BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 61, No. 1, 2013
DOI: 10.2478/bpasts-2013-0025

Asymptotic guarantee of success for multi-agent memetic systems

A. BYRSKI[1*], R. SCHAEFER[1], M. SMOŁKA[1], and C. COTTA[2]

[1] Department of Computer Science, AGH University of Science and Technology, 30 Mickiewicza Av., 30-059 Kraków, Poland
[2] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga ETSI Informática, Campus de Teatinos, 29071-Málaga, Spain

Abstract. The paper introduces a stochastic model for a class of population-based global optimization meta-heuristics, that generalizes existing models in the following ways. First of all, an individual becomes an active software agent characterized by the constant genotype and the meme that may change during the optimization process. Second, the model embraces the asynchronous processing of agent's actions. Third, we consider a vast variety of possible actions that include the conventional mixing operations (e.g. mutation, cloning, crossover) as well as migrations among demes and local optimization methods. Despite the fact that the model fits many popular algorithms and strategies (e.g. genetic algorithms with tournament selection) it is mainly devoted to study memetic algorithms.

The model is composed of two parts: EMAS architecture (data structures and management strategies) allowing to define the space of states and the framework for stochastic agent actions and the stationary Markov chain described in terms of this architecture. The probability transition function has been obtained and the Markov kernels for sample actions have been computed.

The obtained theoretical results are helpful for studying metaheuristics conforming to the EMAS architecture. The designed synchronization allows the safe, coarse-grained parallel implementation and its effective, sub-optimal scheduling in a distributed computer environment. The proved strong ergodicity of the finite state Markov chain results in the asymptotic stochastic guarantee of success, which in turn imposes the liveness of a studied metaheuristic. The Markov chain delivers the sampling measure at an arbitrary step of computations, which allows further asymptotic studies, e.g. on various kinds of the stochastic convergence.

Key words: computational multi-agent systems, asymptotic analysis, global optimization, parallel evolutionary algorithms, Markov chain modeling.

1. Introduction

Meta-heuristic optimization methods have nowadays become one of the weapons of choice to deal with complex optimization problems. The success story of these techniques is marked by our increasing understanding of them. One of the most important lessons learned was put forward by early practitioners such as L. Davis [1] and P. Moscato [2], namely the need for adjusting the solver to the problem by exploiting knowledge available on the latter. Initially, this practical lesson was theoretically backed up initially by [3] and later by [4] in the conspicuous *No-Free-Lunch Theorem*. The realization of this lesson constitutes one of the raisons-d'être of optimization techniques such as memetic algorithms (MAs) [5–7]. These are population-based techniques that blend together ideas from other meta-heuristics, most commonly in terms of integrating local search (LS) within the population-based search engine. This definition of MA was actually popularized by early works such as [8] and paved the way for the vigorous development of optimization algorithms based on this idea, exhibiting a remarkable record of success, check, e.g., [9, 10]. It is also true that seminal works on this topic had a wider perspective, in which an evolutionary algorithm endowed with LS was an appropriate incarnation of a MA, rather than a restrictive definition [2]. Under this wider interpretation of MAs, they can be regarded a population of search agents that alternate periods of cooperation/competition with phases of self-improvement see [11, 12]. The use of the term *agent* here is to emphasize the fact that in the search process individuals are more active actors than mere solution placeholders that passively suffer the application of different variation and selection operations on them [13]. While this interpretation remains compatible with classical MA approaches, it also opens up the door to more complex strategies and suggests an underlying connection of MAs with multi-agent systems. Indeed, the current notion of memetic computing as a paradigm that uses memes as units of information encoded in computational representations for the purpose of problem-solving can easily lead to a co-evolving system of intelligent agents [14], see also [15–17]. In this context, a meme is regarded as a lifetime learning procedure capable of locally improving solutions [18–20], although a more general interpretation is also possible in terms of culturally learned traits [21].

Aside of many important MA engineering applications mentioned above – see also [22] – several interesting examples of such metaheuristics solving difficult problems in area of computational mechanics and computer graphics may be found in [23, 24]. Some other results obtained by artificial agent-based metaheuristics, more affordable than certain classical approaches are the optimization of neural-network architecture [25, 26], multi-objective optimization [27], multimodal optimization [28] and financial optimization [29] to name a few. A survey of the above mentioned results is presented in [30].

*e-mail: olekb@agh.edu.pl

Formal analysis of metacheuristic features is usually a hard task. Foundations of this area of research was delivered in parallel by Vose (see e.g. [31]) and Rudloph (see e.g. [32–34]). They introduced the way of modeling metaheuristic dynamics as a Markov chain with a set of states being the set of all possible populations or their unambiguous representations (e.g. the Vose simplex). This approach allows analyzing of the metaheuristic convergence in the more or less conventional sense (see [33, 35]) and at the same time analyzing of the asymptotic behavior of sample measures and verifying in this way the asymptotic guarantee of success. The second feature (cf. [36, 37]) was frequently drawn as a direct consequence of ergodicity of a Markov chain and justifies the metaheuristics as a well-defined global optimization algorithm.

Other models for single-population, e.g., [38–41] and parallel evolutionary algorithms, e.g., [42–45] were defined as well; however, more advanced optimization techniques like memetic or agent-based computational systems lacked such models, with some notable exceptions, e.g., [46] consider an abstract model of MAs based on applying gradient-based LS to the whole population on a generation-basis and provide 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, [47] consider 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 [48, 49] and cannot be proved in general for local adaptive strategies.

Summing up, there is still a lack of a comprehensive stochastic model of the wide class of memetic algorithms. One of the reason is perhaps that memetic algorithms are still weakly defined and formalized which of course have to be done before any attempt to analyse their features mathematically.

In the course of the paper we introduce the EMAS architecture (see Sec. 2) that generalizes the broad class of metaheuristics including in particular GAs with the tournament selection and most of advanced MA instances. The idea of EMAS came from [50] and was further enhanced by [25, 26]. The main directions in which the classical GA models was relaxed are: an individual become the active software agent characterized by the constant genotype and the meme that can be changed during the optimization process, the multideme structure of population, the asynchronous processing of agent's actions, the broad variety of possible actions that may be the conventional mixing operations (e.g. mutation, cloning, crossover) as well as migrations among demes and local optimization methods (see Sec. 6). Both, genes and memes comes from finite universa. This model allows for defining the space of states and the framework for describing stochastic effect of agent's actions.

Next step consists in detailed analysis of dependencies among the actions (see Sec. 3) which allow to define the safe synchronization among agents (see Sec. 4).

Both above steps enable to define the stationary Markov chain that transforms stochastically EMAS states according to the roles given by agent's actions and inter-agent synchronization (see Sec. 5).

Assuming some non-restrictive features of the sample set of agent's actions defined in Sec. 6 we are able to prove the ergodicity of the Markov chain expressing the dynamics of the metaheuristics being in accordance with the EMAS architecture.

The agent-based architecture of the global optimization metaheuristics makes a metaphor useful by the analyzing of their asymptotic features. Our earlier papers [51, 52] discussed similar metaheuristics with a continuous meme representation and in consequence the set of states was bounded but infinite. The ergodicity of the Markov chain is difficult to study in this case.

The results presented in this paper were partially communicated at PPSN 2010 conference [53]. Moreover, the agent based synchronization among demes was applied for island model for which the Markov chain model was established and its ergodicity was proven (see [54]). The Markov chain model was also obtained for the complex, metaheuristic HGS with a tree-structured set of demes (see [55]).

The results presented in this paper might be helpful by many aspects of metaheuristic analysis especially of memetic type.

- The EMAS framework can help to understand the stochastic dynamics of particular strategy that fall into its specification. We show the way of model its behavior as the stationary Markov chain. In particular, the general form of the space of states (see Subsec. 2.2, formula (2.2)) and the transition probability function (see Observations 5.1, 5.2) are drown.
- We formulate and prove precise mathematical criterion which enable to classify an action as the global one, which has to be executed exclusively in the whole system, or the local one, which can be performed independently with other local action in the separate location (see Definition 3.1, Propositions A.1 and 3.1). This results allows to design proper synchronization among agents or verify the existing one ensuring metaheuristic *safeness*. In particular the safe, coarse-grained parallel computation and its suboptimal scheduling in a distributed computer environment (computer cluster) might be obtained.
- The ergodicity of Markov chain assigned with the studied memetic metaheuristic (see Theorem 7.1, Theorem 7.2, Corollary 7.1, Remark 7.4) leads to its asymptotic stochastic guarantee of success (see [36, 37]). This condition imposes in particular *liveness* of studied metaheuristic allowing to use it as the robust global optimization strategy.
- The Markov chain modeling particular metaheuristic allows to obtain its sampling measure in the arbitrary step by iterating the probability transition function. It constitutes the necessary basis for future asymptotic studies e.g. various kinds of stochastic convergence (see e.g. [33, 35]).

The theoretical deliberations are illustrated with exemplary experimental results obtained with use of EMAS and its memetic variants (see Appendix C).

2. EMAS architecture and behavior

In this section we give a description of the EMAS architecture and refine the model already presented in [52] to apply it to discrete spaces of states.

We focus on solving global optimization problems consisting in finding all global minimizers $\arg\min\{\Phi(x)\}, x \in \mathcal{D}$ 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, $\mathbb{R}_+ \ni M < +\infty$. The positive value of Φ was motivated only by the traditional convention accepted in genetic algorithms which allow us later for easy definition of agent's action based on a fitness function. Of course, each bilaterally bounded objective can be shifted to the weakly positive one and the obtained global optimization problem will be equivalent.

2.1. Characterization of computational agents. Computational agents in EMAS may be perceived as autonomous individuals. Every agent is capable of observing its environment by gathering information that it finds important, making decisions that affect its activity and performing actions that lead to changes in the overall state of the system, see e.g., [56, 57].

Genotypes belong to the finite genotype universum U, $\#U = r < +\infty$ which can be the set of binary strings, real numbers or other codes convenient to solving particular global optimization problem. Agents are assigned to locations (analogous to "islands", see e.g. [58]) and may migrate between them. Genetic operations performed on the agent's genotypes such as crossover and mutation, are similar to those used in classical evolutionary algorithms and lead to create a new agent (see Subsec. 6.2). The EMAS agent can also create its offspring using the local search method starting from the point encoded by its genotype (see Subsec. 6.3). Each agent is transformed asynchronously in the EMAS system. Selection mechanisms correspond to their prototype and are based on the existence of a non-renewable resource called *life energy*, which is gained and lost when agents perform actions, see [59].

The EMAS under consideration is characterized by:

- *Quasi-signature of an agent*: it is composed of its (invariant) genotype gen and the numerical identifier of the copy n changed during the migration.
- *Fitness function*: it is 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 contained 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 topology of channels is determined by the symmetric relation $Top \subset Loc^2$. We assume 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 at every moment can be one-to-one mapped into set $U \times P$, where $P = \{1, \ldots, p\}$ and p is assumed to be the maximum number of agents containing the same genotype. In other words, every agent $ag_{gen,n} \in Ag$ contains one potential solution to a given problem encoded as $gen \in U$, whereas there may be more than one agent present in the system containing this solution and the index $n \in P$ is used to distinguish them. Furthermore, we assume that every 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*: its value is quantized; every agent may possess only one of the following 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 every time-point. Thus in the sequel we shall write *agent* $ag_{gen,n}$ keeping in mind that this notation is time-dependent. Note that in the context of finding the objective function minimizers 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. If, on the other hand, one wanted to align our approach with the Belief-Desire-Intention (BDI) model [60], one could construct a true agent identifier by means of our 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 was active. It is, however, worth noticing that such an identifier would not be very useful unless it was stored in a globally synchronized repository and the need of the global synchronization would in turn prevent the concurrent performance of some crucial EMAS actions.

2.2. EMAS state. Let us introduce the set of three-dimensional incidence and energy matrices $x \in X$ with s layers (corresponding to all locations) $x(i) = \{x(i, gen, n),\ gen \in U,\ n \in P\},\ i \in Loc$. The layer $x(i)$ will contain energies of agents in the i-th location. In other words, condition $x(i, gen, k) > 0$ means that the k-th clone of the agent containing gene $gen \in U$ is active, its energy equals $x(i, gen, k)$ and it is present in i-th location.

We introduce the following coherency conditions:

- $(\cdot, j, k)$-th column contains at most one value greater than zero, which states the fact that the agent with the k-th copy of the j-th genotype may be present 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 the incidence and energy matrices are non-negative $x(i, j, k) \geq 0$ for $1 \leq i \leq s, 1 \leq j \leq r, 1 \leq k \leq p$

and $\sum_{i=1}^{s}\sum_{j=1}^{r}\sum_{k=1}^{p} x(i,j,k) = 1$, which means that total energy contained in the whole system is constant and equal to 1;

- $x(i, gen, n)$ can be positive only for n acceptable in location i, i.e. $n \in P_i$;
- each layer $x(i)$ contains at most q_i values greater than zero, which denotes the maximum 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

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

which allows 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 $\sum_{i=1}^{s} q_i$; we assume that $p = \sum_{i=1}^{s} q_i$, which assures that each configuration of agents in locations is available, respecting the constrained 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 because we want to allow a system state in which a particular location is filled with clones of one agent; on the other hand because of the previous assumption $\#P_i$ cannot also be greater than q_i; therefore finally we assume that $\#P_i = q_i$.

Gathering all these conditions, the considered set of three-dimensional incidence and energy matrices may be described in the following way:

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

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

2.3. Managing agents, EMAS structure and behavior. EMAS may be modeled as the following tuple:

$$\begin{aligned} & < U, Loc, Top, Ag, \{agsel_i\}_{i\in Loc}, \\ & locsel, \{LA_i\}_{i\in Loc}, MA, \omega, Act > \end{aligned} \tag{3}$$

where

MA (master agent) is used to synchronize the work of the locations; it allows to perform actions in particular locations; this agent is also used to introduce the necessary synchronization into the system;

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

LA_i (local agent) is assigned to each location; it is used to synchronize the work of computational agents present in its location, LA_i chooses the computational agent and lets it evaluate a decision and perform the action, at the same time asking MA whether this action may be performed;

$agsel_i : X \to \mathcal{M}(U \times P)$ is a family of functions used by local agents to select the agent that may perform the action, such that every 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 it is present in a location different from i-th one ($n \notin P_i$);

$\omega : X \times U \to \mathcal{M}(Act)$ is the function used by agents for selecting 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, we will use the following convention to describe the discrete probability distributions $a : X \times Par \to \mathcal{M}(A)$ on the set A, depending on state $x \in X$ and an optional parameter $p \in Par$.

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

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

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 $\mathcal{M}(X)$. Every agent starts working immediately after being activated. At every observable moment a certain agent on each location gains the possibility of changing the state of the system by executing its action.

The function $agsel_i$ is used by the Local Agent LA_i to determine which agent present in the i-th location is 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 $\omega(x, gen)$. Notice the relationship of this probability distribution with the concept of fine-grain schedulers introduced into the syntactic model for memetic algorithms in [5]. It must be noted that the selection of action by all agents containing 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 permission is granted, $ag_{gen,n}$ checks whether the associated condition (described later by the formula (5) and the surrounding text) is true, and if so, the agent performs the action. If during an action an agent's energy is brought to 0, this agent suspends its work in the system (it becomes inactive).

The master agent MA manages the activities of LA_i allowing them to grant their agents permission (thus relating to coarse-grain schedulers in [5]). The detailed managing algorithm is described in Sec. 4.

Let us denote by

$$X_{gen,n} = \{x \in X \mid \exists\, l \in Loc : x(l, gen, n) > 0\}, \quad (gen, n) \in U \times P \tag{4}$$

a subset of states in which the agent $ag_{gen,n}$ is active.

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

$$\delta_\alpha^{gen,n} : X \to \mathcal{M}(\{0,1\}) \tag{5}$$

make it possible to take the decision, if the action can be performed. 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

$$\delta_\alpha^{gen,n}(x) = \begin{cases} \delta_\alpha^{gen,n}|X_{gen,n} & x \in X_{gen,n} \\ (1,0) & x \in X \setminus X_{gen,n}. \end{cases} \tag{6}$$

Next, the formula

$$\vartheta_\alpha^{gen,n} : X \to \mathcal{M}(X) \tag{7}$$

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 $ag_{gen,n}$. Because the function is only invoked if the agent is active, 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

$$\vartheta_{null} : X \to \mathcal{M}(X) \tag{8}$$

such that for all $x \in X$

$$\vartheta_{null}(x)(x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

is performed.

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

$$\varrho_\alpha^{gen,n} : X \to \mathcal{M}(X) \tag{10}$$

is defined by the formula

$$\varrho_\alpha^{gen,n}(x)(x') = \delta_\alpha^{gen,n}(x)(0) \cdot \vartheta_{null}(x)(x') + \delta_\alpha^{gen,n}(x)(1) \cdot \vartheta_\alpha^{gen,n}(x)(x'), \tag{11}$$

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

Notice 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 [61] 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 [19]).

Observation 2.1. Given an 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. (10) and Eq. (11).

3. Global and local actions

The agents' 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,
- local – they change the state of the system in one location considering only the state of local agents; only one local action for one location may be performed at a time.

Therefore, we divide the Act set in the following way:

$$Act = Act_{gl} \cup Act_{loc}. \tag{12}$$

Informally speaking, if the location $i \in Loc$ contains the agent performing a certain local action $\alpha \in Act_{loc}$, only 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 formalized as follows:

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 besides 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
$$\forall\, x \in X : \varrho_\alpha^{gen,n}(x)(x_{next}) = 0, \tag{13}$$
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
$$\begin{gathered} \forall\, x_1, x_2 \in X, x_1(l) = x_2(l), \\ \forall\, x_{1,next}, x_{2,next} \in X, x_{1,next}(l) = x_{2,next}(l) \\ \text{and for each } \ i \neq l \\ x_1(i) = x_{1,next}(i),\ x_2(i) = x_{2,next}(i) \\ \varrho_\alpha^{gen,n}(x_1)(x_{1,next}) = \varrho_\alpha^{gen,n}(x_2)(x_{2,next}). \end{gathered} \tag{14}$$

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

The importance of local actions lies in the fact that they are commutative if performed in different locations. The formal proof of the local actions commutativity is provided in Appendix A.

The following feature of the local actions may be drawn:

Proposition 3.1. Any two local actions executed by agents present in different locations commute.

Proof. Take $gen, gen' \in U$, $n, n' \in P$. Let $ag_{gen,n}$ perform a local action α in the layer l and $ag'_{gen',n'}$ perform a local action α' in the layer l' different from l. The transition function of the composite action in which α is performed before α' is given by the following formula.

$$(\varrho_{\alpha'}^{gen',n'} \circ \varrho_{\alpha}^{gen,n})(x)(x') = \sum_{y \in X} \varrho_{\alpha'}^{gen',n'}(y)(x') \cdot \varrho_{\alpha}^{gen,n}(x)(y). \quad (15)$$

Since both actions are local (i.e., they satisfy (13) and (14)) and are performed in different locations, the assumptions of Proposition A.1 are satisfied. From this Proposition we obtain

$$(\varrho_{\alpha'}^{gen',n'} \circ \varrho_{\alpha}^{gen,n})(x)(x') = (\varrho_{\alpha}^{gen,n} \circ \varrho_{\alpha'}^{gen',n'})(x)(x') \quad (16)$$

for any $x, x' \in X$.

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.

4. EMAS management

In order to obtain relaxed synchronization (i.e. agents present in locations may act in parallel), some timing mechanism must be introduced, that is, all state changes must be assigned to subsequent time moments $t_0, t_1, \ldots$ Now, let us consider the algorithmic description for computational agents, $LA_i, i \in Loc$ (4.3) and MA presented in Pseudocodes 4.1, 4.3 and 4.2 respectively. Note, that here and later $\underline{a}(B)$ denotes the effect of random sampling one of the elements from the set B with probability distribution a. We also assume that the sets $localact, globalact \subset Act$ contain the local and global actions' signatures respectively.

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

Both these functions are variadic. First of their parameters is always a target identifier, and the next 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).

When permission is granted and the decision assigned to the considered action is true, the computational agent changes the state of the location (see Pseudocode 4.1). Afterwards the agent suspends its work again in order to get permission to perform subsequent actions.

The local agent (see Pseudocode 4.3) starts by checking whether the location contains any agents, so it sends a message to the master agent and waits for reply. If there are agents in the location, the LA_i receives signals containing actions to be performed from all its agents and puts them into a hash map indexed by genotypes and containing actions. Then the local agent utilizes function $agsel_i$ to choose the computational agent which should try to perform its action. This action is reported to the master agent and after receiving permission, the computational agent can perform the action. All other agents (and the chosen one when permission is not granted) are prevented from performing their actions. Afterwards the agent waits for all local agents to report the readiness to perform subsequent actions, and then reports this fact to the master agent and – after receiving permission – lets them do it.

Pseudocode 4.1: Computational agent's algorithm

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

Pseudocode 4.2: 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** $act \in Act_{gl}$
      - **then** $localglob \leftarrow localglob \cup \{j\}$
      - **else** $localloc \leftarrow localloc \cup \{j\}$
- $lchosen \leftarrow \underline{locsel(x)}(Loc)$
- **if** $lchosen \in localglob$
  - **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 4.3: 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** $send(g)$

The master agent (see Pseudocode 4.2) 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. In the end, the master agent waits once more for all locations to report that their work is finished and let them try to perform a subsequent action.

5. EMAS dynamics

At the observable moment at which the EMAS takes the state $x \in X$ all agents in all locations notify their local agents of their intent to perform an action. All local agents choose an agent using the distribution given by the function $agsel_i(x)$, $i \in Loc$ and then notify the master agent of their intent to let one of their agents perform an action. The master agent chooses the location with the probability distribution given by $locsel(x)$.

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

$$\xi_i(x) = \sum_{gen \in U} \sum_{n \in P_i} agsel_i(x)(gen, n) \cdot \omega(x, gen)(Act_{loc}). \tag{17}$$

The probability that the master agent will choose the location with the agent intending to perform a local action is:

$$\zeta^{loc}(x) = \sum_{i \in Loc} locsel(x)(i) \cdot \xi_i(x). \tag{18}$$

Of course the probability of choosing the global action by the master agent is:

$$\zeta^{gl}(x) = 1 - \zeta^{loc}(x). \tag{19}$$

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 respecting Pseudocodes 4.1, 4.2, 4.3:

$$\tau^{gl}(x)(x') = \sum_{i \in Loc} locsel(x)(i) \left(\sum_{gen \in U} \sum_{n \in P_i} agsel_i(x)(gen, n) \cdot \left(\sum_{\alpha \in Act_{gl}} \omega(x, gen)(\alpha) \cdot \varrho_\alpha^{gen,n}(x)(x') \right) \right). \tag{20}$$

Let us state a set of action sequences containing at least one local action:

$$Act_{+1loc} = \left\{ (\alpha_1, \ldots, \alpha_s) \in Act^s;\ \sum_{i=1}^{s} [\alpha_i \in Act_{loc}] > 0 \right\} \tag{21}$$

Let us define now the family of coefficients $\{\mu_{\alpha_i, gen_i, n_i}(x)\}$, $i \in Loc$, $gen_i \in U$, $n_i \in P_i$, $x \in X$. If the location i is nonempty in the state x, then $\mu_{\alpha_i, gen_i, n_i}(x)$ is equal to the probability that in the i-th location the agent ag_{gen_i, n_i} chooses action α_i:

$$\mu_{\alpha_i, gen_i, n_i}(x) = agsel_i(x)(gen_i, n_i) \cdot \omega(x, gen_i)(\alpha_i). \tag{22}$$

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, we set $\mu_{\alpha_i, gen_i, n_i}(x) = 1$ if the location i is empty in the state x. Next we introduce the multi-index:

$$ind = \big(\alpha_1, \ldots, \alpha_s; (gen_1, n_1), \ldots, (gen_s, n_s)\big) \in IND = Act^s_{+1loc} \times \prod_{i=1}^{s} (U \times P_i). \tag{23}$$

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

$$\mu_{ind}(x) = \prod_{i=1}^{s} \mu_{\alpha_i, gen_i, n_i}(x). \tag{24}$$

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

all possible sampling results respecting Pseudocodes 4.1, 4.2, 4.3:

$$\tau^{loc}(x)(x') = \sum_{ind\in IND} \mu_{ind}(x)(\pi_1^{ind} \circ , \ldots, \circ \pi_s^{ind})(x)(x'), \quad (25)$$

where

$$\pi_i^{ind}(x) = \begin{cases} \varrho_{\alpha_i}^{gen_i,n_i}(x), & \alpha_i \in Act_{loc} \text{ and } i \text{ is nonempty} \\ \vartheta_{null}, & \alpha_i \in Act_{gl} \text{ or } i \text{ is empty.} \end{cases} \quad (26)$$

The definition of the coefficient $\mu_{\alpha_i,gen_i,n_i}(x)$ and the above formula (26) show in particular that the action $null$ is executed in every location instead of the selected global action and formally in all empty locations.

Let us observe 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 Proposition 3.1 and Corollary 6.1). It validates the following observation.

Observation 5.1. The probability transition function for the EMAS model is given by the formula

$$\tau(x)(x') = \zeta^{gl}(x) \cdot \tau^{gl}(x)(x') + \zeta^{loc}(x) \cdot \tau^{loc}(x)(x') \quad (27)$$

and Eqs. (17)–(26).

Observation 5.2. The stochastic state transition of EMAS given by Eq. (27) satisfies the Markov condition. Moreover, the Markov chain defined by these functions is stationary.

Proof. All transition functions and probability distributions given by Eqs. (17)–(26) depend only on the current state of the system, which motivates the Markovian features of the transition function τ given by (27). The transition functions do not depend on the step number at which they are applied, which motivates the stationarity of the chain.

6. EMAS sample actions

Let us consider the following set of actions:

$$Act = \{get, repr, migr, clo, lse\}, \quad (28)$$

where get allows for the life energy exchange between computing agents 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 to incorporate the local search methods into EMAS. Such sample set of actions cover almost all search activities appearing in GA and MA as selection, mutation, crossover and local optimization.

Let us 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. We use lower index "next" for denoting the state which may appear in the next step assuming some current state, e.g., x_{next} is the subsequent possible value of x.

6.1. Action performing distributed selection. The energy transfer action get is based on the idea of agent rendezvous. Agents meet one of their neighbors (it chooses randomly one of its neighbors using the uniform distribution—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. [62]).

Taking into account Observation 2.1 we may get the following

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

$$\delta_{get}^{gen,n}|X_{gen,n}\ (x)(1) = \begin{cases} 1 & \text{if } NBAG_{l,gen,n} \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (29)$$

$$NBAG_{l,gen,n} = \{(j,k) : x(l,j,k) > 0 \text{ and } (j \neq gen \text{ or } k \neq n)\} \quad (30)$$

$$\vartheta_{get}^{gen,n}|X_{gen,n}\ (x)(x') = \frac{1}{\#NBAG_{l,gen,n}} \sum_{(gen,n)\in NBAG_{l,gen,n}} \Big(cmp(gen, gen')(0) \cdot [x' = next(x, gen, n, gen', n')] + cmp(gen, gen')(1) \cdot [x' = next(x, gen', n', gen, n)] \Big) \quad (31)$$

$$cmp : U \times U \to \mathcal{M}(\{0,1\}) \quad (32)$$

$$next(x, a, b, a', b') = x_{next} : x_{next}(i,j,k) = \begin{cases} x(i,j,k) - \Delta e & \text{if } j = a \text{ and } k = b \text{ and } i = l \\ x(i,j,k) + \Delta e & \text{if } j = a' \text{ and } k = b' \text{ and } i = l \\ x(i,j,k) & \text{otherwise} \end{cases} \quad (33)$$

Explanation. The decision of the action get, $\delta_{get}^{gen,n}$, defined by Eq. (29), 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. (30). 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 genotype which has the better fitness has a greater probability of getting the quantum of energy from its neighbor. 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. (31) 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. (33). The incidence matrix $x_{next} \in X$ is obtained from x by changing two entries related to the pair of agents $ag_{gen,n}, ag_{gen',n'}$ that exchanged energy.

Observation 6.2. The value of the probability transition function imposed by Eqs. (10), (11), (29) – (33) performed by 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 introduce changes only in entries of the state associated with the location l. In other words, assuming x as a current state, all states that differ from x outside the l-th layer have a null probability in the next step.

Explanation. Observation 6.1 stems from Eqs. (10), (11), (29) – (33). 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.

6.2. Actions inspired by the genetic operations. A decision on the reproduction action $repr$ is based on the idea of agent rendezvous (similarly to get). An agent with sufficient energy (above a certain predefined threshold e_{repr}) meets one of their neighbors capable of reproducing (choosing it from the neighbors having sufficient energy by random sampling with uniform distribution), and creates an 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 [31]). 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.

Taking again into account Observation 2.1 we obtain the following:

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

$$\delta_{repr}^{gen,n}|X_{gen,n}\ (x)(1) = \begin{cases} 1 & \text{if } x(l,gen,n) > e_{repr} \text{ and } RPAG_{l,gen,n} \neq \emptyset \\ & \text{and } \sum_{j=1}^{r} \sum_{k \in P_l} [x(l,j,k) > 0] < q_l \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

$$RPAG_{l,gen,n} = \{(gen', n') \in NBAG_{l,gen,n}; \\ x(l, gen', n') > e_{repr}\} \tag{35}$$

$$\vartheta_{repr}^{gen,n}|X_{gen,n}\ (x)(x') = \frac{1}{\#RPAG_{l,gen,n}} \sum_{(gen',n') \in RPAG_{l,gen,n}} \sum_{gen'' \in U} mix(gen, gen')(gen'') \\ cpchoose(x, x', gen, n, gen', n', gen'') \tag{36}$$

$$cpchoose(x, x', gen, n, gen', n', gen'') = \begin{cases} [x' = x] \text{ if } FC_{l,gen''} = \emptyset \\ \frac{1}{\#FC_{l,gen''}} \sum_{m \in FC_{l,gen''}} [x' = next(x, gen, n, gen', n', \\ gen'', m)] \text{otherwise} \end{cases} \tag{37}$$

where

$$FC_{l,gen''} = \{o \in P_l \mid x(l, gen'', o) = 0\} \tag{38}$$

$$next(x, a, b, a', b', a'', m) = x_{\text{next}} : \\ x_{\text{next}}(i,j,k) = \begin{cases} x(i,j,k) - \frac{e_0}{2} & \text{if } j \in \{a, a'\} \text{ and } k \in \{b, b'\} \\ & \text{and } i = l \\ e_0 & \text{if } j = a'' \text{ and } k = m \text{ and } i = l \\ x(i,j,k) & \text{otherwise} \end{cases} \tag{39}$$

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. (34)), is based on the condition that there is at least one neighboring 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. (35). 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. (36) 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 neighbor $ag_{gen',n'}$ 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 such a case there is always a free copy number: see Remark 6.2). If there are more than 1 inactive copies, the copy-number n'' of the offspring agent is selected with uniform probability distribution – see Eqs. (37), (38)).

The state transition function is constructed in the way described in Eq. (39). 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 genotype gen'', which is activated in the location l (and whose energy is set to e_0).

Observation 6.4. The value of the probability transition function imposed by Eqs. (10), (11), (34)–(39), 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 as well.

Explanation. Equations (34)–(39) involve only those entries of the incidence matrix that are related to the location l, so both assumptions of Definition 3.1 are satisfied straightforwardly.

A decision on the migration action $migr$ may be undertaken by an 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 its location to another one.

Observation 6.5. The probability transition function $\varrho^{gen,n}_{migr} : X \to \mathcal{M}(X)$ associated with action $migr$ is determined by:

$$\delta^{gen,n}_{migr}|X_{gen,n}\ (x)(1) = \begin{cases} 1 & \text{if } (x(l,gen,n) > e_{migr} \text{ and } \#ACCLOC_l > 0) \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

$$ACCLOC_l = \left\{ Loc \setminus \{l\} \ni l' : ((l,l') \in Top) \text{ and } \left(\sum_{j=1}^{r} \sum_{k \in P_{l'}} [x(l',j,k) > 0] < q_{l'} \right) \right\} \tag{41}$$

$$\vartheta^{gen,n}_{migr}|X_{gen,n}\ (x)(x') = \frac{1}{\#ACCLOC_l} \sum_{loc' \in ACCLOC_l} \frac{1}{\#FC_{loc',gen}} \sum_{m \in FC_{loc',gen}} [x' = next(x,gen,n,loc',m)] \tag{42}$$

where $FC_{loc',gen}$ is given by Eq. (38) and

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

$$x_{next}(i,j,k) = \begin{cases} 0 & \text{if } i = l \text{ and } j = a \text{ and } k = b \\ x(l,a,b) & \textit{if } i = c \textit{ and } j = a \text{ and } k = m \\ x(i,j,k) & \text{otherwise.} \end{cases} \tag{43}$$

Explanation. The decision on the action $migr$, $\delta^{gen,n}_{migr}$ (defined by Eq. (40)) is based on the condition that the agent $ag_{gen,n}$ has sufficient energy to migrate (higher than $e_{migr} \geq e_{repr}$) and there is at least one neighboring location (l') that is able to accept the agent which 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. (41). The arbitrary state of the system when the decision is evaluated by the agent $ag_{gen,n}$ is denoted by $x \in X$.

In case of positive evaluation of this decision, the state transition described by Eq. (42) is performed. Again, the formula comes from Bayes' theorem which allows to check 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 that initiated the action moves from its location (l) to another one (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 6.2) according to the uniform distribution.

The state transition function is constructed as described in Eq. (43). The incidence matrix $x_{next} \in X$ is obtained from x by changing two entries related to the position of the agent $ag_{gen,n}$ in the location.

Observation 6.6. The value of the probability transition function imposed by Eqs. (10), (11), (40)–(43) 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 entries of state associated with the location l and another location.

Explanation. Eqs. (40)–(43) include references to the elements of the system contained in the l-th and other locations.

The EMAS definition presented here is enriched with respect to the one in [51, 52] by endowing it with a new cloning and mutation action clo which allows a single agent to produce an 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 an 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. [31]). 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.

Taking again into account Observation 2.1 we may get the following

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

$$\delta_{clo}^{gen,n}|X_{gen,n}\ (x)(1) = \begin{cases} 1 & \text{if } x(l,gen,n) > e_{repr} \\ & \text{and} \sum_{j=1}^{r} \sum_{k \in P_l} [x(l,j,k) > 0] < q_l \\ 0 & \text{otherwise} \end{cases} \tag{44}$$

$$\vartheta_{clo}^{gen,n}|X_{gen,n}\ (x)(x') = \sum_{gen' \in U} mut(gen)(gen') \cdot cpchoose_1(x,x',gen,n,gen') \tag{45}$$

$$cpchoose_1(x,x',gen,n,gen') = \begin{cases} [x'=x] \text{ if } FC_{l,gen'} = \emptyset \\ \dfrac{1}{\#FC_{l,gen'}} \displaystyle\sum_{m \in FC_{l,gen'}} [x' = \\ next(x,gen,n,gen',m)] \text{ otherwise} \end{cases} \tag{46}$$

where $FC_{l,gen'}$ is given by Eq. (38) and

$$next(x,a,b,a',m) = x_{\text{next}} : x_{\text{next}}(i,j,k) = \begin{cases} x(i,j,k) - e_1 & \text{if } j=a \text{ and } k=b \\ & \text{and } i=l \\ e_1 & \text{if } j=a' \text{ and } k=m \\ & \text{and } i=l \\ x(i,j,k) & \text{otherwise} \end{cases} \tag{47}$$

Explanation. The decision $\delta_{clo}^{gen,n}$ on the action clo defined by Eq. (44) is based on the 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. (45) is performed. As for the previous actions, this formula stems from Bayes' theorem. Depending on the target agent (after applying 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. (47). 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 to the offspring agent with genotype gen', which is activated in location l and whose energy is set to e_0.

Observation 6.8. The value of the probability transition function imposed by Eqs. (10), (11), (44)–(47) 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. (44)–(47) 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.

6.3. Action resulting from the local search activation. Using a mechanism similar to the one included in the definition of action clo we are able to represent local searches invoked from particular points encoded by genotypes in U. The action implementing the local search will be called lse. The agent executing lse produces a new agent containing a new genotype gen' resulting 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. (44).

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 we typically assume 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 landscape defined by the neighborhood structure considered and the fitness function corresponding to the optimization 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 similar way as it is carried out during execution of the action clo. The above assumptions together with the Observation **??** lead to the following:

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. (44) and the actions' kernel by the Eqs. (45)–(47) 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 6.2 formulated and proven for the clo action. Similarly, without additional verification we may accept the following:

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 does not introduce changes in other locations.

6.4. Action's taxonomy. Observations 6.1, 6.4, 6.6, 6.8 and 6.10 may be summarized in the following corollary:

Corollary 6.1. Action $migr$ is global, whereas actions get, $repr$, clo and lse are local, i.e.

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

Observation 6.11. The probability transitions imposed by actions get, $repr$, $migr$, clo and lse satisfy the Markov condition (see, e.g., [63]).

Explanation. The probability transitions of the actions $\varrho_{get}^{gen,n}$, $\varrho_{repr}^{gen,n}$, $\varrho_{migr}^{gen,n}$, $\varrho_{clo}^{gen,n}$, $\varrho_{lse}^{gen,n}$ given by Eqs. (10), (11), (29)–(43) depend only on the current state $x \in X$ of the system.

7. Ergodicity of EMAS

We intend to analyze some asymptotic features of the presented model in order to draw significant conclusions regarding the capabilities of localizing optima of a given function by EMAS with actions definitions given in 6.

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 neighbors).
2. The quantum of energy is lower than or equal to the total energy divided by the maximum number of agents that may be present in the system $\Delta e \leq \frac{1}{\sum_{i=1}^{s} q_i}$. This assumption allows to achieve a maximal population of agents in the system.
3. The reproduction (cloning) energy is lower than two energy quanta $e_{repr} \leq 2\Delta e$.
4. The amount of energy passed from parent to child during the action clo is equal to Δe (so $n_1 = 1$).
5. The maximum number of agents on every location is greater than one, $q_i > 1, i = 1, \ldots, s$.
6. Locations are totally connected, i.e. $Top = Loc^2$.
7. Each active agent can be selected by its local agent with strictly positive probability, i.e.
   $\exists\ \iota_{agsel} > 0;\ \forall\ i \in Loc, \forall\ gen \in U, \forall$
   $n \in P_i,\ \forall\ x \in \{y \in X; y(i, gen, n) > 0\}$,
   $agsel_i(x)(gen, n) \geq \iota_{agsel}$.
8. The families of probability distributions being the parameters of EMAS have uniform, strictly positive lower bounds:
   $\exists\ \iota_{\omega} > 0;\ \forall\ x \in X,\ gen \in U, \alpha \in Act$,
   $\omega(gen, x)(\alpha) \geq \iota_{\omega}$,
   $\exists\ \iota_{cmp} > 0;\ \forall\ gen, gen' \in U$,
   $cmp(gen, gen') \geq \iota_{cmp}$,
   $\exists\ \iota_{mut} > 0; \forall gen, gen' \in U$,
   $mut(gen)(gen') \geq \iota_{mut}$,
   $\exists\ 0 < \iota_{locsel} < 1; \forall\ x \in X$,
   $\forall\ j \in Loc,\ locsel(x)(j) \geq \iota_{locsel}$.

Then the Markov chain modeling EMAS (see Eq. (27)) is irreducible, i.e. any states $x_b, x_e \in X$ communicate.

Remark 7.1. Note that assumption 7 of Theorem 7 is reasonable because the number of possible states of the system is finite and so is the number of locations.

Remark 7.2. The definition of the space of states X (see Eq. (2.2)) immediately implies that there already exists at least one computational agent in EMAS and that at least one location is nonempty at any time.

Because of the complexity, the detailed proof is postponed to the appendix (see Appendix B).

Outline of the proof of Theorem 7.1. 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. We impose that the above mentioned passage may be performed by the following sequence of stages (see Fig. 1).

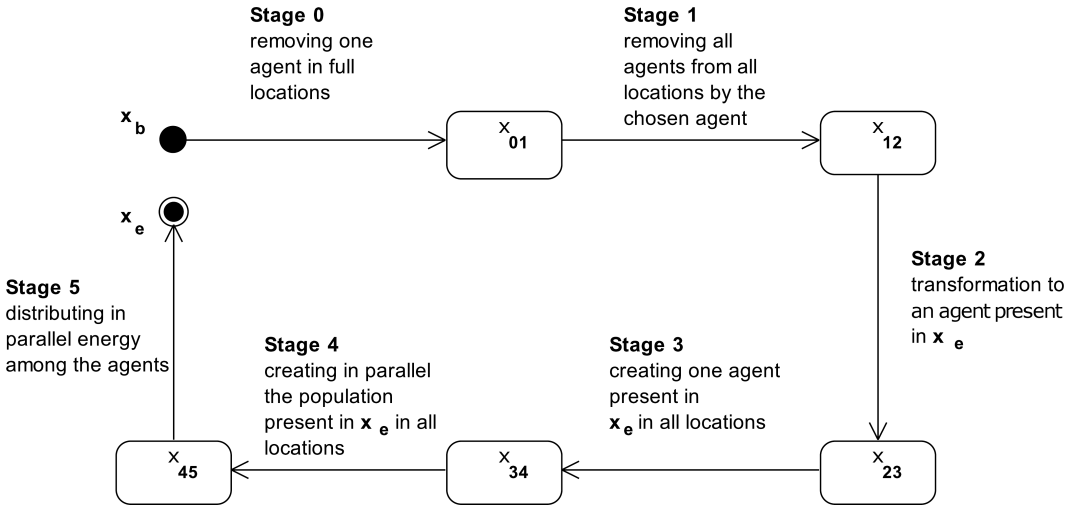


Fig. 1. State transitions in irreducibility proof

- **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 neighbors in order to remove it (to enable incoming migration from any other location, in case this location is full). After removing one of its neighbors 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 (failing to do it) during the whole stage. Final state of the *Stage 0* is denoted by x_{01}.
- **Stage 1**: One location is chosen, at which the sum of agents' energy exceeds the migration threshold in the state x_{01} (based on assumption 1 of Theorem 7 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 neighbors, 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.
- **Stage 2**: The chosen agent ag_{gen_1,n'_1} performs the clo action producing $ag_{gen_2,n_2}, n_2 \in P_{i_1}$, being one of the agents present in the state in the location i_2. The location i_2 will contain 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) visiting 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. The cloned agent in each non-empty location 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 we will further call in the same manner ($ag_{gen_i^{first},n_i^{first}}$) the migrating agent after it reaches the final i_2 location.
- **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 maximum 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 neighbors in order to pass them the sufficient amount of energy required in the state x_e.

We have shown that every of the aforementioned stages requires performing at most a finite number of Markov chain steps. Moreover, we have shown that every aforementioned sequence has non-zero probability. For the detailed proof of the above features refer to Appendix B.

Remark 7.3. Theorem 7 leads us straightforwardly to the statement that every possible state of EMAS is reachable after performing a finite sequence of transitions independently on the initial population. Therefore, also the states containing the extrema are reachable. Thus any metaheuristics respecting EMAS architecture and the assumption of Theorem 7, satisfies the asymptotic guarantee of success [36, 37].

Theorem 7.2. If the assumptions of Theorem 7 hold, then the Markov chain modeling EMAS is aperiodic.

Proof. Let us consider a state of the chain such that every 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, the master agent will allow them all to perform their actions, however the absence of neighbors 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. (25). 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 considered state is aperiodic. Our chain is irreducible (see Theorem 7) and therefore it has only one class of states, the whole state space, which obviously contains the considered aperiodic state. On the other hand, from Theorem 2.2 of [64] we know that aperiodicity is a state class property. In our case it means that all states of EMAS are aperiodic, which concludes the proof.

The following corollary is a consequence of Theorems 7 and 1.

Corollary 7.1. The Markov chain modeling EMAS is *ergodic*.

Remark 7.4. The Markov chain (27) is ergodic in a strong sense, namely it is not only irreducible, but also aperiodic. Such chains are quite often called *regular* (see e.g. [64]). Obviously it is also ergodic in a weaker (and also quite common) sense, meaning that it is simply irreducible.

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

$$Q = \{\tau(x)(y)\},\ x, y \in X \tag{48}$$

where τ is the EMAS probability transition function – see Eq. (27). 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

$$\xi_{t+1} = Q \cdot \xi_t,\ t = 0, 1, \ldots \tag{49}$$

Remark 7.5. From Theorems 7 and 1 as well as the ergodic theorem [63] 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.

8. Conclusions

The paper introduces a general stochastic model related to a wide class of heuristic, population based search algorithms with a special emphasis on the memetic ones. The individuals in these algorithms are active software agents (not passive entities, as in classical EAs), characterized by a genotype and a meme that can be changed during the optimization process. The population can be spread among demes (evolutionary islands) working in parallel. The actions includes a broad variety of optimization activities as the conventional mixing (e.g. mutation, cloning, crossover), migrations among demes and local optimization methods. We assume that both, genes and memes belong to finite universa.

The model is composed of two parts. The first one is EMAS architecture described by the data structures and management strategies described in Secs. 2, 4, 6 and the Pseudocodes 4.1, 4.2 and 4.3. This model allows for defining the space of states (see formula (2.2)) and the framework for describing stochastic effect of agent's actions (see formulas (4)–(10)). The second part is the stationary Markov chain defined in terms of the EMAS architecture (see Sec. 5). In particular the general form of the probability transition function was established (see formula (27)) and the Markov kernels for sample actions were drawn (see Sec. 6 formulas (30)–(47)).

The class of global optimization metaheuristics that coincide with the proposed model is of great practical importance. The application of memetic algorithm being the most advanced ones is depicted in Sec. 1.

We performed a detailed analysis of inter-actions dependencies (see Sec. 3) In particular, we formulated and prove precise mathematical criteria enabling to classify an action as the global one, which has to be executed exclusively in the whole system, or the local one, which can be performed independently with other local action in the separate location (see Definition 3.1, Propositions A.1 and 3.1).

The double indexing of the computing agents $ag_{gen,n}$ with the time-varying number of copy n allows us to describe the concurrent execution of actions representing mutation, crossover and local search activation. This approach extends the previous version of EMAS – see [52] – to the case in which the most burdensome operations involving fitness evaluation can be run in parallel in a separate locations. On the other hand, this kind of time dependent indexing makes it impossible to distinguish agents contained in the set Ag, which is the main drawback of such a solution, making the presented stochastic model less applicable for multi-agent software systems with a predefined static set of agents. However, in the context of memetic algorithms finding an objective's minimizers, the identity of agents turns out to be less important. In this situation the crucial agent's attributes are the genotype and life energy whereas the copy number plays only an auxiliary role. Nevertheless, the distinction of all agents that were active during at least one time step is possible if we compute a true agent identifier as a composition of the genotype with the sequence of the agent's copy numbers at subsequent moments including 0 at the moments when the agent is not active.

The ergodicity of the obtained Markov chain assigned to the studied memetic metaheuristics was proven (see Theorem 7.1, Theorem 7.2, Corollary 7.1, Remark 7.4).

The obtained theoretical results are helpful for studying important features of stochastic global optimization metaheuristics conforming EMAS architecture and implemented as a concurrent system in a distributed computing environment.

The proper synchronization among agents designed respecting the classification of agent's actions assures *safeness*. Moreover the safe, coarse-grained parallel computation and its effective, sub-optimal scheduling in a distributed computer environment (computer cluster) might be obtained.

The proved strong ergodicity of the finite state Markov chain modeling the metaheuristics (see Remark 7.4) 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 (see 7.3). This condition imposes in particular *liveness* of the metaheuristic studied.

The Markov chain allows to obtain its sampling measure in the arbitrary step by iterating the probability transition function. It constitutes the necessary basis for future asymptotic studies e.g. various kinds of stochastic convergence of a studied metaheuristic (see [33, 35]).

Other possibility of further studying the memetic metaheuristics makes the thesis of Remark 7.5. In particular, the behavior of the limit, equilibrium distributions by growing the number of agent's copies might be considered.

The theory presented here may be extended to the case of memetic algorithms including immune mechanisms. Such a model called iEMAS was also applied to the case of a continuous state space and to very restricted agent's features – see [51, 52]. The formal verification of iEMAS ergodicity similar to those presented in Theorems 7.1 and 7.2 will be an object of our future research.

Appendix A. Commutativity of local actions

Proposition A.1. Let $\varrho_1, \varrho_2 : X \to \mathcal{M}(X)$ satisfy (13) and (14) with l_1 and l_2, $l_1 \neq l_2$ respectively (i.e. $\varrho_1 = \varrho_\alpha^{gen,n}$ and $\varrho_2 = \varrho_{\alpha'}^{gen',n'}$ for some $gen, gen' \in U,\ n, n' \in P$ so that $(gen, n) \neq (gen', n')$, $x(l_1, gen, n) > 0,\ x(l_2, gen', n') > 0$, $x(l_1)^T x(l_2) = 0$ and for some $\alpha, \alpha' \in Act$). Then

$$\sum_{y \in X} \varrho_1(x)(y) \cdot \varrho_2(y)(x') = \sum_{y \in X} \varrho_2(x)(y) \cdot \varrho_1(y)(x') \quad (50)$$

for all $x, x' \in X$.

Proof. Let us fix $x \in X$. Denote by $\varrho_{2,1}(x)(x')$ the left-hand side of (50) and by $\varrho_{1,2}(x)(x')$ the right-hand side of the equation. First let us note that for $x' \in X$ such that $x(l) \neq x'(l)$ for some $l \notin \{l_1, l_2\}$ from (13) we obtain

$$\varrho_1(x)(x') = \varrho_2(x)(x') = 0$$

as well as

$$\varrho_{2,1}(x)(x') = \sum_{\substack{y \in X \\ y(i) = x(i),\ i \neq l_1}} \varrho_1(x)(y) \cdot \varrho_2(y)(x'). \quad (51)$$

But if $y(i) = x(i)$ for $i \neq l_1$ then $x'(l) \neq y(l)$, hence we obtain

$$\varrho_2(y)(x') = 0.$$

Therefore for $x, x' \in X$ such that $x(l) \neq x'(l)$ for some $l \notin \{l_1, l_2\}$ we have

$$\varrho_{2,1}(x)(x') = 0. \quad (52)$$

Exchanging indices 1 and 2 in the previous considerations we easily obtain formula (52) for $\varrho_{1,2}$.

It remains to prove (50) for $x, x' \in X$ such that $x(i) = x'(i)$ for all $i \notin \{l_1, l_2\}$. To this end, we can once again apply (13) to (51) obtaining

$$\varrho_{2,1}(x)(x') = \sum_{\substack{y \in X \\ y(i) = x(i),\ i \neq l_1 \\ y(i) = x'(i),\ i \neq l_2}} \varrho_1(x)(y) \cdot \varrho_2(y)(x') \quad (53)$$

It is easy to see that in this case there is exactly one $\hat{y}$ satisfying the conditions under the sum. It is given by the following formula:

$$\hat{y}(i) = \begin{cases} x(i) = x'(i) & \text{for } i \notin \{l_1, l_2\}, \\ x(l_2) & \text{for } i = l_2, \\ x'(l_1) & \text{for } i = l_1. \end{cases}$$

Hence (53) takes the following form.

$$\varrho_{2,1}(x)(x') = \varrho_1(x)(\hat{y}) \cdot \varrho_2(\hat{y})(x') \quad (54)$$

Since $x(l_2) = \hat{y}(l_2)$ we can apply (14) to obtain

$$\varrho_2(\hat{y})(x') = \varrho_2(x)(\hat{y}')$$

with

$$\hat{y}'(i) = \begin{cases} x(i) = x'(i) & \text{for } i \notin \{l_1, l_2\}, \\ x(l_1) & \text{for } i = l_1, \\ x'(l_2) & \text{for } i = l_2. \end{cases}$$

Therefore

$$\varrho_{2,1}(x)(x') = \varrho_1(x)(\hat{y}) \cdot \varrho_2(x)(\hat{y}') \quad (55)$$

As this equality is symmetric with respect to the layer number, we can repeat the same reasoning obtaining exactly the same result for $\varrho_{1,2}(x)(x')$, which concludes the proof.

Appendix B. Proof of the Theorem 7.1

We precede the proof of Theorem 7.1 by a series of useful technical lemmas.

Lemma B.1. Given the assumptions of Theorem 7.1, there exists a positive constant $0 < \zeta_0 \leq \frac{1}{2}$ such, that $\zeta_0 \leq \zeta^{gl}(x)$ and $\zeta_0 \leq \zeta^{loc}(x)$ for all $x \in X$.

Proof. According to Formula (18)

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

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. (17). Because $Act_{loc} \neq \emptyset$ and there always exists at least one agent (see Remark 7.2) we have

$$\xi_i(x) \geq \iota_{agsel} \cdot \iota_\omega$$

for all $x \in X$ and $i = 1, \ldots, s$. Finally, because at least one location contains an agent (see Remark 7.2 again) we may evaluate

$$\zeta^{loc}(x) \geq \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_\omega = \zeta_0$$

for all $x \in X$. Replacing Act_{loc} by Act_{gl} ($Act_{gl} \neq \emptyset$) we similarly obtain

$$\zeta^{gl}(x) \geq \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_\omega = \zeta_0$$

for all $x \in X$. The constant ζ_0 is strictly positive as it is the product of strictly positive numbers. Moreover $2\zeta_0 \leq \zeta^{gl}(x) + \zeta^{loc}(x) = 1$ for all $x \in X$, so $\zeta_0 \leq \frac{1}{2}$.

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. (42) describing the $migr$ action's kernel $\varrho_{migr}^{gen,n}$.

Lemma B.3. If $\exists \iota_{mut} > 0;\ mut(gen)(gen') \geq \iota_{mut}\ \forall gen, gen' \in U$ then $\exists\ \iota_{clo} > 0$ such that $\varrho_{clo}^{gen,n}(x)(x') \geq \iota_{clo}$ 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. (38) the set $FC_{l,gen'}$ would be nonempty. Furthermore, taking into account (46), (45) 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.4. If $\exists\ \iota_{cmp} > 0$; $cmp(gen, gen') \geq \iota_{cmp}\ \forall gen, gen' \in U$ then $\exists\ \iota_{get} > 0$ such that $\varrho_{get}^{gen,n}(x)(x') \geq \iota_{get}$ and $\varrho_{get}^{gen,n}(x)(x'') \geq \iota_{get}$ 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. (30)) is nonempty. Then according to Eqs. (31)–(33)

$$\varrho_{get}^{gen,n}(x)(x') = \vartheta_{get}^{gen,n}(x)(x') > \frac{\iota_{cmp}}{(\max_{i \in Loc}\{q_i\}) - 1} = \iota_{get} > 0.$$

The same reasoning leads to $\varrho_{get}^{gen,n}(x)(x'') > \iota_{get}$.

Lemma B.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

$$P\left(\bigcap_{i=1}^{k} A_i\right) \geq \prod_{i=1}^{k} \lambda_i > 0. \tag{56}$$

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

$$P\left(\bigcap_{i=1}^{k} A_i\right) = P(A_1) \cdot \prod_{i=2}^{k} P\left(A_i \,\middle|\, \bigcap_{j=1}^{i-1} A_j\right) \tag{57}$$

may be proven inductively, which is enough to prove the lemma.

Proof of Theorem 7.1 – 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. We have already shown (see *Outline of the proof of Theorem* 7.1, Sec. 7) that the passage mentioned above may be performed by the sequence of stages 1 – 5 described there and illustrated in Fig. 7. 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.6. Given the assumptions of Theorem 7.1, 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 Subsec. 2.1).

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 maximum 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 neighbors $ag_{gen'_i,n'_i}$. Both agents are fixed during the *Stage 0*. After removing its neighbor, the agent begins to perform the global action $migr$ and fails (rejected to do it by the master agent), 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 the master agent), until the end of the Stage 0.

The probability of performing a single step of the described sequence is given by:

$$\zeta^{loc}(z) \prod_{i\in Loc_{get}} \big(agsel_i(z)(gen_i,n_i)\cdot\omega(gen_i,z)(get)\cdot \delta_{get}^{gen_i,n_i}(z)(1)\cdot\varrho_{get}^{gen_i,n_i}(z_i')(z_i'')\big) \prod_{i\in Loc_{migr}} \big(agsel_i(z)(gen_i,n_i)\cdot\omega(gen_i,z)(migr)\big) \geq \zeta_0 \prod_{i\in Loc_{get}} (\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get})\cdot \prod_{i\in Loc_{migr}} (\iota_{agsel}\cdot\iota_\omega) \geq \zeta_0\left(\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\right)^s \quad (58)$$

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. (58) is arbitrary, because the get action is local (see Observation 6.1). $z_i' = z$ for the first location from Loc_{get}, $z_j' = z_i''$ where j is a location next to i in Eq. (58). 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. (58) is uniform for all steps in *Stage 0*. The maximum 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 8, the probability of the whole sequence can be bounded from below by:

$$pr_0 = \left(\zeta_0\left(\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\right)^s\right)^{m-1} > 0. \quad (59)$$

Lemma B.7. Given the assumptions of Theorem 7 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 7.1 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 8 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 the master agent.

The probability of performing one action get in the sequence discussed is given by:

$$\zeta^{loc}(z)\Big(agsel_i(z)(gen_1,n_\xi)\cdot\omega(gen_1,z)(get)\cdot \delta_{get}^{gen_1,n_\xi}(z)(1)\cdot\varrho_{get}^{gen_1,n_\xi}(z)(z')\Big) \cdot \prod_{j\in Loc_{migr}} \big(agsel_j(x)(gen_j,n_j)\cdot\omega(gen_j,z)(migr)\big) \geq \zeta_0\left(\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\right) \cdot \prod_{i\in Loc_{migr}} (\iota_{agsel}\cdot\iota_\omega) \geq \zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get} \quad (60)$$

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. (60) 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:

$$\left(\zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\right)^{m-1}. \quad (61)$$

After the removal of all its neighbors the chosen agent has to perform migration. The probability of this step is given by:

$$\zeta^{gl}(z)\cdot locsel(z)(i) \Big(agsel_i(z)(gen_1,n_\xi)\cdot\omega(gen_1,z)(migr) \delta_{migr}^{gen_1,n_\xi}(z)(1)\cdot\varrho_{migr}^{gen_1,n_\xi}(z)(z'')\Big) \geq \zeta_0\cdot\iota_{locsel}\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{migr} \quad (62)$$

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. (61)–(62). According to Lemma B.5, the probability of the whole sequence can be bounded from below by:

$$pr_1 = \left(\zeta_0\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{get}\right)^{s\cdot(m-1)} \left(\zeta_0\cdot\iota_{locsel}\cdot\iota_{agsel}\cdot\iota_\omega\cdot\iota_{migr}\right)^{s-1} \quad (63)$$

Moreover, the number of steps in the sequence may be estimated by the constant $st_1 = s(m-1)+s-1$.

Lemma B.8. Given the assumptions of Theorem 7 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:

$$\zeta^{loc}(x_{12}) \cdot agsel_{i_1}(x_{12})(gen_1, n_1') \cdot \omega(gen_1, x_{12})(clo) \cdot \delta_{clo}^{gen_1,n_1'}(x_{12})(1) \cdot \varrho_{clo}^{gen_1,n_1'}(x_{12})(z') \geq \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo} \quad (64)$$

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:

$$\zeta^{loc}(z)\Big(agsel_{i_1}(z)(gen_1, n_1') \cdot \omega(gen_1, z)(get) \cdot \delta_{get}^{gen_1,n_1'}(1) \cdot \varrho_{get}^{gen_1,n_1'}(z)(z'')\Big) \geq \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{get} \quad (65)$$

where z is the current state (at the end of the stage z is 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 8, the probability of *Stage 2* is evaluated from below by:

$$pr_2 = \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo} \cdot (\zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{get})^{m-1} > 0 \quad (66)$$

and $st_2 = 1 + (m-1) = m$.

Lemma B.9. Given the assumptions of Theorem 7 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:

$$\zeta^{gl}(z) \cdot locsel(z)(i) \Big(agsel_i(z)(gen_2, n_\xi) \cdot \omega(gen_2, z)(migr) \delta_{migr}^{gen_2,n_\xi}(z)(1) \cdot \varrho_{migr}^{gen_2,n_\xi}(z)(z')\Big) \geq \zeta_0 \cdot \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{migr} \quad (67)$$

where z is the current state (in the beginning $z = x_{23}$ at the end of the stage z is 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.

Let us denote now by $Loc_{migr} \subset Loc$ a nonempty 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 master agent.

The probability of each step in which the new agent $ag_{gen_i^{first},n_i^{first}}$ is produced is evaluated by:

$$\zeta^{loc}(z) \cdot agsel_i(z)(gen_2, n_\xi) \cdot \omega(gen_2, z)(clo) \cdot \delta_{clo}^{gen_2,n_\xi}(z)(1) \cdot \varrho_{clo}^{gen_2,n_\xi}(z)(z') \prod_{j \in Loc_{migr}} (agsel_j(z)(gen_j, n_j) \cdot \omega(gen_j, z)(migr)) \geq \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo} \cdot (\iota_{agsel} \cdot \iota_\omega)^{s-1} \quad (68)$$

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:

$$\zeta^{loc}(z)\Big(agsel_i(z)(gen_2, n_\xi) \cdot \omega(gen_2, z)(get) \cdot \delta_{get}^{gen_2,n_\xi}(z)(1) \cdot \varrho_{get}^{gen_2,n_\xi}(z)(z'')\Big) \prod_{j \in Loc_{migr}} (agsel_j(z)(gen_j, n_j) \cdot \omega(gen_j, z)(migr)) \geq \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{get} \cdot (\iota_{agsel} \cdot \iota_\omega)^{s-1} \quad (69)$$

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. (68).

Assuming A_t to be the consecutive t-th event described above, according to Lemma B.5, the probability of the whole *Stage 3* is bounded from below by:

$$pr_3 = \Big(\zeta_0 \cdot \iota_{locsel} \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{migr} \cdot \zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo} \cdot (\iota_{agsel} \cdot \iota_\omega)^{s-1}\Big)^{s-1} \big(\zeta_0 \cdot \iota_{agsel} \cdot \iota_\omega \cdot \iota_{get} \cdot (\iota_{agsel} \cdot \iota_\omega)^{s-1}\big)^{m-1} > 0. \quad (70)$$

The number of steps required for performing the whole sequence is $st_3 = 2 \cdot (s-1) + (m-1)$.

Lemma B.10. Given the assumptions of Theorem 7 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:

$$\begin{aligned}
&\zeta^{loc}(z) \prod_{i\in Loc_{clo}} \Big(agsel_i(z)(gen_i^{first}, n_i^{first}) \cdot \\
&\omega(gen_i^{first}, z)(clo) \cdot \\
&\delta_{clo}^{gen_i^{first},n_i^{first}}(z)(1) \cdot \varrho_{clo}^{gen_i^{first},n_i^{first}}(z')(z'') \Big) \\
&\prod_{i\in Loc_{migr}} \Big(agsel_i(z)(gen_i, n_i) \cdot \omega(gen_i, z)(migr) \Big) \\
&\geq \zeta_0 \prod_{i\in Loc_{clo}} (\iota_{agsel} \cdot \iota_\omega \cdot \iota_{clo}) \prod_{i\in Loc_{migr}} (\iota_{agsel} \cdot \iota_\omega) \\
&\geq \zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{clo}, \iota_\omega\}\right)^s \qquad (71)
\end{aligned}$$

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.5, the probability of the whole sequence will be bounded from below by:

$$pr_4 = \left(\zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{clo}, \iota_\omega\}\right)^s\right)^{\max_{i\in Loc}\{q_i\}} > 0. \qquad (72)$$

Lemma B.11. Given the assumptions of Theorem 7, 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 the master agent (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 neighbors (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:

$$\begin{aligned}
&\zeta^{loc}(z) \prod_{i\in Loc_{get}} \Big(agsel_i(z)(gen_i^{first}, n_i^{first}) \cdot \\
&\omega(gen_i^{first}, z)(get) \cdot \\
&\delta_{get}^{gen_i^{first},n_i^{first}}(z)(1) \cdot \varrho_{get}^{gen_i^{first},n_i^{first}}(z')(z'') \Big) \\
&\prod_{i\in Loc_{migr}} \Big(agsel_i(z)(gen_i, n_i) \cdot \omega(gen_i, z)(migr) \Big) \\
&\geq \zeta_0 \prod_{i\in Loc_{get}} (\iota_{agsel} \cdot \iota_\omega \cdot \iota_{get}) \prod_{i\in Loc_{migr}} (\iota_{agsel} \cdot \iota_\omega) \\
&\geq \zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{get}, \iota_\omega\}\right)^s \qquad (73)
\end{aligned}$$

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.5, the probability of the whole sequence will be bounded from below by:

$$pr_5 = \left(\zeta_0 \left(\iota_{agsel} \cdot \iota_\omega \cdot \min\{\iota_{get}, \iota_\omega\}\right)^s\right)^{m-1} > 0. \qquad (74)$$

To conclude the proof of Theorem 7.1 let us note that lemmas B.6–B.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

$$st = \sum_{a=0}^{5} st_a < +\infty. \qquad (75)$$

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

$$pr = \prod_{a=0}^{5} pr_a > 0 \qquad (76)$$

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.

Appendix C. Computational example

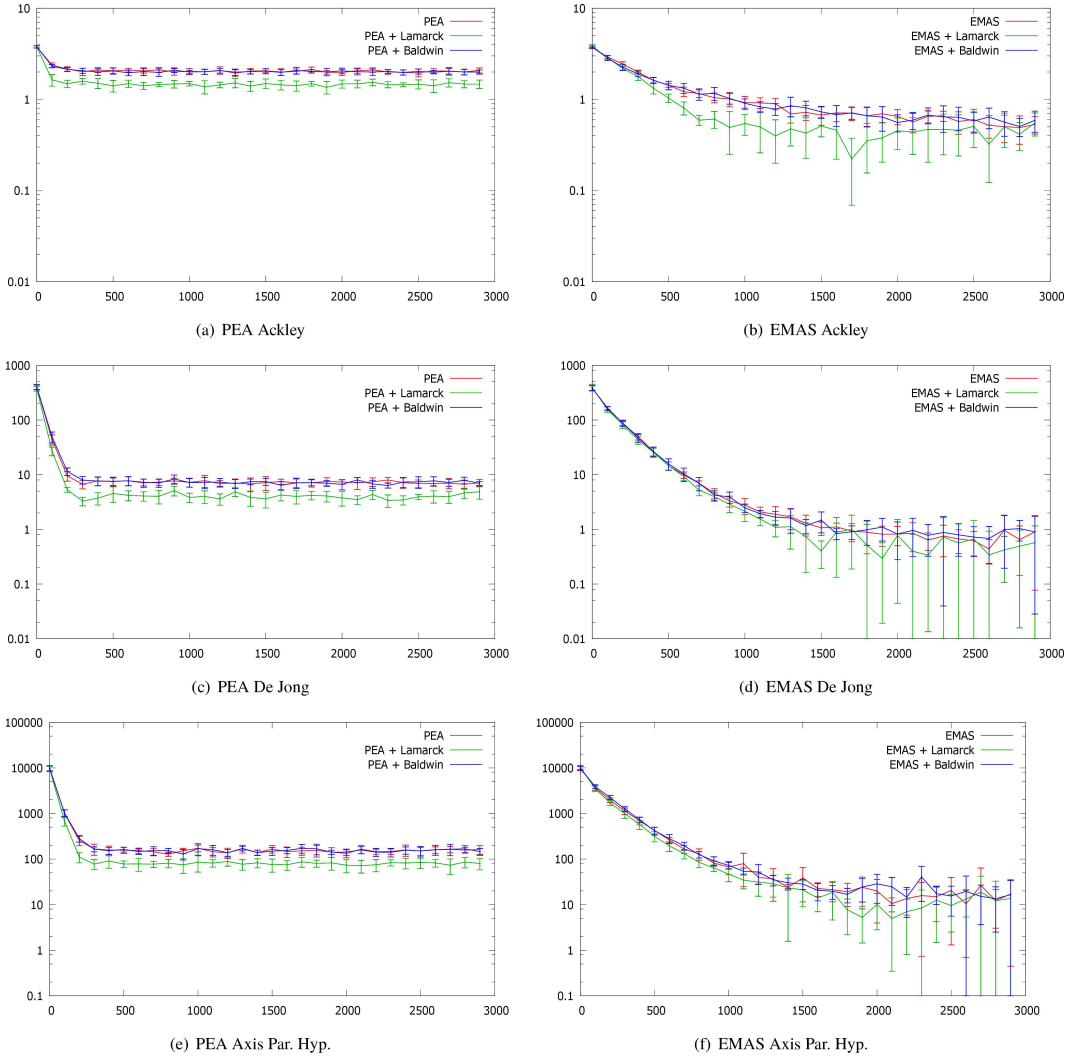
Here, some highlights of experimental results are presented, solely for illustration purposes of the applicability of the constructed model. Based on a dedicated software environment implemented using Python technology, both EMAS (evolutionary multi-agent system) and PEA (parallel evolutionary algorithm) systems were implemented and used to generate

the presented results. The configuration of the both systems is presented as follows.

- Common parameters: normal distribution-based mutation of one randomly chosen gene (described by the probability distribution $mut(\cdot)(\cdot)$ in the model), single-point crossover, the descendant gets parts of its parents genotype after dividing them in one randomly chosen point (described by the probability distribution $mix(\cdot)(\cdot)$ in the model), $q_i = 30, 1 \leq i \leq s$ individuals located on each island. All experiments were repeated 10 times and standard deviation was computed. Allopatric speciation (island model), $s = 3$ 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: $e_0 = 100$, received by the agents in the beginning of their lives, minimal reproduction energy: $e_{repr} = 90$, required to reproduce, evaluation energy win/loose: $e_{get} = 40$, passed from the loser to the winner. The values presented in the model should be computed in relation to the total energy in the system: $e_{total} = e_0 \cdot \sum_{i=1}^{s} q_i = 100 \cdot 90 = 9000$.

Memetic operators were implemented according to gradient-free steepest descent algorithm based on choosing the best from 10 potential mutated individuals. Such a procedure was repeated 30 times and the best result was returned. These memetic operators have been hybridized with EMAS in the following way:

- Baldwinian memetics: the evaluation operator is enhanced with local search algorithm. The evaluation of a certain individual starts the local search from this individual and returns the fitness of the solution found instead of the original fitness value.
- Lamarckian memetics: a dedicated mutation operator is called in the course of agent's life, therefore its genotype may be changed whenever this action is undertaken. It may be used either during the mutation or at the arbitrarily chosen moments of agent's life, depending solely on agent's own decision.



(a) PEA Ackley (b) EMAS Ackley

(c) PEA De Jong (d) EMAS De Jong

(e) PEA Axis Par. Hyp. (f) EMAS Axis Par. Hyp.

Fig. 2. Comparison of results of PEA and EMAS computations Ackley, Axis Parallel Hyperellipsoid and De Jong benchmark functions). Note the logarithmic scale on the Y-axis

The problems under considerations were follows 50-dimensional benchmark functions [65]:

- De Jong:
  $f(x) = \sum_{i=1}^{n} x_i^2, -5.12 \leq x_i \leq 5.12,$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$
- Ackley:
  $f(x) = -a \cdot e^{-b\frac{\sqrt{\sum_{i=1}^{n} x_i^2}}{n}} - e^{\frac{\sum_{i=1}^{n} cos(c \cdot x_i)}{n}} + a + e;$
  $a = 20, b = 0.2, c = 2 \cdot \pi, i \in [1, n],$
  $-32.768 \leq x_i \leq 32.768,$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$
- Axis Parallel Hyperellipsoid:
  $f(x) = \sum_{i=1}^{n} i \cdot x_i^2, -5.12 \leq x_i \leq 5.12,$
  global minimum: $f(x) = 0, x_i = 0, i \in [1, n]$

In Fig. 2, graphical comparison of EMAS and PEA computations were shown. All the graphs show the averaged experimental results along with standard deviation to visualize the dispersion of the experiments. It is easy to see, that in the all presented cases, EMAS results were better than these obtained from PEA, moreover, EMAS retained exploration and exploitation balance, that may be observed as constant improving of the result, while PEA apparently stuck in the local extremum. It is to note that Lamarckian versions of both algorithms found better results than the classical ones. Baldwinian versions were somewhat worse, their results can be easily compared to these obtained by classical versions.

Acknowledgements. The work presented in this paper was partially supported by the Polish National Science Centre research projects No. DEC-2011/03/B/ST6/01393 "Multimodel, multi-criterial and multi-adaptive strategies for solving the inverse tasks" (Robert Schaefer, Maciej Smołka) and No. N N516 500039 "Biologically inspired mechanisms in planning and management of dynamic environments" (Aleksander Byrski). Carlos Cotta is supported by Spanish MICINN under project ANYSELF (TIN2011-28627-C04-01) and by Junta de Andalucía under project DNEMESIS (TIC-6083). The authors would like to thank Wojciech Korczyński for working out the presented experimental results.

REFERENCES

[1] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold Computer Library, New York, 1991.

[2] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms", in *Technical Report* 826, California Institute of Technology, Pasadena, 1989.

[3] W.E. Hart and R.K. Belew, "Optimizing an arbitrary function is hard for the genetic algorithm", *Proc. 4th Int. Conf. on Genetic Algorithms* 4, 190–195 (1991).

[4] D.H. Wolpert and W.G. Macready, "No free lunch theorems for optimization", *IEEE Trans. on Evol. Comp.* 67 (1), CD-ROM (1997).

[5] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy, and design issues", *IEEE Trans. on Evol. Comp.* 9 (5), 474–488 (2005).

[6] P. Moscato and C. Cotta, "A modern introduction to memetic algorithms", in eds. M. Gendrau and J.-Y. Potvin, *Handbook of Metaheuristics*, vol. 146, pp. 141–183, Springer, Berlin, 2010.

[7] F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms*, *Studies in Computational Intelligence*, vol. 379, Springer-Verlag, Berlin, 2012.

[8] N.J. Radcliffe and P.D. Surry, "Formal memetic algorithms". in ed. T. Fogarty, *Evolutionary Computing: AISB Workshop Lecture Notes in Computer Science*, vol. 865, pp. 1–16, Springer-Verlag, Berlin, 1994.

[9] W.E. Hart, N. Krasnogor, and J.E. Smith, "Memetic evolutionary algorithms" in *Recent Advances in Memetic Algorithms Studies in Fuzziness and Soft Computing*, vol. 166, pp. 3–27, Springer-Verlag, Berlin, 2005.

[10] F. Neri and C. Cotta. "Memetic algorithms and memetic computing optimization: a literature review", *Swarm and Evolutionary Computation* 2, 1–14 (2012).

[11] M.G. Norman and P. Moscato, "A competitive and cooperative approach to complex combinatorial search", in *Technical Report 790, California Institute of Technology*, Pasadena, 1989.

[12] P. Moscato, "Memetic algorithms: a short introduction", in *New Ideas in Optimization*, pp. 219–234, McGraw-Hill, Maidenhead, 1999.

[13] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms", in eds. F. Glover and G. Kochenberger, *Handbook of Metaheuristics*, pp. 105–144, Kluwer Academic Publishers, Boston, 2003.

[14] Y.-S. Ong, M.-H. Lim, and X. Che, "Memetic computation – past, present & future", *IEEE Comp. Intel. Mag.* 5 (2), 24–36 (2010).

[15] W. Zhong, J. Liu, M. Xue, and Licheng Jiao, "A multiagent genetic algorithm for global numerical optimization", *IEEE Trans. on Systems, Man, and Cybernetics* B 34 (2), 1128–1141 (2004).

[16] A.S.S.M. Barkat Ullah, R. Sarker, and C. Lokan, "An agent-based memetic algorithm (AMA) for nonlinear optimization with equality constraints", *Proc. 11th Congress on Evolutionary Computation* 1, 70–77 (2009).

[17] G. Acampora, M. Gaeta, V. Loia, and A. Vitiello, "Multi-agent memetic computing for adaptive learning experiences", *2010 Proc. IEEE Int. Conf.on Fuzzy Systems* 1, 1–8 (2010).

[18] N. Krasnogor, B.P. Blackburne, E.K. Burke, and J.D. Hirst, "Multimeme algorithms for protein structure prediction", in ed. J.J. Merelo, *Parallel Problem Solving From Nature VII* of *Lecture Notes in Computer Science*, vol. 2439, pp. 769–778, Springer-Verlag, Berlin, 2002.

[19] N. Krasnogor and S. Gustafson, "A study on the use of "self-generation in memetic algorithms", *Natural Computing* 3, 53–76 (2004).

[20] J.E. Smith, "Coevolving memetic algorithms: a review and progress report", *IEEE Trans. on Systems, Man, and Cybernetics* B 37 (1), 6–17 (2007).

[21] D. Hales, "Selfish memes and selfless agents-altruism in the swap shop", *Proc. Int. Conf. on Multi Agent Systems* 1, 431–432 (1998).

[22] A. Caponio and F. Neri, "Memetic algorithms in engineering and design", in eds. F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms Studies in Computational Intelligence*, vol. 379, pp. 241–260, Springer-Verlag, Berlin, 2012.

[23] A. Długosz and T. Burczyński, "Multiobjective shape optimization of selected coupled problems by means of evolutionary algorithms", *Bull. Pol. Ac.: Tech.* 60 (2), 205–222 (2012).

[24] Ł. Chomatek and M. Rudnicki, "Application of genetically evolved neural networks to dynamic terrain generation", *Bull. Pol. Ac.: Tech.* 59 (1), 3–8 (2011).

[25] A. Byrski and M. Kisiel-Dorohinicki, "Immunological selection mechanism in agent-based evolutionary computation", *Proc. IIS: IIPWM '05 Conf., Advances in Soft Computing* 1, 411–415 (2005).

[26] A. Byrski and M. Kisiel-Dorohinicki, "Agent-based evolutionary and immunological optimization", *Proc. Computational Science – ICCS 2007, 7th Int. Conf.* 1, CD-ROM (2007).

[27] L. Siwik and R. Dreżewski, "Agent-based multi-objective evolutionary algorithms with cultural and immunological mechanisms", in *Evolutionary Computation*, pp. 541–556, In-Teh, London, 2009.

[28] R. Dreżewski, "Co-evolutionary multi-agent system with speciation and resource sharing mechanisms", *Computing and Informatics* 25 (4), 305–331 (2006).

[29] R. Dreżewski, J. Sepielak, and L. Siwik, "Classical and agent-based evolutionary algorithms for investment strategies generation", in *Natural Computing in Computational Finance Studies in Computational Intelligence*, vol. 185, pp. 181–205, Springer-Verlag, Berlin, 2009.

[30] A. Byrski, R. Dreżewski, L. Siwik, and M. Kisiel-Dorohinicki, "Evolutionary multi-agent systems", *The Knowledge Engineering Review*, (2013), to be published.

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

[32] G. Rudolph, "Evolution strategies", in *Handbook of Evolutionary Computations*, Oxford University Press, Oxford, 1997.

[33] G. Rudolph, "Models of stochastic convergence", in *Handbook of Evolutionary Computations*, Oxford University Press, Oxford, 1997.

[34] G. Rudolph, "Stochastic processes", in *Handbook of Evolutionary Computations*, Oxford University Press, Oxford, 1997.

[35] P. Pardalos and E. Romeijn, in *Handbook of Global Optimization*, vol. 2, Kluwer Academic Publisher, Dortrecht, 2002.

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

[37] A. Rinnoy Kan and G. Timmer, "Stochastic global optimization methods", *Mathematical Programming* 39, 27–56 (1987).

[38] T.E. Davis and J.C. Principe, "A simulated annealing like convergence theory for the simple genetic algorithm", *Proc. Fourth Int. Conf. on Genetic Algorithms* 1, 174–181 (1991).

[39] J. Suzuki, "A Markov chain analysis on a genetic algorithm", *Proc. 5th Int. Conf. on Genetic Algorithms, Urbana-Champaign* 1, 146–154 (1993).

[40] D.E. Goldberg and P. Segrest, "Finite Markov chain analysis of genetic algorithms", *Proc. 2nd ICGAG* 1, 1–8 (1987).

[41] S. Mahfoud, "Finite Markov chain models of an alternative selection strategy for the genetic algorithm", *Complex Systems* 7, 155–170 (1991).

[42] E. Alba and M. Tomassini, Parallelism and evolutionary algorithms, *IEEE Trans. on Evolutionary Computation* 6 (5), 443–462 (2002).

[43] E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, Kluwer Academic Publishers, Norwell, 2000.

[44] F. Fernández de Vega, and E. Cantú-Paz, *Parallel and Distributed Computational Intelligence*, vol. 269, Springer, Berlin, 2010.

[45] M. Tomassini, *Spatially Structured Evolutionary Algorithms*, Springer, Berlin, 2005.

[46] X. Xu and H. He, "A theoretical model and convergence analysis of memetic evolutionary algorithms", *Advances in Natural Computation. Lecture Notes in Computer Science* 1, 1035–1043 (2005).

[47] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong, "Classification of adaptive memetic algorithms: a comparative study", *Systems, Man, and Cybernetics, IEEE Trans. on Cybernetics* B 36 (1), 141–152 (2006).

[48] G. Rudolph, "Convergence analysis of canonical genetic algorithms", *IEEE Trans. on Neural Networks* 5 (1), 96–101 (1994).

[49] Y.J. Cao and Q.H. Wu, "Convergence analysis of adaptive genetic algorithm", *Proc. Genetic Algorithms in Engineering Systems: Innovations and Applications* 1, 85–89 (1997).

[50] K. Cetnarowicz, M. Kisiel-Dorohinicki/ and E. Nawarecki, "The application of evolution process in multi-agent world (MAW) to the prediction system", *Proc. on 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96)* 1, CD-ROM (1996).

[51] A. Byrski and R. Schaefer, "Stochastic model of evolutionary and immunological multi-agent systems: mutually exclusive actions", *Fundamenta Informaticae* 95 (2–3), 263–285 (2009).

[52] R. Schaefer, A. Byrski, and M. Smołka, "Stochastic model of evolutionary and immunological multi-agent systems: parallel execution of local actions", *Fundamenta Informaticae* 95 (2–3), 325–348 (2009).

[53] A. Byrski, R. Schaefer, M. Smołka, and C. Cotta, "Asymptotic analysis of computational multi-agent systems", *Proc. 11th Int. Conf. on Parallel Problem Solving from Nature – PPSN XI* 68, 475–484 (2011).

[54] R. Schaefer, A. Byrski, and M. Smołka, "Island model as markov dynamic system", *Int. J. Applied Mathematics & Computer Science* 22 (4), 971–984 (2012).

[55] R. Schaefer, A. Byrski, J. Kołodziej, and M. Smołka, "An agent-based model of hierarchic genetic search", *Computers & Mathematics with Applications (CAMWA)* 64 (12), 3763–3776 (2012).

[56] N.R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development", *J. Autonomous Agents and Multi-Agent Systems* 1 (1), 7–38 (1998).

[57] N.R. Jennings and M.J. Wooldridge, "Software agents", *IEE Review* 1, 17–20 (1996).

[58] E. Cantú-Paz, "A summary of research on parallel genetic algorithms", *Report* 95007, CD-ROM (1995).

[59] M. Kisiel-Dorohinicki, "Agent-oriented model of simulated evolution", in *Theory and Practice of Informatics*, vol. 2540, Springer-Verlag, Berlin, 2002.

[60] A.S. Rao and M.P. Georgeff, "Bdi agents: from theory to practice", *Proc. First Int. Conf. on Multi-Agent Systems* 1, 312–319 (1995).

[61] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I", *Monatsheft für Math. und Physik* 38, 173–198 (1931).

[62] Z. Michalewicz, *Genetic Algorithms Plus Data Structures Equals Evolution Programs*, Springer-Verlag New York, 1994.

[63] P. Billingsley, *Probability and Measure*, John Wiley and Sons, London, 1987.

[64] M. Iosifescu, *Finite Markov Processes and Their Applications* John Wiley and Sons, London, 1980.

[65] J. Digalakis and K. Margaritis, "An experimental study of benchmarking functions for evolutionary algorithms", *Int. J. Computer Mathematics* 79 (4), 403–416 2002.