eq. (2.27)) is based on condition that the agent $ag_{gen,n}$ has sufficient energy to migrate (greater than $e_{migr} \geqslant e_{repr}$) and there is at least one neighbouring location (l') capable of accepting the agent that wants to migrate (the number of agents in this location does not exceed $q_{l'}$), which is stated by checking the contents of the $ACCLOC_l$ set defined by Eq. (2.28). The arbitrary state of the system when the decision is evaluated by the agent $ag_{gen,n}$ is denoted by $x \in X$.

In the case of positive evaluation (of this decision), the state transition described by Eq. (2.29) is performed. Again, the formula comes from Bayes' theorem, which checks all possible locations from $ACCLOC_l$. For each location contained in this

set, a different state transition is performed as described by the function $next(\cdot,\cdot,\cdot,\cdot)$. The probability of choosing a location is equal to $(\#ACCLOC_l)^{-1}$. The agent initiating the action moves from its location (l) to location l' that is uniformly chosen from the set $ACCLOC_l$. The change of location requires a change of the migrating agent's copy number. A new number is chosen from the set of available copy numbers for the target location (if the location is not full, this set is not empty, see Remark 2.2.1) according to the uniform distribution.

The state transition function is constructed as described in Eq. (2.30). The incidence matrix $x_{next} \in X$ is obtained from x by changing two entries related to a position of the agent $ag_{gen,n}$ in the location.

Observation 2.2.7 ([25, Observation 6.6])**.** *The value of the probability transition function imposed by Eqs. (2.10), (2.11), (2.27)–(2.30) and performed by the agent $ag_{j,k}$ present in the location l does not depend only on the elements of the system state contained in its location. The action $migr$ may introduce changes in the state entries associated with the location l and another location.*

Explanation. Eqs. (2.27)–(2.30) include references to the elements of the system contained in the l-th and other locations.

The EMAS definition given here is enriched (with respect to the one in [29, 163]) by endowing it with a new cloning and mutation action clo, which allows a single agent to produce offspring.

A decision on the cloning and mutation action clo is based on checking the amount of agent's energy only. An agent with sufficient energy may create offspring agent based on its solution using the predefined family of probability distributions $mut : U \to \mathcal{M}(U)$, associated with genetic mutation (see, e.g. [187]). In particular, $mut(gen)(gen')$ denotes the probability that gen' is born of the parent gen. Part of the parent's energy ($e_1 = n_1 \cdot \Delta e, n_1 \in \mathbb{N}$) is passed onto the offspring agent.

By Observation 2.2.1 the following may be obtained:

Observation 2.2.8 ([25, Observation 6.7])**.** *The probability transition function $\varrho_{clo}^{gen,n} : X \to \mathcal{M}(X)$ associated with the action clo is determined by:*

$$\delta_{clo}^{gen,n}|X_{gen,n}\,(x)(1) = \begin{cases} 1 & \text{if } x(l,gen,n) > e_{repr} \\ & \text{and } \sum_{j=1}^{r}\sum_{k\in P_l}[x(l,j,k) > 0] < q_l \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

$$\vartheta_{clo}^{gen,n}|X_{gen,n}\,(x)(x') = \sum_{gen'\in U} mut(gen)(gen') \cdot cpchoose_1(x,x',gen,n,gen') \tag{2.32}$$

$$cpchoose_1(x, x', gen, n, gen') = \begin{cases} [x' = x] \text{ if } FC_{l,gen'} = \emptyset \\ \dfrac{1}{\#FC_{l,gen'}} \displaystyle\sum_{m \in FC_{l,gen'}} [x' = \\ next(x, gen, n, gen', m)] \text{ otherwise,} \end{cases} \tag{2.33}$$

where $FC_{l,gen'}$ is given by Eq. (2.25) and

$$next(x, a, b, a', m) = x_{next} :$$
$$x_{next}(i,j,k) = \begin{cases} x(i,j,k) - e_1 & \text{if } j = a \text{ and } k = b \\ & \text{and } i = l \\ e_1 & \text{if } j = a' \text{ and } k = m \\ & \text{and } i = l \\ x(i,j,k) & \text{otherwise.} \end{cases} \tag{2.34}$$

Explanation. The decision $\delta_{clo}^{gen,n}$ on the action clo defined by Eq. (2.31) is based on condition that the energy of the agent exceeds the predefined threshold e_{clo}, where $x \in X$ denotes the arbitrary state of the system when the decision is evaluated by the agent $ag_{gen,n}$.

In the case of positive evaluation of this decision and enough space in the location (the number of agents does not exceed q_l), the state transition described by Eq. (2.32) is performed. As for the previous actions, this formula stems from Bayes' theorem. Based on the target agent (after applying a mutation operator), the state transition is performed as described by the function $next(\cdot, \cdot, \cdot, \cdot, \cdot)$.

The state transition function is constructed in the way described in Eq. (2.34). The incidence matrix $x_{next} \in X$ is obtained from x by changing entries related to agents $ag_{gen,n}$ and $ag_{gen',n'}$. Part of the parent's energy is passed onto the offspring agent with genotype gen' which is activated in location l and whose energy is set to e_0.

Observation 2.2.9 ([25, Observation 6.8])**.** *The value of the probability transition function imposed by Eqs. (2.10), (2.11), (2.31)–(2.34) and performed by the agent $ag_{gen,n}$ present in the location l, depends only on the elements of the system state contained in its location. The action clo will not introduce any changes in the other location.*

Explanation. In eqs. (2.31)–(2.34) entries of the incidence matrix not related to the current l-th location remain intact. Also the right-hand sides of these equations do not involve these entries.

Action resulting from the local search activation Using a mechanism similar to the one included in the definition of action clo, it is possible to represent local searches invoked from particular points encoded by genotypes in U. The action which implements the local search will be called lse. The agent executing lse produces a new agent with a new genotype gen' which results from the application of the local search procedure loc starting from the parental genotype gen. The local search may be invoked by an agent with sufficient energy (greater than e_{repr}), thus the decision function $\delta_{lse}^{gen,n}|X_{gen,n}$ will have the same form as determined by Eq. (2.31).

The result of running the local method is characterized by the function $loc : U \to \mathcal{M}(U)$. In the case of stochastic local search (e.g. a strictly ascending random walk), the probability distribution $loc(gen)$ characterizes the result of running such method starting from the parental genotype gen. Of course, $loc(gen)$ need not (and in general will not) be strictly positive as it might be assumed in the case of the genetic mutation distribution $mut(gen)$. In the case of deterministic local method, $loc(gen)$ takes strictly one positive value for the genotype gen' obtained from applying this local method to gen. Of course, the loc function depends on both the local search algorithm and the fitness function corresponding to the optimisation problem at hand.

Part $e_1 = n_1 \cdot \Delta e, n_1 \in \mathbb{N}$ of the parent's energy is passed to the offspring in much the same way as it is carried out during execution of the action clo. The above assumptions together with the Observation 2.2.1 lead to the following:

Observation 2.2.10 ([25, Observation 6.9])**.** *The probability transition function $\varrho_{lse}^{gen,n} : X \to \mathcal{M}(X)$ associated with the action lse is determined by the decision function $\delta_{lse}^{gen,n}|X_{gen,n}$ described by the Eq. (2.31) and the actions' kernel by the Eqs. (2.32)–(2.34) in which the function $mut : U \to \mathcal{M}(U)$ is replaced by the function $loc : U \to \mathcal{M}(U)$.*

The above observation might be verified in the same way as Observation 2.2.8 formulated and proved for the clo action. Similarly, without additional verification we may accept the following:

Observation 2.2.11 ([25, Observation 6.10])**.** *The value of the probability transition function $\varrho_{lse}^{gen,n} : X \to \mathcal{M}(X)$ imposed by the action lse executed by the agent $ag_{gen,n}$ present in the location l depends only on the elements of the system state contained in its location. The action lse will not introduce changes in other locations.*

Action's taxonomy Observations 2.2.3, 2.2.5, 2.2.7, 2.2.9 and 2.2.11 may be summarised in the following corollary:

Corollary 2.2.1 ([25, Corollary 6.1])**.** *The action $migr$ is global, whereas the actions get, $repr$, clo and lse are local, i.e.*

$$\begin{aligned} Act_{loc} = &\ \{get, repr, clo, lse\}, \\ Act_{gl} = &\ \{migr\}. \end{aligned}$$

Observation 2.2.12 ([25, Observation 6.11])**.** *The probability transitions imposed by actions get, $repr$, $migr$, clo and lse satisfy the Markov condition (see, e.g. [12]).*

Explanation. The probability transitions of the actions $\varrho^{gen,n}_{get}$, $\varrho^{gen,n}_{repr}$, $\varrho^{gen,n}_{migr}$, $\varrho^{gen,n}_{clo}$, $\varrho^{gen,n}_{lse}$ given by Eqs. (2.10), (2.11), (2.16)–(2.30) depend only on the current state $x \in X$ of the system.

2.2.4. EMAS management

In order to obtain relaxed synchronisation (i.e., agents present in locations may act in parallel), a dedicated timing mechanism must be introduced, which means that all state changes must be assigned to subsequent time moments $t_0, t_1, \ldots$ Now, the algorithmic description for CA will be considered, $LA_i, i \in Loc$ and MA presented in Pseudocodes 2.2.1, 2.2.2 and 2.2.3, respectively [25] constituting Hoare's rendezvous-like synchronisation mechanism [86]. Note that here and later $\underline{a}(B)$ denotes the effect of randomly sampling one of the elements from the set B with the random distribution a; it is also assumed that the sets $localact, globalact \subset Act$ contain the local and global actions' signatures respectively.

In Figure 2.5 the scheme of the synchronisation mechanism is shown.

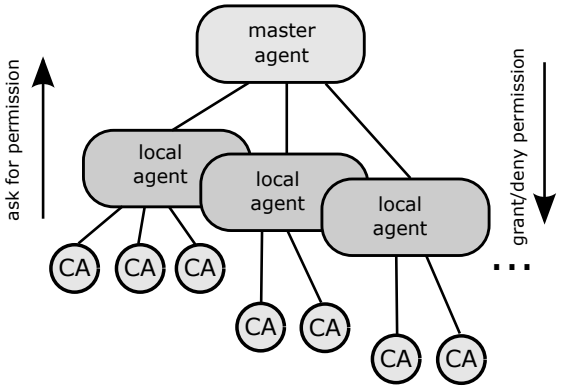


Figure 2.5. EMAS management structure

$CA = ag_{gen,n}$, present in the location i at every observable time moment chooses an action it wants to perform. It uses the probability distribution ω to choose from Act and asks its supervisor (LA_i) for permission, and using function $send()$, it sends a message with a chosen action. Then, it suspends its work and waits for permission (or denial) from LA_i using blocking function $b\_receive()$.

Both these functions are variadic. The first parameter in each function is always a target identifier, and the other parameters may be one or more values to be passed. In this particular case, the target either receives a certain value or just receives a signal from the sender (in this case no value is required).

Once permission is granted and the decision assigned to the action is true, the CA changes the state of the location (see Pseudocode 2.2.1). Then the agent suspends its work again in order to get permission to perform subsequent actions.

Pseudocode 2.2.1: COMPUTING AGENT'S ALGORITHM

while true
$\begin{cases} reply \leftarrow 0 \\ \alpha \leftarrow \underline{\omega(x, gen)}(Act) \\ send(LA_i, \alpha) \\ b\_receive(LA_i, reply) \\ \textbf{if } reply \textbf{ and } \underline{\delta_\alpha(x, gen, n)}(\{0, 1\}) \\ \quad \textbf{then } x_{next} \leftarrow \underline{\vartheta_\alpha^{gen,n}(x)}(X) \\ send(LA_i) \\ b\_receive(LA_i) \end{cases}$

The LA_i (see Pseudocode 2.2.2) starts its work by checking whether the location contains any agents, so it sends a message to MA and waits for a reply. If there are any agents in the location, the LA_i receives signals containing identifiers of actions to be performed from all its agents and puts them into a hash map indexed by genotypes and containing actions identifiers. Then the LA_i utilises the function $agsel_i$ to choose CA which should try to perform its action. This action is reported to MA and as soon as it has received permission, the computing agent can perform the action. All other agents (and the chosen one when permission is not granted) are stopped from performing their actions. Afterwards the LA waits for all CA to report the readiness to perform subsequent actions, and then reports this fact to the MA, and as soon as it has received permission, CA can perform their actions.

MA (see Pseudocode 2.2.3) waits for all requests from each location and then chooses randomly one location. If this location asks for permission to perform a global action, then this permission is granted and all other locations are rejected. Otherwise all locations asking for permission to perform a global action are rejected

and all those asking for permission to perform local actions—are granted. Finally, MA waits once more for all locations to report that their work has been finished and let them try to perform a subsequent action.

Pseudocode 2.2.2: LOCAL AGENT'S ALGORITHM

while true
$localgen \leftarrow \{(j,k) \in U \times P_i;\ x(i,j,k) > 0\}$
$genact \leftarrow hashmap(U \times P_i, Act)$
$act \leftarrow 0$
$reply \leftarrow 0$
if $\#localgen = 0$
  then $send(MA, null)$; $b\_receive(MA)$; $send(MA, null)$; $b\_receive(MA)$
  else **for each** $g \in localgen$ **do** $b\_receive(g, act)$; $genact[g] \leftarrow act$
    $gchosen \leftarrow \underline{agsel_i(x)}(Act)$
    REPORT $(genact[gchosen], gchosen)$

function REPORT$(act, chosen)$
$send(MA, act)$
$b\_receive(MA, reply)$
if $reply$
  then $send(chosen, 1)$
  else $send(chosen, 0)$
for each $g \in (localgen \setminus chosen)$ **do** $send(g, 0)$
for each $g \in localgen$ **do** $b\_receive(g)$
$send(MA)$
$b\_receive(MA)$
for each $g \in localgen$ **do** $send(g)$

Pseudocode 2.2.3: MASTER AGENT'S ALGORITHM

while true
$\left\{\begin{array}{l}
local \leftarrow \{i : i \in [1, s]\} \\
localloc \leftarrow \emptyset \\
localglob \leftarrow \emptyset \\
act \leftarrow 0 \\
rep \leftarrow 0 \\
\textbf{for each } j \in local \\
\quad \textbf{do} \left\{\begin{array}{l} b_receive(j, act) \\ \textbf{if } act \in Act_{gl} \\ \quad \textbf{then } localglob \leftarrow localglob \cup \{j\} \\ \quad \textbf{else } localloc \leftarrow localloc \cup \{j\} \end{array}\right. \\
lchosen \leftarrow \overline{locsel(x)}(Loc) \\
\textbf{if } lchosen \in \overline{localglob} \\
\quad \textbf{then} \left\{\begin{array}{l} send(lchosen, 1) \\ \textbf{for each } j \in (local \setminus \{lchosen\}) \\ \quad \textbf{do } send(j, 0) \end{array}\right. \\
\quad \textbf{else} \left\{\begin{array}{l} \textbf{for each } j \in localloc \textbf{ do } send(j, 1) \\ \textbf{for each } j \in localglob \textbf{ do } send(j, 0) \end{array}\right. \\
\textbf{for each } j \in local \textbf{ do } b_receive(j) \\
\textbf{for each } j \in local \textbf{ do } send(j)
\end{array}\right.$

2.3. Immunological evolutionary multi-agent system

Solving difficult search problems with population-based approaches, especially those with costly evaluation of their candidate solutions (e.g. inverse problems [5]), requires looking for techniques that may make it possible to increase the search efficiency. One method can be reducing the number of fitness function calls. It may be done in several ways, e.g. by applying tabu search [3] or, strictly in a technical layer, by caching fitness values generated for identical (or similar) genotypes. In this section, an immunological selection mechanism designed for EMAS is discussed.

The formal model presented in this section is recalled after Byrski, Schaefer and Smołka [26].

2.3.1. iEMAS concept

The main idea of applying immunological inspirations to speed up the process of selection in EMAS is based on the assumption that "bad" phenotypes come from "bad" genotypes. Thus, a new group of agents (acting as lymphocyte T-cells) may be introduced (see works of Byrski and Kisiel-Dorohinicki [21, 20] and Ph.D. thesis of Byrski [18]). They are responsible for recognising and removing agents with genotypes similar to the genotype pattern possessed by these lymphocytes. Another approach may introduce specific penalty applied by T-cells for recognised agents (certain amount of the agent's energy disappears) instead of removing them from the system. The general structure of iEMAS (immunological EMAS) is presented in Figure 2.6.

Of course, there must exist some predefined affinity (lymphocyte-agent matching) function which may be based, e.g. on the percentage difference between corresponding genes. Lymphocytes are created in the system after the action of death. The late agent genotype is transformed into lymphocyte patterns by means of mutation operator, and the new lymphocyte (or a group of lymphocytes) is introduced into the system.

In both cases, new lymphocytes must undergo a process of negative selection. Within a specified period of time, the affinity of immature lymphocytes' patterns towards "good" agents (possessing relatively high amount of energy) is tested. If it is high (lymphocytes recognise "good" agents as nonself), they are removed from the system. If the affinity is low, it is assumed that they will be able to recognise nonself individuals ("bad" agents) leaving agents with high energy intact. The lifespan of lymphocytes is controlled by specific, renewable resource (strength), used as a counter by the lymphocyte agent. (see Fig. 2.6).

iEMAS is an example of a cultural algorithm, which is an extension of EMAS. However, based on the agent energy-related information the population of lympho-

cytes is modified (they are considered mature or not). Therefore, the agent energy-related knowledge serves as situational knowledge (see Section 1.3.2). Lymphocyte energy serves as temporal knowledge, so the lymphocyte may be removed after a certain period of time. Generally speaking, the information contained in lymphocytes may be seen as distributed "tabu" list, and therefore considered to be temporal knowledge.

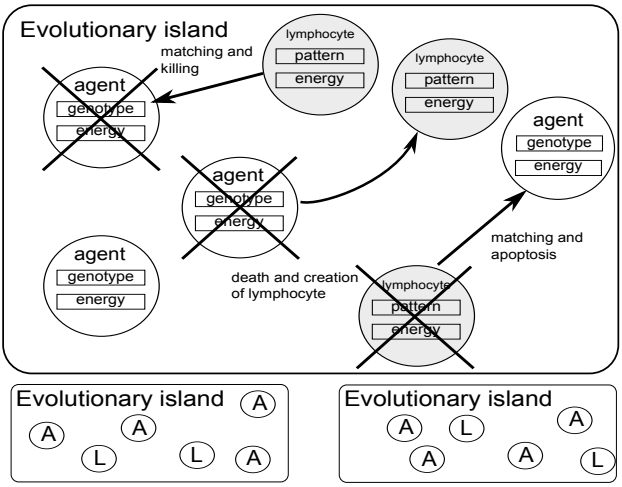


Figure 2.6. Immunological evolutionary multi-agent system (iEMAS)

The early removing of "bad" solutions and a decrease in the number of the individuals in the computing populations (see the computing results later in this section) makes iEMAS a weapon of choice to deal with problems, where a complex fitness function is used. Indeed, an interesting optimisation task was approached with iEMAS, namely the evolution of neural network architecture [20, 23] and benchmark function optimisation [21].

Immunological selection mechanism may seem as a dangerous, potentially harmful extension of the already effective system, as its main principle of work is removing certain, not fully evaluated individuals, even more, by excluding certain areas from the solution space. However, as mentioned before, this mechanism should be seen as a distributed tabu list that temporarily excludes non-promising areas by "guarding" them over a certain time period with the use of lymphocytes. As soon as the time of lymphocytes activity has passed, these areas are available again for computing agents.

2.3.2. Formal definition of iEMAS

In this section, a description of EMAS state is cited and extended by adding a matrix describing iEMAS state (following [26]).

EMAS state In order to define the iEMAS state, the EMAS state, which is a crucial part of iEMAS, must be outlined first. Here, similarly to Section 2.2.2, a set of three-dimensional, incidence and energy matrices is introduced, as $\lambda \in \Lambda$ with s layers (corresponding to all locations) $\lambda(i) = \{\lambda(i, gen, n),\ gen \in U,\ n \in P_i\},\ i \in Loc.$ The layer $\lambda(i)$ will contain energies of agents in the i-th location. In other words, if $\lambda(i, gen, k) > 0$, it means that the k-th clone of the agent with the gene $gen \in U$ is active, its energy equals $\lambda(i, gen, k)$ and it is present in the i-th location.

Now, the following coherency conditions are given:

- $(\cdot, j, k)$-th column contains at most one value greater than zero, which means that the agent with k-th copy of j-th genotype may be present in only one location at a time, whereas other agents containing copies of j-th genotype may be present in other locations;
- entries incidence and energy matrices are non-negative $\lambda(i, j, k) \geq 0,\ \forall\ i = 1, \ldots, s,\ j = 1, \ldots, r,\ k = 1, \ldots, p$ and $\sum_{i=1}^{s} \sum_{j=1}^{r} \sum_{k=1}^{p} \lambda(i, j, k) = 1$, which means that the total energy contained of the whole system is constant, equal to 1;
- each layer $\lambda(i)$ has at most q_i values greater than zero, which denotes the maximal capacity of the i-th location and moreover, the quantum of energy Δe is less than or equal to the total energy divided by the maximal number of individuals that may be present in the system $\Delta e \leq \frac{1}{\sum_{i=1}^{s} q_i}$, which allows us to achieve maximal population of agents in the system;
- reasonable values of p should be greater than or equal to 1 and less than or equal to $\sum_{i=1}^{s} q_i$; it is assumed that $p = \sum_{i=1}^{s} q_i$, which assures that each configuration of agents in locations is available, in respect of the total number of active agents $\sum_{i=1}^{s} q_i$; increasing p over this value does not enhance the descriptive power of the model;
- the maximal number of copies for each location $\#P_i$ should not be less than q_i, because a system state in which a particular location is filled with clones of one agent should be allowed; increasing $\#P_i$ over q_i is only a formal constraint relaxation, so finally it is assumed that $\#P_i = q_i$.

Putting all these conditions together, a set of three-dimensional incidence and energy matrices may be described in the following way, giving the EMAS part of the

system state.

$$\Lambda = \Big\{ince \in \{0, \Delta e, 2 \cdot \Delta e, 3 \cdot \Delta e, \ldots, m \cdot \Delta e\}^{s \cdot r \cdot p}, \Delta e \cdot m = 1,$$
$$\sum_{i=1}^{s}\sum_{j=1}^{r}\sum_{k=1}^{p} x(i,j,k) = 1, \forall\, i = 1, \ldots, s : \sum_{j=1}^{r}\sum_{k=1}^{p}[x(i,j,k) > 0] \leq q_i, \quad (2.35)$$
$$\forall\, i = 1, \ldots, s,\ j = 1, \ldots, r,\ k \notin P_i :\ x(i,j,k) = 0,$$
$$\forall j = 1, \ldots, r,\ k = 1, \ldots, p : \sum_{i=1}^{s}[x(i,j,k) > 0] \leq 1\Big\}$$

where $[\cdot]$ denotes the value of the logical expression in parentheses.

iEMAS state To continue considerations presented in, e.g. [29, 163, 26] iEMAS has a dynamic collection of lymphocytes that belong to the finite set Tc. Lymphocytes are unambiguously indexed by the genotypes from U, so that $\#Tc = \#U$ = r. Lymphocytes have a similar structure to the computing agents defined in the previous paragraph, however, their actions are different (because their goals are different from the computing agents' goals) and their total energy does not have to be constant.

In addition to the above-given EMAS state describing the location and energy of agents, a set of matrices containing similar information for lymphocytes must be considered. Yet there is no need to ensure the constant total energy for lymphocytes. This additional set of lymphocyte incidence and energy matrices is described in the following way:

$$\Gamma = \Big\{tcince \in [0, \Delta e, \ldots, n \cdot \Delta e]^{r \cdot s} : \forall\, i = 1, \ldots, s \ \sum_{j=1}^{r}[tcince(i,j) > 0] \leq tcq_j$$
$$\text{and } \forall\, j = 1, \ldots, r \ \sum_{i=1}^{s}[tcince(i,j) > 0] \leq 1\Big\}, \quad (2.36)$$

where $tcince(i,j)$ stands for energy of tc_j which is active in the location i. The integers tcq_j, $j = 1, \ldots, s$ stand for the maximal number of lymphocytes in particular locations. It is most convenient to assume $tcq_j = q_j$, $\forall j = 1, \ldots, s$.

The space of iEMAS states is defined as follows:

$$X = \Lambda \times \Gamma. \quad (2.37)$$

Structure and behaviour of iEMAS iEMAS may be modelled as the following tuple:

$$\langle U, \{P_i\}_{i\in Loc}, Loc, Top, Ag, \{agsel_i\}_{i\in Loc}, locsel, \{LA_i\}_{i\in Loc}, MA, \omega, Act, \{typesel_i\}_{i\in Loc}, \{tcsel_i\}_{i\in Loc}, Tc, Tcact\rangle, \tag{2.38}$$

where:

- MA (Master Agent) is used to synchronise the work of locations; it allows performing actions in particular locations. This agent is also used to introduce necessary synchronisation into the system;
- $locsel : X \to \mathcal{M}(Loc)$ is a function used by MA to determine which location should be permitted to perform the next action,
- LA_i (Local Agent) is assigned to each location; it is used to synchronize the work of computing agents present in the location; LA_i chooses a Computing Agent (CA) or a T-Cell (TC) and lets it evaluate a decision and perform the action, at the same time asking permission from MA to perform this action;
- $agsel_i : X \to \mathcal{M}(U \times P_i)$ is a family of functions used by LA_i to select the agent that may perform the action, such that each location $i \in Loc$ has its own function $agsel_i$.
- $\omega : X \times U \to \mathcal{M}(Act)$ is a function used by agents to select actions from the set Act.
- Act is a predefined, finite set of actions.
- $typesel_i$ is a function used to select the type of agent in i-th location to interact with the system in the current step,
- $tcsel_i$ is used to choose a lymphocyte in the i-th location to interact with the system in the current step,
- φ is the decision function for lymphocytes, which chooses an action for them to perform,
- $Tcact$ is a set of actions that may be performed by lymphocytes.

Similarly to the case of EMAS (see Section 2.2.2) the population of agents is initialized by means of the introductory sampling. This may be regarded as a one-time sampling from X according to a predefined probability distribution (possibly the uniform one) from $\mathcal{M}(X)$. Each agent starts its work immediately after being activated. At every observable moment only one agent present in each location gains the possibility of changing the state of the system by executing its action.

The function $typesel_i$ is used by LA_i to decide, whether CA or a TC should be chosen. Then, one of the functions $agsel_i$ or $tcsel_i$ is used to determine which

agent (or lymphocyte) present in the i-th location will be the next one to interact with the system. After being chosen, the agent $ag_{gen,n}$ chooses one of the possible actions according to the probability distribution $\omega(x, gen)$. In the case of lymphocyte tc_{gen}, the probability distribution $\varphi(x, gen)$ is used.

It must be noted that the selection of action by all agents, which carry the same genotype gen in the same state x, is performed according to the same probability distribution $\omega(x, gen)$ and does not depend on the the genotype copy number n. In the simplest case, ω returns the uniform probability distribution over Act for all $(x, gen) \in X \times U$. Similarly to the case of a lymphocyte, φ returns the uniform probability distribution over $Tcact$ for all $(x, gen) \in X \times U$.

Next, the computing agent or lymphocyte applies to LA_i for the permission to perform this action. When the necessary permission is granted, the agent $ag_{gen,n}$ (or the lymphocyte tc_{gen}) performs the action after checking that a condition defined by formulas (2.39) and (2.48) has been fulfilled. If during the action an agent's or lymphocyte's energy is brought to 0, this agent suspends its work in the system (it becomes inactive).

MA manages the activities of LA_i and allows them to grant their agents permissions to carry out requested tasks. The detailed managing algorithm based on the rendezvous mechanism [86] is described in Section 2.3.3.

Denote by X_{gen} a subset of states in which there are active agents with the genotype $gen \in U$ or an active lymphocyte. Again, as the first step in defining the iEMAS dynamics, the EMAS part of the system must be addressed.

Each action $\alpha \in Act$ will be represented as the a of function families $(\{\delta_\alpha^{gen}\}_{gen \in U}, \{\vartheta_\alpha^{gen}\}_{gen \in U})$. The functions

$$\delta_\alpha^{gen} : X \to \mathcal{M}(\{0, 1\}) \tag{2.39}$$

represent the decision to be taken: whether the action can be performed or not. The action α is performed with the probability $\delta_\alpha^{gen}(x)(1)$ by the agent $ag_{gen,n}$ at the state $x \in X$ and rejected with the probability $\delta_\alpha^{gen}(x)(0)$.

Next, the formula

$$\vartheta_\alpha^{gen} : X \to \mathcal{M}(X) \tag{2.40}$$

defines the non-deterministic state transition functions, so that ϑ_α^{gen} is caused by the execution of the action α by the agent $ag_{gen,n}$. The function is only invoked if the agent in active, therefore it is enough to define its restriction $\vartheta_\alpha^{gen}|X_{gen}$ and take an arbitrary value on $X \setminus X_{gen}$.

If any action is rejected, the trivial state transition

$$\vartheta_{null} : X \to \mathcal{M}(X) \tag{2.41}$$

such that for all $x \in X$

$$\vartheta_{null}(x)(x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{2.42}$$

is performed.

The probability transition function for the action α performed by the agent carrying the genotype gen

$$\varrho_{\alpha}^{gen} : X \to \mathcal{M}(X) \tag{2.43}$$

is given by the formula

$$\begin{aligned} \varrho_{\alpha}^{gen}(x)(x') = \delta_{\alpha}^{gen}(x)(0) \cdot \vartheta_{null}(x)(x') \\ +\delta_{\alpha}^{gen}(x)(1) \cdot \vartheta_{\alpha}^{gen}(x)(x'), \end{aligned} \tag{2.44}$$

where $x \in X$ denotes a current state and $x' \in X$ a consecutive state resulted from the conditional execution of α.

The function $typesel_i$ is introduced, to choose which type of agents will be able to perform the action:

$$typesel_i : X \to \mathcal{M}(\{0, 1\}). \tag{2.45}$$

When 0 is chosen, one of the agents is activated, when 1—the lymphocyte is activated.

The function $agsel_i$ that chooses an agent to be activated is similar as in the case of EMAS but it now depends in some way on the extended state from X defined by (2.37). Now a new function that will choose a lymphocyte to be activated is introduced:

$$tcsel_i : X \to \mathcal{M}(Tc). \tag{2.46}$$

The function ω which chooses the action for the active agent remains intact, though its domain changes (because of the new state definition, see (2.37)).

The function which chooses the action for the active lymphocyte is the following:

$$\varphi : U \times X \to \mathcal{M}(Tcact) \tag{2.47}$$

Each action $\alpha \in Tcact$ will be represented as a pair of function families $(\{\gamma_{\alpha}^{gen}\}_{gen \in U}, \{\kappa_{\alpha}^{gen}\}_{gen \in U})$. The functions

$$\gamma_{\alpha}^{gen} : X \to \mathcal{M}(\{0, 1\}) \tag{2.48}$$

represent the decision to be taken: whether the action can be performed or not. The action α is performed with the probability $\gamma\alpha^{gen}(x)(1)$ by the lymphocyte tc_{gen} at the state $x \in X$, and rejected with the probability $\gamma_{\alpha}^{gen}(x)(0)$.

The following family of functions $\eta_\alpha^{gen} : X \to \mathcal{M}(X)$ will be used, where $gen \in U$, $\alpha \in Tcact$. Each of them expresses the probability transition imposed by the lymphocyte tc_{gen} that performs the action $\alpha \in Tcact$. They are given by the general formula:

$$\eta_\alpha^{gen}(x)(x') = \gamma_\alpha(gen, x)(\{0\}) \cdot \vartheta_{null}(x)(x') + \gamma_\alpha(gen, x)(\{1\}) \cdot \kappa_\alpha^{gen,n}(x)(x') \tag{2.49}$$

The agents' and lymphocytes' actions may be divided into two distinct types: global—they change the state of the system in two or more locations, so only one global action may be performed at a time, and local—they change the state of the system inside one location respecting only the state of local agents; only one local action for one location may be performed at a time.

Therefore the Act set is divided in the following way: $Act = Act_{gl} \cup Act_{loc}$ and $Tcact : Tcact = Tcact_{gl} \cup Tcact_{loc}$ accordingly. Speaking informally, local actions (elements of $Act_{loc}, Tcact_{loc}$) change only the entries of the layer $x(i)$ of the incidence and energy matrices if the location $i \in Loc$ contains the agent performing a certain action. Moreover, these actions do not depend on other layers of x. The action $null$ is obviously "the most local one", because it does not change anything at all.

In the case of EMAS and iEMAS, actions such as evaluation or lymphocyte pattern matching may be perceived as local, whereas the action of migration is perceived as global. The above-stated conditions may be defined formally and may be used to prove commutativity of iEMAS (cf. [29, 163]), as in the case of EMAS in [25].

2.3.3. iEMAS management

Similarly to the case of EMAS described in Section 2.2.4, in order to design a Markov model of the system with relaxed synchronisation (i.e. so that agents present in different locations may act concurrently), a timing mechanism must be introduced, i.e. all state changes must be assigned to subsequent time moments $t_0, t_1, \ldots$ [26].

In Figure 2.7, a scheme of the synchronisation mechanism is presented constituting Hoare's rendezvous-like synchronisation mechanism [86], similarly to EMAS.

The CA (see Pseudocode 2.3.1) and TC (see Pseudocode 2.3.2) present in the location i at every observable time moment choose an action they want to perform and ask their supervisor (LA_i) for permission to carry on, sending a message with a chosen action identifier using function $send()$ (similarly to the case of EMAS, see Section 2.2.4). Then they suspend their work and wait for permission (or denial) from LA_i using blocking function $b\_receive()$. Both these functions are variadic. The first parameter in each function is always a target identifier, and the other parameters may be one or more values to be passed. In this particular case, the target either receives

a certain value or just receives a signal from the sender (in this case no value is required).

Once the permission is granted and the decision assigned to the action is positive, the computational agent changes the state of the location. Then, the agent suspends its work again in order to get permission to perform a subsequent action.

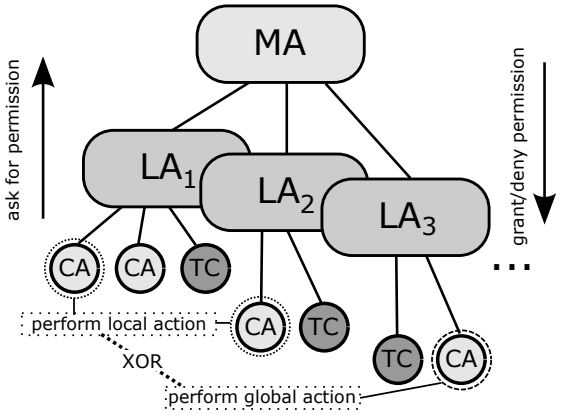


Figure 2.7. iEMAS management structure

LA_i (see Pseudocode 2.3.4) receives signals containing actions to be performed from all its agents. Then chooses one CA which should try to perform its action. This action is reported to MA and after receiving permission, the CA can perform the action. All other agents are stopped from performing their actions.

MA (see Pseudocode 2.3.3) waits for all requests from location and then chooses randomly one location. If this location asks for permission to perform a global action, then permission is granted this and all other locations are rejected. Otherwise all locations which asked for the permission to perform global action are rejected and all those asking for permission to perform local action are granted.

Pseudocode 2.3.1: COMPUTATIONAL AGENT'S ALGORITHM

while true

$$\begin{cases} reply \leftarrow 0; \alpha \leftarrow \underline{\omega(x, gen)}(Act) \\ send(LA_i, \alpha); \underline{b\_receive}(LA_i, rep) \\ \textbf{if } rep \textbf{ and } \underline{\delta_\alpha^{gen}(x)}(\{0,1\}) \\ \quad \textbf{then } x_{next} \leftarrow \underline{\vartheta_\alpha^{gen}(x)}(X) \\ send(LA_i); \underline{b\_receive}(LA_i) \end{cases}$$

Pseudocode 2.3.2: LYMPHOCYTE'S ALGORITHM

while true
- $reply \leftarrow 0$
- $\alpha \leftarrow \underline{\varphi(x, tc_{gen})}(Tcact)$
- $send(LA_i, \alpha)$
- $b\_receive(LA_i, rep)$
- **if** rep **and** $\underline{\delta_{\alpha}^{gen}(x)}(\{0,1\})$
  - **then** $x_{next} \leftarrow \underline{\eta_{\alpha}^{gen}(x)}(X)$
- $send(LA_i)$
- $b\_receive(LA_i)$

Pseudocode 2.3.3: MASTER AGENT'S ALGORITHM

while true
- $local \leftarrow \{i : i \in [1, s]\}$
- $localloc \leftarrow \emptyset$
- $localglob \leftarrow \emptyset$
- $act \leftarrow 0$
- $rep \leftarrow 0$
- **for each** $j \in local$
  - **do**
    - $b\_receive(j, act)$
    - **if** $act \in \{Act_{gl} \cup Tcact_{gl}\}$
      - **then** $localglob \leftarrow localglob \cup \{j\}$
      - **else** $localloc \leftarrow localloc \cup \{j\}$
- $lchosen \leftarrow \underline{locsel(x)}(Loc)$
- **if** $lchosen \in \underline{localglob}$
  - **then**
    - $send(lchosen, 1)$
    - **for each** $j \in (local \setminus \{lchosen\})$ **do** $send(j, 0)$
  - **else**
    - **for each** $j \in localloc$ **do** $send(j, 1)$
    - **for each** $j \in localglob$ **do** $send(j, 0)$
- **for each** $j \in local$ **do** $b\_receive(j)$
- **for each** $j \in local$ **do** $send(j)$

Pseudocode 2.3.4: LA_i ALGORITHM

while true

$localgen \leftarrow \{(j,k) \in U \times P_i;\ x(i,j,k) > 0\}$
$localtc \leftarrow \{U \ni j : tcince(i,j) > 0\} genact \leftarrow hashmap(U \times P_i, Act)$
$tcact \leftarrow hashmap(U \times P_i, Tcact)$
$act \leftarrow 0$
$reply \leftarrow 0$
if $\#\{localgen \cup localtc\} = 0$

then
- $send(MA, null)$
- $b\_receive(MA)$
- $send(MA, null)$
- $b\_receive(MA)$

else
- **for each** $g \in localgen$ **do**
  - $b\_receive(g, act)$
  - $genact[g] \leftarrow act$
- **for each** $g \in localtc$ **do**
  - $b\_receive(g, act)$
  - $tcact[g] \leftarrow act$
- **if** typesel(x)
  - **then** $gchosen \leftarrow \underline{agsel_i(x)}(Act)$
- $REPORT(genact[\overline{gchosen}], gchosen)$
  - **else** $gchosen \leftarrow \underline{tcsel_i(x)}(Tcact)$
- $REPORT(tcact[\overline{gchosen}], gchosen)$

function REPORT$(act, chosen)$
$send(MA, act)$
$b\_receive(MA, reply)$
if $reply$
  then $send(chosen, 1)$
  else $send(chosen, 0)$
for each $g \in (localgen \setminus chosen)$ **do** $send(g, 0)$
for each $g \in localgen$ **do** $b\_receive(g)$
$send(MA)$
$b\_receive(MA)$
for each $g \in localgen$ **do** $send(g)$

2.4. Towards verification of EMAS

As stated in Sections 2.2.1 and 2.3, agent-based metaheuristics, such as EMAS and iEMAS have already proved to be good techniques for solving difficult search problems (see, e.g. [22, 169, 54, 20, 23, 21] to point out a few). However, in order to make sure that such a complex tool may be useful, both formal analysis and experimental verification should be conducted.

The model presented here was not only constructed for the sole purpose of EMAS, but also to constitute a framework that may be further used to describe other computing systems (therefore, EMAS model was extended in order to support iEMAS). Research may be continued, and other similar computing systems may be defined using this style of modelling.

Using these models for the sole purpose of definition of computing systems will not make it possible utilise its descriptive power to the full extent. In the next chapter, a detailed analysis of EMAS stochastic features based on the model presented in this chapter is given. The Markov-chain modelling EMAS has turned out to be ergodic, which gives for this class of systems asymptotic guarantee of success (cf. [150, 162]). For iEMAS, appropriate ergodic conjecture is given and the proof is outlined.

The formal model of EMAS together with the ergodicity proof may be seen as unique phenomenon in the field of computing, as there is a lack of a comprehensive stochastic model of the wide class of population-based, general evolutionary or memetic algorithms.

At the same time, a detailed experimental analysis of EMAS is required to give a perceptible proof of correctness of the advocated computing approach. A series of experiments were conducted and the outcome is discussed in detail in Chapter 4, which shows that these systems are capable of solving difficult problems. It has also been proved that agent-based computing is better than classical approaches when applied to selected problems.

3. Formal aspects of agent-based metaheuristics

In this chapter a formal model for EMAS and iEMAS dynamics based on Markov chains is presented. This way of modelling is inspired by the works of Vose [187].

Byrski, Schaefer and Smołka [29, 163] proposed a construction of a Markov-chain formal model for EMAS and iEMAS utilising the continuous space of states. In these works, a globally synchronised and parallel version of the system were described.

After detailed analysis of EMAS dynamics, consisting in formulation of a Markovian transition function for the discrete model of EMAS is given. Next, the necessary conditions for feasible space of states are formulated, along with the ergodicity theorem for EMAS and its formal proof (recalled after Byrski, Schaefer, Smołka, Cotta [25]). The technical details of this proof may be found in Appendix B.

Later, the Markovian transition function for the discrete model of iEMAS is given, along with ergodic conjecture and the outline of its proof (recalled after Byrski, Schaefer, Smołka [26]).

Various types of convergence of the solutions of the global optimisation problem can be verified using the approach presented here, especially the probabilistic guarantee of success (see, e.g. [150, 162]) is an outcome of the ergodicity of the model. This proof marks so-called asymptotic guarantee of success [150, 162] that may also be associated with probabilistic completeness defined by Hoos and Stützle [91].

3.1. Formal analysis of EMAS

Following the concept of EMAS and its structural model provided in Section 2.2.1, discussion on asymptotic features of this system is given here. Starting with a detailed description of the system transition function, the ergodicity theorem is formulated and proved [25].

3.1.1. EMAS dynamics

At every observable moment at which EMAS has the state $x \in X$, all CA in all locations notify their LA_i of their intent to perform an action. All LA_i choose an agent using the distribution given by the function $agsel_i(x)$ and then notify the Master Agent of their intent to let one of their agents perform an action. MA chooses the location with the probability distribution $locsel(x)$.

The probability that in the chosen location $i \in Loc$ the agent wants to perform a local action is as follows:

$$\xi_i(x) = \sum_{gen \in U} \sum_{n \in P_i} agsel_i(x)(gen, n) \cdot \omega(x, gen)(Act_{loc}) \tag{3.1}$$

The probability that MA will choose the location with the agent which intends to perform a local action is:

$$\zeta^{loc}(x) = \sum_{i \in Loc} locsel(x)(i) \cdot \xi_i(x) \tag{3.2}$$

Of course, the probability that MA will choose the global action is:

$$\zeta^{gl}(x) = 1 - \zeta^{loc}(x) \tag{3.3}$$

If the global action has been chosen then the probability of passing from the state $x \in X$ to $x' \in X$ can be computed using Bayes rule as the sum over all possible sampling results:

$$\tau^{gl}(x)(x') = \sum_{i \in Loc} locsel(x)(i)$$
$$\left(\sum_{gen \in U} \sum_{n \in P_i} agsel_i(x)(gen, n) \cdot \left(\sum_{\alpha \in Act_{gl}} \omega(x, gen)(\alpha) \cdot \varrho_{\alpha}^{gen,n}(x)(x') \right) \right) \tag{3.4}$$

Now, a set of action sequences containing at least one local action is defined:

$$Act_{+1loc} = \left\{ (\alpha_1, \ldots, \alpha_s) \in Act^s;\ \sum_{i=1}^{s} [\alpha_i \in Act_{loc}] > 0 \right\} \tag{3.5}$$

The following family of coefficients is defined $\{\mu_{\alpha_i, gen_i, n_i}(x)\},\ i \in Loc,\ gen_i \in U,\ n_i \in P_i,\ x \in X$. If the location i is non-empty in the state x, then $\mu_{\alpha_i, gen_i, n_i}(x)$ is equal to the probability that the agent ag_{gen_i, n_i} residing in the i-th location will choose action α_i:

$$\mu_{\alpha_i, gen_i, n_i}(x) = agsel_i(x)(gen_i, n_i) \cdot \omega(x, gen_i)(\alpha_i). \tag{3.6}$$

Of course, $\mu_{\alpha_i,gen_i,n_i}(x) = 0$ if the agent ag_{gen_i,n_i} does not exist in the location i in the state x, because $agsel_i(x)(gen_i, n_i) = 0$ in this case. Moreover, $\mu_{\alpha_i,gen_i,n_i}(x) = 1$, if the location i is empty in the state x. Next, the multi-index is introduced:

$$ind = \big(\alpha_1, \ldots, \alpha_s; (gen_1, n_1), \ldots, (gen_s, n_s)\big) \in IND = Act^s_{+1loc} \times \prod_{i=1}^{s} (U \times P_i). \tag{3.7}$$

The probability that in the state x, agents ag_{gen_i,n_i} will choose the actions α_i in consecutive locations is given by:

$$\mu_{ind}(x) = \prod_{i=1}^{s} \mu_{\alpha_i,gen_i,n_i}(x) \tag{3.8}$$

Similarly to the previous case the probability of passing from the state $x \in X$ to $x' \in X$ for the parallel system can be computed using Bayes' rule as the sum over all possible sampling results:

$$\tau^{loc}(x)(x') = \sum_{ind \in IND} \mu_{ind}(x)(\pi_1^{ind} \circ , \ldots, \circ \pi_s^{ind})(x)(x'), \tag{3.9}$$

where

$$\pi_i^{ind}(x) = \begin{cases} \varrho_{\alpha_i}^{gen_i,n_i}(x), & \alpha_i \in Act_{loc} \quad \text{and} \quad i \text{ is non-empty} \\ \vartheta_{null}, & \alpha_i \in Act_{gl} \quad \text{or} \quad i \text{ is empty.} \end{cases} \tag{3.10}$$

The definition of the coefficient $\mu_{\alpha_i,gen_i,n_i}(x)$ and the above formula (3.10) show in particular that the action $null$ is executed in every location instead of a selected global action, and formally in all empty locations.

It is easy to see that the value of $(\pi_1^{ind} \circ , \ldots, \circ \pi_s^{ind})(x)(x')$ does not depend on the composition order because transition functions associated with local actions commutate pairwise (see Corollary 2.2.1). It validates the following observation.

Observation 3.1.1 ([25, Observation 5.1])**.** *The probability transition function for the EMAS model is given by the formula*

$$\tau(x)(x') = \zeta^{gl}(x) \cdot \tau^{gl}(x)(x') + \zeta^{loc}(x) \cdot \tau^{loc}(x)(x') \tag{3.11}$$

and Eqs. (3.1)–(3.10).

Observation 3.1.2 ([25, Observation 5.2])**.** *The stochastic state transition of EMAS given by Eq. (3.11) satisfies the Markov condition. Moreover, the Markov chain defined by these functions is stationary.*

Proof. All transition functions and probability distributions given by Eqs. (3.1)–(3.10) depend only on the current state of the system, which motivates the Markovian features of the transition function τ given by (3.11). The transition functions do not depend on the step number at which they are applied, which motivates the stationarity of the chain. □

3.1.2. Ergodicity of EMAS

It is intended to analyse some asymptotic features of the model in order to draw significant conclusions on capabilities of finding the optimum of a given function by EMAS with actions definitions given in 2.2.3.

The actions defined in Section 2.2.3, forming the following sets:

$$\begin{aligned} Act_{loc} = &\quad \{get, repr, clo, lse\}, \\ Act_{gl} = &\quad \{migr\}. \end{aligned}$$

are used in this proof.

Theorem 3.1.1 ([25, Theorem 7.1])**.** *Assume the following conditions hold.*

1. *The migration energy threshold is lower than the total energy divided by the number of locations $e_{migr} < \frac{1}{s}$. This assumption ensures that there will be at least one location in the system in which an agent is capable of performing migration (by gathering enough energy from its neighbours).*

2. *The quantum of energy is lower than or equal to the total energy divided by the maximal number of agents that may be present in the system $\Delta e \leqslant \frac{1}{\sum_{i=1}^{s} q_i}$. This assumption allows to achieve a maximal population of agents in the system.*

3. *The reproduction (cloning) energy is lower than two energy quanta $e_{repr} \leqslant 2\Delta e$.*

4. *The amount of energy passed from parent to child during the action* clo *is equal to Δe (so $n_1 = 1$).*

5. *The maximal number of agents on every location is greater than* 1*, $q_i > 1, i = 1, \ldots, s$.*

6. *Locations are connected to each other, i.e. $Top = Loc^2$.*

7. *Each active agent can be selected by its LA_i with strictly positive probability, i.e.*
   $\exists\, \iota_{agsel} > 0;\ \forall\, i \in Loc, \forall\, gen \in U,$
   $\forall n \in P_i,\ \forall\, x \in \{y \in X; y(i, gen, n) > 0\}, agsel_i(x)(gen, n) \geqslant \iota_{agsel}.$

8. *Families of probability distributions which are parameters of EMAS have uniform, strictly positive lower bounds*

$\exists\, \iota_\omega > 0;\ \forall\, x \in X,\ gen \in U, \alpha \in Act, \omega(gen, x)(\alpha) \geqslant \iota_\omega,$

$\exists\, \iota_{cmp} > 0;\ \forall\, gen, gen' \in U, cmp(gen, gen') \geqslant \iota_{cmp},$

$\exists\, \iota_{mut} > 0; \forall gen, gen' \in U, mut(gen)(gen') \geqslant \iota_{mut},$

$\exists\, 0 < \iota_{locsel} < 1; \forall\, x \in X, \forall\, j \in Loc,\ locsel(x)(j) \geqslant \iota_{locsel}.$

Then the Markov chain modeling EMAS (see Eq. (3.11)*) is irreducible, i.e. the system state may be transformed between any two arbitrarily chosen states* $x_b, x_e \in X$.

Remark 3.1.1 ([25, Remark 7.1])**.** *Note that assumption 7 of Theorem 3.1.1 is reasonable because the number of possible states of the system is finite and so is the number of locations.*

Remark 3.1.2 ([25, Remark 7.2])**.** *The definition of the state space* X *(see Eq.* (2.3.2)*) implies that there already exists at least one computing agent in EMAS and that at least one location is non-empty at any time.*

Because of the complexity, the technical details of the proof are transferred to Appendix B.

Proof of Theorem 3.1.1. It will suffice to show that the transformation between two arbitrary EMAS states ($x_b, x_e \in X$) may be performed in a finite number of steps with the positive probability. The following sequence of stages (see Fig. 3.1) defines such transformation.

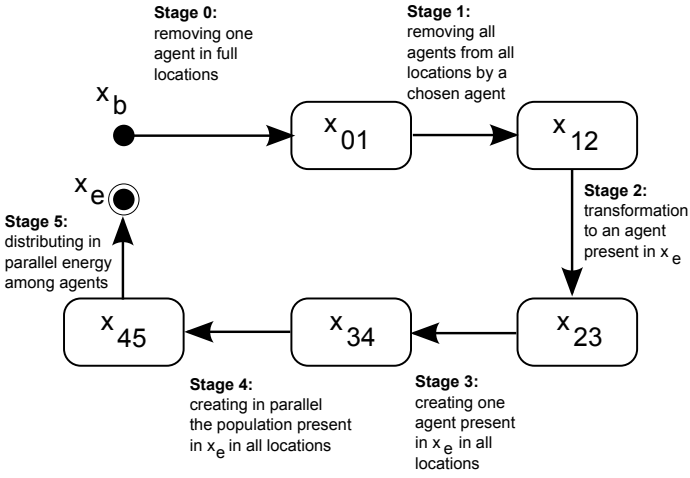


Figure 3.1. State transitions in EMAS irreducibility proof

- **Stage 0**: In every location in parallel:
  - If the location is full, the agent is chosen and it performs sequentially the action get with one of its neighbours in order to remove it (to enable incoming migration from any other location). After removing one of its neighbours the agent tries to perform any global action, e.g. $migr$ (and fails) until the end of the stage.
  - If the location is empty, the trivial null state transition is performed.
  - If the population in the location contains at least one agent, but is not full, this agent also attempts to perform the $migr$ action (but fails to do it) during the whole stage. Final state of the *Stage 0* is denoted by x_{01}.
- **Stage 1**: A single location is chosen, in which the sum of agents' energy exceeds the migration threshold in the state x_{01} (based on assumption 1 of Theorem 3.1.1 there must be at least one agent). Then, the agent ag_{gen_1,n_1} from this location (possibly possessing the largest energy in the state x_{0e}) is chosen. This agent performs a sequence of actions get in order to gather all energy from all its neighbours, finally removing them from the system (by bringing their energy to zero). Now ag_{gen_1,n_1} begins the first migration round in order to visit all locations and to remove the agents (overtaking their energy by performing multiple get actions). This round is finished in the location i_1. Now, the agent possesses the total energy of the system which equals 1. The final state of *Stage* 1 is denoted by x_{12}. Note that the state matrix has only one positive entry $x_{12}(i_1, gen_1, n'_1) = 1$ where $n'_1 \in P_{i_1}$ is the new copy number of the selected agent after all migrations performed during the first round have occurred.
- **Stage 2**: The chosen agent ag_{gen_1,n'_1} performs the clo action producing $ag_{gen_2,n_2}, n_2 \in P_{i_1}$, which is one of the agents present in the state in the location i_2. The location i_2 will contain the total energy greater than the migration threshold e_{migr} in the state x_e. Next, the agent ag_{gen_1,n'_1} passes all its energy to this newly produced agent, finally being removed from the system. The purpose of *Stage 2* is to ensure that the agent recreating the population in the last location i_2 will be one of the agents present in this location in the state x_e.
- **Stage 3**: Next, the agent ag_{gen_2,n_2} begins the second migration round (starting migration from the location i_1), and visits all locations. In every visited location i it performs the clo operation producing one of the agents $ag_{gen_i^{first},n_i^{first}}$ that will be present in this location in the state x_e. In each non-empty location the cloned agent will receive total energy that should be assigned to its location in the state x_e by the sequence of get operations. The agent finishes migration in the location i_2 (one of the islands containing a total energy in the state x_e greater than the migration threshold e_{migr}). For the sake of simplicity the migrating agent

after it reaches the final i_2 location will be further called in the same manner ($ag_{gen_i^{first},n_i^{first}}$).

- **Stage 4**: Every agent $ag_{gen_i^{first},n_i^{first}}$ present in each non-empty location performs in parallel a sequence of cloning actions recreating the population of agents in its location in the state x_e. The total number of parallel steps is not greater than the maximal number of agents in the single location in the state x_e. Some agents may finish recreation earlier, and in this case they will choose the action $migr$ (and fail to perform it) until the end of the stage.
- **Stage 5**: In every location in parallel: the agent active in *Stage 4* performs a sequence of get actions with its neighbours in order to pass them the a sufficient amount of energy required in the state x_e.

It was shown that each of the aforementioned stages requires performing at most a finite number of Markov chain steps. Moreover, it was shown that every aforementioned sequence has non-zero probability. For the details of the these features refer to Appendix B including detailed estimates of the lower probability bounds and upper bounds of the number of steps for every stage of the proof. □

Remark 3.1.3 ([25, Remark 7.3])**.** *Theorem 3.1.1 leads straightforwardly to the statement that each possible state of EMAS is reachable after performing a finite sequence of transitions independently of the initial population. Therefore, also the states containing the extrema are reachable. Thus any metaheuristic respecting EMAS architecture and the assumption of Theorem 3.1.1 satisfies the asymptotic guarantee of success [92, 150].*

Theorem 3.1.2 ([25, Theorem 7.2])**.** *If the assumptions of Theorem 3.1.1 hold, then the Markov chain modelling EMAS is aperiodic.*

Proof. Consider a state of the chain such that each location contains a single agent. In this case let us assume that each agent chooses get as its next action. Because all agents have chosen local actions, MA will allow them all to perform their actions, however, the absence of neighbours will force all the agents to perform the trivial (i.e. null) action. The transition probability function is then the s-fold composition of ϑ_{null}—see Eq. (3.9). Therefore, in this case the system will return to the same state in one step. The probability of such transition is not less than $(\iota_{get})^s > 0$. It means that the state is aperiodic. The chain is irreducible (see Theorem 3.1.1) and therefore has only one class of states, the whole state space, which obviously contains the aperiodic state. On the other hand, from Theorem 2.2 of [94] it is clear that aperiodicity is a state class property. In this case it means that all states of EMAS are aperiodic, which concludes the proof. □

The following corollary is a consequence of Theorems 3.1.1 and 3.1.2.

Corollary 3.1.1 ([25, Corollary 7.1])**.** *The Markov chain modelling EMAS is* ergodic.

Remark 3.1.4 ([25, Remark 7.4])**.** *The Markov chain* (3.11) *is ergodic in its strong sense, namely it is not only irreducible, but also aperiodic. Such chains are often called* regular *(see e.g. [94]). Obviously it is also ergodic in its weaker (and also wide) sense, meaning that it is simply irreducible.*

Since the space of states X is finite we may introduce the probability transition matrix:

$$Q = \{\tau(x)(y)\},\ x, y \in X, \tag{3.12}$$

where τ is the EMAS probability transition function—see Eq. (3.11). The Markov chain describing the EMAS dynamics is a sequence of random variables (or, equivalently, probability distributions) $\{\xi_t\} \subset \mathcal{M}(X), t = 0, 1, \ldots$ where ξ_0 should be a given initial probability distribution. Of course we have that

$$\xi_{t+1} = Q \cdot \xi_t,\ t = 0, 1, \ldots \tag{3.13}$$

Remark 3.1.5 ([25, Remark 7.5])**.** *By Theorems 3.1.1 and 3.1.2 as well as the ergodic theorem [12] there exists a strictly positive limit $\widehat{\xi} \in \mathcal{M}(X)$ (i.e., $\widehat{\xi}(x) > 0, \forall\, x \in X$) of the sequence $\{\xi_t\}$ as $t \to +\infty$. This equilibrium distribution does not depend on the initial probability distribution ξ_0.*

3.2. Formal analysis of iEMAS

Following the concept of iEMAS and its structural model provided in Section 2.2.1 deliberations on asymptotic features of this system are given here. Starting with a detailed derivation of the system transition function, the ergodicity conjecture is formulated and its proof is outlined [26].

3.2.1. iEMAS dynamics

At every observable moment at which EMAS has the state $x \in X$ all agents in all locations notify their LA_i their intent to perform an action, all LA_i choose an agent with the distribution given by the function $agsel_i(x)$ and then notify the MA of their intent to let one of their agents to perform an action. MA chooses the location with the probability distribution $locsel(x)$.

The model of EMAS dynamics is extended here in order to model the behaviour of iEMAS. The probability that in the chosen location $i \in Loc$ the agent or the

lymphocyte wants to perform a local action is:

$$\xi_i(x) = typesel(x)(\{0\}) \sum_{gen\in U} \sum_{n=1}^{p} (agsel_i(x)(\{gen, n\}) \cdot \omega(gen, x)(Act_{loc})) + typesel(x)(\{1\}). \tag{3.14}$$

The probability that MA will chose the location with the agent intending to perform the local action is:

$$\zeta^{loc}(x) = \sum_{i\in Loc} locsel(x)(\{i\})\xi_i(x). \tag{3.15}$$

Of course, the probability that MA will choose the global action is:

$$(1 - \zeta^{loc}(x)) = \zeta^{gl}(x). \tag{3.16}$$

If the global action has been chosen, the state transition is given by:

$$\tau^{gl}(x)(x') = \sum_{i\in Loc} locsel(x)(\{i\}) \cdot \left(\sum_{gen\in U} \sum_{n=1}^{p} agsel(x)(\{gen, n\}) \cdot \left(\sum_{\alpha\in Act_{gl}} \omega(gen, x)(\{\alpha\}) \cdot \varrho_{\alpha}^{gen,n}(x)(x') \right) \right). \tag{3.17}$$

The set of action sequences containing at least one local action is now defined:

$$Act_{+1loc} = \left\{ (\alpha_1, \ldots, \alpha_s) \in (Act \cup Tcact)^s; \sum_{i=1}^{s} [\alpha_i \in (Act_{loc} \cup Tcact)] > 0 \right\} \tag{3.18}$$

The probability that in the location i-th the agent ag_{gen_i,n_i} or the lymphocyte $tc_{\widetilde{gen_i}}$ will choose the action α_i is:

$$\mu_{\alpha_i,gen_i,n_i,\widetilde{gen_i}}(x) = typesel(x)(\{0\}) \cdot agsel_i(x)(\{gen_i, n_i\})\omega(gen_i, x)(\{\alpha_i\}) + typesel(x)(\{1\})tcsel_i(x)(\{\widetilde{gen_i}\})\varphi(\widetilde{gen_i}, x)(\{\alpha_i\}). \tag{3.19}$$

Now, a multi-index is defined:

$$ind = (\alpha_1, \ldots, \alpha_s; (gen_1, n_1), \ldots, (gen_s, n_s); \ (\widetilde{gen_1}), \ldots, (\widetilde{gen_s})) \in IND = (Act \cup Tcact)^s \times (U \times \{1, \ldots, p\})^s \times U^s, \tag{3.20}$$

the probability that in consecutive locations agents ag_{gen_i,n_i} or lymphocytes $tc_{\widetilde{gen_i}}$ will choose the actions α_i is:

$$\mu_{ind}(x) = \prod_{i=1}^{s} \mu_{\alpha_i,gen_i,n_i,\widetilde{gen_i}}(x). \tag{3.21}$$

The transition function for parallel system is the following:

$$\tau^{loc}(x)(x') = \sum_{(\alpha_1,\ldots,\alpha_s)\in Act_{+1loc}} \sum_{ind\in IND} \mu_{ind}(x)(\pi_1^{ind}(x) \circ, \ldots, \circ \pi_s^{ind}(x))(x'), \tag{3.22}$$

where π_i is defined as:

$$\pi_i^{ind}(x) = \begin{cases} \varrho_{\alpha_i}^{gen_i,n_i}(x), & \alpha_i \in Act_{loc} \\ \eta_{\alpha_i}^{\widetilde{gen_i}}(x), & \alpha_i \in Tcact \\ \vartheta_{null}, & \alpha_i \in Act_{gl}. \end{cases} \tag{3.23}$$

The value of $(\pi_1^{ind}(x) \circ, \ldots, \circ \pi_s^{ind}(x))(x')$ does not depend on the composition order, because transition functions associated with local actions commutate pairwise (this feature of iEMAS actions may be proved similarly to the case of EMAS, cf. [25]) . Finally, the following observation may be derived:

Observation 3.2.1 ([26, Observation 1])**.** *The probability transition function for the parallel iEMAS model is given by the formula*

$$\tau(x)(x') = \zeta^{gl}(x)\tau^{gl}(x)(x') + \zeta^{loc}(x)\tau^{loc}(x)(x') \tag{3.24}$$

and formulas (3.14)–(3.23).

Observation 3.2.2 ([26, Observation 2])**.** *The stochastic state transition of iEMAS given by formula (3.24) satisfies the Markov condition.*

Proof. All transition functions and probability distributions given by formulas (3.14)–(3.23) depend only on the current state of the system, which motivates the Markovian features of the transition function τ given by (3.24). The transition functions do not depend on the step number at which they are applied, which motivates the stationarity of the chain. □

3.2.2. Ergodicity of iEMAS

In this section, an ergodic conjecture of the Markov chain describing the behaviour of iEMAS is presented. A sequence of proof stages is also given, it is shown that after estimating the upper bounds of their steps and lower bounds for their probabilities may become a full formal proof of iEMAS ergodicity in much the same way as in Section 3 and Appendix B for EMAS [26].

The system which uses the following actions (they may be defined in details in similar way as it is shown in Section 2.2.3), is considered in this section:

- $repr$, clo, lse, $migr$ – these actions are inherited unchanged from EMAS.
- get – this action is modified. When the agent performs this action, and its energy (or energy of evaluated agent) reaches zero, it activates the lymphocyte containing the genotype of the inactivated agent.
- $give$ – this is an action executed solely by lymphocytes. It is performed every time the lymphocyte is activated and decreases the lymphocyte's energy, which makes the lymphocyte be deactivated (when its energy reaches zero).
- $kill$ – another lymphocyte's action, removing (or penalising) the computing agent (performed when the genotype of the tested agent matches the pattern contained in the lymphocyte).

Features of these actions, in particular if they are global or local, may be proved similarly as in 2.2.3, resulting in the following actions taxonomy:

$$\begin{aligned} Act_{loc} = &\quad \{get, repr, clo, lse, give, kill\}, \\ Act_{gl} = &\quad \{migr\}. \end{aligned}$$

Conjecture 3.2.1 ([26, Theorem 1])**.** *Assume that the following assumptions hold.*

1. *The migration energy threshold is lower than the total energy divided by the number of locations $e_{migr} < \frac{1}{s}$. This assumption ensures that there will be at least one location in the system in which an agent is capable of performing migration (by gathering enough energy from its neighbours).*
2. *The quantum of energy is lower than or equal to the total energy divided by the maximal number of agents that may be present in the system $\Delta e \leqslant \frac{1}{\sum_{i=1}^{s} q_i}$. This assumption makes it possible to achieve a maximal population of agents in the system.*
3. *Reproduction (cloning) energy is lower than two energy quanta $e_{repr} \leqslant 2\Delta e$.*
4. *The amount of energy passed from parent to child during a cloning action is equal to Δe (so $n_1 = 1$).*
5. *The maximal number of agents in every location is greater than one, $q_i > 1, i = 1, \ldots, s$.*

6. *Locations are all connected, i.e.* $Top = Loc^2$.
7. *Each active agent can be selected by its* LA_i *with strictly positive probability.*
8. *The families of probability distributions which are the parameters of EMAS have uniform, strictly positive lower bounds.*

Then the Markov chain modelling iEMAS (see equation (3.24)) is irreducible, i.e. all its states communicate.

Proof outline 3.2.1 ([26, Section 7])**.** *In order to prove Conjecture 3.2.1, it is enough to show that the passage from* x_b *to* x_e *(two arbitrarily chosen states from* X*) may be performed in a finite number of steps with the probability strictly greater than zero (see Fig. 3.2).*

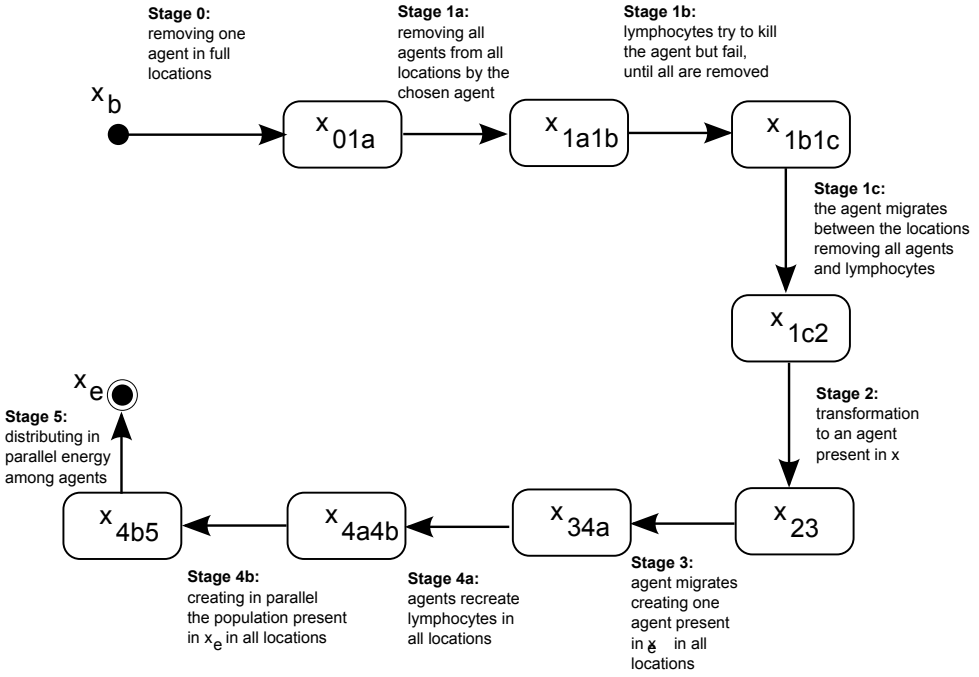


Figure 3.2. State transitions in iEMAS irreducibility proof outline

Consider the following sequence of stages [26].

- ***Stage 0****: In every location in parallel: If the location is full, an agent is chosen, and it performs sequentially an evaluation action with one of its neighbours in order to remove it (to make incoming migration possible from any other location, in case this location is full). After removing one of its neighbours the agent tries to perform any global action, e.g. migration (and fails), until the end of the stage.*

Otherwise, the trivial null state transition is performed. Final state of the Stage 0 is denoted by x_{0e}.

- ***Stage 1 a****: One location is chosen, at which the sum of agents' energy exceeds the migration threshold in the state* x_{0e} *(based on assumption 1 of Theorem 3.2.1 there must be at least one). Then one agent from this location* ag_{gen_1,n_1} *(possibly with the largest energy in the state* x_{0e}*) is chosen. This agent performs a sequence of evaluation actions in order to gather all energy from all its neighbours, finally removing them from the system (by bringing their energy to zero).*
- ***Stage 1 b****: If there are any lymphocytes in the current location, they perform killing action, one by one, on the agent* ag_{gen_1,n_1}*, but fails to remove it from the system until all lymphocytes are removed. In the end, only one agent is present in the location.*
- ***Stage 1 c****: Now this agent begins the first migration round in order to visit all locations and remove the agents (overtaking their energy by performing multiple* get *actions) and all lymphocytes. This round is finished at the location* i_1*. Now, agent* ag_{gen_1,n_1} *possesses the total energy of the system which equals* 1*. Final state of Stage 1 is denoted by* x_{1e}*. Note that the state matrix has only one positive entry* $x_{1e}(i_1, gen_1, n_1) = 1$.
- ***Stage 2****: The agent performs a cloning action producing one of the agents (*ag_{gen_2,n_2}*) that will be present in the location* i_2*, which is one of the locations in the state* x_e *containing the total energy greater than the migration threshold. Having passed all of its energy to this new agent, it is finally removed from the system. The purpose of Stage 2 is to ensure that the agent recreating the population in last location* i_2 *will be one of the agents present in this location in the state* x_e*. Otherwise, if* i_2 *is full in the state* x_e*,* ag_{gen_1,n_1} *cannot recreate this population. If* ag_{gen_1,n_1} *is active in the location* i_2 *at the state* x_e *(i.e.* $x_e(i_2, gen_1, n_1) > 0$*), Stage 2 may be omitted (in this case* ag_{gen_1,n_1} *takes the role of* ag_{gen_2,n_2} *in the consecutive stages).*
- ***Stage 3****: Next, the agent* ag_{gen_2,n_2} *begins the second migration round (starting migration from the location* i_1*) and visits all locations. In each visited location it performs a cloning action and produces one of the agents that will be present in this location in the state* x_e*. The cloned agent in each non-empty location (denoted by* $ag_{gen_i^{first},n_i^{first}}$*) will receive the total energy that should be assigned to its location, by the sequence of evaluation actions. The agent finishes migration after recreating the population in the location* i_2 *(one of the islands containing the total energy in the state* x_e *greater than the migration threshold).*
- ***Stage 4 a****: In the system, the following sequence of actions assigned with consecutive locations labelled* $i \in Loc$*, non-empty in the state* x_e*, is performed:*

each agent $ag_{gen_i^{first},n_i^{first}}$ performs a cloning action to produce an agent with a genotype of one of the lymphocytes existing in the location in the state x_e. Now it performs a sequence of evaluation actions to remove the created agent, so the desired lymphocyte is produced. The lymphocyte performs a sequence of energy lowering actions to adjust its energy to the level observed in the state x_e. This is repeated until all the lymphocytes present in x_e are recreated.

- ***Stage 4 b**: In the system, the following sequence of actions assigned with the consecutive locations labelled $i \in Loc$, non-empty in the state x_e, is performed: every agent $ag_{gen_i^{first},n_i^{first}}$ performs a sequence of cloning actions, recreating the population of agents in its location in the state x_e.*
- ***Stage 5**: In every location in parallel: agent $ag_{gen_i^{first},n_i^{first}}$ performs a sequence of evaluation actions with its neighbours in order to pass to them a sufficient amount of energy, required in the state x_e.*

Assuming that Conjecture 3.2.1 is true, similarly to the EMAS case, the following features may be proved, which lead directly to the statement that every possible state of iEMAS is reachable (with positive probability) after performing a finite sequence of transitions independently of the initial population.

1. All states containing the extrema are reachable from an arbitrary initial state. Thus iEMAS satisfies asymptotic guarantee of success in the sense of [162, 92, 150] [26, Corollary 1].
2. If the assumptions of Theorem 3.2.1 hold, then the Markov chain modelling EMAS is aperiodic [26, Theorem 2].
3. As a consequence of features 1 and 2, the Markov chain modelling EMAS is *ergodic* [26, Corollary 2].
4. It is noteworthy that the Markov chain (3.24) is ergodic in its strong sense (not only irreducible but also aperiodic). Such chains are quite often called *regular* (see e.g. [94]) [26, Remark 1].
5. Since the space of states X is finite the probability transition matrix is introduced:
$$Q = \{\tau(x)(y)\},\ x, y \in X, \tag{3.25}$$
where τ is the iEMAS probability transition function (see Eq. (3.24)). The Markov chain describing the iEMAS dynamics is a sequence of random variables (or, equivalently, probability distributions) $\{\xi_t\} \subset \mathcal{M}(X), t = 0, 1, \ldots$, where ξ_0 should be a given initial probability distribution. Of course, the following condition holds:
$$\xi_{t+1} = Q \cdot \xi_t,\ t = 0, 1, \ldots \tag{3.26}$$

6. By Theorems 3.2.1 and feature 4, as well as the ergodic theorem [12] there exists a strictly positive limit $\widehat{\xi} \in \mathcal{M}(X)$ (i.e., $\widehat{\xi}(x) > 0, \forall\, x \in X$) of the sequence $\{\xi_t\}$ as $t \to +\infty$. This equilibrium distribution does not depend on the initial probability distribution ξ_0 [26, Remark 2].

3.3. Goals attained in formal analysis

The theoretical results obtained in this research, especially the most important feature of ergodicity proved for EMAS (including its memetic versions) and iEMAS, are crucial for studying features of stochastic global optimisation metaheuristics.

The strong ergodicity of the finite state Markov chain modelling the metaheuristics shows that these systems can reach an arbitrary state (arbitrary population) in the finite number of iteration with the probability equal to 1 which implies the asymptotic stochastic guarantee of success (see 3.1.3).

The formal framework constructed made it possible to analyse similar systems (e.g. HGS [160] or parallel version of Vose's algorithm [161]) and will be considered for future analysis of other novel and classical computing systems (e.g. in the near future, an analysis of evolutionary algorithm using tournament selection is envisaged).

It is worth noting that apart from formal models focused on particular aspects of population-based metaheuristics (mostly Simple Genetic Algorithm [187, 191] and selected Evolution Strategies [156, 154]), there are no similar approaches to modelling such complex computing systems, as agent-based ones. It seems to be one of the greatest advantages of the research presented in this monograph.

4. Experimental verification of EMAS

In this chapter, selected experiments with a detailed description of the researched problems and discussion of the results are presented. A wider selection of experimental results, along with a description of experimental configurations may be found in Appendix A. The experimental results were obtained using a dedicated software framework, which was implemented on the basis of experience gained in developing computing and simulation agent-based frameworks [60]. Based on this framework, a series of experiments was conducted, considering efficiency verification of EMAS and PEA, their memetic and immunological variations, using selected high-dimensional benchmark functions. Later the influence of different parameters on the computing results is examined, and finally, the overall efficiency of these systems is tested, using selected real-world problems.

In captions of the graphs, short symbolic explanation of the presented function is given in an easily understandable function-like form, e.g. when a graph presents the best fitness value in the subsequent steps, the caption contains the following: $bestFitness(step)$

4.1. EMAS in solving benchmark problems

Considering the "no free lunch theorem" [193], it is still important to try to test how the examined metaheuristic works when applied to different well-known problems, i.e., benchmark functions [49].

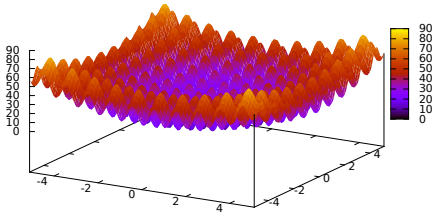
In order to obtain plausible results, the systems compared should be parametrised in the most similar way. So it is the case presented in this section, as EMAS, and its memetic variants (Baldwinian and Lamarckian) are compared with PEA (along with respective memetic modifications). Later, a comparison between EMAS and its immunological variant (iEMAS) is made. Finally two real-world problems are tackled: optimisation of parameters of game advisory strategy and step and flash imprint lithography inverse problem.

4.1.1. Definition of benchmark problems

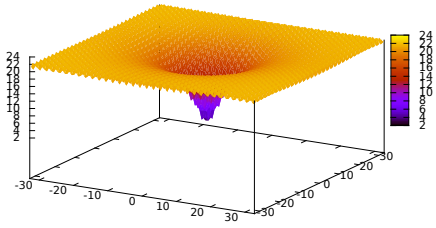
The continuous benchmark functions considered in this monograph were selected from the set described in [49], their visualisations are presented in Figure 4.1:

- Rastrigin:
  $f(x) = 10 \cdot n + \sum_{i=1}^{n}(x_i^2 - Acos(2\pi x_i))$
  $-5.12 \leq x_i \leq 5.12$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$, see Fig. 4.1a.
- Ackley:
  $f(x) = -a \cdot e^{-b\frac{\sqrt{\sum_{i=1}^{n} x_i^2}}{n}} - e^{\frac{\sum_{i=1}^{n} cos(c \cdot x_i)}{n}} + a + e;$
  $a = 20, b = 0.2, c = 2 \cdot \pi, i \in [1, n],$
  $-1 \leq x_i \leq 1,$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$, see Fig. 4.1b.
- De Jong:
  $f(x) = \sum_{i=1}^{n} x_i^2$
  $-5.12 \leq x_i \leq 5.12,$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$, see Fig. 4.1c.
- Rosenbrock:
  $f(x) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2$
  $-2.048 \leq x_i \leq 2.048,$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$, see Fig. 4.1d.
- Schwefel:
  $f(x) = \sum_{i=1}^{n} -x_i \cdot sin(\sqrt{|x_i|})$
  $-500 \leq x_i \leq 500,$
  global minimum: $f(x) = -n \cdot 418.9829, x_i = 420.9678, i \in [1, n]$, see Fig. 4.1e.
- Axis Parallel Hyperellipsoid:
  $f(x) = \sum_{i=1}^{n} i \cdot x_i^2$
  $-5.12 \leq x_i \leq 5.12,$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$, see Fig. 4.1f.
- Moved Axis Parallel Hyper Ellipsoid:
  $f(x) = \sum_{i=1}^{n} 5 \cdot i \cdot x_i^2$
  $-5.12 \leq x_i \leq 5.12,$
  global minimum: $f(x) = 0, x_i = 5 \cdot i, i \in [1, n]$, see Fig. 4.1g.

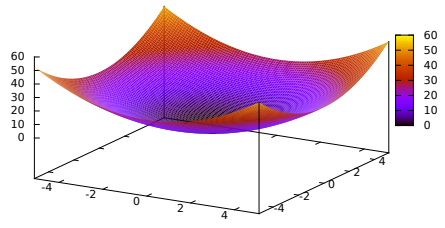
All of the selected functions are multimodal or deceptive (where gradient information may lead the search astray), except for De Jong benchmark that is a convex function. This selection of course does not deplete the benchmarks, however, it seems to present a subjective, but reasonable set of reference problems, frequently used in testing population-based metaheuristics.



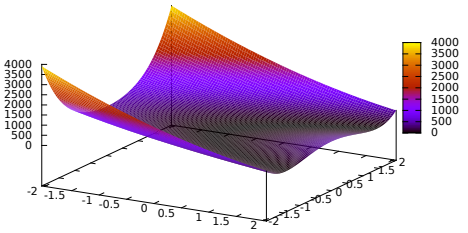
(a) Rastrigin



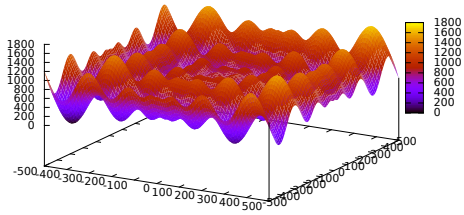
(b) Ackley



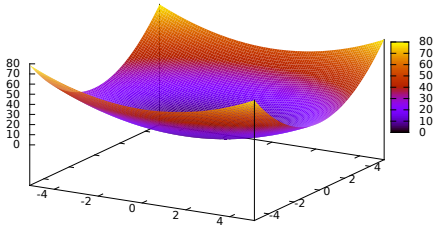
(c) De Jong



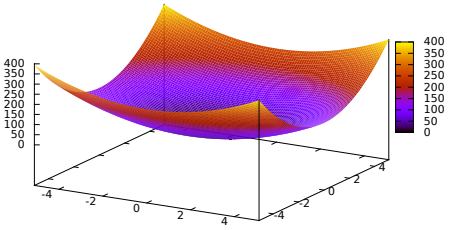
(d) Rosenbrock



(e) Schwefel



(f) Axis parallel hyperellipsoid



(g) Moved axis parallel hyperellipsoid

Figure 4.1. Visualisation of the 2-dimensional cases of continuous benchmark functions used

4.1.2. Classical EMAS and PEA

Here, before going to more sophisticated versions of these systems, the comparison results of classical EMAS with PEA are given. It is to note that PEA was implemented according to a scheme proposed by Michalewicz [131], namely population initialisation, and looping through evaluation, selection, crossover and mutation of the individuals. In the discussed case real-valued encoding is used and allopatric speciation is introduced according to so-called island-model of evolution [30]. The problem considered in this section is optimisation of 100-dimensional Rastrigin benchmark for several different configurations (after Pisarski, Rugała, Byrski and Kisiel-Dorohiniki [145]).

In Figure 4.2, a simple comparison of the computation results, obtained for one-population configuration (1 island, 40 individuals), namely observation of the best fitness in each step of computation may be seen.

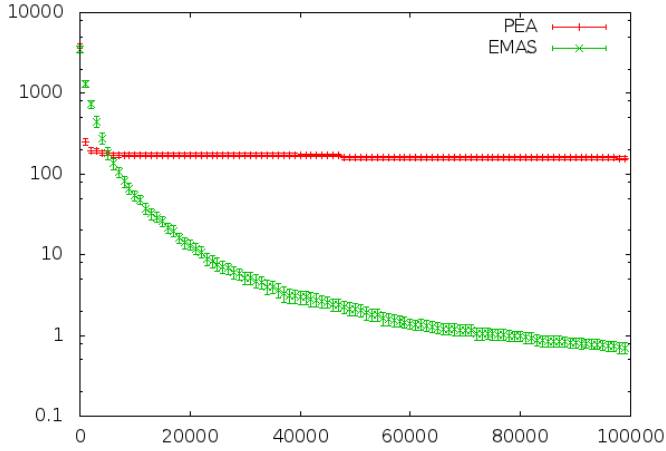


Figure 4.2. EMAS vs. PEA (1 island, 40 agents/individuals) $bestFitness(step)$

It turns out that EMAS outperforms PEA for over two orders of magnitude. This is a very promising result, and it may be further verified by checking the results obtained in both experiments in the last step presented in Table 4.1.

In this table, the results obtained for other configurations (3 islands, 40 individuals) are also presented. In these cases, the domination of EMAS is still retained. It is easy to see that the results are repeatable (as standard deviation is relatively low).

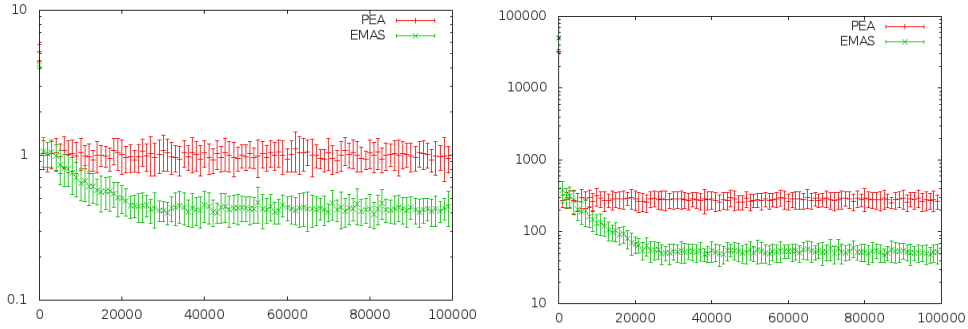
Another important observation is that the multi-population models of computation tends to be better than single-population ones, as the former have better capabilities of exploration (as the population is decomposed to subpopulations), still retaining the capability of exploitation (in each single subpopulation). It should also be noted

Table 4.1. EMAS and PEA optimisation results obtained for 25 and 40 individuals

Number of islands	1 island		3 islands	
Computing system	EMAS	PEA	EMAS	PEA
25 individuals				
Result	1.77	242.96	0.71	156.34
St. Dev.	0.22	6.54	0.10	6.84
St. Dev. %	12.24	2.69	14.46	4.38
40 individuals				
Result	0.90	180.48	0.37	111.45
St. Dev.	0.13	6.45	0.05	4.16
St. Dev. %	15.24	3.57	13.56	3.73

that increasing the number of individuals (from 25 to 40) improved the final result (although this requires a further proving and many more experimental runs).

In Figures 4.3a and 4.3b, diversity computed according to MSD and MOI schemes are shown. It is easy to see that EMAS has lower diversity than PEA in both cases. However, it turns out that it does not hamper the computing efficiency (as EMAS outperforms PEA). Moreover, diversity, though lower, it is still quite stable (see standard deviation range marked on the graphs), which leads to the conclusion that though the population in EMAS is not so diverse as in PEA, the exploration and exploitation features of the system are balanced.



(a) $divMSD(step)$ **(b)** $divMOI(step)$

Figure 4.3. Diversity of EMAS and PEA computed according to MOI and MSD schemes, 1 island 40 agents/individuals with standard deviation

The final values of both diversity measures (MOI and MSD) shown in Tables 4.2, 4.3 confirm the observation of the computation behaviour shown in Figures 4.3a and 4.3b.

Table 4.2. EMAS and PEA MOI diversity for 25 and 40 individuals

Number of islands	1 island		3 islands	
Computing system	EMAS	PEA	EMAS	PEA
25 individuals				
Result	0.56	0.99	0.65	1.31
St. Dev.	0.22	0.14	0.29	0.14
St. Dev. %	38.67	14.04	44.05	10.87
40 individuals				
Result	0.50	1.00	0.57	1.30
St. Dev.	0.23	0.14	0.30	0.15
St. Dev. %	45.01	13.84	52.35	11.16

Table 4.3. EMAS and PEA MOI diversity for 25 and 40 individuals

Number of islands	1 island		3 islands	
Computing system	EMAS	PEA	EMAS	PEA
25 individuals				
Result	77.51	287.63	271.64	1253.43
St. Dev.	903.60	1004.92	2883.95	3000.84
St. Dev. %	1165.86	349.37	1061.69	239.41
40 individuals				
Result	135.22	313.53	460.94	1340.27
St. Dev.	1590.73	1030.57	4962.78	3066.69
St. Dev. %	1176.37	328.69	1076.66	228.81

In order to examine the dynamics of the computing process for PEA and EMAS, an average number of steps between subsequent improvements of the best fitness observed are presented in Table 4.4. It is easy to see that also in this case, EMAS outperforms PEA. This result confirms that lower diversity of EMAS as compared to PEA does not hamper the capability of improving the result of the computation.

Another confirmation of the above observation may be found when looking at Table 4.5. There, a maximal number of steps needed for improving the value of the fitness function are shown. It is easy to see that EMAS outperforms PEA also in this case.

Table 4.4. Average number of steps between subsequent improvements of the best fitness for 25 and 40 individuals

Number of islands	1 island		3 islands	
Computing system	EMAS	PEA	EMAS	PEA
25 individuals				
Result	116.90	133.74	142.61	195.33
St. Dev.	4.26	72.37	5.12	72.90
St. Dev. %	3.64	54.12	3.59	37.32
40 individuals				
Result	119.07	166.13	147.16	234.57
St. Dev.	4.43	87.15	5.97	77.02
St. Dev. %	3.72	52.46	4.05	32.84

Table 4.5. Maximal number of steps between subsequent improvements of the best fitness for 25 and 40 individuals

Number of islands	1 island		3 islands	
Computing system	EMAS	PEA	EMAS	PEA
25 individuals				
Result	4042.37	27441.57	3940.23	42225.17
St. Dev.	1580.74	17264.97	1330.81	17906.04
St. Dev. %	39.10	62.92	33.77	42.41
40 individuals				
Result	3850.93	33019.87	3596.30	47966.66
St. Dev.	1173.24	23218.82	889.32	20118.41
St. Dev. %	30.47	70.32	24.73	41.94

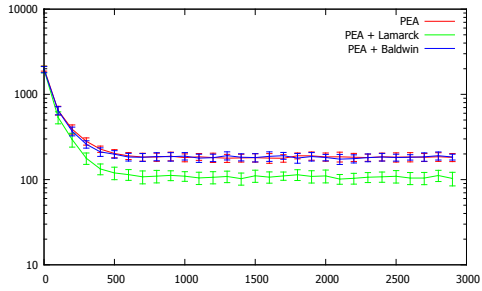
After checking that classical EMAS may become a reliable weapon of choice to deal with a difficult, e.g. black-box problem, it is time to consider other versions of EMAS and PEA capable of bringing substantial improvements to the computing process (e.g. balancing exploration and exploitation capabilities or reducing the number of fitness function calls).

4.1.3. Memetic EMAS and PEA

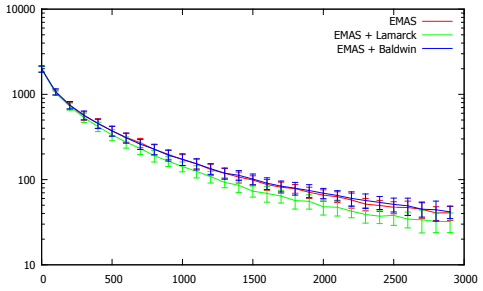
The first and the most important thing to consider is the efficiency of the systems being compared, measured with the classical means (after Byrski, Korczyński and Kisiel-Dorohinicki [24]). Therefore, the fitness value of the best individual reached in certain generation (PEA) compared to a certain step of computation (EMAS) was examined. In Fig. 4.4, both PEA and EMAS fitnesses are shown, for all variants of these systems (evolutionary and memetic ones). The problem considered in this graph was 50-dimensional Rastrigin benchmark function.

It is easy to see that EMAS turns out to be generally better than PEA at evading the local extrema and continuing the exploration, while PEA is apparently already stuck. Drawing comparison between the memetic and evolutionary versions of the systems shows that Lamarckian memetics improves both PEA and EMAS, while Baldwinian memetics in the case considered produces results quite similar to the ones produced by the basic evolutionary versions.

Memetic operators It may be quite interesting to check how the application of particular memetic operators changes the outcomes of the experiments. In the given experiments, gradient-free steepest descent algorithm based on choosing the best one from 10 potential mutated individuals was used. Such a procedure was repeated 10 times and the best up-to-date result was returned. The results of the experiments are shown in Figure 4.5.

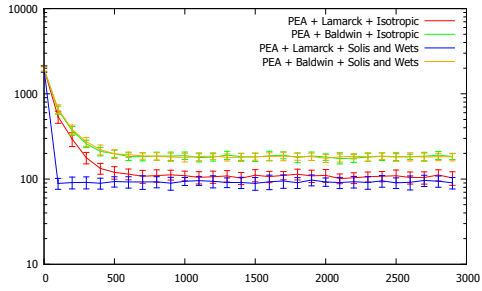


(a) PEA $bestFitness(step)$

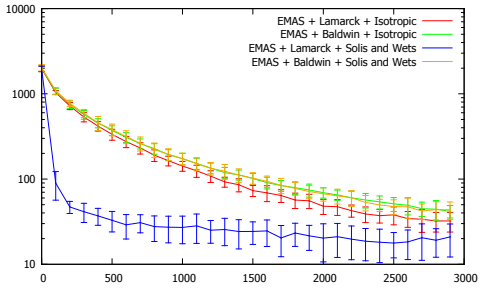


(b) EMAS $bestFitness(step)$

Figure 4.4. PEA and EMAS (memetic and evolutionary versions) fitness for 50-dimensional Rastrigin benchmark



(a) PEA *bestFitness(step)*



(b) EMAS *bestFitness(step)*

Figure 4.5. PEA and EMAS fitness for two different memetic operators (Solis Wets and isotropic mutation) applied

The local search algorithms were implemented according to the two following strategies:

- Isotropic mutation—it is a method aimed at generating uniform sampling points on and within N-dimensional hyperspheres. The idea of the Isotropic method algorithm is as follows: firstly the N normal distributed numbers z_i are generated. Then the vectors x are computed by making a projection onto surface by dividing each generated number z_i by $r = \sqrt{\sum_{i=1}^{N} z_i^2}$. Since the z vectors are isotropically distributed, the vectors x will be of norm 1 and also isotropically distributed. Therefore, the points will be distributed uniformly on the hypersphere. The generation of points inside the hypersphere may be achieved by rescaling the coordinates obtained in the previous steps [122].
- Solis and Wets' algorithm—it is a randomised version of optimisation technique which belongs to a hill climbing family. Every step size can be adapted as fol-

lows: it starts at a current point $\mathbf{x}$ and checks if either $\mathbf{x} + \mathbf{d}$ or $\mathbf{x} - \mathbf{d}$ is better, where $\mathbf{d}$ is a deviate chosen from a normal distribution whose standard deviation is given by a parameter ρ. If the answer is positive then a move to a better point is made and a success is recorded. Otherwise, a failure is recorded. If several successes in a row happen, parameter ρ is increased to make longer moves. However, if several failures in a row are recorded, the ρ is decreased to focus the search [170].

It is easy to see that applying dedicated complex local search method (Solis-Wets operator) helps in quick approach to the final candidate solution for all the experiments. PEA gets stuck again in a local extremum, while EMAS retains better exploration capabilities which is pointed out by the curvature of the graphs, and also by higher standard deviation of the results. It is noteworthy that enhancing EMAS with dedicated local-search method (Solis-Wets), yielded very good results for Lamarckian memetics, while Baldwinian search again was the worst.

Number of fitness function calls The results observed at the beginning of this section, in particular PEA being outperformed by EMAS (see Fig. 4.4 and Table 4.7) require several important aspects to be discussed. First, it should be noted that in the comparison between these two algorithms, based on the number of steps or generations, EMAS is somewhat more handicapped. In EMAS, distributed selection mechanism allows for parallel ontogeny so that at one observable moment a certain number of individuals in the population are about to die, another group can be almost ready for reproduction etc. In PEA global synchronisation is used and the whole population of individuals is processed at once. Therefore, PEA seems to be potentially better suited for exploration purposes, as it processes significantly more individuals than EMAS: conversely EMAS turns out to be more efficient.

Distributed selection mechanism in EMAS results in the fact that in one step of system work, the number of fitness function calls is far lower than in PEA. In Figure 4.6, the number of new individuals produced in each step for PEA, EMAS and for their modifications is shown. It should be noted that in PEA, the number of fitness function calls per generation is constant and equals the number of individuals that was 90 per one generation (30 per one island) in the experiments. Lamarckian and Baldwinian modifications lead to multiplication of this number in the case of the experimental results by 100, therefore the number of fitness computations for memetic PEAs equals 900 per generation.

The number of fitness function calls in the case of EMAS, which oscillates around 10, is a significant advantage of this computing method. Moreover, Lamarckian modification of EMAS leads to obtaining about 200 fitness computations per step (still significantly lower than in the case of PEA), whereas Baldwinian is the

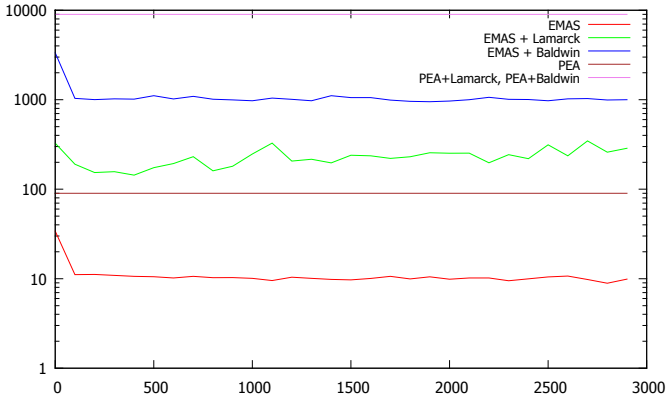


Figure 4.6. Number of fitness function calls in EMAS and its memetic variations

most costly one with 1000 as estimate of the number of fitness computations. It is easy to see that the low number of fitness function calls for EMAS makes it an interesting weapon of choice for dealing with problems characterised by a costly fitness function (e.g. inverse problems). In this case, the complexity of the implementation of the whole system supporting the notion of agency, communication, naming services etc. is overwhelmed by the complexity of the fitness function. Therefore, looking for a more intelligent search algorithm becomes reasonable, despite the intrinsic complexities imposed by its implementation.

Step execution time The most accurate evaluation of efficiency of the systems is observation of the execution time of computation, or as in the case of this analysis, of one computation step. Average times of step execution gathered during one experiment are shown in Table 4.6, along with dispersion estimation[1].

Table 4.6. Execution time of steps

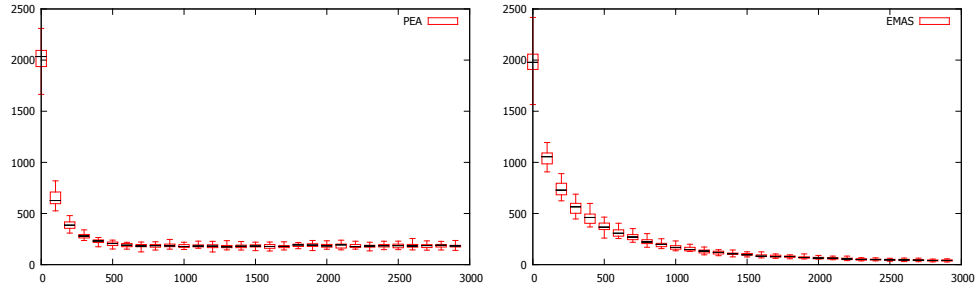
System	Avg. time [ms]	Std. dev	Std. dev %
EMAS	82.46	25.43	30
EMAS + Lamarck	138.48	45.03	32
EMAS + Baldwin	120.84	34.02	28
PEA	75.63	6.05	7
PEA + Lamarck	487.84	127.48	26
PEA + Baldwin	79.41	10.71	13

At first glance, computing with EMAS bears higher time cost than with PEA, however one must remember that these results were collected for the case of optim-

[1]These results were gathered using a server-class hardware (SUN FIRE X2100: Dual-Core AMD Opteron® Processor 1220 2.8GHz, 4GB RAM (2 x 2GB), 1 x 250GB SATA).

isation of simple benchmark function, and the total execution time will surely be much higher for PEA than for EMAS in the case of complex fitness function (cf. the paragraph describing the average number of fitness function calls in the current section). Moreover, these results clearly show that computing with EMAS is more unpredictable than with PEA, as the standard deviations are higher.

Experiments repeatability High dispersion of the results calls for additional analysis of repeatability of the experiments. Therefore, *box-and-whiskers* plots (containing minimum, maximum value, median, first and third quartiles) were prepared for selected experimental runs. In Figure 4.7 these plots are presented for PEA and EMAS fitnesses. Based on standard deviation, it is easy to see that these experiments are repeatable.

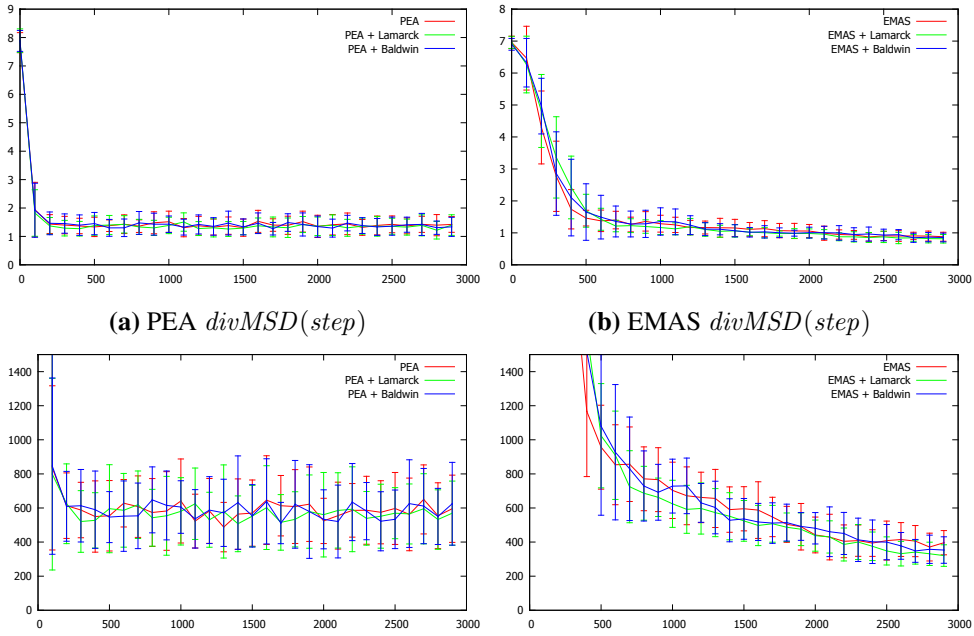


(a) PEA $bestFitness(step)$ **(b)** EMAS $bestFitness(step)$

Figure 4.7. Box-and-whiskers plot for EMAS and PEA fitness

Diversity Observation of diversity measures for the systems show that the non-zero diversity is still retained, though falls down from the beginning of computation. However, the curvature of the graphs show that loss of diversity is much quicker in PEA than in EMAS (see Fig. 4.8). The same observation holds for all memetic variants of the system. Fortunately, it never reaches zero (compare experiments presented in [136]). Both diversity measures (MOI and MSD) yield generally similar results (loss of diversity is clearly seen, and EMAS losses diversity significantly later than PEA).

Dimensionality of the problem The degree of difficulty of the problems may be measured in different ways, considering e.g. the number of local extrema, features of the search space, dimensionality etc. In order to check, how the systems behave when they are faced with problems of different difficulty, dimensionality was chosen as its determinant. Therefore, an experiment based on conducting search for optima for



(a) PEA $divMSD(step)$ **(b)** EMAS $divMSD(step)$

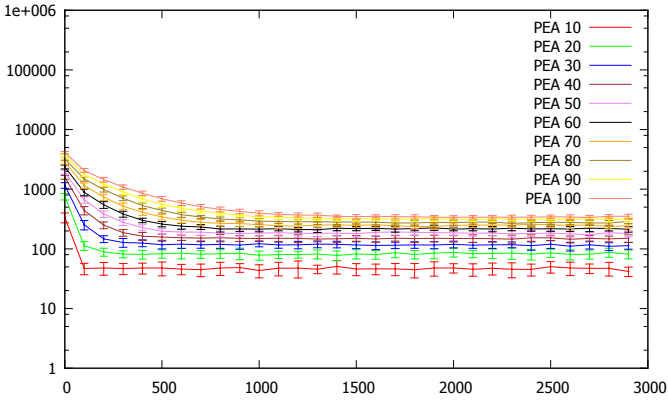
(c) PEA $divMOI(step)$ **(d)** EMAS $divMOI(step)$

Figure 4.8. PEA and EMAS diversity measurements

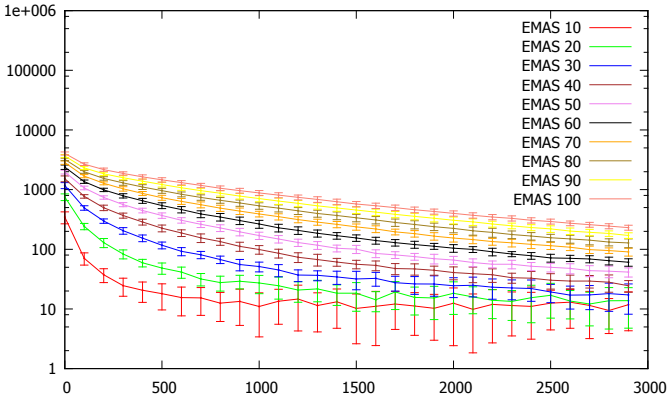
benchmarks described in spaces of different dimensionality was conducted. It should be noted that such a test is useful to assess features of the search systems, as the high dimensionality of input and output variables presents an exponential difficulty (i.e., the effort grows exponentially with dimensions) for both problem modelling and optimisation [110].

Figure 4.9 shows the graphs depicting the fitness value dependent on the simulation step for PEA and EMAS beginning from the 10-dimensional Rastrigin problem, and finishing on 100-dimensional one. It is easy to see that dimensionality of the problem greatly influences the efficiency of the techniques researched. It is easy to see that the higher dimension, the lower efficiency of the search technique.

Another important feature that may be found is the exploration capability. For PEA experiments, all the examined methods seem to get stuck in a local extremum, starting quite early (about the 1000-th generation in the case of higher dimensions, and even the 100-th generation when considering lower ones). In the case of EMAS techniques, a similar feature is a little bit harder to observe (besides the lowest dimensions) for both memetic operators used, and capabilities of improving the search result for PEA and EMAS using Lamarckian variation operators are not clearly visible.



(a) PEA, 10D–100D, $bestFitness(step)$



(b) EMAS, 10D–100D, $bestFitness(step)$

Figure 4.9. PEA and EMAS for 10–100 dimensional Rastrigin problem

Outcome of the computation In Table 4.7, the final values obtained in the 3000-th step are presented. It seems that EMAS outperformed all other systems (the best final results are pointed out using bold font). However, as the selected benchmarks are difficult, which is caused by its nature and high dimensionality, dispersion of the results is quite high. The diversity-related measures presented in Table 4.8 show that the final diversity is the highest in the case of PEA experiments. However (see Figure 4.8), the curvature of the graphs point out that though the final result of the diversity for EMAS may be worse than for PEA, the diversity is lost more slowly, helping to reach better final results than those obtained for PEA.

Table 4.7. The results obtained in the 3000-th step of computation for all the examined systems.

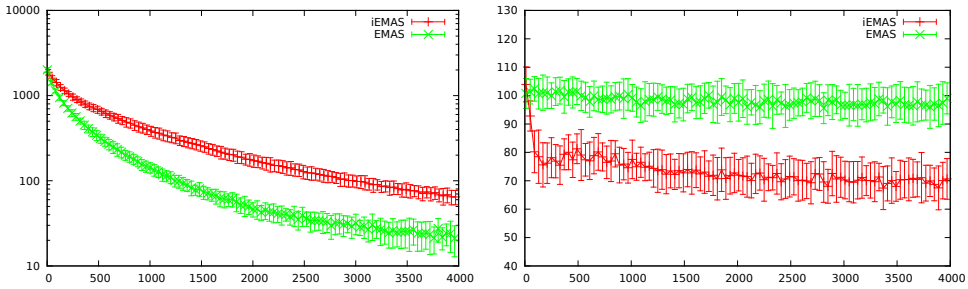
	Result	St. Dev.	Min value	Median	Max value
Ackley					
PEA	2.03	0.16	1.66	2.09	2.26
EMAS	0.58	0.17	0.27	0.59	0.93
PEA + Lamarck	1.47	0.19	0.95	1.44	1.83
EMAS + Lamarck	**0.43**	0.20	0.09	0.44	0.85
PEA + Baldwin	2.03	0.14	1.77	2.03	2.38
EMAS + Baldwin	0.58	0.19	0.26	0.59	0.92
de Jong					
PEA	7.14	1.57	5.09	7.02	10.78
EMAS	0.81	0.65	0.18	0.49	2.79
PEA + Lamarck	4.34	1.20	2.25	4.03	7.17
EMAS + Lamarck	**0.67**	0.87	0.03	0.35	4.31
PEA + Baldwin	7.46	1.30	5.40	7.26	10.65
EMAS + Baldwin	0.70	0.60	0.18	0.56	3.17
Rosenbrock					
PEA	1219.37	346.69	741.61	1162.32	2478.04
EMAS	206.46	115.42	70.02	168.26	572.19
PEA + Lamarck	694.01	222.61	391.96	668.68	1557.13
EMAS + Lamarck	**190.49**	137.38	49.99	151.99	629.38
PEA + Baldwin	1233.87	285.51	766.65	1184.23	1760.46
EMAS + Baldwin	264.51	262.63	78.72	195.85	1266.85
Schwefel					
PEA	-12942.72	790.64	-14669.28	-12840.60	-11152.09
EMAS	**-13999.87**	1023.44	-15942.66	-14194.64	-11767.72
PEA + Lamarck	-13557.15	725.19	-14560.63	-13640.47	-11800.74
EMAS + Lamarck	-13680.38	901.95	-15517.86	-13682.55	-11695.06
PEA + Baldwin	-13051.08	795.88	-14274.21	-13157.16	-11370.22
EMAS + Baldwin	-13684.54	827.41	-15756.03	-13621.34	-12112.40
Axis Parallel Hyper Ellipsoid					
PEA	155.83	30.89	93.62	152.67	229.39
EMAS	16.11	15.05	3.46	9.96	63.12
PEA + Lamarck	82.56	26.84	22.06	80.32	146.14
EMAS + Lamarck	**9.90**	13.87	0.62	7.11	78.17
PEA + Baldwin	153.35	25.93	99.18	153.01	205.77
EMAS + Baldwin	18.51	17.24	3.69	11.29	66.82
Moved Axis Parallel Hyper Ellipsoid					
PEA	784.53	162.22	471.90	770.15	1170.59
EMAS	**50.69**	39.10	15.54	35.19	181.81
PEA + Lamarck	445.18	126.97	261.73	402.70	769.96
EMAS + Lamarck	58.69	64.73	3.12	32.81	273.17
PEA + Baldwin	780.84	166.14	493.92	775.91	1119.10
EMAS + Baldwin	78.09	65.54	15.46	45.96	218.38

Table 4.8. The diversity measures obtained in the 3000-th step of computation for all the examined systems

	MSD diversity	MSD diversity st. dev.	MOI diversity	MOI diversity st. dev.
Ackley				
PEA	0.70	0.15	173.02	40.47
EMAS	0.31	0.03	46.45	7.23
PEA + Lamarck	0.65	0.12	182.92	35.83
EMAS + Lamarck	0.32	0.05	43.62	7.32
PEA + Baldwin	**0.74**	0.16	**195.87**	42.56
EMAS + Baldwin	0.33	0.03	46.41	7.20
de Jong				
PEA	0.75	0.13	**254.35**	56.52
EMAS	0.52	0.11	74.19	14.37
PEA + Lamarck	0.72	0.10	226.00	39.86
EMAS + Lamarck	0.51	0.09	70.41	14.95
PEA + Baldwin	**0.77**	0.16	248.11	37.53
EMAS + Baldwin	0.54	0.10	78.35	15.45
Rosenbrock				
PEA	0.77	0.13	264.41	55.31
EMAS	0.51	0.12	81.17	23.04
PEA + Lamarck	0.69	0.10	232.09	34.17
EMAS + Lamarck	0.45	0.07	69.17	13.73
PEA + Baldwin	**0.81**	0.15	**276.26**	61.61
EMAS + Baldwin	0.49	0.09	73.06	14.87
Schwefel				
PEA	0.78	0.12	255.55	44.81
EMAS	1.82	0.42	1615.33	356.51
PEA + Lamarck	**6.67**	4.06	**3589.18**	2945.89
EMAS + Lamarck	1.69	0.29	1426.96	298.61
PEA + Baldwin	0.88	0.16	266.75	51.20
EMAS + Baldwin	1.81	0.37	1615.79	303.59
Axis Parallel Hyper Ellipsoid				
PEA	**1.05**	0.38	298.34	73.49
EMAS	0.57	0.13	90.35	20.05
PEA + Lamarck	0.98	0.33	271.58	56.41
EMAS + Lamarck	0.56	0.13	81.67	17.92
PEA + Baldwin	1.04	0.23	**315.22**	71.53
EMAS + Baldwin	0.56	0.11	84.28	17.67
Moved Axis Parallel Hyper Ellipsoid				
PEA	0.94	0.18	297.35	57.34
EMAS	0.53	0.10	82.68	17.58
PEA + Lamarck	**1.08**	0.43	304.14	94.99
EMAS + Lamarck	0.53	0.09	78.85	15.87
PEA + Baldwin	1.07	0.33	**324.78**	77.17
EMAS + Baldwin	0.54	0.10	88.75	22.06

4.1.4. Classical and immunological EMAS

Introduction of the immunological selection, which is in fact a distributed tabu list, leads to an important question, as to whether this mechanism hampers the efficiency of the system and whether it has any benefits at all? Indeed, despite first doubts, after observing the graphs presented in Figure 4.10, it is easy to see that though the fitness seems to be a little worse in iEMAS than in EMAS, the number of agents is significantly lower during the whole computation. Therefore, this mechanism may produce interesting results, when the complexity of the fitness function is high and any means of decreasing the number of fitness function evaluation are crucial. In order to compare EMAS and iEMAS, 50-dimensional Rastrigin problem was used. The results presented in this section are recalled after Byrski [27].

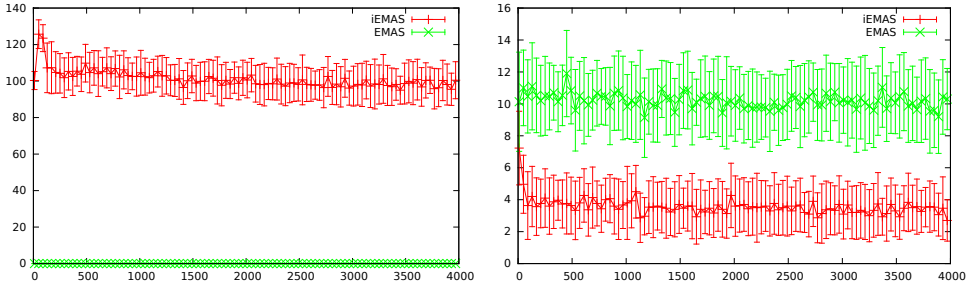


(a) $bestFitness(step)$ **(b)** $agentCount(step)$

Figure 4.10. Fitness and agent count for EMAS and iEMAS

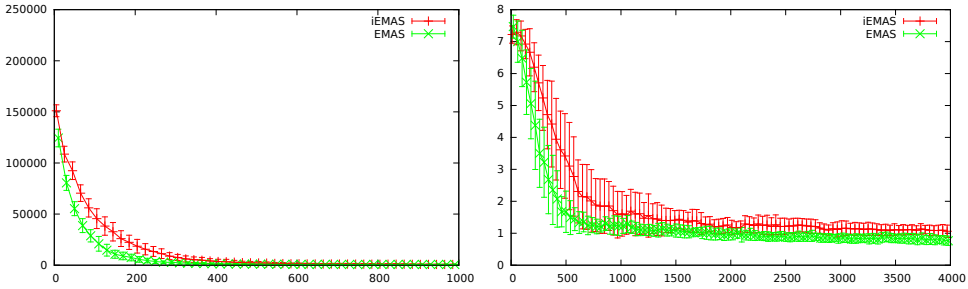
The above observation gets confirmation when looking at Figure 4.11, which shows the number of lymphocytes and the number of fitness function calls. Of course, as the number of agents is lower, it directly affects the number of fitness function calls, as predicted. Moreover, the number of lymphocytes is stable, therefore this feature of the computation may be easily predictable and adequate efforts may be made to tune the computing framework and the hardware to support an additional group of agents.

An interesting observation may be made after analysing the graphs shown in Figure 4.12. Both diversity measures point out that the diversity in iEMAS is a little better than in EMAS. This may result from the fact that the population of agents is affected by the lymphocytes. As similar agents are removed from the system (in fact, those similar to the ones recently removed), the diversity rises (this may be perceived as an effect similar to fitness sharing [125]).



(a) $lymphocyteCount(step)$ **(b)** $fitnessCalls(step)$

Figure 4.11. Lymphocyte count and number of fitness function calls for EMAS and iEMAS



(a) $divMOI(step)$ **(b)** $divMSD(step)$

Figure 4.12. MOI and MSD diversity for EMAS and iEMAS

4.2. EMAS parameters tuning

Having shown that EMAS approaches are effective in solving selected benchmark and real-life problems, it would be interesting to take an insight into the exact features of the most important mechanism of EMAS, i.e. the distributed selection based on existence of non-renewable resource. Such experiments could help to understand it and tune the computation based on this knowledge. The problem is not trivial, because EMAS, similar to other metaheuristics, utilises many parameters imposing on the user the setting dozens of degrees of freedom. The results presented in this section are recalled after Byrski [27].

4.2.1. Energy-related parameters

Energy-based distributed selection mechanism is an immanent feature of EMAS. Therefore a detailed examination of its parameters is crucial for better understanding

of the search process, and for being able to effectively tune them in order to adopt them to solving particular problems.

Energy exchange rate The most crucial parameter of the distributed selection mechanism in EMAS is the rate of energy exchange between the meeting agents. The influence of changing this parameter on the fitness and agent count in the population is shown in Figures 4.13, 4.14.

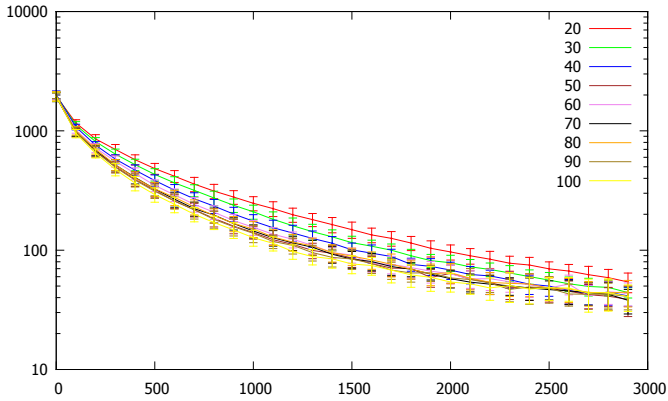


Figure 4.13. Influence of agent exchange energy on EMAS fitness $bestFitness(step)$

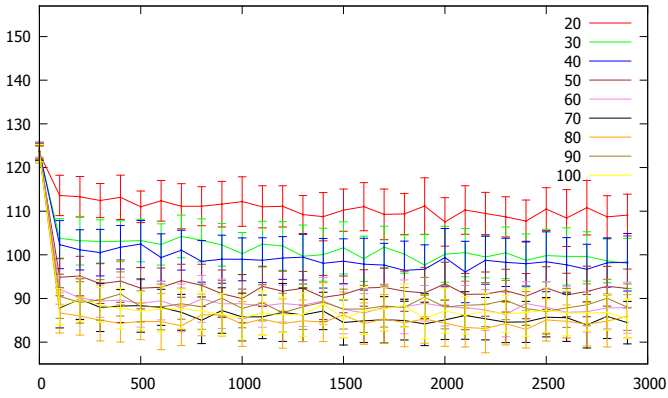


Figure 4.14. Influence of agent exchange energy on EMAS agent count $agentCount(step)$

It is easy to see that increasing this parameter makes the final result of computation better, but due to a logarithmic scale applied, this advantage does not seem to be

significant. As predicted, this parameter greatly affects the agent count in the system. The higher the energy exchange rate, the lower the average agent count in the system.

Initial energy level Influence of initial agent energy on the features of the system is presented in Figures 4.15, 4.16. Initial energy of the agents in the system is supposed to have a significant influence on the features of the agent population, as it is the main component of the total energy which is a base for distributed selection mechanism. In fact, looking at Figure 4.15, the influence seems to be strong and straightforward.

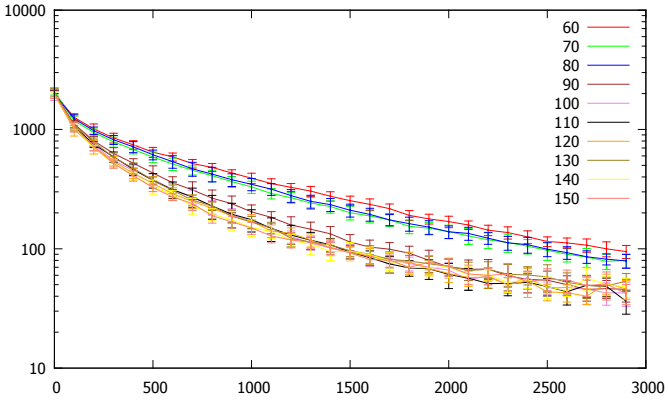


Figure 4.15. Influence of agent initial energy on EMAS fitness $bestFitness(step)$

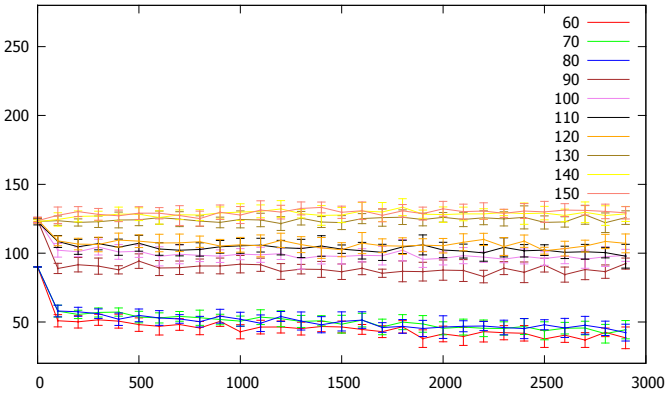


Figure 4.16. Influence of agent initial energy on EMAS agent count $agentCount(step)$

The higher initial energy, the greater the number of agents during the computation. It should be noted that the selection mechanism is stable, as the number of

agent does not grow indefinitely nor does it fall to zero during the whole computing process. It is easy to see that changing the initial energy affects indirectly the fitness in the system (see Fig. 4.16), changing the actual number of the agents in the system that are capable of exploring and exploiting the search space. Generally speaking, increasing the initial energy helps to reach better results, though this effect is not very apparent.

Minimal reproduction energy Influence of minimal reproduction energy on the features of the system is presented in Figures 4.17, 4.18.

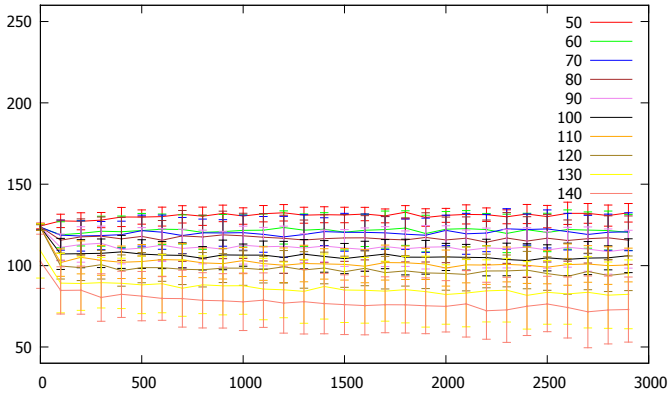


Figure 4.17. Influence of minimum reproduction energy on EMAS agent count $agentCount(step)$

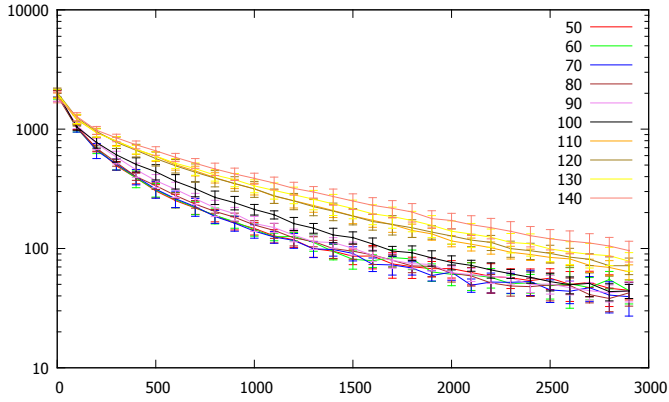


Figure 4.18. Influence of minimum reproduction energy on EMAS fitness $bestFitness(step)$

Minimal reproduction energy of the agents is supposed to have a significant influence on the features of the agent population, as it directly affects the distributed selection mechanism by controlling "maturity" of agents capable of reproduction. If this parameter value is low agents that performed few rendezvous will reproduce, while for its high value only the agents which live longer may generate offspring. In fact, looking at Figure 4.18, the influence seems to be strong and straightforward, just opposite to the case of initial energy.

It is easy to see that the higher the minimal reproduction level, the lower the number of the agents during the computation, as it is more difficult for them to reproduce. Again, the selection mechanism is stable, as the number of agent does not grow indefinitely nor does it fall to zero, during the whole computing process.

The fitness is also affected (see Fig. 4.17) because the number of the agents varies for different values of the minimal reproduction energy. The system is able to better and quicker explore the search space for lower levels of this parameter (final results of the search are better for lower values of minimal reproduction energy, and the search is quicker as the graph curvature is higher).

4.2.2. Probabilistic decision parameters

Stochastic nature of the systems brings flexibility into the computing, but if EMAS and related techniques are to be used effectively, a detailed examination of the most important probabilistic decision parameters is necessary.

Migration probability The existence of migration phenomenon between the subpopulations should affect positively the value of fitness. It seems to be straightforward because such techniques as niching and speciation are meant to increase the exploration efficiency of the algorithm. Indeed, it is easy to see that introducing migration into the system enhances the quality of results (see Fig. 4.19), despite the fact effect is almost discrete—if the probability is non-zero, the results are significantly better, however, increasing of this parameter does not produce distinguishable changes in the fitness value. This may result from the fact that the evolutionary islands were fully connected; perhaps introducing more sophisticated topology would relax the influence of this parameter on the overall efficiency of the computation.

Meeting probability This parameter affects the frequency of meetings between the agents (as the decision whether or not the agent meets another agent is based on the outcome of probabilistic sampling). The higher the meeting probability is, the more frequently agents will meet and exchange their energy.

However, this parameter does not influence the number of agents in the population (see Fig. 4.20) because the same number of agents simply exchanges the energy

faster or slower, also in the memetic versions of EMAS. Again the selection mechanism is stable, as the number of agent does not grow indefinitely nor does it fall to zero during the whole computing process.

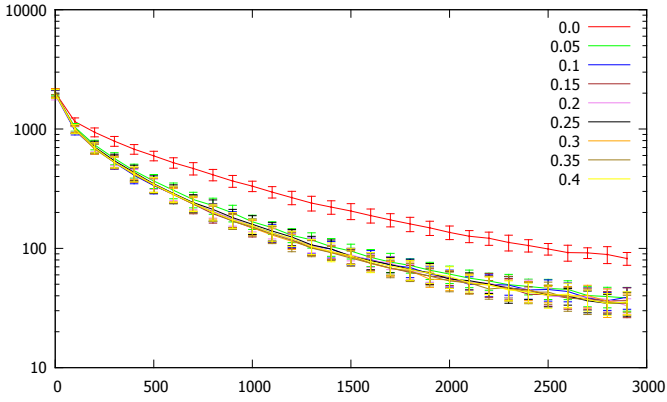


Figure 4.19. Influence of migration probability on EMAS fitness $bestFitness(step)$

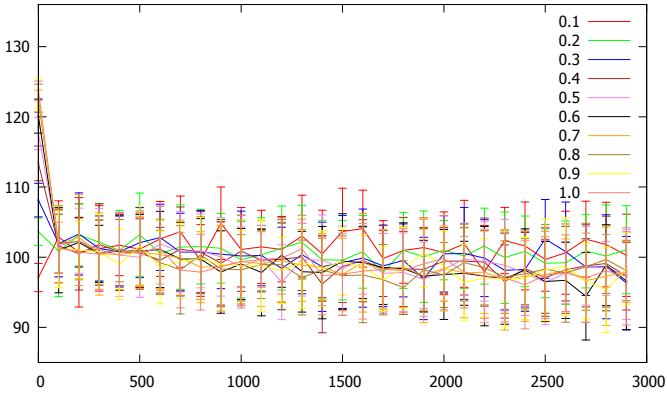


Figure 4.20. Influence of meeting probability on EMAS agent count $agentCount(step)$

Increasing the meeting probability makes it possible to reach the desired solutions quicker (see Fig. 4.21), as the energy flow from "worse" agents to "better" ones is faster, so "better" agents may reproduce quicker. Therefore, the final results of the search are better for higher values of meeting probability, and the search is quicker as the graph curvature is higher. Again, changing this parameter, does not greatly affect memetic modifications of EMAS.

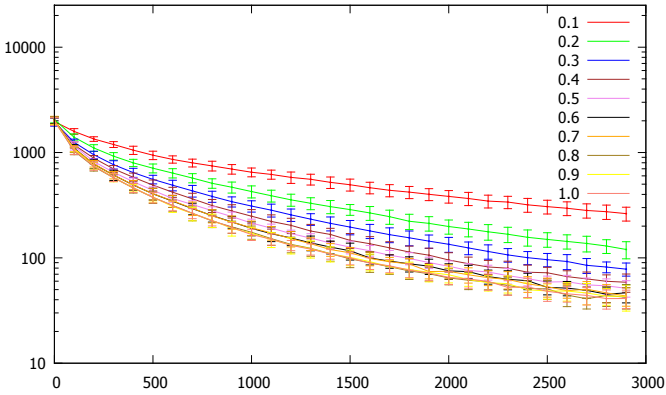


Figure 4.21. Influence of meeting probability on EMAS fitness $bestFitness(step)$

Very important information may be obtained when observing diversity shown in Figures 4.22, 4.23. Increasing the meeting probability decreases diversity. As the presence of a diverse population is an important thing in the population-based search [30], one should choose the value of this parameter in such a way that the desired solution is approached as quickly as planned (as a result of exploitation), and diversity is high enough to maintain the exploration. Choosing an appropriate value of this parameter seems to be crucial to maintain balance between exploration and exploitation for EMAS and its variations.

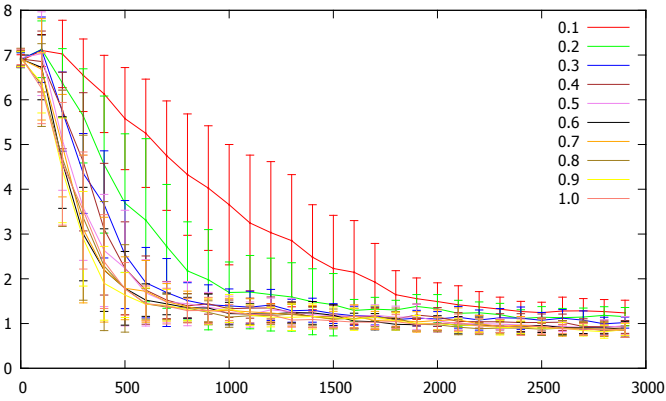


Figure 4.22. Influence of meeting probability on EMAS MSD diversity $divMSD(step)$

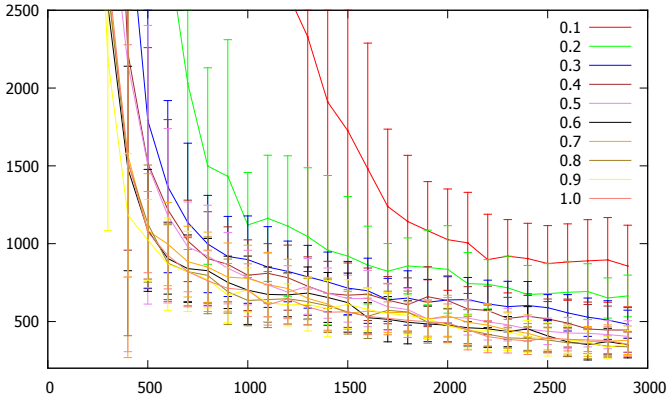
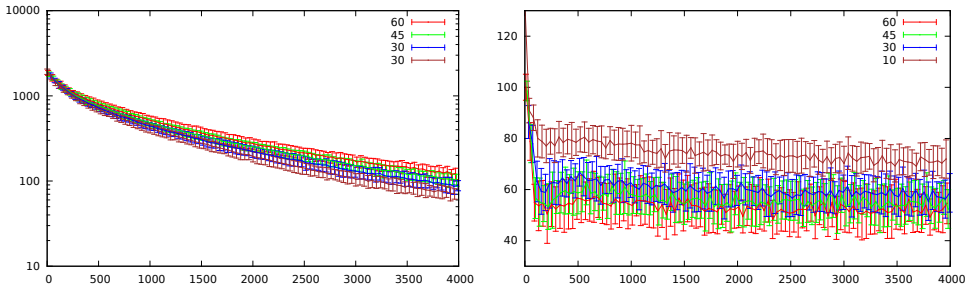


Figure 4.23. Influence of meeting probability on EMAS MOI diversity $divMOI(step)$

4.2.3. Immunological parameters

As in Section 4.1.4, immunological variant of EMAS (iEMAS) is an important weapon of choice when dealing with problems which have a complex fitness function. Therefore, an examination of selected parameters influencing the immunological selection is necessary.

Penalty threshold This parameter may be described as a quantity of energy taken from the agent, which turns out to be similar to a lymphocyte during affinity testing. It is easy to see that changing this parameter significantly influences the number of agents in the system and yet the fitness remains almost unchanged (see Fig. 4.24a), which is a very interesting fact.

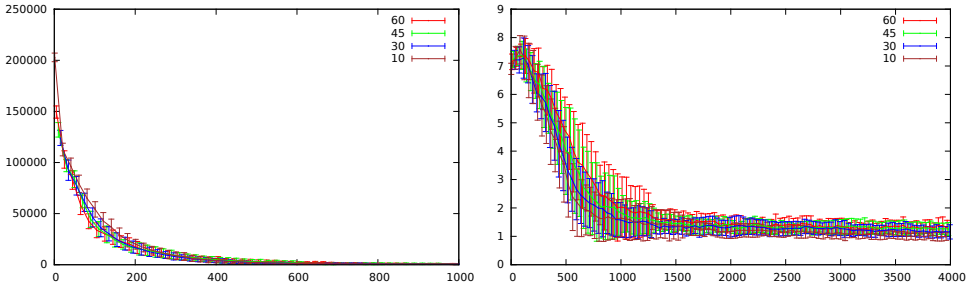


(a) $bestFitness(step)$ **(b)** $agentCount(step)$

Figure 4.24. Influence of penalty threshold on fitness and agent count in iEMAS

This observation clearly indicates that introducing distributed tabu mechanism defined in this way does not hamper the search capabilities of the system. Of course, the higher penalty is, the more agents are removed from the system, therefore, the relation shown in (Fig. 4.24b) is predictable.

Observation of the diversity measures (see Fig. 4.25) shows that changing the penalty threshold (at the same time changing the immunological selection pressure) does not hamper the diversity. Moreover, as reported in Section 4.1.4, quicker removing of "bad" agents makes the system more diverse (in terms of MSD metric).



(a) $divMOI(step)$ **(b)** $divMSD(step)$

Figure 4.25. Influence of penalty threshold on diversity in iEMAS

Penalty threshold has also a predictable influence on the number of lymphocytes in the system (see Fig. 4.26) closely connected with reducing of the agent population.

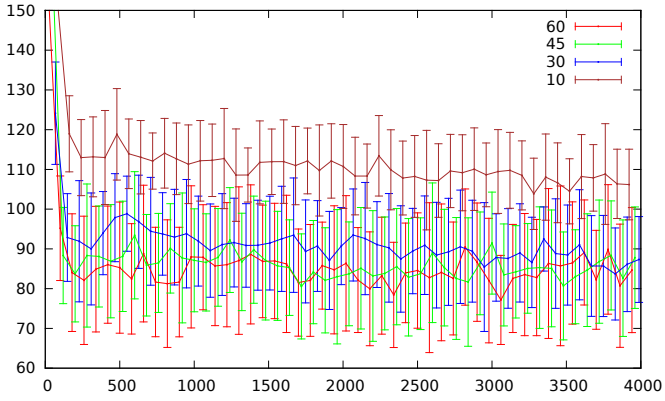
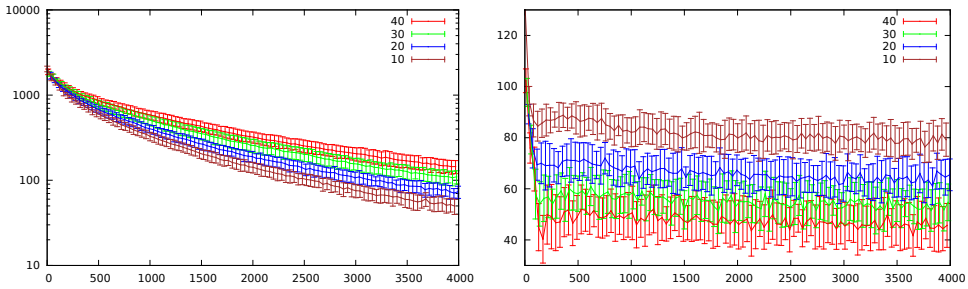


Figure 4.26. Influence of penalty threshold on lymphocyte count

When the number of agents is lower, the same total sum of energy is distributed among individuals of a smaller population, therefore, the average value of energy per agent is higher and agents do not die so often as in the case of a bigger population. In effect, when the number of individuals in the population is low, also the smaller number of lymphocytes is generated.

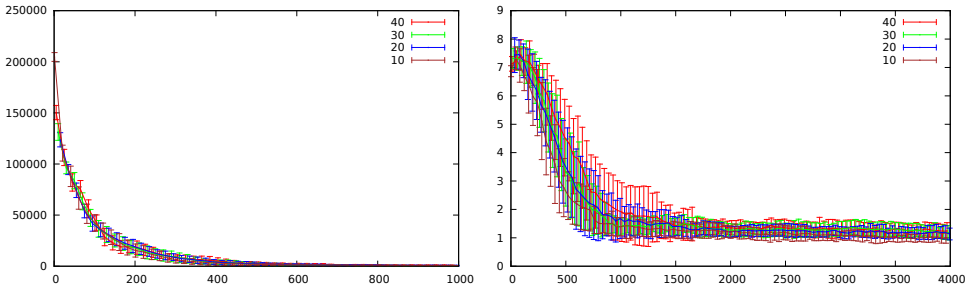
Lymphocyte life length Longer lymphocyte life (see Fig. 4.27) again does not significantly worsen the fitness, however certain influence may be observed, as the fitness becomes little better in the case of a shorter lymphocyte life. At the same time, of course, the agent count decreases with the rise of lymphocyte life as the lymphocytes may act longer removing the individuals from the population.



(a) $bestFitness(step)$ **(b)** $agentCount(step)$

Figure 4.27. Influence of lymphocytes' lifespan on fitness and agent count

At the same time, manipulating the lifespan of lymphocytes does not hamper the diversity measures, though a little positive influence may be observed in the case of MSD diversity, when the lymphocyte lifespan is longer (see Fig. 4.28).



(a) $divMOI(step)$ **(b)** $divMSD(step)$

Figure 4.28. Influence of lymphocytes' life length on diversity

It is interesting that the length of lymphocyte lifespan does not affect at all the number of lymphocytes in the system (see Fig. 4.29). It shows that the immunological selection mechanism is stable and lymphocytes do not tend to overpopulate agents, though the average number count has a significant diversity, due to full stochastic nature of the selection mechanism.

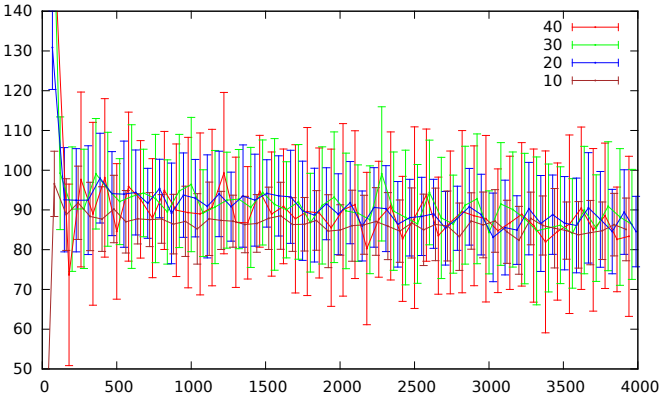
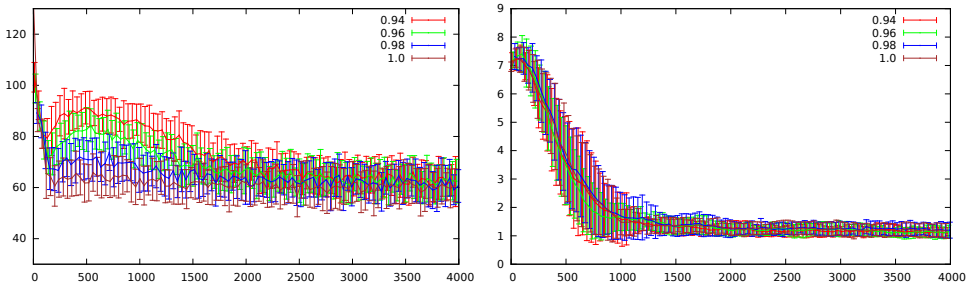


Figure 4.29. Influence of lymphocytes' life length on lymphocyte count

Percentage of "good" fitness During a negative selection process, lymphocytes are removed when they are still considered immature, though they match a "good" agent in the population. This is the case when an immature lymphocyte matches an agent that has fitness related in some way to the average fitness in the population (an appropriate percentage is considered). In Figure 4.30 the results of changing this percentage are shown, along with the MSD diversity of the population.

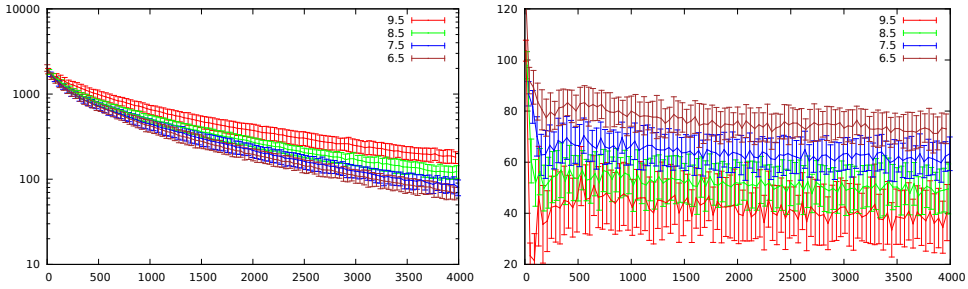


(a) $agentCount(step)$ **(b)** $divMSD(step)$

Figure 4.30. Influence of "good" fitness percentage on agent count and diversity

It is easy to see that these two graph sets are related. When the population is diverse (mostly at the beginning of the computation) the level of "good" fitness is lower than later when the diversity falls down. So, lymphocytes tend to be removed more often, and therefore the population of agents is larger. Other important parameters such as fitness, MOI diversity and lymphocyte count are quite similar to those discussed before and remain unchanged in the relation with a given parameter.

Affinity measure In order to measure affinity between lymphocytes and agents, Mahalanobis distance was used [123]. Lower distance means that the lymphocyte must match a closer agent before penalising it (and vice versa). Therefore, it is easy to see that increasing the distance hampers a little the obtained fitness, and of course decreases the number of agents in the system (see Fig. 4.31).



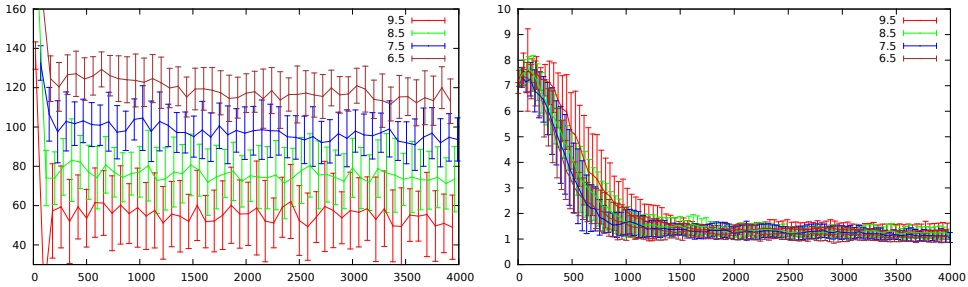
(a) $bestFitness(step)$ **(b)** $agentCount(step)$

Figure 4.31. Influence of similarity threshold on fitness and agent count

At the same time, observation of the lymphocyte count reveals that if the distance is lower, more lymphocytes are created in the system, as it is easier to remove the agent. It is connected of course with the similarity measure (this effect has been already observed before) that removing lymphocytes increases the MSD similarity measure (see Fig. 4.32).

4.2.4. Parameters tuning recapitulation

The above-mentioned experiments may surely provide a basis for researchers who are willing to apply the EMAS-like computing to their problems. In order to make this easier, a summary of the parameters tuning is presented in Table 4.9. Based on the results presented in this table, in order to appropriately parametrise computation, one must focus not only on attaining a specified goal (e.g. good fitness) but also check whether other parameters comply with this goal.



(a) Lymphocyte count **(b)** MSD diversity

Figure 4.32. Influence of similarity threshold on lymphocyte count and MSD diversity

Table 4.9. Parameters tuning summary

Increase of the parameter	Fitness	Agent count	MOI	MSD	Lymphocyte count
Energy exchange rate	↗	↘	↗	—	
Initial energy level	↗	↗	—	—	
Minimal reproduction energy	↘	↘	↘	—	
Migration probability	↗	—	↗	—	
Meeting probability	↗	—	↘	↘	
Penalty level	—	↘	—	↗	↘
Good fitness percentage	—	↘	—	—	—
Lymphocyte life length	↘	↘	—	↗	—
Similarity distance	↘	↘	—	↗	↘

Therefore a desire to increase one parameter of the computation (e.g., fitness), must be verified with other criteria, for example a need to retain relatively small population of the agents or high diversity.

Synthetic results as these presented in Table 4.9 will surely help in such activities, creating a starting point for continuing of the research. That is, after finding interesting parameters interdependencies, the researcher should refer to one of the previous sections, describing the details of the experiments, and start to prepare and test his own system configurations.

4.3. EMAS in real-world problems

Conducting experiments for renowned benchmark functions is always important, as this is a way to promote a given computing algorithm by making it possible to compare it to other methods researched by scientists active in the field. However, research work which concentrates solely on benchmarks lacks an important aspect: the real-life application. Due to this fact, selected real-life problems were also considered, and efficiency obtained for EMAS and PEA is presented here.

4.3.1. Step and flash imprint lithography inverse problem

One of the black-box problems with costly fitness function is an inverse problem defined as simulation of deformation in step and flash imprint lithography (SFIL). Here, following the definition of the problem, the results obtained by applying EMAS and PEA to solve it are presented. For details on experimental configuration refer to Appendix A. The results presented in this sections are recalled after Wróbel, Torba, Paszyński and Byrski [197].

Step and flash imprint lithography is an important patterning framework used in silicon industry [130, 144]. This process consists of the following phases: dispense—depositing a low viscosity silicon containing, photo-curable etch barrier onto a substrate; imprint—bringing the template into contact with the etch barrier; expose—exposing the etch barrier to UV in order to cure it; separate—releasing the template.

Photopolymerisation, however, is often accompanied by densification. The shrinkage of the feature can be modelled by linear elasticity with thermal expansion coefficient.

Linear elasticity model with thermal expansion coefficient Following [93] strong and weak formulations for the linear elasticity problem with thermal expansion coefficient are given as follows. The computational domain Ω is defined in the following way:

$$\Omega = \{(x_1, x_2, x_3) : x_i \in (0, 1)\} \tag{4.1}$$

The bottom of the Ω constitutes the Dirichlet boundary

$$\Gamma_D = \{(x_1, x_2, x_3) : x_1, x_2 \in (0, 1), x_3 = 0\} \tag{4.2}$$

and the remaining parts of the boundary of Ω constitute the Neumann boundary

$$\Gamma_N = \partial\Omega - \Gamma_D \tag{4.3}$$

Strong formulation. Given $g_i : \Gamma_D \ni \mathbf{x} \to g_i(\mathbf{x}) \in R$, θ and α_{kl}, find the displacement vector field $u_i : \bar{\Omega} \ni \mathbf{x} \to u_i(\mathbf{x}) \in R, i = 1, 2, 3$, such that

$$\sigma_{ij,j} = 0 \text{ in } \Omega, \tag{4.4}$$

$$u_i = g_i \text{ on } \Gamma_D, \tag{4.5}$$

$$\sigma_{ij} n_j = 0 \text{ on } \Gamma_N, \tag{4.6}$$

where σ_{ij} is the stress tensor, defined in terms of the generalised Hook's law

$$\sigma_{ij} = c_{ijkl}(\epsilon_{kl} + \theta\alpha_{kl}), \tag{4.7}$$

here c_{ijkl} are elastic coefficients (known for a given material), θ is temperature, α_{kl} are thermal expansion coefficients, and $\epsilon_{ij} = u_{(i,j)} = \frac{u_{i,j}+u_{j,i}}{2}$ is the strain tensor, where $u_{i,j}$ are displacement gradients.

Weak formulation. The weak formulation is obtained by multiplying (4.4) by test functions $w_i \in H_0^1(\Omega)$ and integrating by parts over Ω:

$$-\int_\Omega w_{i,j}\sigma_{ij}d\Omega + \int_\Gamma w_i\sigma_{ij}n_j d\Omega = 0. \tag{4.8}$$

Since σ_{ij} is the symmetric tensor, then $w_{i,j}\sigma_{ij} = w_{(i,j)}\sigma_{ij}$ and

$$\int_\Omega w_{(i,j)}\sigma_{ij}d\Omega = 0, \tag{4.9}$$

which uses the fact that $w_i = 0$ on Γ_D and $\sigma_{ij}n_j = 0$ on Γ_N. Finally, by utilizing (4.7) the following is obtained

$$\int_\Omega w_{(i,j)}c_{ijkl}u_{(k,l)}d\Omega = -\theta\int_\Omega w_{(i,j)}c_{ijkl}\alpha_{kl}d\Omega. \tag{4.10}$$

Reformulation for the SFIL modelling. For the convenient implementation of the algorithm, the following equivalent weak formulation is utilised. Find $\mathbf{u} \in \mathbf{V}$, such that

$$\mathbf{a}(\mathbf{u},\mathbf{w}) = -\mathbf{A}(\mathbf{w})\,\forall\mathbf{w} \in \mathbf{V} \tag{4.11}$$

$$\mathbf{a}(\mathbf{u},\mathbf{w}) = \int_\Omega \epsilon(\mathbf{w})^{\mathbf{T}}\mathbf{D}\epsilon(\mathbf{u})\,\mathbf{d\Omega} \tag{4.12}$$

$$\mathbf{A}(\mathbf{w}) = \theta\int_\Omega \epsilon(\mathbf{w})^{\mathbf{T}}\mathbf{D}\alpha\mathbf{d\Omega}, \tag{4.13}$$

where $\mathbf{V} = \{\mathbf{V} \in \left(H^1(\Omega)\right)^3 : tr\mathbf{v} = \mathbf{0} \text{ on } \Gamma_D\}$, and Γ_D is defined as the bottom of the 3D cube. Here

$$\epsilon(\mathbf{u}) = \begin{pmatrix} u_{1,1} \\ u_{2,2} \\ u_{3,3} \\ u_{2,3}+u_{3,2} \\ u_{1,3}+u_{3,1} \\ u_{1,2}+u_{2,1} \end{pmatrix}, \tag{4.14}$$

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)}\begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{pmatrix}. \tag{4.15}$$

Inverse problem In order to calibrate the direct problem model it is necessary to find out the model parameters, which involves the Young modulus, Poisson ratio and thermal expansion coefficient.

Following [130] the Poisson ratio is set to $\nu = 0.3$. It is also assumed that $g_i : \Gamma_D \ni \mathbf{x} \to g_i(\mathbf{x}) = 0$ (the feature is fixed at the bottom, with free boundary conditions on all other sides), $\theta = 1$ (the thermal expansion coefficient α expresses the volumetric contraction of the feature when the temperature gradient is equal to 1 Celsius), $\alpha_{ij} = -\alpha\delta_{ij}$, where $\alpha = -0.0615$ is based on the inverse analysis [142].

The aim of the study is to find out the non-uniform Young modulus of the feature, resulting in slight lean of the feature, presented in Figure 4.33. It is assumed that there are 27 Young moduli for each of 27 subparts of the feature, summarised in Figure 4.34. The goal of the inverse analysis is to localise these 27 Young moduli, resulting in the measured deformation of the feature.

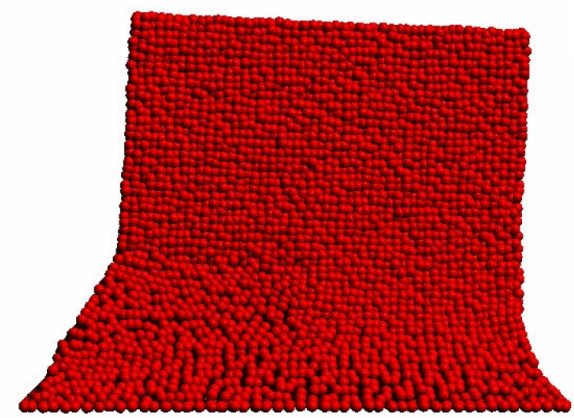


Figure 4.33. Slight lean of the feature

For the direct problem solution the self-adaptive hp finite element method application hp3d [143, 46] was used, implementing the linear elasticity with the thermal expansion coefficient.

Computing results As stated in [197], EMAS and EA were configured to solve an inverse problem, which consists in identifying 27 parameters (Young's moduli) that are required by the solver to perform the Finite Element Method-based simulation.

The number of steps were arbitrary chosen to satisfy the final average fitness value on level about 1.0. This parameter is different for EMAS and EA for one more reason: the most important observations (e.g. best fitness) were noted in relation to

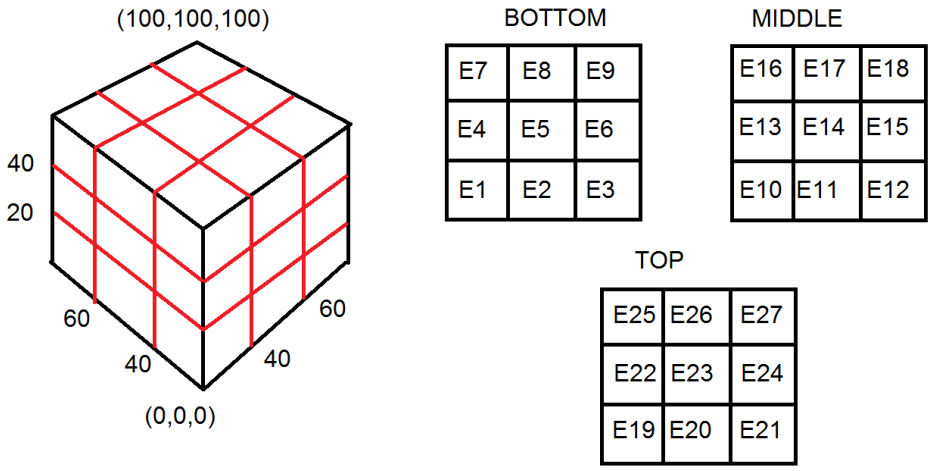


Figure 4.34. The problem considered

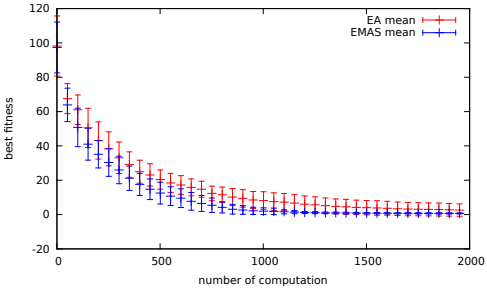
the number of fitness computation (instead of subsequent step of computation or arbitrarily measured time).

Initial values of genes were randomly generated from continuous space defined by boundaries. They represent 27 Young moduli and with precision are input to hp3d application, which outputs deformation values.

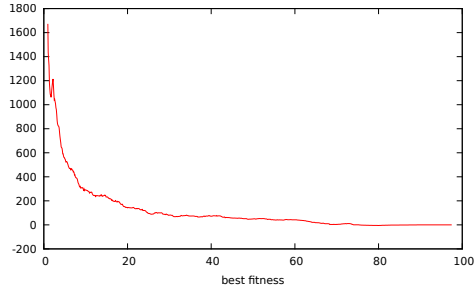
The output from the solver is the minimum and maximum displacements of the feature along x, y and z axis. The fitness function is calculated using mean squared error between output values of these minimum/maximum displacements obtained from hp3d code and the minimum / maximum displacements obtained from the experiment.

This simulation as it was mentioned is costly. The exact time of one fitness function call (involving the simulation by the solver) of course depends on the hardware configuration and also on accuracy of the solver. The higher the accuracy, the longer the computing time of course (see Tables 4.10 and 4.11). The experiments with the precision set to 60 were repeated 30 times, experiments with the precision set to 25 were repeated 3 times, and there was one-time approach conducted to compute the results for the precision 8. Of course, the memetic versions of the systems were avoided.

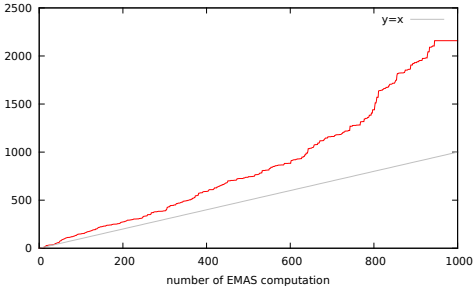
The detailed results obtained for a given precision (60, 25, 8) are presented in Figures 4.35, 4.36, 4.37, respectively.



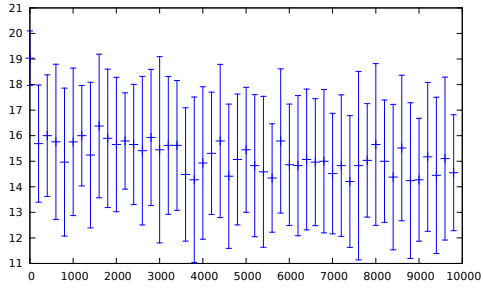
(a) Fitness



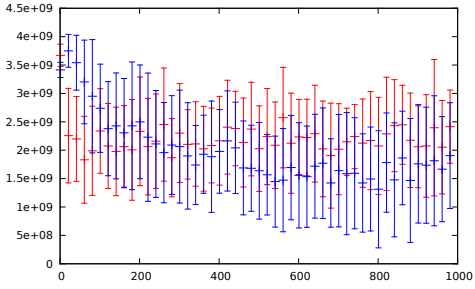
(b) Difference between computing of certain fitness value (EMAS vs EA)



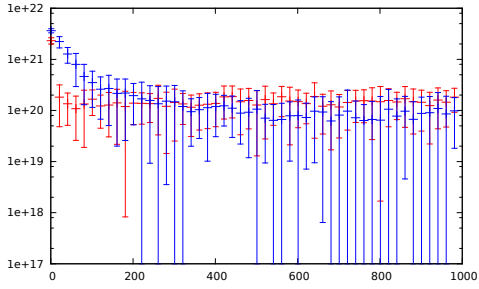
(c) Number of EA fitness computation of the same value as EMAS



(d) Agent count in EMAS

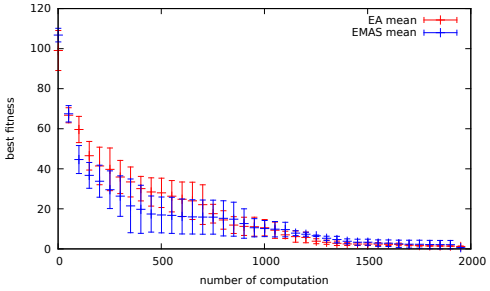


(e) MOI diversity for EA and EMAS

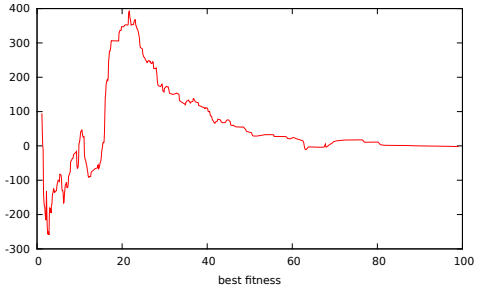


(f) MSD diversity for EA and EMAS

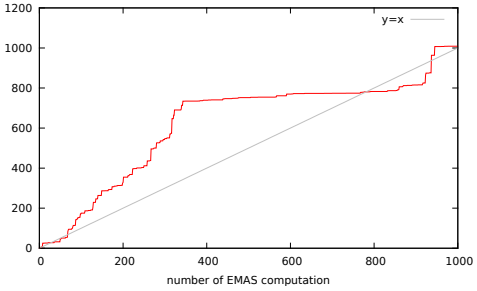
Figure 4.35. EMAS produces better results faster than EA maintaining stable population and diversity (precision 60)



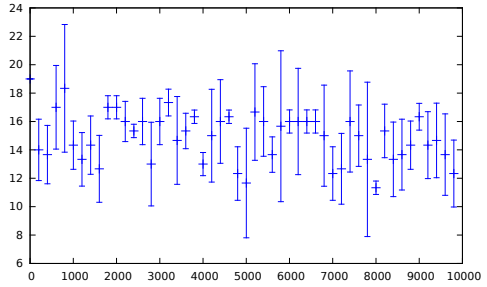
(a) Fitness



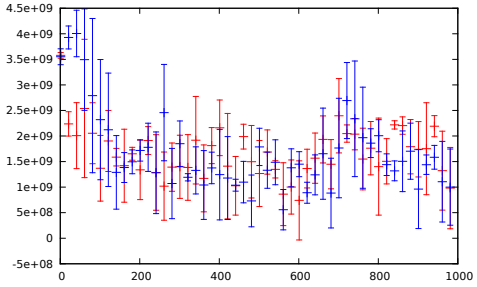
(b) Difference between computing of certain fitness value (EMAS vs EA)



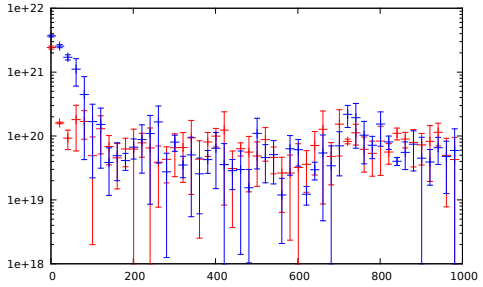
(c) Number of EA fitness computation of the same value as EMAS



(d) Agent count in EMAS

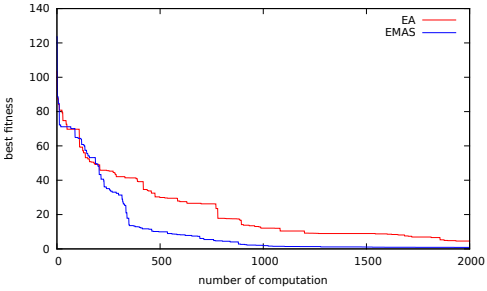


(e) MOI diversity for EA and EMAS

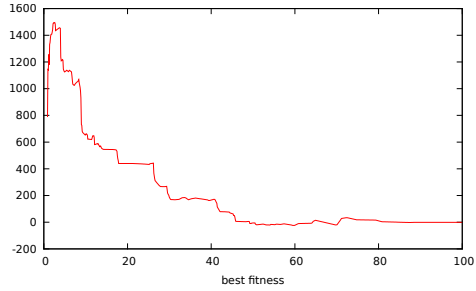


(f) MSD diversity for EA and EMAS

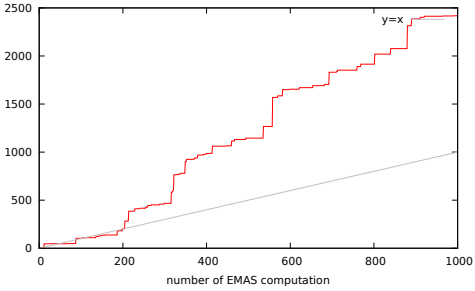
Figure 4.36. EMAS produces better results faster than EA maintaining stable population and diversity (precision 25)



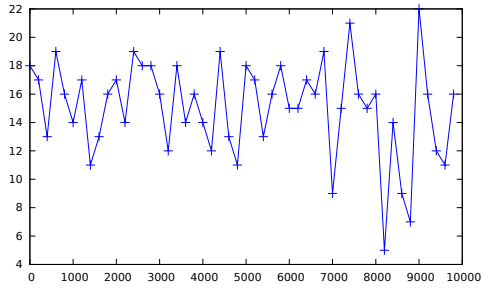
(a) Fitness



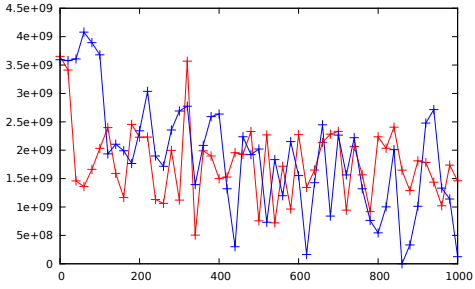
(b) Difference between computing of certain fitness value (EMAS vs EA)



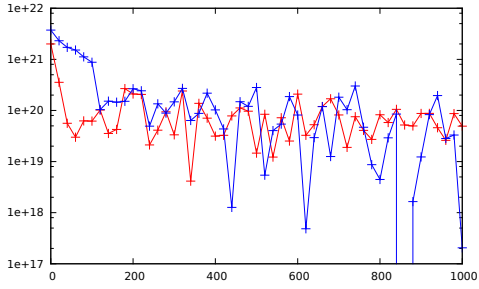
(c) Number of EA fitness computation of the same value as EMAS



(d) Agent count in EMAS



(e) MOI diversity for EA and EMAS



(f) MSD diversity for EA and EMAS

Figure 4.37. EMAS produces better results faster than EA maintaining stable population and diversity (precision 8)

Table 4.10. Execution time (in minutes) of fitness single function call using FEM-based SFIL solver on different processors

Precision	Intel i7 2.2GHz	AMD Opteron 1220 2.8GHz
60	0:0.533	0:1.053
59-25	0:53.234	2:37.105
24-8	2:31.210	7:41.221
7-5	4:11.729	12:49.887
4-1	6:26.175	19:7.572

Table 4.11. Execution time of experiments on AMD Opteron 1220 2.8GHz

Precision	Repetitions	EMAS	EA
60	30	17h30m	26h20m
25	3	10d22h	16d8h
8	1	10d16h	16d

Experiments were repeated to ensure the independence of the initial values. The numbers of experiments repetitions were chosen to finish them within a reasonable time as shown in Table 4.11, therefore the most reliable results from the statistical point of view are shown for the precision of 60%.

All these experiments revealed that EMAS produced better results much earlier than EA (see Figs. 4.35, 4.36, 4.37). These differences were visualised in Figs. 4.37b, 4.36b, 4.35b. Only for the precision 25%, EA reached the same fitness level as EMAS, but not until the 800th fitness function call was executed.

The final solution found on acceptable precision (8%) has the fitness value of 0.876222 and 27 Young moduli are (see Fig. 4.38): $1.9 \cdot 10^{09}$, $5.1 \cdot 10^{08}$, $1.6 \cdot 10^{09}$, $3.4 \cdot 10^{08}$, $1.4 \cdot 10^{08}$, $5.1 \cdot 10^{08}$, $1.2 \cdot 10^{08}$, $1.1 \cdot 10^{09}$, $1.1 \cdot 10^{09}$, $1.2 \cdot 10^{09}$, $4.9 \cdot 10^{08}$, $6.5 \cdot 10^{08}$, $1.7 \cdot 10^{09}$, $1.4 \cdot 10^{09}$, $3.7 \cdot 10^{07}$, $5.2 \cdot 10^{08}$, $9.1 \cdot 10^{08}$, $3.8 \cdot 10^{08}$, $6.6 \cdot 10^{07}$, $1.3 \cdot 10^{09}$, $4.8 \cdot 10^{08}$, $8.7 \cdot 10^{08}$, $1.8 \cdot 10^{09}$, $2.9 \cdot 10^{08}$, $1.0 \cdot 10^{09}$, $7.4 \cdot 10^{08}$, $2.4 \cdot 10^{08}$.

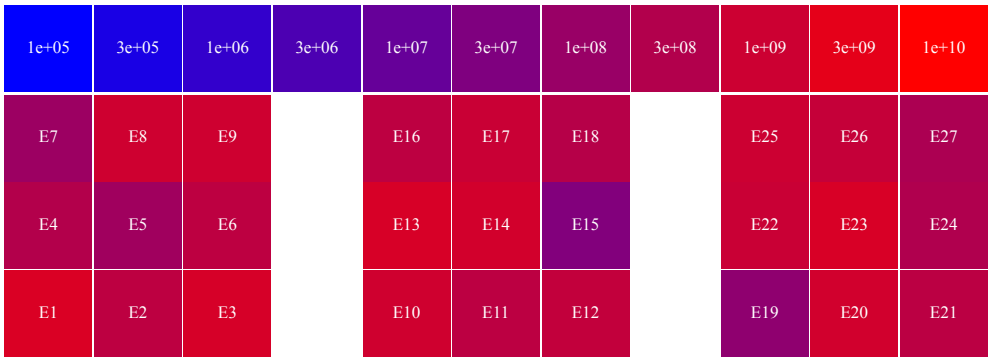


Figure 4.38. Visualisation of found solution (fitness 0.876222 with precision 8)

The presented graphs depicting the difference between computing of certain fitness values and number of EA computation of the same fitness value as EMAS, clearly show that EMAS tends to produce better results quicker, though they may become similar to the ones produced by EA later. Note, that because of high cost of fitness function calls, this outcome is a significant advantage of EMAS.

Agent count measurement showed that the population is stable similarly to other EMAS-related experiments. A similar conclusion may be drawn when observing the diversity measures—they all are quite high at the beginning of the computation and they fall in the end.

4.3.2. Advisory strategy parameters optimisation

Decision support, which is mostly related to artificial intelligence, constitutes a broad range of different techniques aimed at helping the human (decision-maker) in different activities, such as choosing the most feasible strategy for investing in financial instruments, performing a diagnosis of a faulty system or predicting a product revenue in the market [159].

Sometimes, valuable advice may be given using a predefined model that will help to simulate or solve a certain task. Such a model may become a source of knowledge, straightforwardly supporting the user in performing certain decision-making tasks. It may be proposed, e.g. in the form of a set of equations constructed by an expert, but it can also be constructed in an automated way [22].

As an example, the process of optimisation of neural network architecture may be mentioned (as neural networks may in a natural way become a part of decision support system and serve as a means of solving approximation problems, e.g. classification and prediction, as well as control problems, i.e. management of some process or device). Even though the use of neural networks replaces the necessity to solve the problem in a deterministic way, one still needs to define network parameters such as its structure, learning coefficients etc., which should be suitable for a given problem. This usually requires carrying out numerous experiments, so it is a time consuming job and can be performed only by specialists [83].

At the same time, techniques of evolutionary computation were successfully used for solving difficult search and optimisation problems. Moreover, they may be useful in supporting the search for optimal parameters of a certain model (e.g. optimal neural network architecture). Although the classical evolutionary algorithms can easily be applied to the search for optimal parameters of a certain model, additional advantages may be expressed by applying more complex search methods such as agent-based computing.

In this section, an application of EMAS and its memetic version to the optimisation of parameters of the advisory strategy is presented. The case study is based on

an original Sudoku solving strategy. Sudoku solving belongs to a NP-complete class of problems, therefore, the problem stated in this paper can be perceived as a good benchmark for the agent-based metaheuristics. For details of experimental configuration refer to Appendix A.

The results presented in this sections are recalled after Polnik, Kumięga and Byrski [146].

Sudoku advisory strategies Sudoku is a number-placement puzzle known worldwide, in which the user is given a task to fill a 9×9 lattice with digits so that each column, row and each of nine 3×3 sub-lattices that compose the lattice contains all of the digits from 1 to 9 (see Fig. 4.39). A partially completed lattice is provided, which typically has an unique solution [168, 72]. Sudoku is a NP-complete constraint satisfaction problem. The proof can be found in [59]. The fact of Sudoku's NP-completeness makes solvers using solely brute-force techniques infeasible.

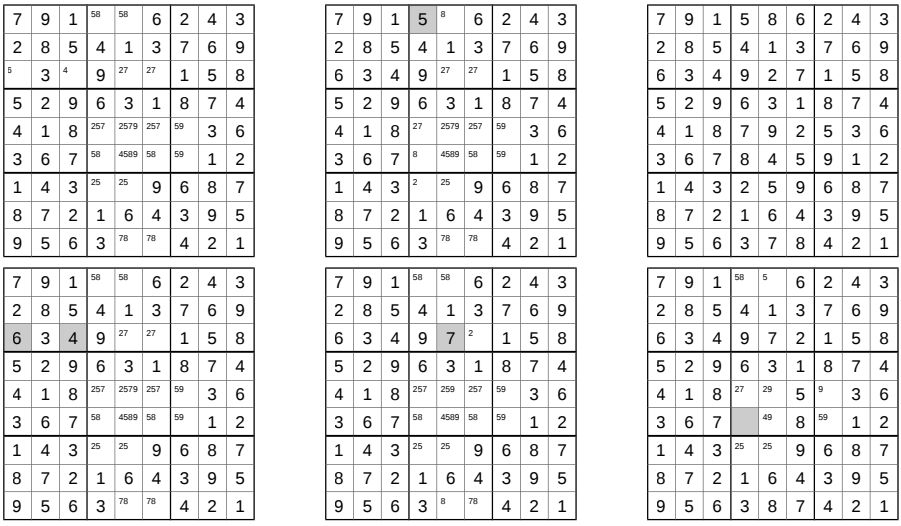
In this paragraph, dedicated advisory strategies for Sudoku problem are discussed. They should not be treated as approaches to solving this puzzle directly, rather than a means of suggesting a strategy that may be used by the human. Open Source Sudoku solvers [87] implement various strategies, formulated by Sudoku community to detect incorrect movements in advance and reduce the number of backtracks [174]. These strategies make it possible either to exclude some movement possibilities or make a deterministic move to satisfy Sudoku's constraints in a specific board setting.

Paragraph below provides a summary of a few popular strategies described in [87, 172].

- *Naked Pairs* – let A and B be the only candidate movements for 2 empty fields in a row, column or 3×3 block. No other empty field within the same unit can be filled with A or B.
- *X-Wing* – let A be a candidate movement in 4 empty fields that are located in the vertices of a rectangle. Any other empty field that share a row or column with the rectangle vertices cannot be filled with A.
- *Sword Fish* – let A be a candidate movement in empty fields that share 3 different rows or columns. Any other empty field in each of the rows or columns cannot be filled with A.

The detailed overview of Sudoku advisory strategies is available in [173]. The aforementioned strategies can also be used to assess the hardness of Sudoku boards [174]. Different approach to solve and estimate Sudoku boards using continuous-time dynamical system and Richter type scale respectively is described in [59].

The proposed strategy An original strategy of solving Sudoku puzzle based on the way in which human usually resolves this puzzle [168] is defined as follows.



(a) Top image shows a board state with two possible deterministic moves; both individuals perform these moves, as shown in the bottom image

(b) As no more deterministic moves were possible, individuals had to perform a guess according to their genotype; **A**'s choice is shown in top image, **B**'s in the bottom one

(c) After a series of deterministic moves following a guess **A** was able to fill the board; **B**'s move proved to be wrong and it had to backtrack

Figure 4.39. Example illustrating a portion of board solving process with moves performed by two different individuals, **A** and **B**; small numbers represent $S(p)$ sets

The strategy is supposed to point out the subsequent field of the lattice to fill it out with one of the feasible digits following that no other movement enforced by Sudoku constraints is known. Therefore, for each field of the lattice, denoted here by (x, y), for all empty fields located in the row x and column y, and all feasible digits i, the value of the following weight function is computed:

$$W(x, y, i) = a_1 * Fill33(x, y) + a_2 * FillRow(x) \\ +a_3 * FillCol(y) + a_4 * Occ(i), \tag{4.16}$$

where:

- $Fill33(x, y)$ is the function computing the filling level of the 3×3 block where (x, y) field is located.
- $FillRow(x)$ computes the value describing the filling level of the row x.

- $FillCol(y)$ computes the value describing the filling level of the column y.
- $Occ(i)$ computes the count of the fields with the number i.

Then the move consisting in putting the digit i into the field (x, y) is made for the field bearing the extremal value of the function $W(x, y, i)$ (minimum or maximum, depending on the exact definition of the $Fill33$, $FillRow$, $FillCol$ and Occ functions).

The problem of optimisation of the proposed Sudoku advisory strategy may be treated as the parametric optimisation [151] (maximisation in this case) of the function $W(x, y, i)$ depending on the parameters $a_k \in [-3, 3], k \in [1, 4]$, used for advising on subsequent moves in Sudoku solving. This may be accomplished with an evolutionary approach. For this purpose, the pattern sought is encoded as a following weight vector:

$$[a_1, a_2, a_3, a_4], a_k \in [-3, 3], k \in [1, 4]$$

and the fitness function is defined as a multiplicative inverse of a number of non-feasible decision undertaken by the individual in the course of solving a series of lattices according to the following procedure:

1. Make all deterministic moves:
   - A deterministic move is the one that complies with the rules of Sudoku (in each column, row and block at most one number of a certain value can be located, without backtracking or contradictions).
   - For each field p of the Sudoku board a set of numbers $S(p)$ is determined, which can be filled into this field without violating the Sudoku rules.
   - If there exists p for which $S(p)$ contains only one symbol s, then it is removed from all other $S(\cdot)$ located in the same 3×3 block, column or row.
   - If in the course of reducing $S(\cdot)$ sets, a new set of cardinality 1 is obtained, the procedure is repeated for this new set.
   - The algorithm is finished when all sets $S(\cdot)$ contain only one element or during the actualisation of the $S(\cdot)$, no one-element set was obtained.
2. If the board is not solved, make a move according to the current strategy, otherwise finish the move according to the strategy and increase the counter of non-feasible decisions for the evaluated solution.
3. If the board is not solved, go to 1.

Several possible decisions undertaken during this procedure are presented and commented in Figure 4.39.

Computing results The detailed results obtained in the course of the experiments are presented in Figures 4.40, 4.41, including fitness, diversity measures computed for EMAS and PEA and agent count for EMAS.

It is easy to see that for a given problem, EMAS obtained better or similar results while maintaining stable population of agents, except for experiments focused on Lamarckian versions of both systems where PEA prevails.

At the same time diversity measures clearly indicate that diversity is significantly better in EMAS, at least at the beginning of computation, so the exploration phase is apparently longer.

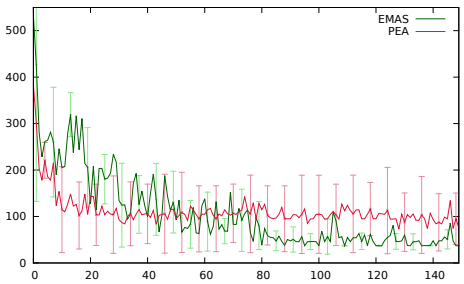
Relatively high dispersion of the results obtained calls for a detailed analysis of the problem stated, and possibly to employ more sophisticated methods. For example, Krzywicki in [116] has already tried enhancing EMAS with niching techniques, so this approach may be utilised in future research, enhancing both classical and memetic versions of the examined systems, in order to reach and clearly report more than one extremum of the optimised problem.

Apart from visual assessment of the fitness results, an insight into the actual solutions is also necessary. In Table 4.12 the fitness value in the last (150th) step of the computation was presented. It should be noted that the best result was reached by EMAS without modifications. The next one was apparently EMAS with Baldwinian memetics. However high dispersion of this results points out that it should be disqualified. In the case of Lamarckian memetics, the final result obtained by EMAS seems to be worse than in the case of PEA.

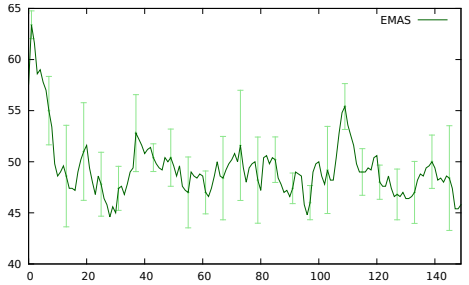
Table 4.12. Final results obtained by the researched systems

System	Fitness	Standard deviation
PEA	75.6	35.51
PEA + Baldwin	93.6	68.91
PEA + Lamarck	77.4	13.82
EMAS	**37.2**	**0.4**
EMAS + Baldwin	47	59.39
EMAS + Lamarck	80.4	33.83

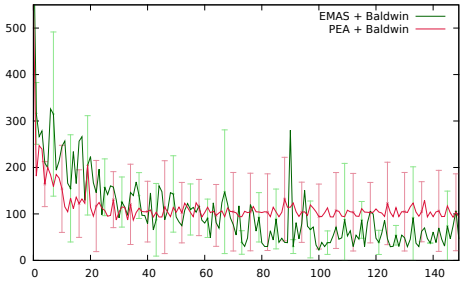
To sum up, of optimisation of game advisory strategy considered in the case of Sudoku solving turned out to be an interesting black-box problem that is apparently well-suited for testing such complex computing systems as EMAS and its modifications. Computing with classical PEA turned out to be worse than the proposed agent-based metaheuristics, besides one test case (Lamarckian memetics).



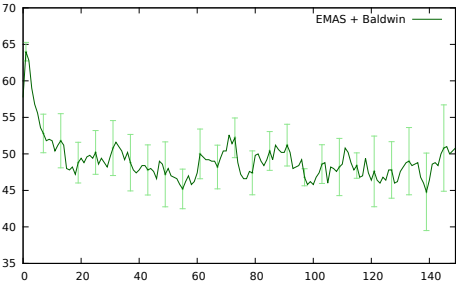
(a) EMAS and PEA fitness



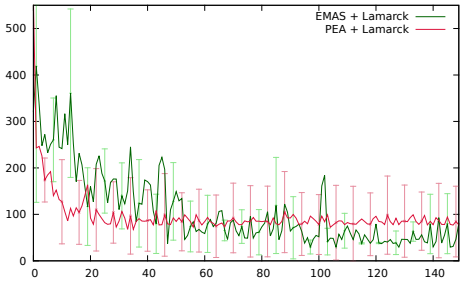
(b) EMAS agent count



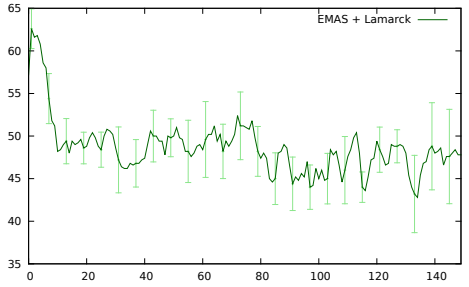
(c) EMAS and PEA (Baldwin) fitness



(d) EMAS + Baldwin agent count

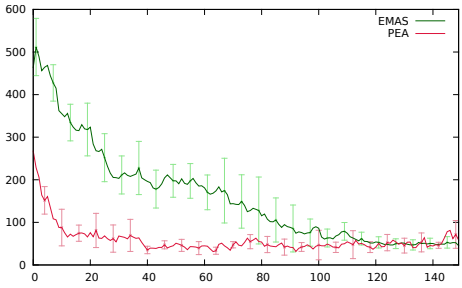


(e) EMAS and PEA (Lamarck) fitness

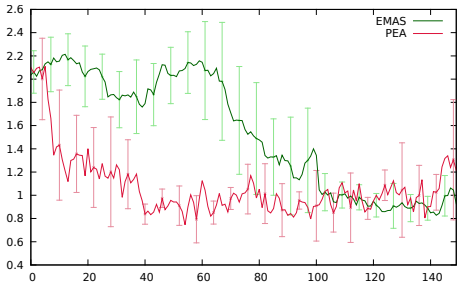


(f) EMAS + Lamarck agent count

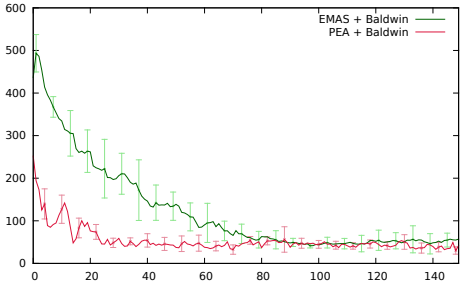
Figure 4.40. Fitness and agent count obtained for all tested systems



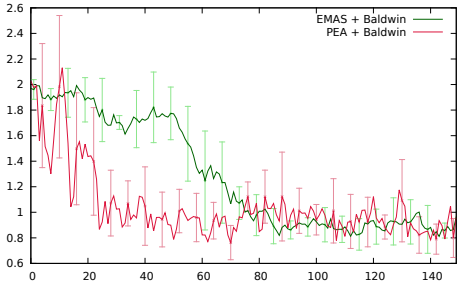
(a) EMAS and PEA MSD diversity



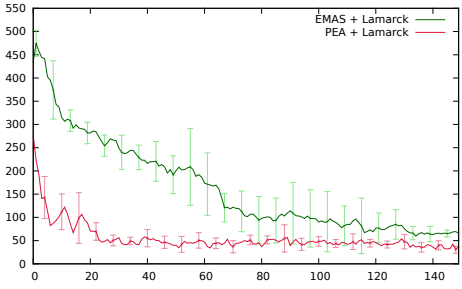
(b) EMAS and PEA MOI diversity



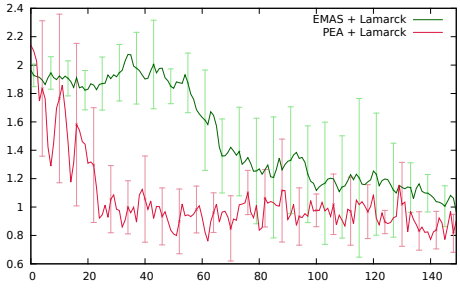
(c) EMAS and PEA (Baldwin) MSD diversity



(d) EMAS and PEA (Baldwin) MOI diversity



(e) EMAS and PEA (Lamarck) MSD diversity



(f) EMAS and PEA (Lamarck) MOI diversity

Figure 4.41. Diversity obtained for all tested systems

4.4. Goals attained in experimental verification

In this chapter, experimental analysis of different aspects of agent-based metaheuristics was performed. Having evaluated the systems of interest, which use selected classical multidimensional benchmark functions (versus parallel evolutionary algorithm), attention was focused on a comparison between memetic versions of EMAS and similarly configured PEA. Both series of experiments showed that EMAS outperforms its competitor by yielding better results for the problems. Moreover, it utilises significantly lower number of individuals.

The immunological selection mechanism was also tested, and the outcome of a comparison between EMAS and iEMAS resulted in the observation that though iEMAS produces results with a little worse fitness, the number of agents in the population is significantly lower, making this a good tool for solving problems with costly fitness functions.

Next, an evaluation of the influence of different parameters of EMAS and iEMAS was made. The results allow the formulation of concise guidelines on the tuning of the computation based on EMAS or iEMAS and applying them to certain problems.

Finally, two real-world problems: Sudoku strategy parameters optimisation and step and flash imprint lithography inverse problem were considered. Again, EMAS turned out to produce better results than PEA for both problems.

Summary

Solving difficult search and optimisation problems (e.g. "black-box") requires non-deterministic approaches, in particular metaheuristics. It may be said that these methods trade-off precision, quality, accuracy and execution time in favour of computational effort. Such methods usually referred to as the *methods of last resort* are necessary for dealing with difficult problems.

Following *no free lunch theorem* mentioned earlier, it turns out that looking for novel metaheuristics for particular problems will always be necessary. Therefore, the existing metaheuristics are proposed to be enhanced with agency, constituting EMAS (proposed by Cetnarowicz) and its modifications (memetic and immunological), which yielded original results presented in this monograph. Moreover, it should be noted that developing novel computing methods calls for the development of a detailed formal model that will ease the understanding of the method, and provide a base for further analytical deliberations, helpful in getting a deeper insight into the features of the system. Accurate testing of the proposed metaheuristics is also necessary, in particular tuning the system parameters may be fruitful for further application of this metaheuristics to other problems.

This monograph starts with a state-of-the-art review of difficult search problems and existing methods of dealing with them, in particular recalling the ones that serve as an inspiration for metaheuristics discussed later. After discussing popular evolutionary and non-evolutionary methods, highlighting the hybrid ones, the architectures of agent-based computing systems is outlined. Deliberations on existing needs in the domain of theory and practice finish the first part of the work, giving a sound base for further discussion.

EMAS is treated as a starting point for further hybridisation and extensions, so in this way memetic EMAS hybrids and immunological EMAS are discussed, which have already proved to be general-purpose search algorithms.

A general Markov-chain based stochastic model related to the agent search and optimisation, inspired by Michael Vose's SGA model is constructed. The description begins with giving data structures and management strategies, which results in defin-

ing the space of states and the framework for describing stochastic agents' actions. Later, the stationary Markov chain for memetic EMAS is defined, including the definition of probabilistic transition function and Markov kernels for sample actions. Then an analysis of dependencies between the actions is performed and notions of local and global actions are given. The same structure is repeated for defining iEMAS.

Theoretical results obtained in this research are helpful in studying important features of stochastic global optimisation metaheuristics, conforming EMAS architecture and implemented as a concurrent system in a distributed computing environment. The proper synchronisation among agents respecting the classification of the agent's actions ensures *safeness*. Moreover, the safe, coarse-grained parallel computation and its effective, suboptimal scheduling in a distributed computer environment (computer cluster) may be obtained. The proved strong ergodicity of the finite state Markov chain modelling the metaheuristics causes that it can reach an arbitrary state (arbitrary population) in the finite number of iteration with the probability one which implies the asymptotic stochastic guarantee of success. This condition mainly imposes *liveness* of the metaheuristic. Similar conclusions may be drawn from the ergodic conjecture prepared for iEMAS and an outline of the proof. This shows that the proposed way of modelling is extensible and applicable to more similar metaheuristics.

The experimental results were conducted for EMAS and PEA, their memetic versions and iEMAS. General outcome of these experiments is that EMAS tends to be better at solving difficult (e.g. multi-dimensional) problems than a parallel evolutionary algorithm (the experiments were prepared in such a way that both these systems could be compared—the selection, population structure, variation operators etc.). Memetic versions of EMAS and PEA were also examined and the results show that effects of Lamarckian memetics makes EMAS better in almost all tested problems, while Baldwinian memetics proved an effective method only in several cases. As far as efficiency of the examined systems is concerned, EMAS turned out to be much more effective in terms of fitness function calls. This makes these systems a promising weapon in dealing with problems with costly fitness function.

The experiments were devoted to solving not only classical benchmark problems, but also several tasks that may be perceived as real-life. Namely, solving inverse problem of step and flash imprint lithography (this was one of the cases when the fitness function was costly), and a parametric optimisation of Sudoku solving strategies were considered. EMAS prevailed in all these problems as a more effective search technique. The final series of experiments considered examination of the influence of different parameters of EMAS and its modifications on its efficiency.

To sum up, the original results presented in this monograph should be divided into two aspects: formal analysis and experimental research.

- Formal analysis:
  - Structure and behaviour models for EMAS (Section 2.2.2) and iEMAS (Section 2.3.2).
  - Synchronisation mechanism for EMAS (Section 2.2.4) and iEMAS 2.3.3).
  - Model of dynamics for EMAS (Section 3.1.1) and iEMAS (Section 3.2.1).
  - Detailed definition of EMAS actions (Section 2.2.3).
  - Formulation of ergodic theorems for EMAS (Section 3.1.2) and iEMAS (Section 3.2.2).
  - Formal proof of ergodicity for EMAS (Section 3.1.2).
  - Formulation of ergodic conjecture for iEMAS (Section 3.2.2).
  - Estimation of lower bounds for probabilities and upper bounds for the number of steps for particular stages in EMAS ergodicity proof (Appendix B).
- Experimental research:
  - Comparing EMAS and PEA (classical and memetic versions) efficiency for selected benchmark problems (Section 4.1).
  - Comparing selected additional parameters such as computing step execution time, diversity, number of fitness function calls per step etc. (Section 4.1).
  - Comparing EMAS and iEMAS efficiency (Section 4.1).
  - Tuning of energetic and probabilistic decision parameters for EMAS (Section 4.2).
  - Tuning of iEMAS parameters (Section 4.2).
  - EMAS and PEA efficiency for selected real-world problems such as step and flash imprint lithography inverse problem and advisory strategy parameters optimisation (Section 4.3).

In the future, the stochastic models presented in this monograph are planned to be further extended to cover other algorithms (e.g. parallel evolutionary algorithm). Conducting the full formal proof for iEMAS ergodicity is also envisaged. Regarding the experimental verification of the systems, other difficult problems are planned to be approached with EMAS and related search methods. For example, combinatorial optimisation or dynamic optimisation will be tackled.

Both practical and theoretical results of this monograph are believed to be an important starting point for adaptation and appropriate configuration of agent-based metaheuristics for solving other problems, which have not been considered here, and for further development of stochastic models for complex systems.

A. Experimental configuration details

Classical EMAS and PEA

Common configuration of the algorithms

- Representation: real-valued.
- Mutation: normal distribution-based modification of one randomly chosen gene.
- Crossover: single-point.
- Migration topology: 3 fully connected islands.
- Migration probability: 0.01 per agent/individual (each one migrates independently—possibly to different islands).

EMAS configuration

- Initial energy: 100 units received by the agents in the beginning of their lives.
- Evaluation energy win/loose: 20 units passed from the looser to the winner.
- Minimal reproduction energy: 90 units required to reproduce.
- Death energy level: 0, such agents should be removed from the system.
- Boundary condition for the intra-island lattice: fixed, the agents cannot cross the borders.
- Intra-island neighbourhood: Moore's, each agent's neighbourhood consists of 8 surrounding cells.
- Size of 2-dimensional lattice as an environment: 10x10.
- Stop condition: 100000 steps of experiment.
- Benchmark problem: Rastrigin function [49].
- Problem size: 100 dimensions.
- Population size configurations:
  - 25 individuals on 1 island,
  - 25 individuals on 3 islands,
  - 40 individuals on 1 island,
  - 40 individuals on 3 islands.

EMAS variants and parameters tuning

The configuration of the tested systems is presented as follows.

- Common parameters: normal distribution-based mutation of one randomly chosen gene, single-point crossover, the descendant gets parts of its parents genotype after dividing them in one randomly chosen point, 30 individuals located on each island, all experiments were repeated 30 times and standard deviation (or other statistical measures, such as median and appropriate quartiles for box-and-whiskers plots) was computed; allopatric speciation (island model), 3 fully connected islands, 3000 steps of experiment, genotype of length 50, agent/individual migration probability 0.01.
- PEA-only parameters: mating pool size equals to the number of individuals, individuals migrate independently (to different islands).
- EMAS-only parameters: initial energy: 100, received by the agents in the beginning of their lives, minimal reproduction energy: 90, required to reproduce, evaluation energy win/loose: $40/-40$, passed from the looser to the winner, death energy level: 0, used to decide which agent should be removed from the system, boundary condition for the intra-island lattice: fixed, the agents cannot cross the borders, intra-island neighbourhood: Moore's, each agent's neighbourhood consists of 8 surrounding cells, size of 2-dimensional lattice as an environment: 10×10, all agents that decided to emigrate from one island, will immigrate to another island together (the same for all of them).

iEMAS

The following parameters were chosen for iEMAS:

- Energy taken by a lymphocyte from similar agent: 30
- Good fitness factor: 0.97 (percentage of the agent fitness related to average fitness in the population, as minimisation is considered, if fitness is smaller than average fitness, it is considered as "good").
- Similarity measure: Mahalanobis distance [123].
- Similarity threshold: 7.3, if similarity is smaller than this the lymphocyte is considered to be similar to the tested agent.
- Immaturity duration for lymphocyte: 10.
- Maturity duration for lymphocyte: 20.
- Lymphocytes cannot migrate between the islands.

Step and flash imprint lithography

Common parameters:

- Mutation: uniform distribution modification of one randomly chosen gene.
- Crossover: single-point, the descendant gets parts of its parents genotype after dividing them in one randomly chosen point.
- Population size: 15.
- As a platform technical feature, global cache for computing fitness values to speed up calculations.

EA parameters:

- number of steps: 1000,
- mating pool size equals to number of individuals,

EMAS parameters:

- number of steps: 10000,
- initial energy: 100, received by the agents in the beginning of their lives,
- minimal reproduction energy: 90, required to reproduce,
- evaluation energy win/loose: 20/-20, passed from the looser to the winner,
- death energy level: 0, used to decide which agent should be removed from the system.
- intra-island neighbourhood: Moore's, each agent's neighbourhood consists of 8 surrounding cells.
- size of 2-dimensional lattice as an environment: 10x10.

Sudoku solving strategy parameters optimisation

The configuration of the tested systems is presented as follows.

- Common parameters: normal distribution-based mutation of one randomly chosen gene, single-point crossover, the descendant gets parts of its parents genotype after dividing them in one randomly chosen point, 15 individuals located on each island, all experiments were repeated 30 times and standard deviation (or other statistical measures, such as median and appropriate quartiles for box-and-whiskers plots) was computed; allopatric speciation (island model), 3 fully connected islands, 150 steps of experiment, genotype of length 4, agent/individual migration probability 0.01.
- PEA-only parameters: mating pool size: 8, individuals migrate independently (to different islands).

- EMAS-only parameters: initial energy: 100, received by the agents in the beginning of their lives, minimal reproduction energy: 90, required to reproduce, evaluation energy win/loose: $40/-40$, passed from the looser to the winner, death energy level: 0, used to decide which agent should be removed from the system, boundary condition for the intra-island lattice: fixed, the agents cannot cross the borders, intra-island neighbourhood: Moore's, each agent's neighbourhood consists of 8 surrounding cells, size of 2-dimensional lattice as an environment: 10×10, all agents that decided to emigrate from one island, will immigrate to another island together (the same for all of them).

B. Technical details of EMAS ergodicity proof

The detailed estimation of lower bound for probabilities and upper bound for number of steps for the stages of proof of Theorem 3.1.1 is preceded by a series of useful technical lemmas.

Lemma B.0.1 ([25, Lemma B.1])**.** *Given the assumptions of Theorem 3.1.1, there exists a positive constant* $0 < \zeta_0 \leqslant \frac{1}{2}$ *such that* $\zeta_0 \leqslant \zeta^{gl}(x)$ *and* $\zeta_0 \leqslant \zeta^{loc}(x)$ *for all* $x \in X$.

Proof. According to Formula (3.2)

$$\zeta^{loc}(x) = \sum_{i \in Loc} locsel(x)(i) \cdot \xi_i(x)$$

where

$$\xi_i(x) = \sum_{gen \in U} \sum_{n \in P_i} agsel_i(x)(gen, n) \cdot \omega(x, gen)(Act_{loc})$$

See Eq. (3.1). Because $Act_{loc} \neq \emptyset$ and there always exists at least one agent (see Remark 3.1.2) we have

$$\xi_i(x) \geqslant \iota_{agsel} \cdot \iota_{\omega}$$

for all $x \in X$ and $i = 1, \ldots, s$. Finally, because at least one location contains an agent (see Remark 3.1.2 again) we may evaluate

$$\zeta^{loc}(x) \geqslant \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_{\omega} = \zeta_0$$

for all $x \in X$. Replacing Act_{loc} by Act_{gl} ($Act_{gl} \neq \emptyset$) we similarly obtain

$$\zeta^{gl}(x) \geqslant \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_{\omega} = \zeta_0$$

for all $x \in X$. The constant ζ_0 is strictly positive as it is the product of strictly positive numbers. Moreover $2\zeta_0 \leqslant \zeta^{gl}(x) + \zeta^{loc}(x) = 1$ for all $x \in X$, so $\zeta_0 \leqslant \frac{1}{2}$. □

Lemma B.0.2 ([25, Lemma B.2]). *$\forall gen \in U,\ l, l' \in Loc; l \neq l',\ n \in P_l,\ n' \in P_{l'}, x, x' \in X$ so that $x(l, gen, n) > e_{migr},\ x(l', gen, n') = 0$, $x'(l', gen, n') = x(l, gen, n)$ we have*

$$\varrho_{migr}^{gen,n}(x)(x') \geq \iota_{migr} = \frac{1}{(s-1)\max_{i \in Loc}\{q_i\}}.$$

In other words, assuming the current state x, and the particular agent $ag_{gen,n}$ ready to migrate from this state, all states x' containing this agent located in other locations than in x are reachable with probability greater than or equal to ι_{migr}.

Proof. It follows straightforwardly from Eq. (2.29) describing the $migr$ action's kernel $\varrho_{migr}^{gen,n}$. □

Lemma B.0.3 ([25, Lemma B.3]). *If $\exists \iota_{mut} > 0;\ mut(gen)(gen') \geqslant \iota_{mut}\ \forall gen, gen' \in U$ then $\exists\ \iota_{clo} > 0$ such that $\varrho_{clo}^{gen,n}(x)(x') \geqslant \iota_{clo}$ for each quadruple $(gen, n, x, x'),\ gen \in U,\ x, x' \in X$ and for a certain location $l \in Loc$ satisfying:*

- *$x(l, gen, n) > e_{repr},\ n \in P_l$,*
- *$\exists\ gen' \in U, n' \in P_l$; $x'(l, gen', n') = \Delta e, x(l, gen', n') = 0$,*
- *$\sum_{a \in U, b \in P_l}[x(l, a, b) > 0] < q_l$.*

Roughly speaking, assuming the current state x, and the particular agent $ag_{gen,n}$ ready for cloning in this state, all states x' containing an additional agent (cloned by $ag_{gen,n}$) are reachable with probability greater than or equal to ι_{clo} independently upon x and x'. Of course, the set of possible x' may be empty if the location of the cloning agent is full in the state x.

Proof. First of all, we may observe that if x satisfies the assumptions of the lemma, then the decision is positively evaluated, i.e. $\delta_{clo}^{gen,n}(x)(1) = 1$. On the other hand, the third assumption implies that there is at least one inactive agent in the system. Let us denote the signature of this agent by (gen', n'). If it was activated in the i-th location by the cloning operation, then the next state would satisfy $x'(l, gen', n') = \Delta e$ with probability 1. Then according to Eq. (2.25) the set $FC_{l,gen'}$ would be non-empty. Furthermore, taking into account (2.33), (2.32) we obtain

$$\varrho_{clo}^{gen,n}(x)(x') = \vartheta_{clo}^{gen,n}(x)(x') > \frac{\iota_{mut}}{\max_{i \in Loc}\{q_i\} - 1} = \iota_{clo} > 0.$$

□

Lemma B.0.4 ([25, Lemma B.4])**.** *If* $\exists\ \iota_{cmp} > 0$; $cmp(gen, gen') \geqslant \iota_{cmp}\ \forall gen, gen' \in U$ *then* $\exists\ \iota_{get} > 0$ *such that* $\varrho_{get}^{gen,n}(x)(x') \geqslant \iota_{get}$ *and* $\varrho_{get}^{gen,n}(x)(x'') \geqslant \iota_{get}$ *for each tuple* (gen, n, x, x', x''), $gen \in U$, $n \in P_l$, $x, x', x'' \in X$ *and for a certain location* $l \in Loc$ *satisfying:*

- $x(l, gen, n) > 0,\ n \in P_l$,
- $\exists\, gen' \in U, n' \in P_l$;
  $(gen, n) \neq (gen', n'),\ x(l, gen', n') > 0$,
- $x'(l, gen', n') = x(l, gen', n') + \Delta e$,
  $x'(l, gen, n) = x(l, gen, n) - \Delta e$,
  $x''(l, gen', n') = x(l, gen', n') - \Delta e$,
  $x''(l, gen, n) = x(l, gen, n) + \Delta e$,
  $x''(l, j, k) = x'(l, j, k) = x(l, j, k)$,
  $\forall\, j \neq gen, j \neq gen', k \neq n, k \neq n'$.

In other words, assuming the current state x*, and a pair of agents* $ag_{gen,n}, ag_{gen',n'}$ *active in this state in the same location, both states* x', x'' *in which the agent* $ag_{gen,n}$ *takes or gives the quantum* Δe *of energy to/from its neighbor* $ag_{gen',n'}$ *are reachable with probability greater than or equal to* ι_{get}*. This probability does not depend on the state* x *or pair of the neighboring agents* $ag_{gen,n}, ag_{gen',n'}$*.*

Proof. First of all, we may observe that if x satisfies the conditions assumed in the lemma then the decision is positively evaluated $\delta_{get}^{gen,n}(x)(1) = 1$, because $NBAG_{l,gen,n}$ (see Eq. (2.17)) is non-empty. Then according to Eqs. (2.18)–(2.20)

$$\varrho_{get}^{gen,n}(x)(x') = \vartheta_{get}^{gen,n}(x)(x') > \frac{\iota_{cmp}}{(\max_{i \in Loc}\{q_i\}) - 1} = \iota_{get} > 0.$$

The same reasoning leads to $\varrho_{get}^{gen,n}(x)(x'') > \iota_{get}$. □

Lemma B.0.5 ([25, Lemma B.5])**.** *Let* A_i *be an event (e.g. denoting that certain agents perform certain actions) in the* i*-th step. Then* $A_1 \cap \ldots \cap A_k$ *is an event consisting of events* $A_1, \ldots, A_k$ *taking place consecutively in subsequent steps* $1, \ldots, k$*. If* $P(A_1) > \lambda_1 > 0$ *and the conditional probabilities* $P(A_i | \bigcap_{j=1}^{i-1} A_j)$ *are bounded from below by* $\lambda_i > 0$ *for* $i = 2, \ldots, k$*, then*

$$P\left(\bigcap_{i=1}^{k} A_i\right) \geqslant \prod_{i=1}^{k} \lambda_i > 0. \tag{B.1}$$

Proof. Considering the sequence of (possibly dependent) events $A_1, \ldots, A_k$ and starting from the well-known conditional probability formula $P(A_1 \cap A_2) = P(A_1) \cdot$

$P(A_2|A_1)$, the following equation

$$P\left(\bigcap_{i=1}^{k} A_i\right) = P(A_1) \cdot \prod_{i=2}^{k} P\left(A_i \middle| \bigcap_{j=1}^{i-1} A_j\right) \tag{B.2}$$

may be proved inductively, which is enough to prove the lemma. □

Proof of Theorem 3.1.1 – detailed estimations. It will suffice to show that the passage between two arbitrary EMAS states ($x_b, x_e \in X$) may be performed in a finite number of steps with a positive probability. It was already shown (see *Outline od the proof of Theorem* 3.1.1, Section 3.1.2) that the passage mentioned above may be performed by the sequence of stages 1–5 described there and illustrated in Fig. 3.1. Now it is enough to estimate the upper bounds for the number of steps required to perform each of these stages and then the lower bound of the probability of series of actions executed in these steps.

Lemma B.0.6 ([25, Lemma B.6])**.** *Given the assumptions of Theorem 3.1.1, Stage 0 requires at most $st_0 = m - 1$ steps in parallel taken with probability greater than or equal to $pr_0 > 0$, where m stands for the number of possible energy values that might be possessed by the agent (see Section 2.2.2).*

Proof. In each parallel step performed during this stage we divide the set of locations into three distinct sets:

- $Loc_{\emptyset}$: empty locations (containing no active agents) – there is no activity there.

- Loc_{get}: locations containing the maximal number of agents (q_i): one of the existing agents ag_{gen_i,n_i} is selected and it performs a sequence of get actions in order to remove one of its neighbours $ag_{gen'_i,n'_i}$. Both agents are fixed during the *Stage 0*. After removing its neighbour, the agent begins to perform the global action $migr$ and fails (rejected to do it by MA), until the end of the *Stage 0*.

- $Loc_{migr} = Loc \setminus (Loc_{\emptyset} \cup Loc_{get})$: other locations in which one of the existing agents performs the global action $migr$ and fails (rejected by MA), until the end of the Stage 0.

The probability of performing a single step of the described sequence is given by:

$$\begin{aligned}
&\zeta^{loc}(z) \prod_{i \in Loc_{get}} \big(agsel_i(z)(gen_i, n_i) \cdot \omega(gen_i, z)(get) \cdot \\
&\qquad \delta_{get}^{gen_i, n_i}(z)(1) \cdot \varrho_{get}^{gen_i, n_i}(z_i')(z_i'')\big) \\
&\prod_{i \in Loc_{migr}} \big(agsel_i(z)(gen_i, n_i) \cdot \omega(gen_i, z)(migr)\big) \\
&\geqslant \zeta_0 \prod_{i \in Loc_{get}} (\iota_{agsel} \cdot \iota_{\omega} \cdot \iota_{get}) \cdot \prod_{i \in Loc_{migr}} (\iota_{agsel} \cdot \iota_{\omega}) \\
&\qquad \geqslant \zeta_0 \left(\iota_{agsel} \cdot \iota_{\omega} \cdot \iota_{get}\right)^s
\end{aligned} \tag{B.3}$$

where:

- z is the current state of the system. It is set to x_b in the beginning of *Stage 0* and the final state of this stage z equals to x_{01},
- z_i', z_i'' are intermediate states that make it possible to express the parallel execution of the action get in all locations in Loc_{get}. The state z_i'' is a result of execution of the action get in the i-th location, starting from the state z_i'. The order of locations from Loc_{get} in Eq. (B.3) is arbitrary, because the get action is local (see Observation 2.2.3). $z_i' = z$ for the first location from Loc_{get}, $z_j' = z_i''$ where j is a location next to i in Eq. (B.3). The state z becomes z_i'' at the end of the sequence of intermediate steps.

It is easy to see that the estimation given in Eq. (B.3) is uniform for all steps in *Stage 0*. The maximal number of steps is bounded from above by $st_0 = m - 1$ because it is maximum possible energy for an agent.

Assume A_t is an event that consists in performing the t-th step described above, according to Lemma B.0.5, the probability of the whole sequence can be bounded from below by:

$$pr_0 = \left(\zeta_0 \left(\iota_{agsel} \cdot \iota_{\omega} \cdot \iota_{get}\right)^s\right)^{m-1} > 0 \tag{B.4}$$

□

Lemma B.0.7 ([25, Observation B.7])**.** *Given the assumptions of Theorem 3.1.1 and assuming Stage 0 to have been completed, Stage 1 requires at most* $st_1 = s(m-1) + s - 1$ *steps taken with probability greater than or equal to* $pr_1 > 0$.

Proof. Following the assumptions of Theorem 3.1.1 there must be at least one location $i \in Loc$ with the total amount of energy greater than the migration threshold

e_{migr} (of course at least one agent must be present there). An agent ag_{gen_1,n_1} is chosen from this location.

At each step of this stage the set of locations can be divided into three distinct sets:

- $Loc_{\emptyset}$: empty locations (containing no active agents)— there is no activity there.
- Loc_{get}: $\#Loc_{get} = 1$, single location containing the agent chosen at the beginning of the stage that performs a sequence of get actions in order to remove its neighbors (if they exist) in this location. Following the proof of Lemma B.0.6 we may estimate the number of steps necessary to perform this sequence as $m - 1$.
- Loc_{migr}: other locations. The agents present in other non-empty locations are trying to perform the global action $migr$ but their requests are rejected by MA.

The probability of performing one action get in the sequence discussed is given by:

$$\begin{aligned}\zeta^{loc}(z)\Big(agsel_i(z)(gen_1,n_\xi)\cdot\omega(gen_1,z)(get)\cdot\delta^{gen_1,n_\xi}_{get}(z)(1)\cdot\varrho^{gen_1,n_\xi}_{get}(z)(z')\Big)\\ \cdot\prod_{j\in Loc_{migr}}\big(agsel_j(x)(gen_j,n_j)\cdot\omega(gen_j,z)(migr)\big)\\ \geqslant\zeta_0\left(\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\right)\cdot\prod_{i\in Loc_{migr}}\left(\iota_{agsel}\cdot\iota_\omega\right)\geqslant\zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\end{aligned}\tag{B.5}$$

where z is the current state of the system. It is set to x_{01} at the beginning of *Stage 1* and the final state of this stage z equals x_{12}. The copy number of the chosen agent n_ξ is equal to n_1 at the beginning of *Stage 1* and then changes according to the migration rule becoming $n_1' \in P_{i_1}$ at the end of this stage. Note that the estimation given by Eq. (B.5) does not depend on the number of the step in the *Stage 1*. Then, the probability of removing all agents from a single location is given by:

$$\left(\zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\right)^{m-1}\tag{B.6}$$

After the removal of all its neighbours the chosen agent has to perform migration. The probability of this step is given by:

$$\begin{aligned}&\zeta^{gl}(z)\cdot locsel(z)(i)\\ &\left(agsel_i(z)(gen_1,n_\xi)\cdot\omega(gen_1,z)(migr)\delta^{gen_1,n_\xi}_{migr}(z)(1)\cdot\varrho^{gen_1,n_\xi}_{migr}(z)(z'')\right)\\ &\geqslant\zeta_0\cdot\iota_{locsel}\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{migr}\end{aligned}\tag{B.7}$$

where z'' is a state obtained after migrating of the chosen agent from one location to another.

Let us assume that A_t is an event that consists in removing all agents and migrating between the locations in the consecutive steps. The probability of each A_t may be evaluated by Eqs. (B.6)–(B.7). According to Lemma B.0.5, the probability of the whole sequence can be bounded from below by:

$$pr_1 = (\zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{get})^{s\cdot(m-1)} (\zeta_0 \cdot \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{migr})^{s-1} \tag{B.8}$$

Moreover, the number of steps in the sequence may be estimated by the constant $st_1 = s(m-1) + s - 1$. □

Lemma B.0.8 ([25, Lemma B.8])**.** *Given the assumptions of Theorem 3.1.1 and assuming Stage 1 to have been completed, Stage 2 requires at most $st_2 = m$ steps taken with probability greater or equal to $pr_2 > 0$.*

Proof. There is only one agent $ag_{gen_1,n_1'}$ in the system, so in order to produce another agent using clo it needs to perform 1 step. Then, it passes all its energy to its offspring by performing get action $(m-1)$ times.

The probability of the first step of this sequence is as follows:

$$\begin{aligned} \zeta^{loc}(x_{12}) \cdot agsel_{i_1}(x_{12})(gen_1, n_1') \cdot \omega(gen_1, x_{12})(clo) \cdot \\ \delta_{clo}^{gen_1,n_1'}(x_{12})(1) \cdot \varrho_{clo}^{gen_1,n_1'}(x_{12})(z') \\ \geqslant \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo} \end{aligned} \tag{B.9}$$

where z' is the state where ag_{gen_2,n_2} was introduced into the system after performing the cloning action by $ag_{gen_1,n_1'}$.

Now after cloning, the agent $ag_{gen_1,n_1'}$ performs at most $(m-1)$ times get action to pass all its energy to ag_{gen_2,n_2}. A single step of this sequence has the following probability:

$$\begin{aligned} \zeta^{loc}(z)\Big(agsel_{i_1}(z)(gen_1, n_1') \cdot \omega(gen_1, z)(get) \cdot \delta_{get}^{gen_1,n_1'}(1) \cdot \varrho_{get}^{gen_1,n_1'}(z)(z'') \Big) \\ \geqslant \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{get} \end{aligned} \tag{B.10}$$

where z is the current state (at the end of the stage z will be equal to x_{23}) and z'' is the state after $ag_{gen_1,n_1'}$ passed a part of its energy to ag_{gen_2,n_2}.

Assuming A_t is an event that consists in performing the t-th step of the sequence described above, according to Lemma B.0.5, the probability of *Stage 2* will be evaluated from below by:

$$pr_2 = \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo} \cdot (\zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{get})^{m-1} > 0 \tag{B.11}$$

and $st_2 = 1 + (m-1) = m$. □

Lemma B.0.9 ([25, Lemma B.9])**.** *Given the assumptions of Theorem 3.1.1 and assuming Stage 2 to have been completed, Stage 3 requires at most* $st_3 = (m+1)^s$ *steps taken with probability greater than or equal to* $pr_3 > 0$*.*

Proof. Assuming *Stage 2* to have been completed, there is only one non-empty location i_1 containing agent ag_{gen_2,n_2}. The agent starts the second round of migration by performing the action $migr$. It is assumed that the path of this round is composed of a sequence of locations that ends at the i_2-th location. The location i_2 is one of the non-empty locations in the state x_e that contains total energy higher than the migration threshold e_{migr}. The length of this path is at most s. Note that the path is not intersecting, with the possible exception of the last location.

The probability of migration of the agent ag_{gen_2,n_2} between current location i to the next location of this path is given by:

$$\begin{gathered}\zeta^{gl}(z) \cdot locsel(z)(i) \\ \left(agsel_i(z)(gen_2, n_\xi) \cdot \omega(gen_2, z)(migr) \delta_{migr}^{gen_2,n_\xi}(z)(1) \cdot \varrho_{migr}^{gen_2,n_\xi}(z)(z') \right) \\ \geqslant \zeta_0 \cdot \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{migr} \end{gathered} \tag{B.12}$$

where z is the current state (in the beginning $z = x_{23}$ at the end of the stage z will be equal to x_{34}), z' is a state obtained after migrating of the agent ag_{gen_2,n_ξ} from current location i to the next one along the mentioned path, $n_\xi \in P_i$ is the current copy number of the migrating agent.

$Loc_{migr} \subset Loc$ will denote a non-empty subset of locations which does not contain the current location in which $ag_{gen_i^{first},n_i^{first}}$ is cloned or feeded by the life energy ($i \notin Loc_{migr}$). We assume that certain agents ag_{gen_j,n_j}, $j \in Loc_{migr}$ try to perform global action $migr$ and their requests are rejected by the MA.

The probability of each step in which the new agent $ag_{gen_i^{first},n_i^{first}}$ is produced is evaluated by:

$$\begin{gathered}\zeta^{loc}(z) \cdot agsel_i(z)(gen_2, n_\xi) \\ \cdot \omega(gen_2, z)(clo) \cdot \delta_{clo}^{gen_2,n_\xi}(z)(1) \cdot \varrho_{clo}^{gen_2,n_\xi}(z)(z') \\ \prod_{j \in Loc_{migr}} (agsel_j(z)(gen_j, n_j) \cdot \omega(gen_j, z)(migr)) \\ \geqslant \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo} \cdot (\iota_{agsel} \cdot \iota_\omega)^{s-1} \end{gathered} \tag{B.13}$$

where z is the current state and z' is the state in which $ag_{gen_i^{first},n_i^{first}}$ is created in the system and again $n_\xi \in P_i$ is the current number of copy of the migrating agent.

Now the agent passes the sufficient amount of energy (required to bring the total sum of energy of the i-th location to the value perceived in the system state x_e) to

agent $ag_{gen_i^{first},n_i^{first}}$ by performing at most $(m-1)$ times get. The probability of one get action is here as follows:

$$\zeta^{loc}(z)\Bigg(agsel_i(z)(gen_2,n_\xi)\cdot\omega(gen_2,z)(get)\cdot\delta_{get}^{gen_2,n_\xi}(z)(1)\cdot\varrho_{get}^{gen_2,n_\xi}(z)(z'')\Bigg)$$
$$\prod_{j\in Loc_{migr}}(agsel_j(z)(gen_j,n_j)\cdot\omega(gen_j,z)(migr))$$
$$\geqslant\zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\cdot(\iota_{agsel}\cdot\iota_\omega)^{s-1} \tag{B.14}$$

where z is the current state, z'' is the state where ag_{gen_2,n_ξ} passed a part of its energy to $ag_{gen_i^{first},n_i^{first}}$ and n_ξ as in the previous Eq. (B.13).

Assuming A_t to be the consecutive t-th event described above, according to Lemma B.0.5, the probability of the whole *Stage 3* will be bounded from below by:

$$pr_3=\Bigg(\zeta_0\cdot\iota_{locsel}\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{migr}\cdot\zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{clo}\cdot(\iota_{agsel}\cdot\iota_\omega)^{s-1}\Bigg)^{s-1}$$
$$\Big(\zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\cdot(\iota_{agsel}\cdot\iota_\omega)^{s-1}\Big)^{m-1}>0. \tag{B.15}$$

The number of steps required for performing the whole sequence is $st_3=2\cdot(s-1)+(m-1)$. □

Lemma B.0.10 ([25, Lemma B.10])**.** *Given the assumptions of Theorem 3.1.1 and assuming Stage 3 to have been completed, Stage 4 requires at most* $st_4=\max_{i\in Loc}\{q_i\}$ *steps taken with probability greater than or equal to* $pr_4>0$.

Proof. We divide the set of locations into three distinct sets:

- Empty locations $Loc_\emptyset\subset Loc$ (containing no active agents): there is no activity there.
- Locations $Loc_{migr}\subset Loc$ in which one agent attempts to perform $migr$ action and fails (these locations contain one agent in the final state x_e or have just finished recreation of the agents in the final state).
- Other locations $Loc_{clo}\subset Loc$ in which one agent $ag_{gen_i^{first},n_i^{first}}$ performs a sequence of the clo actions in order to recreate the population of its neighbors (required in the state x_e).

The probability of this step may be evaluated by:

$$\begin{aligned}
&\zeta^{loc}(z) \prod_{i\in Loc_{clo}} \Big(agsel_i(z)(gen_i^{first}, n_i^{first}) \cdot \omega(gen_i^{first}, z)(clo) \cdot \\
&\qquad \delta_{clo}^{gen_i^{first}, n_i^{first}}(z)(1) \cdot \varrho_{clo}^{gen_i^{first}, n_i^{first}}(z')(z'') \Big) \\
&\prod_{i\in Loc_{migr}} \Big(agsel_i(z)(gen_i, n_i) \cdot \omega(gen_i, z)(migr) \Big) \\
&\geqslant \zeta_0 \prod_{i\in Loc_{clo}} (\iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo}) \prod_{i\in Loc_{migr}} (\iota_{agsel} \cdot \iota_\omega) \\
&\geqslant \zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{clo}, \iota_\omega\}\right)^s
\end{aligned} \tag{B.16}$$

where z is the current state, $z = x_{34}$ at the beginning and $z = x_{45}$ at the end of the stage, gen_i, n_i is the quasi signature of an arbitrary agent in the location $i \in Loc_{migr}$. Note that the whole sequence of the clo actions on one location may have length $\max_{i\in Loc}\{q_i\}$ in the worst case, so $st_4 = \max_{i\in Loc}\{q_i\}$.

Assuming A_t to be an event that consists in performing the t-th step of the sequence described above, according to Lemma B.0.5, the probability of the whole sequence will be bounded from below by:

$$pr_4 = \left(\zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{clo}, \iota_\omega\}\right)^s\right)^{\max_{i\in Loc}\{q_i\}} > 0. \tag{B.17}$$

□

Lemma B.0.11 ([25, Lemma B.11])**.** *Given the assumptions of Theorem 3.1.1, Stage 5 requires at most $st_5 = m - 1$ steps in parallel taken with probability greater than or equal to $pr_5 > 0$.*

Proof. We divide the set of locations into three distinct sets:

- Empty locations $Loc_\emptyset \subset Loc$ (containing no active agents): there is no activity there.
- $Loc_{migr} \subset Loc$: in these locations, the agent tries to perform the global action $migr$ but its requests are rejected by MA (these locations contain one agent in the final state x_e or have just finished redistribution of the agents' energy in the final state).
- Other locations $Loc_{get} \subset Loc$: the agent containing the highest amount of energy ($ag_{gen_i^{first}, n_i^{first}}$) performs a sequence of get actions in order to pass the sufficient amount of energy to all its neighbours (required in the state x_e).

After each agent $ag_{gen_i^{first},n_i^{first}}$ has finished distributing energy it starts to perform the global action $migr$ but its requests are rejected, and waits for other agents to finish their sequences. The locations containing exactly one agent behave in the same way. In the worst case there will be $(s-1)$ agents performing this global action.

The probability of performing one of the get action of the sequence described above is given by:

$$\begin{aligned}
&\zeta^{loc}(z) \prod_{i \in Loc_{get}} \Big(agsel_i(z)(gen_i^{first}, n_i^{first}) \cdot \\
&\qquad \omega(gen_i^{first}, z)(get) \cdot \\
&\delta_{get}^{gen_i^{first},n_i^{first}}(z)(1) \cdot \varrho_{get}^{gen_i^{first},n_i^{first}}(z')(z'') \Big) \\
&\prod_{i \in Loc_{migr}} \Big(agsel_i(z)(gen_i, n_i) \cdot \omega(gen_i, z)(migr) \Big) \\
&\geqslant \zeta_0 \prod_{i \in Loc_{get}} (\iota_{agsel} \cdot \iota_\omega \cdot \iota_{get}) \prod_{i \in Loc_{migr}} (\iota_{agsel} \cdot \iota_\omega) \\
&\geqslant \zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{get}, \iota_\omega\}\right)^s
\end{aligned} \tag{B.18}$$

where z is the current state, $z = x_{45}$ at the beginning and $z = x_e$ at the end of this stage and gen_i, n_i is the quasi signature of an arbitrary agent present in the location $i \in Loc_{migr}$.

Assuming A_t to be an event that consists in performing the t-th step of the sequence described above, according to Lemma B.0.5, the probability of the whole sequence will be bounded from below by:

$$pr_5 = \left(\zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{get}, \iota_\omega\}\right)^s\right)^{m-1} > 0. \tag{B.19}$$

□

To conclude the proof of Theorem 3.1.1 let us note that lemmas B.0.6–B.0.11 together state the fact that the total number of steps necessary for passing between states x_b and x_e is not greater than

$$st = \sum_{a=0}^{5} st_a < +\infty. \tag{B.20}$$

Let us recall that all actions taken in the consecutive stages 0 – 5 are executable, assuming the completion of the previous stages. The probability of completing each

stage $a = 1, \ldots, 5$ was bounded from below independently on the state imposed by the previous stage by $pr_a > 0$. Thus the following positive real number

$$pr = \prod_{a=0}^{5} pr_a > 0 \tag{B.21}$$

estimates from below the probability of passing from x_b to x_e. As the states were taken arbitrarily, we can show analogously that one can pass from x_e to x_b with a positive probability, which concludes the proof. □

Bibliography

[1] Alba E., Tomassini M.: *Parallelism and Evolutionary Algorithms*. IEEE Transactions on Evolutionary Computation, vol. 6, no. 5, 2002, 443–462

[2] Ali M., Storey C., Törn A.: *Application of Stochastic Global Optimization Algorithms to Practical Problems*. Journal of Optimization Theory and Applications, vol. 95, 1997, 545–563

[3] An S., Yang S., Ho S., Li T., Fu W.: *A Modified Tabu Search Method Applied to Inverse Problems*. Magnetics, IEEE Transactions on, vol. 47, no. 5, may 2011, 1234 –1237

[4] Arabas J.: *Wykłady z algorytmów ewolucyjnych (in Polish, Lectures on Evolutionary Algorithms)*. WNT, 2001

[5] Aster R., Borchers B., Thurber C.: *Parameter Estimation and Inverse Problems*. Elsevier, 2012

[6] Atallah M.: *Algorithms and Theory of Computation Handbook*. CRC Press LLC, 1999

[7] Bäck T.: *Evolutionary algorithms in theory and practice*. New York, Oxford University Press, 1996

[8] Back T., Hammel U., Schwefel H.-P.: *Evolutionary computation: Comments on the history and current state*. IEEE Trans. on Evolutionary Computation, vol. 1(1), 1997

[9] Baldwin J.: *A new factor in evolution*. American Naturalist, vol. 30, 1896, 441–451

[10] Bersini H., Varela F.: *Hints for adaptive problem solving gleaned from immune networks*. [In:] Schwefel H.-P., Männer R., (Eds.): *Parallel Problem Solving*

from Nature. Vol. 496, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1991, 343–354

[11] Beume N., Laumanns M., Rudolph G.: *Convergence Rates of (1+1) Evolutionary Multiobjective Optimization Algorithms*. [In:] Schaefer R., Cotta C., Kolodziej J., Rudolph G., (Eds.): *PPSN (1)*. Vol. 6238, Lecture Notes in Computer Science. Springer, 2010, 597–606

[12] Billingsley P.: *Probability and Measure*. John Wiley and Sons, 1987

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

[14] Bouvry P., González-Vélez H., Kołodziej J.: *Intelligent Decision Systems in Large-Scale Distributed Environments*. Springer, 2011

[15] Brownlee J.: *Clever Algorithms: Nature-Inspired Programming Recipes*. lulu.com, 2012

[16] Burke E., Kendall G., Newall J., Hart E., Ross P., Schulenburg S.: *Hyper-Heuristics: An Emerging Direction in Modern Search Technology*. [In:] Glover F., Kochenberger G., (Eds.): *Handbook of Metaheuristics*. Vol. 57, International Series in Operations Research & Management Science, Springer US, 2003, 457–474

[17] Burnet F.: *The clonal selection theory of acquired immunity*. Vanderbilt University Press, 1959

[18] Byrski A.: *Immunological Selection Mechanism in Agent-based Evolutionary Computation (in Polish: Immunologiczny mechanizm selekcji w agentowych obliczeniach ewolucyjnych)*. Ph.D. thesis, AGH University of Science and Technology, 2007

[19] Byrski A., Dreżewski R., Siwik L., Kisiel-Dorohinicki M.: *Evolutionary Multi-agent Systems*. The Knowledge Engineering Review, 2013 (accepted for printing)

[20] Byrski A., Kisiel-Dorohinicki M.: *Immunological selection mechanism in agent-based evolutionary computation*. [In:] Klopotek M. A., Wierzchon S. T., Trojanowski K., (Eds.): *Intelligent Information Processing and Web Mining : proceedings of the international IIS: IIPWM '05 conference : Gdansk, Poland*. Advances in Soft Computing. Springer Verlag, 2005, 411–415

[21] Byrski A., Kisiel-Dorohinicki M.: *Agent-Based Evolutionary and Immunological Optimization*. [In:] *Computational Science – ICCS 2007, 7th International Conference, Beijing, China, May 27–30, 2007, Proceedings*. Springer, 2007

[22] Byrski A., Kisiel-Dorohinicki M., Nawarecki E.: *Agent-Based Evolution of Neural Network Architecture*. [In:] Hamza M., (Ed.): *Proc. of the IASTED Int. Symp.: Applied Informatics*. IASTED/ACTA Press, 2002

[23] Byrski A., Kisiel-Dorohinicki M., Nawarecki E.: *Immunological selection in agent-based optimization of neural network parameters*. [In:] *Proc. of the 5th Atlantic Web Intelligent Conference – AWIC'2007 : Fontainebleau, France, June 25–27, 2007*. Springer Advances in Soft Computing 43, 2007, 62–67

[24] Byrski A., Korczyński W., Kisiel-Dorohinicki M.: *Memetic Multi-Agent Computing in Difficult Continuous Optimisation*. [In:] *Proceedings of 6th International KES Conference on Agents and Multi-agent Systems Technologies and Applications, 2013, Hue City, Vietnam, IOS Press (accepted in 2013)*. Springer

[25] Byrski A., Schaefer R., Smołka M.: *Asymptotic Guarantee of Success for Multi-Agent Memetic Systems*. Bulletin of the Polish Academy of Sciences—Technical Sciences, vol. 61, no. 1, 2013

[26] Byrski A., Schaefer R., Smołka M.: *Markov Chain Based Analysis of Agent-Based Immunological System*. Transactions on Computational Collective Intelligence, vol. 10, 2013

[27] Byrski A.: *Tuning of Agent-based Computing*. Computer Science (accepted), 2013

[28] Byrski A., Debski R., Kisiel-Dorohinicki M.: *Agent-based computing in an augmented cloud environment*. Computer Systems Science and Engineering, vol. 27, no. 1, 2012

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

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

[31] Cantú-Paz E.: *Efficient and Accurate Parallel Genetic Algorithms*. Norwell, MA, Kluwer Academic Publishers, 2000

[32] Cao Y., Wu Q.: *Convergence analysis of adaptive genetic algorithm.* [In:] *Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA 1997).* 1997, 85—89

[33] Cavicchio D.: *Adaptive search using simulated evolution.* Ph.D. thesis, University of Michigan, 1970

[34] Cetnarowicz K., Kisiel-Dorohinicki M., Nawarecki E.: *The application of evolution process in multi-agent world (MAW) to the prediction system.* [In:] Tokoro M., (Ed.): *Proc. of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96).* AAAI Press, 1996

[35] Chen S.-H., Kambayashi Y., Sato H.: *Multi-Agent Applications with Evolutionary Computation and Biologically Inspired Technologies.* IGI Global, 2011

[36] Crepinsek M., Liu S.-H., Mernik M.: *Exploration and Exploitation in Evolutionary Algorithms: A Survey.* ACM Computing Surveys, vol. 45, no. 3, 2013 (in press)

[37] Cutello V., Nicosia G.: *The clonal selection principle for in silico and in vitro computing.* [In:] *Recent Developments in Biologically Inspired Computing.* Idea Group Publishing, 2004, 104–146

[38] Darwin C.: *The Origin of Species.* Gramercy Books, 1998

[39] Dasgupta D.: *Artificial Immune Systems and Their Applications.* Springer-Verlag, 1998

[40] Davis T. E., Principe J. C.: *A Simulated Annealing Like Convergence Theory for the Simple Genetic Algorithm.* [In:] *Proc. of the Fourth International Conference on Genetic Algorithms.* San Diego, CA, 1991, 174–181

[41] Dawkins R.: *Selfish gene.* Oxford University Press, 1989

[42] de Castro L., Timmis J.: *Artificial Immune Systems: A New Computational Intelligence Approach.* Springer Verlag, 2002

[43] de Castro L., Timmis J.: *Artificial Immune Systems: A Novel Paradigm to Pattern Recognition.* [In:] *Artificial Neural Networks in Pattern Recognition.* University of Paisley, UK, 2002, 67–84

[44] De Jong K.: *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.D. thesis, University of Michigan, Ann Arbor, 1975

[45] Demazeau Y.: *Systèmes multi-agents*. OFTA Paris, 2004

[46] Demkowicz L., Kurtz J., Pardo D., Paszyński M., Rachowicz W., Zdunek A.: *Computing with Hp-Adaptive Finite Elements*. [In:] *Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications*. Chapman & Hall/CRC, 2007

[47] D'haeseleer P.: *An immunological approach to change detection: theoretical results*. [In:] *Proceedings of the 9th IEEE Computer Security Foundations Workshop*. 1996

[48] D'haeseleer P., Forrest S., Helman P.: *An immunological approach to change detection: algorithms, analysis and implications*. [In:] *Proceedings of the IEEE Symposium on Security and Privacy*. 1996

[49] Digalakis J., Margaritis K.: *An Experimental study of Benchmarking Functions for Evolutionary Algorithms*. International Journal of Computer Mathematics, vol. 79, no. 4, April 2002, 403–416

[50] Dobrowolsi G., Kisiel-Dorohinicki M., Nawarecki E.: *Some approach to design and realisation of mass multi-agent systems*. [In:] Schaefer R., Sędziwy S., (Eds.): *Advances in Multi-Agent Systems*. Jagiellonian University, 2001

[51] Dorigo M.: *Optimization, Learning and Natural Algorithms*. Ph.D. thesis, Politecnico di Milano, Italy, 1992

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

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

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

[55] Dreżewski R., Sepielak J., Siwik L.: *Classical and Agent-Based Evolutionary Algorithms for Investment Strategies Generation*. [In:] Brabazon A., O'Neill M., (Eds.): *Natural Computing in Computational Finance*. Vol. 185, Studies in Computational Intelligence, Springer-Verlag, 2009, 181–205

[56] Droste S., Jansen T., Wegener I.: *Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization*. Theory of Computing Systems, vol. 39, 2006, 525–544

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

[58] Eldridge N., Gould S.: *Punctuated equilibria: An alternative to phyletic gradualism*. [In:] Schopf T., (Ed.): *Models in Paleobiology*. Freeman, Cooper and Co., 1972

[59] Ercsey-Ravasz M., Toroczkai Z.: *The Chaos Within Sudoku*. Aug 2012

[60] Faber L., Piętak K., Byrski A., Kisiel-Dorohinicki M.: *Agent-Based Simulation in AgE Framework*. [In:] Byrski A., Oplatková Z., Carvalho M., Kisiel-Dorohinicki M., (Eds.): *Advances in Intelligent Modelling and Simulation*. Vol. 416, Studies in Computational Intelligence, Springer Berlin Heidelberg, 2012, 55–83

[61] Farmer J., Packard N., Perelson A.: *The immune system, adaptation, and machine learning*. Physica D: Nonlinear Phenomena, vol. 22, no. 1–3, 1986, 187–204

[62] Ferber J.: *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999

[63] Fernández de Vega F., Cantú-Paz E.: *Parallel and Distributed Computational Intelligence*. Vol. 269, Studies in Computational Intelligence, Springer, 2010

[64] FIPA, http://fipa.org/: *The Foundation for Intelligent Physical Agents*

[65] Fogel L.: *Autonomous Automata*. Industrial Research, vol. 4, 1962, 14–19

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

[67] Forrest S., Hofmeyr S., Somayaji A.: *Computer immunology*. Communications of the ACM, 1997

[68] Forrest S., Javornik B., Smith R., Perelson A.: *Using genetic algorithms to explore pattern recognition in the immune system*. Evolutionary Computation, vol. 1, no. 3, 191–211

[69] Forrest S., Perelson A., Allen L., Cherukuri R.: *Self-Nonself Discrimination in a Computer*. [In:] *Proceedings of the 1992 IEEE Symposium on Security and Privacy*. 1994

[70] Forrest S., Mitchell M.: *Relative Building-Block Fitness and the Building-Block Hypothesis*. [In:] Whitley D., (Ed.): *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, 1993

[71] Franklin S., Graesser A.: *Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*. [In:] *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*. ECAI '96, London, UK, UK, 1997, Springer-Verlag, 21–35

[72] Gary McGuireBastian Tugemann G. C.: *There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem*. Jan 2012

[73] George J., Gleizes M., P.Glize, 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*. University of Wales, 2003

[74] Glover F., G.A. K.: *Handbook of Metaheuristics*. Springer, 2003

[75] Glover F.: *Future paths for integer programming and links to artificial intelligence*. Comput. Oper. Res., vol. 13, no. 5, May 1986, 533–549

[76] Glover F., McMillan C.: *The general employee scheduling problem: an integration of MS and AI*. Comput. Oper. Res., vol. 13, no. 5, May 1986, 563–573

[77] Gödel K.: *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*. Monatsheft für Math. und Physik, vol. 38, 1931, 173–198

[78] Goldberg D. E.: *Algorytmy genetyczne i ich zastosowania*. WNT Warszawa, 1996

[79] Goldberg D. E., Segrest P.: *Finite Markov chain analysis of genetic algorithms*. [In:] *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. Hillsdale, NJ, USA, 1987, L. Erlbaum Associates Inc., 1–8

[80] Grefenstette J.: *Optimization of Control Parameters for Genetic Algorithms*. IEEE Trans. on Systems, Man and Cybernetics, vol. 16, no. 1, 1986, 122–128

[81] Grochowski M., Schaefer R., Uhruski P.: *Diffusion based scheduling in the agent-oriented computing systems*. Lecture Notes in Computer Science, vol. 3019, 2004, 97–104

[82] Hastings W.: *Monte Carlo sampling methods using Markov chains and their applications*. Biometrika, vol. 57, no. 1, 1970

[83] Haykin S.: *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 2 edition, July 1998

[84] He J., Yao X.: *From an Individual to a Population: An Analysis of the First Hitting Time of Population-Based Evolutionary Algorithms*. IEEE Transactions on Evolutionary Computation, vol. 6, no. 5, 2002, 495–511

[85] Hinton G., Nolan S.: *How learning can guide evolution*. Complex Systems, vol. 1, 1987, 495–502

[86] Hoare C.: *Communicating sequential processes*. Commun. ACM, vol. 21, no. 8, August 1978, 666–677

[87] Hobiger B. Hodoku, a sudoku generator, solver, trainer and analyzer. `http://hodoku.sourceforge.net/en/index.php` (accessed 17.04.2013)

[88] Hoffmann G.: *A Neural Network Model Based on the Analogy with the Immune System*. Journal of Theorectical Biology, vol. 122, no. 1, 1986, 33–67

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

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

[91] Hoos H., Stützle T.: *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2004

[92] Horst R., Pardalos P.: *Handbook of Global Optimization*. Kluwer, 1995

[93] Hughes T.: *The Finite Element Method. Linear Statics and Dynamics Finite Element Method Analysis.* Dover, 2000

[94] Iosifescu M.: *Finite Markov Processes and Their Applications*. John Wiley and Sons, 1980

[95] Ishida Y.: *Fully distributed diagnosis by PDP learning algorithm: towards immune network PDP models*. [In:] *IJCNN International Joint Conference on Neural Networks*. 1990

[96] Jennings N. R., Sycara K., Wooldridge M.: *A Roadmap of Agent Research and Development*. Journal of Autonomous Agents and Multi-Agent Systems, vol. 1, no. 1, 1998, 7–38

[97] Jennings N. R., Wooldridge M. J.: *Software Agents*. IEE Review, 1996, 17–20

[98] Jennings N. R., Wooldridge M. J.: *Applications of Intelligent Agents*. [In:] Jennings N. R., Wooldridge M. J., (Eds.): *Agent Technology: Foundations, Applications, and Markets*. Springer Verlag: Heidelberg, Germany, 1998, 3–28

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

[100] Juels A., Wattenberg M.: *Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms*. Technical report, University of California at Berkeley, 1994

[101] Jun-qing L., Quan-ke P., Yun-Chia L.: *An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems*. Computers & Industrial Engineering, vol. 59, no. 4, 2010, 647–662

[102] Kalos M., Whitlock P.: *Monte Carlo Methods*. Wiley-VCH, 2008

[103] Kelsey J., Timmis J.: *Immune inspired somatic contiguous hypermutation for function optimization*. [In:] Cantú-Paz E., (Ed.): *GECCO Genetic and Evolutionary Computation Conference, LNCS Vol. 2723*. Springer, 2003, 207–218

[104] Kennedy J., Eberhart R.: *Particle swarm optimization*. [In:] *Proc. of IEEE Int. Conf. on Neural Networks*. Vol. 4, 1995, 1942–1948

[105] Kephart J.: *A Biologically Inspired Immune System for Computers*. [In:] *Artificial Life IV*. 1994

[106] Kephart J., Sorkin G., Arnold W., Chess D. M., Tesauro G., White S.: *Biologically inspired defences against computer viruses*. [In:] *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. 1995

[107] Kirkpatrick S., Gelatt C. D., Vecchi M. P.: *Optimization by simulated annealing*. Science, vol. 220, 1983, 671–680

[108] Kisiel-Dorohinicki M.: *Agent-Oriented Model of Simulated Evolution*. [In:] Grosky W. I., Plasil F., (Eds.): *SofSem 2002: Theory and Practice of Informatics*. Vol. 2540, LNCS. Springer-Verlag, 2002

[109] Kisiel-Dorohinicki M., Dobrowolski G., Nawarecki E.: *Agent Populations as Computational Intelligence*. [In:] Rutkowski L., Kacprzyk J., (Eds.): *Neural Networks and Soft Computing*. Physica Verlag, 2002, 608–614

[110] Koch P., Simpson T., Allen J., Mistree F.: *Statistical approximations for multidisciplinary design optimization: the problem of size*. Journal of Aircraft, vol. 36, no. 1, 1999, 275–286

[111] Kołodziej J.: *A Simple Markov Model and Asymptotic Properties of a Hierarchical Genetic Strategy*. Schedae Informaticae, vol. 11, 2002, 41–55

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

[113] Krasnogor N., Gustafson S.: *A Study on the use of "self-generation" in memetic algorithms*. Natural Computing, vol. 3, 2004, 53–76

[114] Krasnogor N., Smith J.: *A tutorial for competent memetic algorithms: Model, taxonomy, and design issues*. IEEE Transactions on Evolutionary Computation, vol. 9, no. 5, 2005, 474–488

[115] Krasnogor N., Smith J.: *A tutorial for competent memetic algorithms: Model, taxonomy and design issues*. IEEE Transactions on Evolutionary Computation, vol. 9, no. 5, 2005, 474–488

[116] Krzywicki D.: *Niching in Evolutionary Multi-agent Systems*. Computer Science, vol. 14, no. 1, 2013

[117] Ku K., Mak M.: *Exploring the effects of Lamarckian and Baldwinian learning in evolving recurrent neural networks*. [In:] *Proc. of 1997 IEEE Int. Conf. on Evolutionary Computation*. IEEE, 1997

[118] Lee J.-S., Park C. H.: *Hybrid Simulated Annealing and Its Application to Optimization of Hidden Markov Models for Visual Speech Recognition*. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 40, no. 4, aug. 2010, 1188 –1196

[119] Li W. D., Ong S. K., Nee A. Y. C.: *Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts*. International Journal of Production Research, vol. 40, no. 8, 2002, 1899–1922

[120] Lobel B., Ozdaglar A., Feijer D.: *Distributed Multi-agent Optimization with State-Dependent Communication*. Mathematical Programming, vol. 129, no. 2, 2011, 255–284

[121] Locatelli M.: *Simulated Annealing Algorithms for Continuous Global Optimization: Convergence Conditions*. Journal of Optimization Theory and Applications, vol. 104, 2000, 121–133

[122] Luban M., Staunton L. P.: *An efficient method for generating a uniform distribution of points within a hypersphere*. Computers in Physics, vol. 2, November 1988, 55–60

[123] Mahalanobis P.: *On the generalised distance in statistics*. Proceedings of the National Institute of Sciences of India, vol. 2, no. 1, 1936, 49–55

[124] Mahfoud S.: *Finite Markov Chain Models of an Alternative Selection Strategy for the Genetic Algorithm*. Complex Systems, vol. 7, 1991, 155–170

[125] Mahfoud S. W.: *A Comparison of Parallel and Sequential Niching Methods*. [In:] *In Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995, 136–143

[126] March J.: *Exploration and Exploitation in Organizational Learning*. Organization Science, vol. 2, 1991, 71–87

[127] Mashinchi M. H., Orgun M. A., Pedrycz W.: *Hybrid optimization with improved tabu search*. Applied Soft Computing, vol. 11, no. 2, 2011, 1993 – 2006, (<ce:title>The Impact of Soft Computing for the Progress of Artificial Intelligence</ce:title>)

[128] McArthur S., Catterson V., Hatziargyriou N.: *Multi-Agent Systems for Power Engineering Applications—Part I: Concepts, Approaches, and Technical Challenges*. IEEE TRANSACTIONS ON POWER SYSTEMS, vol. 22, no. 4, November 2007

[129] McPhee N., Hopper N.: *Analysis of genetic diversity through population history*. [In:] *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 13-17 July, Orlando FL, USA*. 1999, 1112–1120

[130] M.E. C.: *Step and Flash Imprint Lithograpy: A Low Pressure, Room Temperature Nonoimprint Lithography*. Ph.D. thesis, The University of Texas in Austin, 2000

[131] Michalewicz Z.: *Genetic Algorithms Plus Data Structures Equals Evolution Programs*. Secaucus, NJ, USA, Springer-Verlag New York, Inc., 1994

[132] Michalewicz Z., Fogel D.: *How to Solve It: Modern Heuristics*. Springer, 2004

[133] Michalewicz Z.: *Ubiquity symposium: Evolutionary computation and the processes of life: the emperor is naked: evolutionary algorithms for real-world applications*. Ubiquity, vol. 2012, no. November, November 2012, 3:1–3:13

[134] Michalski R.: *Learnable Evolution Model: Evolutionary Processes Guided by Machine Learning*. Machine Learning, vol. 38, 2000, 9–40

[135] Mitchell M., Holland J. H., Forrest S.: *When Will a Genetic Algorithm Outperform Hill Climbing?* [In:] *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 1993, 51–58

[136] Morrison R. W., Jong K. A. D.: *Measurement of population diversity*. [In:] *In 5th International Conference EA 2001*. Springer, 2002, 31–41

[137] Moscato P.: *Memetic algorithms: a short introduction*. [In:] *New ideas in optimization*. Maidenhead, UK, England, 1999, McGraw-Hill Ltd., UK, 219–234

[138] Neri F.: *Diversity Management in Memetic Algorithms*. [In:] Neri F., Cotta C., Moscato P., (Eds.): *Handbook of Memetic Algorithms*. Vol. 379, Studies in Computational Intelligence, Springer Berlin Heidelberg, 2012, 153–165

[139] Ong Y.-S., Lim M.-H., Zhu N., Wong K.-W.: *Classification of adaptive memetic algorithms: a comparative study*. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 36, no. 1, 2006, 141 –152

[140] Papadimitriou C., Steiglitz K.: *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., 1998

[141] Paredis J.: *Coevolutionary Computation*. Artificial Life, vol. 2, no. 4, 1995, 355–375

[142] Paszyński M., Barabasz B., Schaefer R.: *Efficient adaptive strategy for solving inverse problems*. Lecture Notes in Computer Science, vol. 4488, 2007, 342–349

[143] Paszyński M., Demkowicz L.: *Parallel, fully automatic hp-adaptive 3D finite element package*. Engineering with Computers, vol. 22, no. 3, 2006, 255–276

[144] Paszyński M., Romkes A., Collister E., Meiring J., Demkowicz L., Willson C.: *On the Modeling of Step and Flash Imprint Lithography*. Technical report, ICES Report 05-38, 2005

[145] Pisarski S., Rugała A., Byrski A., Kisiel-Dorohinicki M.: *Evolutionary Multi-Agent System in Hard Benchmark Continuous Optimisation*. [In:] *Proc. of EVOSTAR Conference, Vienna*. IEEE (accepted for printing), 2013

[146] Polnik M., Kumięga M., Byrski A.: *Agent-based optimisation of advisory strategy parameters*. Journal of Telecommuniactions and Information Technology, no. 2, 2013

[147] Potter M., De Jong K.: *Cooperative coevolution: An Architecture for Evolving Coadapted Subcomponents*. Evolutionary Computation, vol. 8, no. 1, 2000, 1–29

[148] Rao A. S., Georgeff M. P.: *BDI Agents: From Theory to Practice*. [In:] *First International Conference on Multi-Agent Systems (ICMAS-95*. AAAI, 1995, 312–319

[149] Rechenberg I.: *Evolutionstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Formann-Holzboog, 1973

[150] Rinnoy Kan A., Timmer G.: *Stochastic Global Optimization Methods*. Mathematical Programming, vol. 39, 1987, 27–56

[151] Rockafellar R.: *Nonsmooth analysis and parametric optimization*. [In:] Cellina A., (Ed.): *Methods of Nonconvex Analysis*. Vol. 1446, Lecture Notes in Mathematics, Springer Berlin Heidelberg, 1990, 137–151

[152] Rudolph G.: *Convergence analysis of canonical genetic algorithms*. IEEE Transactions on Neural Networks, vol. 5, no. 1, 1994, 96 –101

[153] Rudolph G.: *Evolution Strategies*. [In:] Bäck T., Fogel D., Michalewicz Z., (Eds.): *Handbook of Evolutionary Computations*. Oxford University Press, 1997

[154] Rudolph G.: *Models of Stochastic Convergence*. [In:] Bäck T., Fogel D., Michalewicz Z., (Eds.): *Handbook of Evolutionary Computations*. Oxford University Press, 1997

[155] Rudolph G.: *Stochastic Processes*. [In:] Bäck T., Fogel D., Michalewicz Z., (Eds.): *Handbook of Evolutionary Computations*. Oxford University Press, 1997

[156] Rudolph G.: *Finite Markov Chain Results in Evolutionary Computation: A Tour d'Horizon*. Fundamenta Informaticae, vol. 35, no. 1–4, 1998, 67–89

[157] Russel S., Norvig P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003

[158] Sarker R., Ray T.: *Agent-Based Evolutionary Search*. Vol. 5, Adaptation, Learning and Optimization, Springer, 1 edition, 2010

[159] Sauter V.: *Decision support systems: an applied managerial approach*. John Wiley, New York, 1997

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

[161] Schaefer R., Byrski A., Smołka M.: *The Island Model as a Markov Dynamic System*. International Journal of Applied Mathematics and Computer Science, vol. 12, no. 4, 2012

[162] Schaefer R.: *Foundations of global genetic optimization*. Springer Verlag, 2007

[163] Schaefer R., Byrski A., Smołka M.: *Stochastic Model of Evolutionary and Immunological Multi-Agent Systems: Parallel Execution of Local Actions*. Fundamenta Informaticae, vol. 95, no. 2-3, 2009, 325–348

[164] Schaefer R., Toporkiewicz W., Grochowski M.: *Rough Partitioning of Lumped Structures*. [In:] *Formal Methods and Intelligent Techniques in Control, Decision Making, Multimedia and Robotics*. Polish-Japanese Institute of Information Technology Press, October 2000, 151–166

[165] Schwefel H.-P.: *Kybernetische evolution als strategie der experimentellen forschung inder strömungstechnik*. Technical report, Technische Universität, Berlin, 1965

[166] Schwefel H.-P., Bäck T.: *Artificial Evolution: How and Why?* [In:] Quagliarella D., Périaux J., Poloni C., Winter G., (Eds.): *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*. Chichester, John Wiley and Sons, 1998, 1–19

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

[168] Shortz W.: *The Little Black Book of Sudoku*. St. Martin's Griffin, New York, NY, 2006

[169] Siwik L., Dreżewski R.: *Agent-based multi-objective evolutionary algorithms with cultural and immunological mechanisms*. [In:] dos Santos W. P., (Ed.): *Evolutionary computation*. In-Teh, 2009, 541–556

[170] Solis F., Wets R.: *Minimization by random search techniques*. Mathematical Methods of Operations Research, vol. 6, 1981, 19–30

[171] Stadnyk I.: *Schema recombination in a pattern recognition process*. [In:] *Proc. of the Second Int. Conf. on Genetic Algorithms*. Cambridge, MA: Lawrence Erlbaum Associates, 1987, 27–35

[172] Stuart A. Strategies for number puzzles of all kinds. `www.sudoku.wiki.org` (accessed 17.04.2013)

[173] Stuart A.: *The Logic of Sudoku*. Michael Mepham Publishing, 2007

[174] Stuart A.: *Sudoku Creation and Grading*. Feb 2007

[175] Suzuki J.: *A Markov Chain Analysis on a Genetic Algorithm*. [In:] Forrest S., (Ed.): *Proceedings of the 5th International Conference on Genetic Algorithms, Urbana-Champaign, IL, USA, June 1993*. Morgan Kaufmann, 1993, 146–154

[176] Talbi E.-G.: *A Taxonomy of Hybrid Metaheuristics*. Journal of Heuristics, vol. 8, 2002, 541–564

[177] Talbi E.-G.: *A taxonomy of hybrid metaheuristics*. Journal of Heuristics, vol. 8, no. 5, 2002, 541–564

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

[179] Talbi N., Belarbi K.: *Optimization of Fuzzy Controller using Hybrid Tabu Search and Particle Swarm Optimization*. [In:] *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*. dec. 2011, 561 –565

[180] Tomassini M.: *Spatially Structured Evolutionary Algorithms*. Springer, 2005

[181] Uhruski P., Grochowski M., Schaefer R.: *A Two-layer Agent-Based System For Large-Scale Distributed Computation*. Computational Intelligence, vol. 24, no. 3, July 2008, 191–212

[182] Uhruski P., Grochowski M., Schaefer R.: *Multi-Agent Computing System in a Heterogeneous Network*. [In:] *Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC 2002)*. Warsaw, Poland, 22–25 September 2002, IEEE Computer Society Press, 233–238

[183] Uhruski P., Grochowski M., Schaefer R.: *Octopus – Computation Agents Environment*. Inteligencia Artificial, Revista Iberoamericana de IA, vol. 9, no. 28, 2005, 55–62

[184] Uhruski P., Grochowski M., Schaefer R.: *A Two-Layer Agent-Based System for Large-Scale Distributed Computation*. Computational Intelligence, vol. 24, no. 3, August 2008, 191–212

[185] Victoire T. A. A., Jeyakumar A. E.: *A tabu search based hybrid optimization approach for a fuzzy modelled unit commitment problem*. Electric Power Systems Research, vol. 76, no. 6–7, 2006, 413 – 425

[186] Victoire T., Jeyakumar A.: *Unit commitment by a tabu-search-based hybrid-optimisation technique*. Generation, Transmission and Distribution, IEE Proceedings-, vol. 152, no. 4, july 2005, 563–574

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

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

[189] Wang H., Wang D., Yang S.: *A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems*. Soft Computing, vol. 13, no. 8–9, 2008, 763–780

[190] Whitley D., Scott Gordon V., Mathias K.: *Lamarckian evolution, the Baldwin effect and function optimization*. [In:] Davidor, Y. and Schwefel, H.-P. and Männer, R., (Ed.): *Proc. of Parallel Problem Solving from Nature III*. Springer, 1994

[191] Whitley D.: *An Executable Model of a Simple Genetic Algorithm*. [In:] *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, 1992, 45–62

[192] Wolpert D., Macready W.: *No free lunch theorems for search*. Technical Report SFI-TR-02-010, Santa Fe Institute, 1995

[193] Wolpert D., Macready W.: *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computation, vol. 67, no. 1, 1997

[194] Wooldridge M.: *An Introduction to Multiagent Systems*. John Wiley & Sons Inc., 2004

[195] Wooldridge M., Jennings N.: *Intelligent Agents*. [In:] *LNAI 890*. Springer Verlag, 1995

[196] Wooldridge M.: *An Introduction to Multiagent Systems*. John Wiley & Sons, 2009

[197] Wróbel K., Torba P., Paszyński M., Byrski A.: *Evolutionary Multi-Agent Computing in Inverse Problems*. Computer Science (accepted for printing), 2013

[198] Xu X., He H.-g.: *A Theoretical Model and Convergence Analysis of Memetic Evolutionary Algorithms*. [In:] Wang L., Chen K., S. Ong Y., (Eds.): *Advances in Natural Computation*. Vol. 3611, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005, 1035–1043

[199] Yao X., Wang F., Padmanabhan K., Salcedo-Sanz S.: *Hybrid Evolutionary Approaches to Terminal Assignment in Communications Networks*. [In:] Hart W., Smith J., Krasnogor N., (Eds.): *Recent Advances in Memetic Algorithms*. Vol. 166, Studies in Fuzziness and Soft Computing, Springer Berlin Heidelberg, 2005, 129–159

[200] Zhong W., Liu J., Xue M., Jiao L.: *A multiagent genetic algorithm for global numerical optimization*. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 34, no. 2, 2004, 1128–1141

[201] Zhong Y., Pan X.: *A Hybrid Optimization Solution to VRPTW Based on Simulated Annealing*. [In:] *Automation and Logistics, 2007 IEEE International Conference on*. aug. 2007, 3113 –3117