Efficiency of Memetic and Evolutionary Computing in Combinatorial Optimisation

Magdalena Kolybacz, Michał Kowol, Łukasz Leśniak
Aleksander Byrski, Marek Kisiel-Dorohinicki
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
magda.kolybacz@gmail.com, michal.kowol@gmail.com, luke.lesniak@gmail.com
olekb@agh.edu.pl, doroh@agh.edu.pl

Abstract— **Difficult search and optimisation problems call for complex techniques for solving them. In particular, in cases when fitness function is costly, applying solutions, such as agent-based computing systems may be fruitful. This approach may yield even better results, in the case of memetic computing, as these algorithms tend to significantly increase the number of fitness function calls, because of their nature. This paper may be treated as a milestone in preparing to tackle combinatorial optimisation problems with memetic approaches in agent-based systems. After discussing the selected problems and details of population-oriented meta-heuristics to solve them, experimental results (with stress put on efficiency) are presented. Then details of applying EMAS-class systems are given and, in the end, preliminary EMAS results obtained for combinatorial optimisation are shown and the work is concluded.**

I. Introduction

Difficult optimization problems (in other words the ones, which domains are very hard or even impossible to be described and explored) cannot be solved using using conventional mathematical apparatuses, like most combinatorial optimization problems [1]. Such problems may be treated as "black-box" problems [2] and may be solved only using general-purpose algorithms, such as meta-heuristics, taking into consideration little, if any information from a problem domain, to devise a solution. As a drawback of such approaches, the one can be mentioned, that the produced results are sub-optimal, and it is impossible to check, whether (and when) they are (if at all) close to the global optimum.

One of the most popular general-purpose meta-heuristics are evolutionary algorithms [3]. Because of their nature, they are very easy to hybridise with other techniques, especially local-search ones constituting so-called memetic systems [4]. Such computing systems have already proven to be effective in solving different difficult combinatorial optimisation problems (see, e.g.,[5], [6]).

A particularly interesting general-purpose approach to optimisation is an evolutionary multi-agent system (EMAS). This technique (along with its variations) has already proven to be an effective tool when dealing with difficult problems in global optimization [7], multi-objective optimization [8], [9], multi-modal optimization [10], and also hybrid methods such as the optimization of neural-network architecture [11], [12]. It was also a subject of theoretical analysis [13] and gained technological support [14]. A valuable feature of EMAS, is a significantly reduced number of fitness function calls needed to obtain similar results, comparing to classical evolutionary algorithms. These results have been obtained for global-optimisation of continuous problems, moreover memetic variations of the tested algorithms have also been researched [7].

However, in the case of combinatorial optimisation, though meta-heuristics applied may be absolutely the same, as in the case of continuous one, the particular configurations, including variation operators applied and their memetic variants pose new challenging problems to the researcher. Therefore, before applying EMAS to solve combinatorial optimisation problems, it might be fruitful for further research to examine evolutionary and memetic solutions of the selected benchmarks. The main aim of this paper is thus to compare experimentally the efficiency of classical evolutionary and memetic algorithms to choose the most promising solutions and reuse them in the preliminary experiments with evolutionary multi-agent systems.

The paper begins with a discussion of three selected combinatorial problems, namely Low Autocorrelation Binary Sequence (LABS), Golomb ruler and Job Shop. After a short review of population-oriented metaheuristics, experimental results (with stress put on efficiency) are presented. In the end, details of applying EMAS-class systems are given together with preliminary results obtained for combinatorial optimisation and the work is concluded.

II. Population-based approaches to combinatorial optimisation

Low Autocorrelation Binary Sequence (LABS) problem was formulated in 1960s by Physics community and has various applications in physics, chemistry and telecommunication. LABS is a very hard optimization problem with simple formulation: find a binary sequences $S = \{s_0, s_1, \ldots, s_{L-1}\}$ with length L where $s_i \in \{-1, 1\}$ which minimizes energy function $E(S)$:

$$C_k(S) = \sum_{i=0}^{L-k-1} s_i s_{i+k} \qquad E(S) = \sum_{i=1}^{n-1} C_k^2(S)$$

The problem has an interesting property: it is symmetric. The energy function $E(S)$ stays the same when sequence is reversed or each element of S is multiplied by -1. The search space for the problem with length L has size 2^L and energy of sequence can be computed in time $O(L^2)$.

LABS problem has no constrains, so S can be represented as a list of binary values, and standard operators for mutation and recombination can be used [15]. In the problem all bits are correlated – there are no blocks of good solutions, so

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

uniform crossover is recommended since it provides a new promising start point for the algorithm.

A few local search techniques can be used in memetic algorithms to solve LABS problems: steepest decent local search (SDLS), tabu search [15], [16], and more. The best results are reported to be obtained using some variations of Tabu Search [17].

Job-shop scheduling problem (JSSP) is a well-known, combinatorial optimisation problem that can be defined as follows: given a set of m machines and n jobs (each job is a sequence of m operations), the aim is to find a schedule of operations that minimizes the finishing time of the last operation in the schedule. Each operation of a job has to be processed on specified machine. Operation processing takes time period of a given length and no preemption is allowed. Each machine can process at most one job at a time and each job can be processed on at most one machine at a time. In the paper we focus on the Open-shop scheduling problem (OSSP) which is a variation of JSSP but — unlike in JSSP — the ordering of operations within a job is not defined and may be changed.

To be able to apply memetic or genetic algorithms to OSSP, the solution has to be properly encoded. Various possible representations are widely discussed in literature [18], [19], [20]. In the *permutation with repetitions* representation a chromosome that encodes a solution for n jobs and m machines consists of numbers from 1 to n (each number occurs exactly m times). Each number represents operations from single job—first occurence of a number in chromosome represents an operation on first machine etc. In *permutation* representation (without repetitions) a chromosome consists of numbers from 1 to $n \times m$. Numbers 1 to m represent operations from the first job, numbers $1 \times m + 1$ to $2 \times m$ represent operations from the second job, etc.

The permutational characteristics of the chromosome requires specific mutation and crossing-over operators [20]. For the above-described representation, LOX for crossing-over and SWAP for mutation may be used. As local optimization method, variations of tabu search are commonly used [21].

In Open-shop problem, specific variation of SWAP can be applied to permutation representation (without repetitions). The variation uses domain knowledge — two operations can be interchanged only if they are assigned to the same machine. We use the method as local optimization in one of our memetic algorithms.

The term **Golomb Ruler** is derived from the work by Professor of Mathematics Solomon W. Golomb [22]. Formally Golomb ruler is an ordered sequence of n distinct, nonnegative integers $\langle a_1, a_2, a_3, \ldots, a_n \rangle$ where $a_i < a_{i+1}$ such that all distances $a_j - a_i$ $(1 \leq i < j \leq n)$ are distinct. By convention $a_1 = 0$ and a_n is the length of the ruler. Golomb rulers have applications in many diverse fields such as radio communications, X-ray crystallography, coding theory and radio astronomy.

A construction of valid n-mark ruler is relatively simple, therefore researches are interested in discovering ruler with minimum length. Figure 1 presents an example of valid 4-mark ruler and all discrete lengths, that it measures.

Finding optimal Golomb ruler is challenging for optimization methods. The most important issue is how to properly encode Golomb ruler into chromosome, because this will reflect the difficulty of good genetic operators design. Commonly 2 representations are used (the conversion between them is simple and has low cost):

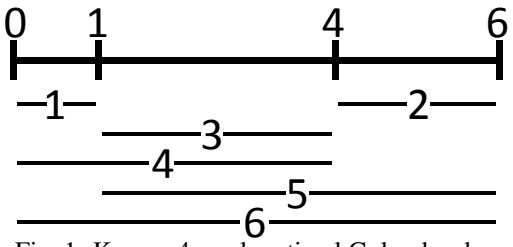

Fig. 1: Known 4-mark optimal Golomb ruler.

- indirect representation – the distance between adjacent segments is stored as array of integers (e.g. ruler $\langle 0, 1, 4, 6 \rangle$ is encoded into $\langle 1, 3, 2 \rangle$ [23];
- direct representation – is consisted of an array of integers corresponding to the marks of the ruler [24].

Other approaches can be found in the literature e.g. with random keys [25] or binary representation [26], [27].

Second important issue is the decision how to handle constraints and properly design genetic operators. We decided to choose static penalty approach, tournament selection with tourney size of 2 and standard variation operators: segment's length mutation [23] and one-point crossover, both working on indirect representation [28]. Other kinds of genetic operators used include shift mutation [27] or two-point crossover [23], [27].

Population-based algorithms can be enhanced by a local-search method, commonly discussed is the tabu-search in this case [28], [29], [30].

Considering different approaches to find optimal Golomb ruler, best results are obtained in algorithms with binary representation and chromosome repair methods [27]. The algorithms which allow to evolve valid and invalid solutions (without penalty methods) obtain the worst results [23]. However memetic approaches' results are situated in the middle [28].

III. Memetic and evolutionary metaheuristics

Population-based meta-heuristics, particularly evolutionary algorithms [3] may be classified as universal optimisation techniques. Their work is based on the following strategy: instead of directly solving the given problem, the problem undergoes encoding using a predefined way (encoded problem is called a genotype). Group of genotypes create population (that is randomly initialised). Population constructed in this way contains potential sub-optimal solutions of the given problem. Now, using a dedicated evaluation function (so called fitness), selection process is conducted (in this way the mating pool is created) and the subsequent population is created by picking the parents from the mating pool and generating offspring based on their genotypes with use of predefined variation operators (such as crossover and mutation). The process continues until a predefined stopping condition is reached (e.g., maximum number of generations, lack of changes in the best solution found so far). This kind of search has some drawbacks (as possibility of

premature convergence), thus several techniques, as multi-deme approaches [31] are applied.

Evolutionary algorithms may be easily hybridised with local-search methods constituting so called memetic algorithms [4]. In these algorithms, two kinds of search-enhancements can be implemented:

- Baldwinian local search is based on Baldwin theory stating that predispositions may be inherited during reproduction [32]. It is implemented usually as calling of local search in the course of evaluation process. The evaluated genotype receives the fitness function value computed for one of its possible descendants (being an outcome of local-search started from this individual).
- Lamarckian local search is based on Lamarck theory stating that characteristics of individuals acquired in the course of life may be inherited by their descendants [33]. It is implemented usually as specific mutation operator. The search for a mutated individual is based not only on stochastic one-time sampling from the solution space, instead a local-search is started from the genotype and the solution reached becomes the result of this process.

Though both theories turned-out to be false, the meta-heuristics based on them are effective in many problems (see, e.g., [34], [35]).

IV. Experimental results

The experimental results were obtained using extensible distributed computing environment AgE[1] that is being developed as an open-source project by the Intelligent Information Systems Group of AGH-UST. AgE provides a platform for the development and execution of distributed agent-based applications in mainly simulation and computational tasks [14], [36].

For each of the described problems both Lamarckian model of memetic (MA) and evolutionary (EA) algorithms were applied in two configurations: a relatively easy and difficult one. Each experiment was repeated 30 times with the same parameters (tuned for the considered case) and mean values with standard deviations of the best-so-far solution quality obtained in fixed time points of the run (measured in seconds) were calculated. The results were scaled to $[0; 1]$ interval, according to maximal and optimal values for each case (minimisation assumed).

LABS

Configuration:

- common parameters—population size: 50, tournament selection, uniform crossover, bit-flip mutation, $2/L$ elements mutated, total execution time: 300 seconds,
- evolutionary algorithm—mutation probability: 0.75, recombination probability: 0.5,
- memetic algorithm—mutation probability: 0.75, recombination probability: 0.5, local search: RMHC [3] (50 steps) and SDLS.

For MAs, a simple algorithm to recompute efficiently fitness in solution's neighbourhood was used [15].

The results obtained for the easy problem ($L = 30$) are shown in Figure 2 and in Table I. Both algorithms were able to found an optimal solution. MA_{RMHC} and MA_{SDLS} reached it in less than 2 seconds. EA needed 300 seconds to obtain the same result. What is more, EA found the desired result in 40% of tested cases only.

TABLE I: Results obtained for LABS ($L = 30$)

T	EA	MA_{RMHC}	MA_{SDLS}
60	0.20 ± 0.10	0.0	0.0
180	0.13 ± 0.10	0.0	0.0
300	0.10 ± 0.09	0.0	0.0

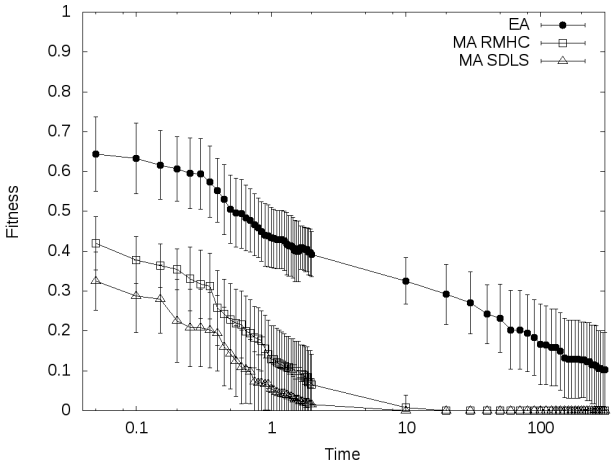

Fig. 2: Simple ($L = 30$) LABS problem in logarithmic scale

The results obtained for harder problem ($L = 50$) are presented in Figure 3 and in Table II. Only MA_{SDLS} was able to reach an optimum after 300 seconds (70% of successful computations took place). Note that EA, MA_{RMHC} and MA_{SDLS}'s error bars in Figure 3 are disjointed—when more sophisticated local search method was used, the results yielded to be significantly better. It is to note that both MAs produced their results in about 10 times less iterations then EA in the same time.

TABLE II: Results obtained for LABS ($L = 50$)

T	EA	MA_{RMHC}	MA_{SDLS}
60	0.52 ± 0.03	0.28 ± 0.05	0.10 ± 0.06
180	0.50 ± 0.02	0.23 ± 0.05	0.04 ± 0.05
300	0.47 ± 0.03	0.21 ± 0.05	0.02 ± 0.04

There has been a try to cache fitness, unfortunately in both EA and MA hit rate of cache was low and there was no improvement in efficiency of algorithms. Hit rate in EA was about 13%, in MA_{RMHC} 26% and in MA_{SDLS} 3%.

There was a Tabu Search implementation, but it turned out to be better than RMHC but worse than SDLS.

Job Shop

Configuration:

- common parameters:
  - population size: 50,
  - tournament selection,
  - swap mutation,
  - LOX recombinantion,
  - first memetic algorithm (MA): local search based on RMHC,

[1] http://age.iisg.agh.edu.pl

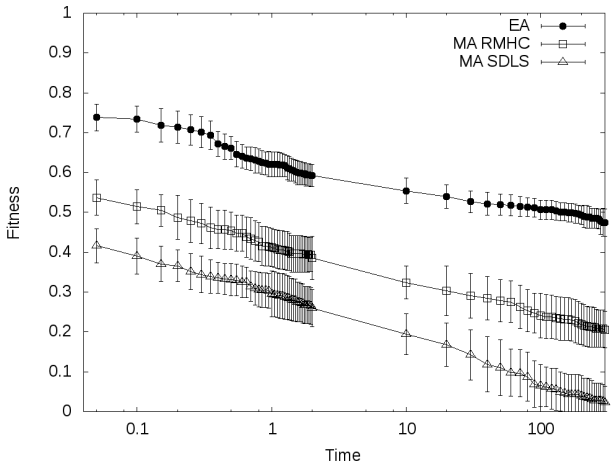

Fig. 3: The hard ($L = 50$) LABS problem in logarithmic scale

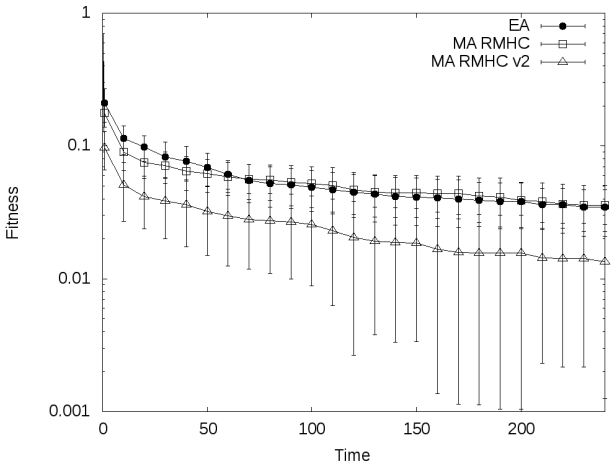

Fig. 4: Comparison of EA and MA for 7x7 OpenShop problem

– second memetic algorithm (MAv2): local search based on RMHC with swaping operations only on single machine mutation

- easy problem (7 jobs and 7 machines):
  – mutation probability: 0.1,
  – recombinantion probability: 0.3 (EA), 0.33 (MA) and 0.35(MAv2)
  – local search: 10 steps,
  – total execution time: 240 seconds,
- difficult problem (15 jobs and 15 machines):
  – mutation probability: 0.03 (EA) and 0.1 (MA and MAv2),
  – recombinantion probability: 0.28 (EA), 0.3 (MA), 0.35 (MAv2)
  – local search: 15 steps,
  – total execution time: 420 seconds.

Both problems were taken from Taillard benchmarks for OSSP [37].

Permutation with repetiton representation was used in evolutionary algorithm and first memetic algorithm. For the second memetic algorithm with single machine swap mutation we used normal permutation representation.

Results for 7×7 problem are presented in Figure 4 and in Table III. As one may notice, evolutionary algorithm and first memetic algorithm gave similar results, however in terms of generations memetic algorithm is much slower—after 60 seconds EA reached ca. 130 000 generation, while MA only 20 000.

Changing mutation in local optimalization for single machine swap (MAv2) gave noticeably better results, even though in terms of generations this algoritm is the slowest—after 60 seconds reached ca. 16 000 generation.

TABLE III: Results for 7×7 problem.

T	EA	MA	MAv2
10	0.114 ± 0.027	0.090 ± 0.025	0.051 ± 0.024
60	0.061 ± 0.016	0.059 ± 0.016	0.030 ± 0.017
240	0.035 ± 0.012	0.036 ± 0.015	0.013 ± 0.012

Results for 15x15 problem are presented in Figure 5 and in Table IV. In this case still the second memetic algorithm (MAv2) gave the best results. However the evolutionary algorithm gave better results than the first memetic algorithm (MA). Admittedly in terms of generations both memetic algorithms gave good results much faster, however one generation in memetic algorithms still takes much more time.

TABLE IV: Results for 15x15 problem.

T	EA	MA	MAv2
10	0.365 ± 0.075	0.322 ± 0.058	0.113 ± 0.029
60	0.218 ± 0.040	0.222 ± 0.053	0.058 ± 0.019
420	0.100 ± 0.030	0.132 ± 0.028	0.021 ± 0.015

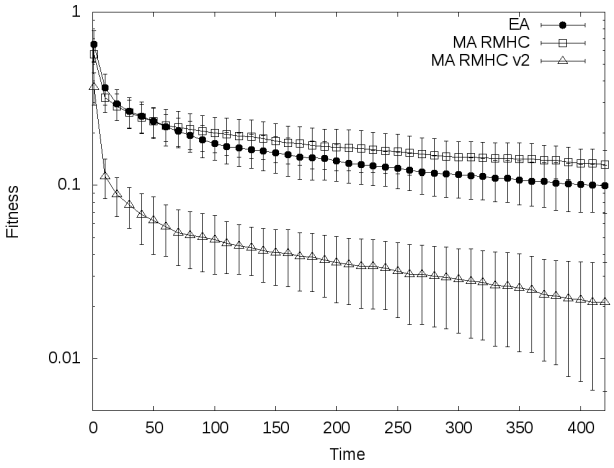

Fig. 5: Comparison of EA and MA for 15x15 OpenShop problem

Golomb Ruler

Configuration:

- common parameters—population size: 100, tournament selection, segment's length mutation with probability of 0.2, one-point crossover with probability of 0.8, total execution time: 300 seconds,
- memetic algorithm—local search based on tabu-search with 10 steps.

The results obtained for simple (7-mark ruler) problem are presented in Figure 6 and in Table V. Both algorithms have found optimal result after approximately the same

range of iterations (10^5), but memetic version completed the computation in less than 10 seconds. Evolutionary algorithm reached optimal result only 46.67% of the all tested cases, not earlier than after 300 seconds.

TABLE V: Results for finding 7-mark OGR.

T	EA result	MA result
60	0.030 ± 0.022	0.000
180	0.024 ± 0.021	0.000
300	0.019 ± 0.020	0.000

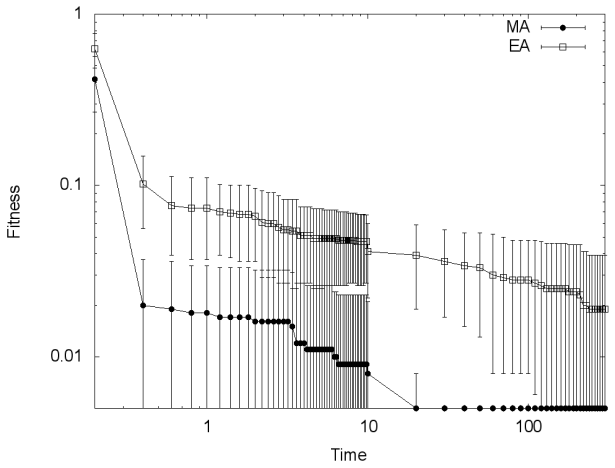

Fig. 6: Comparison of finding 7-mark optimal Golomb ruler by evolutionary (top) and memetic (bottom) algorithm. Both axis are in logarithmic scale.

The results obtained for high difficulty problem (14-mark ruler) are presented in Figure 7 and in Table VI. In this case neither of the algorithms reached optimal result, however both were stable. The memetic algorithm in every case reached better results, but it was much slower. After 300 s., it crossed 10^3 range of iterations, for evolutionary algorithm it was 10^5.

TABLE VI: Results for finding 14-mark OGR.

T	EA result	MA result
60	0.038 ± 0.007	0.022 ± 0.003
180	0.035 ± 0.006	0.020 ± 0.003
300	0.033 ± 0.007	0.018 ± 0.003

V. Preliminary results for memetic and evolutionary agent-based computing

In memetic [7] and evolutionary multi-agent systems (EMAS), agents represent solutions for a given problem, which are inherited from its parent(s) with the use of mutation and recombination. Assuming that no global knowledge is available and autonomy of the agents, selection is based on non-renewable resource, most often called *life energy*. The agents exchange energy meeting one another and comparing the quality of their solutions. At the same time a decisive factor of the agent's activity is the amount of energy it possesses—agents with high energy level are more likely to reproduce, while low energy increases the possibility of death. Each action is attempted randomly with certain probability, and it is performed only some preconditions are met

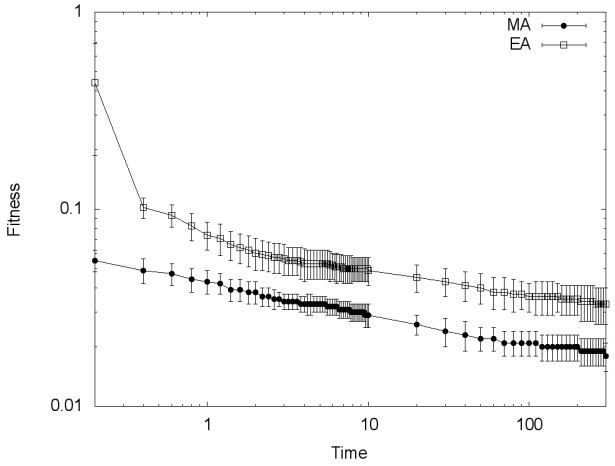

Fig. 7: Comparison of finding 14-mark optimal Golomb ruler by evolutionary (top) and memetic (bottom) algorithm. Both axis are in logarithmic scale.

(e.g. an agent may attempt to perform the action of reproduction, but it will reproduce only if its energy rises above certain level and it meets an appropriate neighbour).

In Fig. 8, preliminary results of optimisation of LABS problem (with lenght 30) were presented. The experiments were repeated 30 times. In both algorithms, the population of 50 individuals was used in the beginning. It is easy to see that in this case EMAS attains significantly worse results than EA. However, consider harder problem, namely

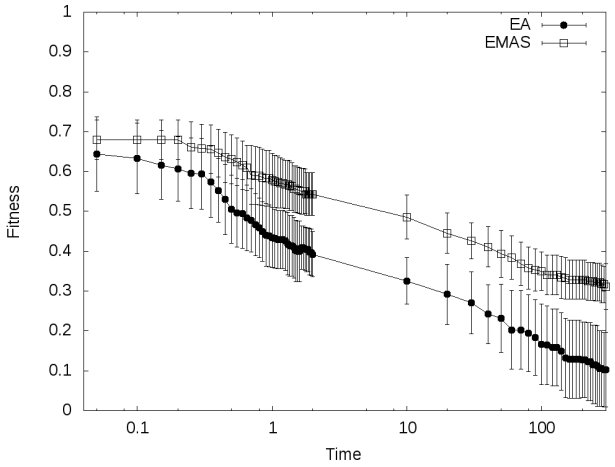

Fig. 8: EMAS and EA fitness for LABS (instance 30)

LABS instance of 50. The results of the solving of this problem with EMAS and EA were presented in Fig. 9. In this case, in the beginning of computation, the results are similar to the ones presented for the easier problem (LABS instance of 30). It is noteworthy, that later EMAS significantly prevails, confirming the well-known statement, that metaheuristics should be used as a *methods of last resort*, only for the complex problems.

Implementation of memetics in EMAS is carried out in a similar way as in classical evolutionary computing. In the case of 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

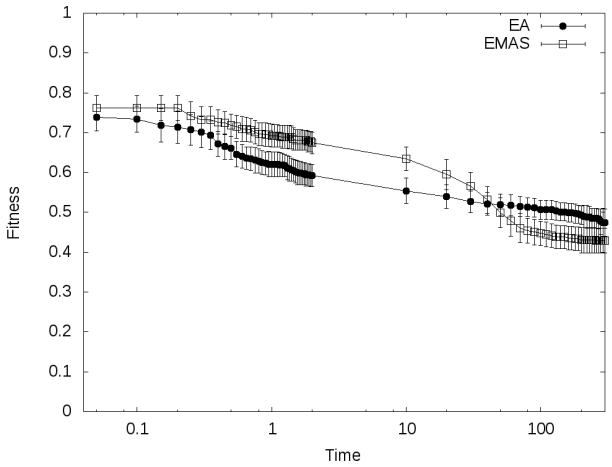

Fig. 9: EMAS and EA fitness for LABS (instance 50)

original fitness value. In Lamarckian model of 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.

In EMAS, distributed selection mechanism allows for parallel ontogeny, so in one observable moment, certain number of individuals in the population might be about to die, other group could be almost ready for reproduction etc. In EA total synchronisation is maintained, so the whole population of individuals is processed at one time. Therefore, potential possibilities of finding new promising solutions seem to be better in the case of EA, yet the EMAS turns out to be better in this comparison.

To sum up, in Fig. 10, the number of newly produced individuals in each step for EMAS and its modifications is shown. The presented results have been gathered for 30 times repeated experiment consisting in optimisation of popular continuous benchmark function. Note, than in EA, the number of fitness function calls per generation is constant and equals the number of individuals that was 90 per one generation in the conducted experiments. Lamarckian and memetic modification lead to multiplication of this number in the case of the presented experimental results by 100, so the number of fitness computations for memetic EAs equals 900 per generation.

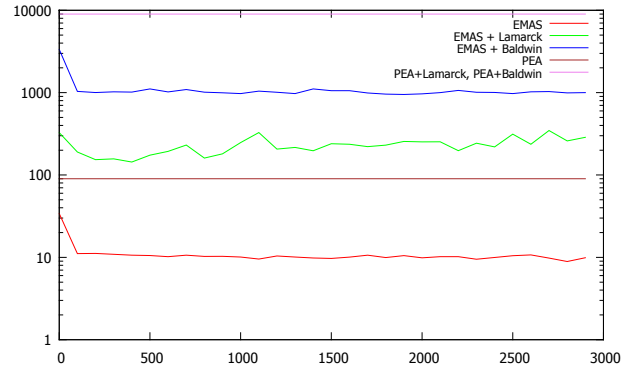

Fig. 10: Number of fitness function calls in EMAS and its memetic variations depending on the step number

However, the number of fitness function calls in the case of EMAS oscillates around 10 is an significant advantage of this computing method, moreover, Lamarckian modification of EMAS leads to obtaining about 200 fitness computations per step (still significantly lower than in the case of EA), while Baldwinian is the most costly one with the 1000 as estimate of the number of fitness computations. It is easy to see that low number of fitness function calls for EMAS makes it an interesting weapon of choice for dealing with problems characterised by a costly fitness function (e.g., inverse problems). In this case, the complexity of the implementation of the whole system, supporting the notion of agency, communication, naming services etc. is overwhelmed by the complexity of fitness function, so looking for a more intelligent search algorithm becomes reasonable, despite the intrinsic complexities imposed by its implementation.

VI. Conclusions

The main aim of the research presented in this paper was to prepare ground for researching agent-based memetic solutions for combinatorial optimisation problems. In order to do this, classical memetic approaches (evolutionary algorithm with Random Mutation Hill Climbing and Steepest Descent Local Search) to solving Low Autocorrelation Binary Sequence, Golomb Ruler and Job Shop were described. Experimental results were also gathered, processed and discussed, with stress on time efficiency.

Almost all the experiments yielded that memetic versions of the researched algorithms are significantly better than classical ones, but one must remember, that the results depend vastly on kind of local search used. However, one of the main problems of memetic algorithms is increased number of fitness function calls, imposed by the nature of this approach.

As a possible answer, moving to agent-based computing is proposed, as the cited results clearly show, that EMAS and its memetic variants significantly reduce the number of fitness function calls. Therefore, this effect merged with the memetic problem pointed out above, may result in fairly effective result, especially in the problems, where the fitness function is complex *per se*. This is preliminary confirmed by the presented experimental results of testing EMAS against EA for LABS problem. The agent-based metaheuristic turned out to be worse than EA for easier instance and significantly better than EA for hard instances of the considered problem.

In the near future the authors plan to further explore the possibilities of dealing with different types of combinatorial optimisation (e.g. more complex benchmarks, dynamic, multi-objective) using agent-based approaches.

Acknowledgment

The research presented here was partially supported by the grant "Biologically inspired mechanisms in planning and management of dynamic environments" funded by the Polish National Science Centre, No. N N516 500039.

References

[1] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., 1998.

[2] S. Droste, T. Jansen, and I. Wegener, "Upper and lower bounds for randomized search heuristics in black-box optimization," *Theory of*

Computing Systems, vol. 39, pp. 525–544, 2006. [Online]. Available: http://dx.doi.org/10.1007/s00224-004-1177-z

[3] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[4] P. Moscato, "Memetic algorithms: a short introduction," in *New ideas in optimization*. Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999, pp. 219–234.

[5] P. Moscato and C. Cotta, "A modern introduction to memetic algorithms," in *Handbook of Metaheuristics*, 2nd ed., ser. International Series in Operations Research and Management Science, M. Gendrau and J.-Y. Potvin, Eds. Springer, 2010, vol. 146, pp. 141–183.

[6] W. Hart, N. Krasnogor, and J. Smith, "Memetic evolutionary algorithms," in *Recent advances in memetic algorithms*, ser. Studies in Fuzziness and Soft Computing. Springer-Verlag, 2005, vol. 166, pp. 3–27.

[7] A. Byrski, W. Korczyński, and M. Kisiel-Dorohinicki, "Memetic multi-agent computing in difficult continuous optimisation," in *Proceedings of 6th International KES Conference on Agents and Multi-agent Systems Technologies and Applications, 2013, Hue City, Vietnam, IOS Press (accepted in 2013)*. Springer.

[8] L. Siwik and M. Kisiel-Dorohinicki, "Semi-elitist evolutionary multi-agent system for multiobjective optimization," in *Computational Science - ICCS 2006: Proc. of 6th International Conference*, ser. LNCS, V. N. Alexandrov, G. D. van Albada, P. Sloot, and J. Dongara, Eds., vol. 3993. Springer-Verlag, 2006.

[9] ——, "Improving the quality of the pareto frontier approximation obtained by semi-elitist evolutionary multi-agent system using distributed and decentralized frontier crowding mechanism," in *Adaptive and natural computing algorithms - ICANNGA 2007, Proc. of 8th international conference*, ser. LNCS, vol. 4431. Springer-Verlag, 2007.

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

[11] M. Kisiel-Dorohinicki, G. Dobrowolski, and E. Nawarecki, "Agent populations as computational intelligence," in *Neural Networks and Soft Computing*, ser. Advances in Soft Computing, L. Rutkowski and J. Kacprzyk, Eds. Physica-Verlag, 2003, pp. 608–613.

[12] A. Byrski and M. Kisiel-Dorohinicki, "Immune-based optimization of predicting neural networks," in *Computational Science - ICCS 2005: Proc. of 5th International Conference*, ser. LNCS, V. S. S. et. al., Ed., vol. 3516. Springer Verlag, 2005, pp. 703–710.

[13] A. Byrski and R. Schaefer, "Formal model for agent-based asynchronous evolutionary computation," in *Proceedings of IEEE Congress on Evolutionary Computation 2009 (IEEE CEC 2009)*, IEEE Computational Intelligence Society. Trondheim, Norway: IEEE Press, 18-21 May 2009.

[14] A. Byrski and M. Kisiel-Dorohinicki, "Agent-based model and computing environment facilitating the development of distributed computational intelligence systems," in *Computational Science - ICCS 2009, Proc. of 9th International Conference*, ser. LNCS, vol. 5545. Springer-Verlag, 2009.

[15] A. J. F. José E. Gallardo, Carlos Cotta, "Finding low autocorrelation binary sequences with memetic algorithms," *Applied Soft Computing*, vol. 9, 2009.

[16] P. V. H. Iván Dotú, "A note on low autocorrelation binary sequences," *Lecture Notes in Computer Science*, vol. 4204, 2006.

[17] F. H. Steven Halim, Roland H. C. Yap, "Engineering stochastic local search for the low autocorrelation binary sequence problem," *Lecture Notes in Computer Science*, vol. 5202, 2008.

[18] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part i: representation," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 983–997, Sep. 1996. [Online]. Available: http://dx.doi.org/10.1016/0360-8352(96)00047-2

[19] T. F. Abdelmaguid, "Representations in genetic algorithm for the job shop scheduling problem: A computational study," *JSEA*, vol. 3, no. 12, pp. 1155–1162, 2010.

[20] C. Bierwirth, D. C. Mattfeld, and H. Kopfer, "On permutation representations for scheduling problems," in *In 4th PPSN*. Springer-Verlag, 1996, pp. 310–318.

[21] C.-F. Liaw, "A hybrid genetic algorithm for the open shop scheduling problem," *European Journal of Operational Research*, vol. 124, no. 1, pp. 28 – 42, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722179900168X

[22] G. Bloom and S. Golomb, "Applications of numbered undirected graphs," *Proceedings of the IEEE*, vol. 65, no. 4, pp. 562 – 570, april 1977.

[23] A. Soliday, S.W. Homaifar and G. Lebby, "Genetic algorithm approach to the search for golomb rulers," in *6th International Conference on Genetic Algorithms (ICGA'95*. Morgan Kaufmann, 1995, pp. 528 – 535.

[24] B. Feeney, "Determining optimum and near-optimum golomb rulers using genetic algorithms," in *Master thesis, Computer Science, University College Cork*, 2003.

[25] J. Pereira, F.B. Tavares and E. Costa, "Golomb rulers: the advantage of evolution," in *Progress in Artificial Intelligence*. Springer Berlin Heidelberg, 2003, pp. 29 – 42.

[26] F. Tavares, J. Pereira and E. Costa, "Understanding the role of insertion and correction in the evolution of golomb rulers," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 1, june 2004, pp. 69 – 76 Vol.1.

[27] J. Tavares, F. Pereira, and E. Costa, "Evolving golomb rulers," in *Genetic and Evolutionary Computation - GECCO 2004*, ser. Lecture Notes in Computer Science, K. Deb, Ed. Springer Berlin Heidelberg, 2004, vol. 3103, pp. 416–417.

[28] I. Dotu and P. Van Hentenryck, "A simple hybrid evolutionary algorithm for finding golomb rulers," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, sept. 2005, pp. 2018 – 2023 Vol. 3.

[29] C. Cotta, I. Dotú, A. Fernández, and P. Hentenryck, "A memetic approach to golomb rulers," in *Parallel Problem Solving from Nature - PPSN IX*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4193, pp. 252–261.

[30] C. Cotta, I. Dotú, A. J. Fernández, and P. Hentenryck, "Local search-based hybrid algorithms for finding golomb rulers," *Constraints*, vol. 12, no. 3, pp. 263–291, Sep. 2007.

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

[32] J. Baldwin, "A new factor in evolution," *American Naturalist*, vol. 30, pp. 441–451, 1896.

[33] E. N. and G. S.J., "Punctuated equilibria: An alternative to phyletic gradualism," in *Models in Paleobiology*, T. Schopf, Ed. Freeman, Cooper and Co., 1972.

[34] K. Ku and M. Mak, "Exploring the effects of lamarckian and baldwinian learning in evolving recurrent neural networks," in *Proc. of 1997 IEEE Int. Conf. on Evolutionary Computation*. IEEE, 1997.

[35] D. Whitley, V. Scott Gordon, and K. Mathias, "Lamarckian evolution, the baldwin effect and function optimization," in *Proc. of Parallel Problem Solving from Nature III*, Y. Davidor, S. H.-P., and M. R., Eds. Springer, 1994.

[36] K. Pietak, A. Woś, A. Byrski, and M. Kisiel-Dorohinicki, "Functional integrity of multi-agent computational system supported by component-based implementation," in *Proc. of the 4th International Conference on Industrial Applications of Holonic and Multi-agent Systems*, ser. LNAI, vol. 5696. Springer-Verlag, 2009.

[37] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278 – 285, 1993, <ce:title>Project Management anf Scheduling</ce:title>. [Online]. Available: http://www.sciencedirect.com/science/article/pii/037722179390182M