Procedia Computer Science

Volume 80, 2016, Pages 845–854

ICCS 2016. The International Conference on Computational Science

Efficient Memetic Continuous Optimization in Agent-based Computing

Wojciech Korczynski[1], Aleksander Byrski[1], and Marek Kisiel-Dorohinicki[1]

AGH University of Science and Technology, Department of Computer Science, Krakow, Poland
{wojciech.korczynski,olekb,doroh}@agh.edu.pl

Abstract

This paper deals with a concept of memetic search in agent-based evolutionary computation. In the presented approach, local search is applied during mutation of an agent. Using memetic algorithms causes increased demand on the computing power as the number of fitness function calls increases, therefore careful planning of the fitness computing (through the proposed local search mechanism based on caching parts of the fitness function) leads to significant lowering of this demand. Moreover, applying local search with care, can lead to gradual improvement of the whole population. In the paper the results obtained for selected high-dimensional (5000 dimensions) benchmark functions are presented. Results obtained by the evolutionary and memetic multi-agent systems are compared with classic evolutionary algorithm.

Keywords: evolutionary algorithms, Evolutionary Multi-Agent System, memetic algorithms

1 Introduction

Some problems are very difficult for common optimization methods as their search space is too big or too complex to be explored efficiently. Such 'black-box' problems [9] may be solved using a general-purpose algorithms, e.g. meta-heuristics, which provide good enough solutions in reasonable time, taking into consideration little information from the problem domain.

For a long time meta-heuristics were not fully understood, until Davis [6] and Moscato [18] conducted successful experimental research and managed to discover the need for adjusting the solver to the problem characteristics. These practical observations were backed up in the so-called 'no-free-lunch' theorem [12, 23], according to which it is not possible to find a meta-heuristic method that will be an ultimate solution for all problems, no matter how excellent it works for a certain problem. Therefore, it is still necessary to look for novel meta-heuristics, adjusted to given problems and often inspired by the various domains of life, such as biology, evolution or genetics.

In 1996 Krzysztof Cetnarowicz [4] introduced a notion of agency to evolutionary algorithms and proposed Evolutionary Multi-Agent System (EMAS), an effective implementation of distributed problem solving. In agent-based systems the main task is decomposed into sub-tasks

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.374

entrusted to agents—intelligent objects which are able to interact one with another, as well as with the environment, and make decisions autonomously. In numerous researches EMAS proved to be an efficient method for solving different problems—classic benchmarks [1], inverse problems [24] and other optimization tasks [7, 8]. Moreover, many modifications and extensions of EMAS have been proposed. Compared to classic evolutionary algorithm, EMAS provides satisfactory results in less computation time, requiring less evaluations.

This paper concerns the idea of hybridization of EMAS with local search algorithms, inspired by the meme theory. Memetic algorithms (MAs) [19, 16, 20] join ideas from the popular metaheuristics and blend together local search with population-based search engine. MAs, initially popularized e.g. by Radcliffe and Surry [21], were proven to provide remarkable success [13]. However, memetic algorithms are very computationally demanding, therefore following [11] of caching partially the fitness results, a similar method is followed in continuous optimization, in order to efficiently realize the local search in the solution space.

Section 2 describes EMAS, along with its main assumptions, in detail. In Section 3 EMAS hybridization with local search memetic algorithms is introduced. Preliminary experiments and their results are presented in Section 4. Section 5 concludes this paper and discusses possible future work.

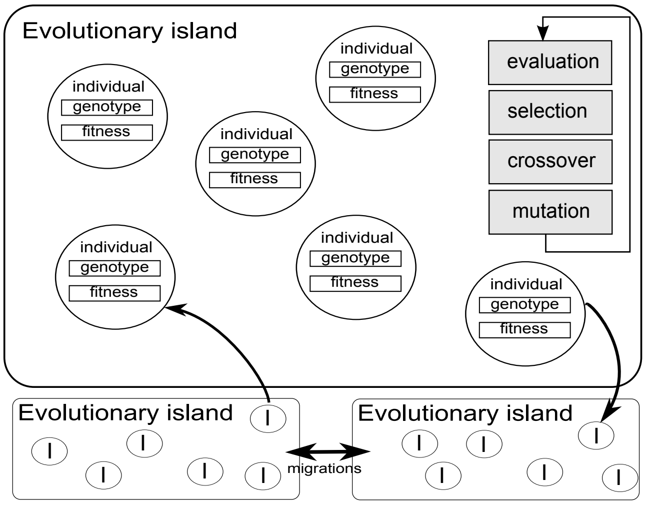
2 From evolutionary algorithms to Evolutionary Multi-Agent Systems

Evolutionary algorithms [17] belong to population-based metaheuristics. In the most popular variant called a genetic algorithm, solutions are encoded into genotypes owned by individuals. These individuals form populations—groups of potential solutions, which are evaluated based on the fitness function. Poor solutions are eliminated in the process of selection and the remaining ones create a mating pool. Subsequent population is created based on that mating pool, using variation operators, such as crossover or mutation. The whole process continues until some stop condition is reached (e.g.: predefined number of iterations or reaching good enough solution).

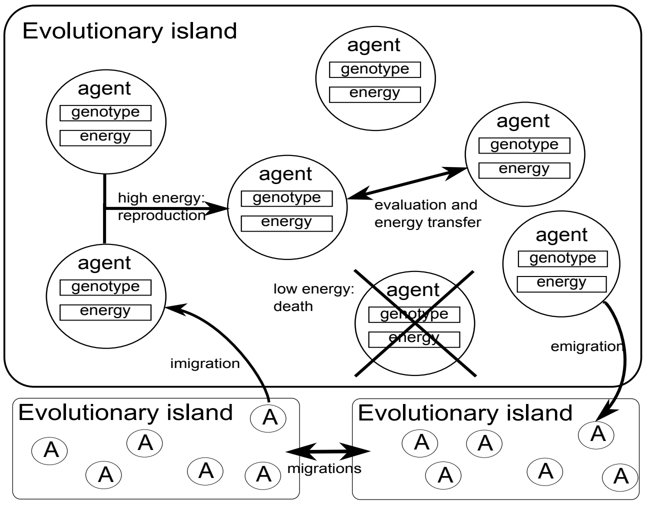
The key issue in practical applications of evolutionary algorithms is the diversity of solutions in the population. To preserve this diversity during the search several techniques may be applied. Following the idea of allopatric speciation, individuals may be distributed among evolutionary islands, which allows for parallelizing the algorithm [3]. Figure 1(a) schematically illustrates parallel evolutionary algorithm (PEA) used as a reference in this paper.

Agency brings to the world of evolutionary metaheuristics decentralization of selection and autonomy of the individuals. Thus the natural process of evolution is mimicked better, and in this way the author of EMAS Krzysztof Cetnarowicz [4], and his followers [2, 7] tend to introduce a new quality into metaheuristics, achieving effective results consisting e.g. in decreasing the computation cost computed as the number of fitness function calls [1].

In EMAS phenomena of inheritance and selection—the main components of evolutionary processes—are modelled via agent actions of death and reproduction (Figure 1(b)). As in the case of classic evolutionary algorithms, inheritance is accomplished by an appropriate definition of reproduction. Core properties of the agent are encoded in its genotype and inherited from its parent(s) with the use of variation operators (mutation and recombination). Moreover, an agent may possess some knowledge acquired during its life, which is not inherited. Both inherited and acquired information (phenotype) determines the behaviour of an agent. It is noteworthy that it is easy to add mechanisms of diversity enhancement, such as allopatric speciation (cf. [3]) to EMAS. It consists in introducing population decomposition and a new action of the agent based on moving from one evolutionary island to another (migration).



(a) PEA



(b) EMAS

Figure 1: Schematic presentation of Parallel Evolutionary Algorithm (PEA) and Evolutionary Multi-Agent System (EMAS)

3 Efficient memetic search in EMAS

Evolutionary algorithms may be enhanced by hybridization with local search memetic algorithms. Local improvements are applied within evolutionary cycle and they may happen in the course of evaluation (according to the Baldwin effect [14]) or mutation (according to the

Lamarckian model [10]). To deal with the problem of efficiency, memetic operators can be used in steady-state evolutionary algorithm [22] and similar ones. But agent-based setting seems to be the most natural as in other metaheuristics, no parallel ontogenesis is observed. Recent our works tried to apply memetic algorithms to EMAS and presented promising results at the expense of efficiency (see, e.g. [2]), since many evaluation events were required. Below a more profound study of the hybridization of EMAS with memetic algorithms is presented.

In this paper Evolutionary Multi-Agent System with memetic algorithms (MemEMAS) is analysed and compared with the memetic version of Parallel Evolutionary Algorithm (MemPEA) and the classic variants of both algorithms. Schematic presentation of MemEMAS and MemPEA are given in Figures 2(a) and 2(b). Local search results in creating different solutions, each of which is evaluated and the best one is selected to replace individual's genotype. Each memetization event (local search realized during mutation) consisted of 10 cycles. At each cycle one random gene was changed. If this change resulted in individual's fitness improvement, it was performed repeatedly as long as fitness value was improved. Moreover, gene change was adapted to the increase of quality of the results obtained on an individual's evolutionary island—if the best fitness on an island does not improve sufficiently, gene change is greater.

As it was stated in Section 3, during each memetization event one gene was changed and fitness value had to be recalculated. Therefore, such local search algorithm is so computationally exhaustive, that without any efficient modification, the number of evolutionary steps performed in a time unit would be disproportionately lower, compared to classic algorithms.

In the presented research the evaluation operator was implemented in a similar way that was presented in [11], so the fitness computation, especially when running a local search, is the most efficient. This amendment bases on the division of fitness function into separable parts, each of them corresponding to the particular gene:

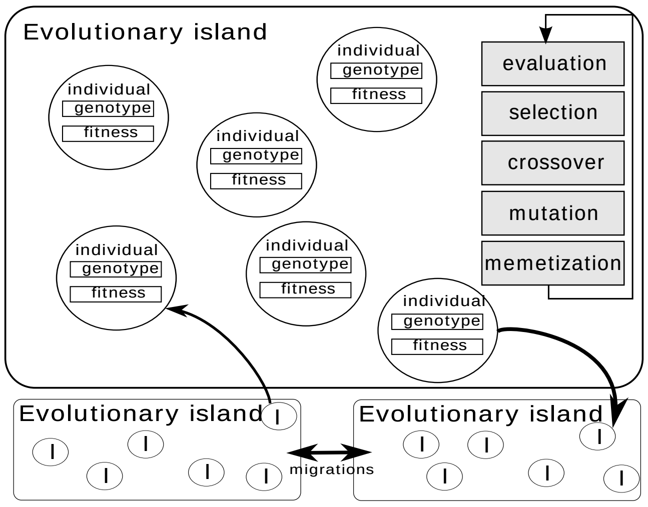
$$f(x) = f_1(x_1) \diamond_1 f_2(x_2) \diamond_2 \ldots \diamond_{n-1} f_n(x_n) \tag{1}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$, $f_i(x_i) : \mathbb{R}^n \to \mathbb{R}, i \in [1, n], i \in \mathbb{N}$, $\diamond_j, j \in [1, n-1]$ is any mathematical operator like sum, subtraction, division or product. Assuming such fitness function, one can easily compute the $n-1$ values of the partial functions $f_j(x_j), j \in [1, n] \wedge j \neq k, j, k \in \mathbb{N}$, leaving the value of $f_k(x_k)$ to be computed once per each mutation, when the single-point mutation is considered. The gain from "caching" the values of the other partial fitness function seems to be inevitable, and it will be confirmed later in the experimental section. This approach may be of course easily extended for other mutation operators (e.g. two-point mutation and others).

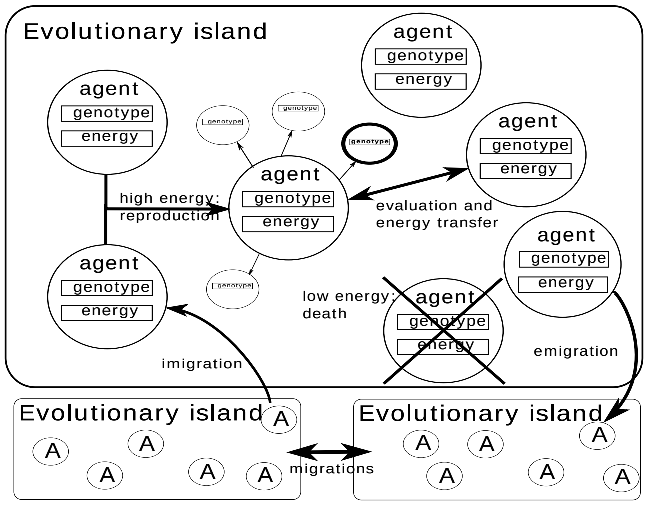
This mechanism can be the most effortlessly applied if a fitness function is separable. However, there are means to implement a similar mechanism in the case of non-separable functions (cf. e.g., discrete optimization with efficient memetization operator presented in [11]), and extending the number of potential benchmarks is one of our future goals.

4 Experimental results

In this Section experiments comparing EMAS, PEA and their memetic variants (MemEMAS and MemPEA) are discussed. These experiments were performed with the use of PyAgE computing and simulation platform [15] and two multidimensional hard continuous benchmark problems were solved: Rastrigin and Ackley functions. These functions are described by Equations 2 and 3, respectively. Global optimum for both of them equals $f(x) = 0.0$. Figures 3(a)



(a) MemPEA



(b) MemEMAS

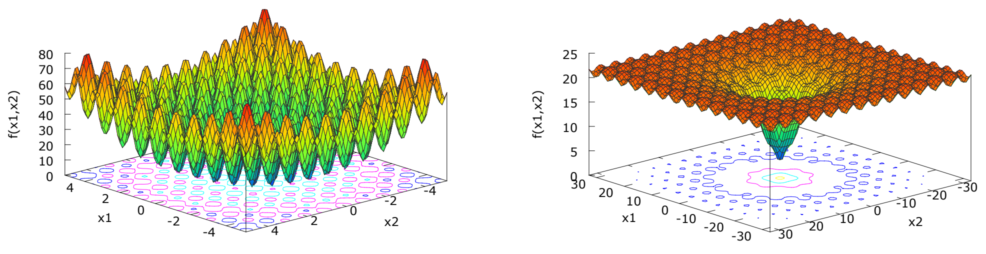
Figure 2: Schematic presentation of PEA and EMAS algorithms enhanced with memetization

and 3(b) illustrate these functions in two dimensions.

$$f(x) = An + \sum_{i=1}^{n}[x_i^2 - A\cos(2\pi x_i)] \tag{2}$$

$$f(x) = -a\exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + a + \exp(1) \tag{3}$$

The tested functions were evaluated in 5000 dimensions on the hypercubes $x_i \in [-5.12, 5.12]$



(a) Two-dimensional Rastrigin function (b) Two-dimensional Ackley function

Figure 3: Rastrigin and Ackley illustration

for Rastrigin and $x_i \in [-32.768, 32.768]$ for Ackley. The constants were assigned the following values: Rastrigin: $A = 10$; Ackley: $a = 20$, $b = 0.2$, $c = 2\pi$. Other configuration parameters are included in Table 1.

Parameter	EMAS	PEA
Mutation	Uniform, of one randomly chosen gene	
Crossover	Single point	
Speciation	Allopatric	
Environment	Torus-shaped, size 10 x 10	
Number of evolutionary islands	3, fully connected	
Numbers of individuals on each island	50	
Agent/individual migration probability	0.05	
Initial energy	100	-
Energy transferred from loser to winner	5	-
Agent's death energy level	0	-
Minimal energy required to reproduce	120	-
Minimal energy required to migrate	130	-
Selection	-	tournament (tournament size: 30)

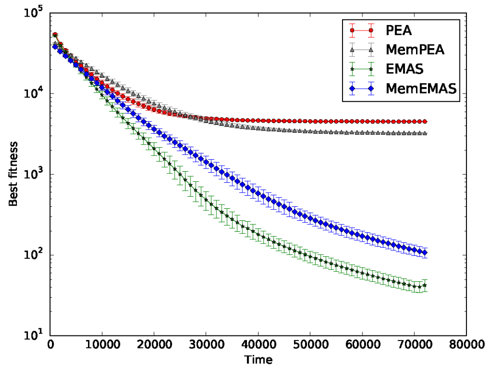
Table 1: Experiments configuration parameters

All experiments lasted exactly 72000 seconds, were repeated 11 times (nature of the problem and the implementation issues made the whole system very computational power demanding, therefore the number of repetitions was decreased) and standard deviation was computed.

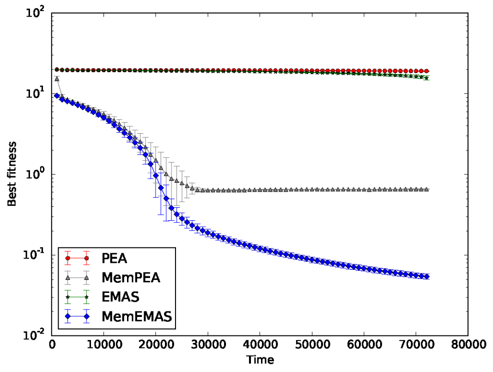
Obtained results have been illustrated by plots in Figures 4(a) and 4(b). Moreover, Table 2 contains precise results at the end of experiments. As one can see, EMAS outperformed PEA for both of the functions (visibly reaching better solutions in the same time), that acknowledges the conclusions of preceding research (e.g. [2]). Focusing on influence of memetization on results obtained by EMAS, two different cases are noticeable:

- Rastrigin function: Memetization allowed to reach a better solution in shorter time (what can be clearly observed in Figure 4(c)), but then it was outperformed by classic EMAS, because of the computational overhead caused by vast amount of fitness evaluation events needed by memetic algorithm (exact data are provided in Table 2).

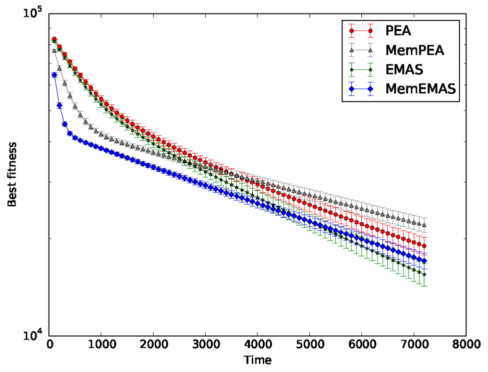
- Ackley function: Local search enabled to further exploit the vicinity of the global optimum, while the classic EMAS clearly stuck in some local optimum.



(a) Rastrigin best fitness



(b) Ackley best fitness



(c) Rastrigin best fitness in the first 7200 seconds

Figure 4: Obtained results

The best fitnesses obtained for all the experiments may be seen in Table 2. It is easy to see, that in case of Rastrigin function the final results provided by EMAS and MemEMAS are not far from the global optimum (mind the high dimensionality of the problem). In case of Ackley function, the optimum was located very accurately by MemEMAS.

The information about generations (for PEA and MemPEA) given in Table 2 clearly shows that introduction of local search slows down the whole search process. E.g. in case of PEA, running the same algorithm with local search enabled makes it run over 60000 less generations (although the result is clearly better). However, one must remember to properly tune the number and frequency of local search events, not to hamper the whole search process.

At the same time the information about the evaluation events shown in Table 2 clearly shows that the implemented, efficient local search mechanism, allows for conducting far more

	Result	St. Dev.	Generations/Agent steps	Evaluation events
			Rastrigin	
PEA	4533.48	27.72	157910.85	23686776.92
MemPEA	3252.40	43.66	93055.62	70682033.00
EMAS	40.28	6.74	3551702.31	13133439.08
MemEMAS	108.45	15.31	2141999.69	169803084.46
			Ackley	
PEA	19.07	0.08	250485.00	37572900.00
MemPEA	0.65	0.01	79066.50	60105471.50
EMAS	15.98	0.89	4105150.20	16112953.93
MemEMAS	0.05	0.01	1693190.79	135641127.21

Table 2: Experiments results in the 72000th second

fitness evaluations as in the classic cases. See e.g. the number of evaluation events for EMAS and MemEMAS in the case of Rastrigin function: local search version executes over $156 \cdot 10^6$ more evaluations as it is in the case of the classic one. One must however remember that these are local fitness evaluations—they may improve the exploitation, but can be fairly useless from the exploration point of view. Again proper tuning of the system's parameters is necessary. Anyway, here the advantages of the efficient memetic operator can be clearly seen.

These different outcomes of the experiments demonstrate how memetization may influence on the obtained results. They can be also explained by the aforementioned *no free lunch theorem*—we cannot provide an ultimate solution for all kinds of problems.

Besides, it is noteworthy that all of the experiments are repeatable as the value of standard deviation is low.

5 Conclusions

The results presented in this paper show that local search realized in EMAS may provide significant improvements in shorter time and should be further developed. Moreover, the efficient implementation of the local search operator allowed in particular cases to increase the number of fitness function evaluations over ten times more than it was realized in the classic version of the tested algorithms. Thus the possibilities of exploration and exploitation of the high-dimensional search spaces become real—and one cannot do there anything without such dedicated mechanisms, because of so called *curse of dimensionality*[5] present there.

The optimized benchmark functions examined in this work were separable, and in fact, such problems are the easiest to tackle with the presented efficient local search mechanism—however implementation of such operator for non-separable problems is also possible (although it will rather not be as efficient as in the case of separable ones). Indeed, one of our future goals is to examine also non-separable problems (starting from e.g. Rosenbrock benchmark).

Of course the detailed examination of parametrization of the system (when and to what extent such local searches should be applied by the agents) is also envisaged, as without such experiments the rationale for using the proposed efficient local search method might not be fully justified.

Acknowledgment

The research presented in this paper was partially supported by the AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications Dean's Grant.

References

[1] Aleksander Byrski. Tuning of agent-based computing. *Computer Science*, 14(3):491, 2013.

[2] Aleksander Byrski, Wojciech Korczynski, and Marek Kisiel-Dorohinicki. Memetic multi-agent computing in difficult continuous optimisation. In *KES-AMSTA*, pages 181–190, 2013.

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

[4] K. Cetnarowicz, M. Kisiel-Dorohinicki, and E. Nawarecki. The application of evolution process in multi-agent world (MAW) to the prediction system. In M. Tokoro, editor, *Proc. of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96).* AAAI Press, 1996.

[5] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, September 2001.

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

[7] R. Drezewski and L. Siwik. Multi-objective optimization technique based on co-evolutionary interactions in multi-agent system. In M. Giacobini, et al., editor, *Applications of Evolutinary Computing, EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Valencia, Spain, April11-13, 2007, Proceedings*, volume 4448 of *LNCS*, pages 179–188, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] R. Drezewski and L. Siwik. Co-evolutionary multi-agent system for portfolio optimization. In A. Brabazon and M. O'Neill, editors, *Natural Computing in Computational Finance*, volume 1, pages 271–299. Springer-Verlag, Berlin, Heidelberg, 2008.

[9] Stefan Droste, Thomas Jansen, and Ingo Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39:525–544, 2006.

[10] N. Eldridge and S.J. Gould. Punctuated equilibria: An alternative to phyletic gradualism. In T.J.M Schopf, editor, *Models in Paleobiology.* Freeman, Cooper and Co., 1972.

[11] José E Gallardo, Carlos Cotta, and Antonio J Fernández. Finding low autocorrelation binary sequences with memetic algorithms. *Applied Soft Computing*, 9(4):1252–1262, 2009.

[12] W.E. Hart and R.K. Belew. Optimizing an arbitrary function is hard for the genetic algorithm. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 190–195, San Mateo CA, 1991. Morgan Kaufmann.

[13] W.E. Hart, N. Krasnogor, and J.E. Smith. Memetic evolutionary algorithms. In *Recent advances in memetic algorithms*, volume 166 of *Studies in Fuzziness and Soft Computing*, pages 3–27. Springer-Verlag, 2005.

[14] G. Hinton and S. Nolan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[15] Maciej Kaziród, Wojciech Korczynski, and Aleksander Byrski. Agent-oriented computing platform in python. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 365–372. IEEE, 2014.

[16] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[17] Z. Michalewicz. *Genetic Algorithms Plus Data Structures Equals Evolution Programs.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1994.

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

[19] P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, 1999.

[20] P. Moscato and C. Cotta. A modern introduction to memetic algorithms. In M. Gendrau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research and Management Science*, pages 141–183. Springer, 2 edition, 2010.

[21] N.J. Radcliffe and P.D. Surry. Formal Memetic Algorithms. In T. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, volume 865 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1994.

[22] Gilbert Syswerda. A study of reproduction in generational and steady state genetic algorithms. *Foundations of genetic algorithms*, 2:94–101, 1991.

[23] D. Wolpert and W. Macready. No free lunch theorems for search. Technical Report SFI-TR-02-010, Santa Fe Institute, 1995.

[24] K. Wróbel, P. Torba, M. Paszyński, and A. Byrski. Evolutionary multi-agent computing in inverse problems. *Computer Science (accepted for printing)*, 2013.