Towards Finding an Effective Way of Discrete Problems Solving: The Particle Swarm Optimization, Genetic Algorithm and Linkage Learning Techniques Hybrydization

Bartosz Andrzej Fidrysiak and Michal Witold Przewozniczek
Department of Computational Intelligence, Wroclaw University of Technology, Wroclaw, Poland

Keywords: PSO for Binary Problems, Genetic Algorithms, Coevolution, Deceptive Functions, Linkage Learning.

Abstract: Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) are well known optimization tools. PSO advantage is its capability for fast convergence to the promising solutions. On the other hand GAs are able to process schemata thanks to the use of crossover operator. However, both methods have also their drawbacks – PSO may fall into the trap of preconvergence, while GA capability of fast finding locally optimal (or close to optimal) solutions seems low when compared to PSO. Relatively new, important research direction in the field of Evolutionary Algorithms is linkage learning. The linkage learning methods gather the information about possible gene dependencies and use it to improve their effectiveness. Recently, the linkage learning evolutionary methods were shown to be effective tools to solve both: theoretical and practical problems. Therefore, this paper proposes a PSO and GA hybrid, improved by the linkage learning mechanisms, dedicated to solve binary problems. The proposed method tries to combine the GA schema processing ability, linkage information processing and uses fast PSO convergence to quickly improve the quality of already known solutions.

1 INTRODUCTION

Particle Swarm Optimization (PSO), proposed in (Eberhart and Kennedy, 1995) is a popular optimization technique, commonly used as a base for proposals of methods designed to solve hard computational problems (Baek et al., 2015; Liu 2015; Lim and Isa 2014; Moubayed et al., 2014; Xua et al. 2015). Usually, the problems solved by PSO-based methods use the floating-point problem encoding. One of the PSO advantages is its capability of reaching fast convergence speed. For instance, in (Baek et al. 2015; Lim and Isa 2014; Liu 2015; Moubayed, et al. 2014) a stop condition is in range of 3100 up to $4.5*10^5$ fitness function evaluations (FFE) and is low when compared to GAs (Kwasnicka et al., 2015). A detailed analysis of PSO convergence speed and swarm diversity preservation can be found in (Bergh 2010). On the other hand the classical PSO analyses the dependencies between particles in the topological way – no schemata or Building Blocks (BB) (Goldberg et al., 1993) are neither found nor processed. The lack of BB processing and exchange mechanisms might limit the genotype-based method capability of solving hard discrete problems (Thierens 1999). Therefore the possible application of PSO-based methods to solve hard combinatorial problems like the Travelling Salesman Problem (TSP) (Rani and Vikas 2014), or the network flow optimization problems (Przewozniczek et al., 2015; Rani and Vikas 2014; Walkowiak et al., 2013) seems to be limited.

The Genetic Algorithm (GA) is a base of many methods designed to solve hard optimization problems of discrete nature (Andrade at al., 2015; Przewozniczek et al. 2015; Walkowiak et al., 2013). Even the simple Genetic Algorithm (sGA) is able to find, populate and exchange the BBs (Thierens 1999). The computational problem, when encoded with the use of genotype, is, at least partially, built from the groups of genes highly dependent on each other. Finding BBs and exchanging them between individuals opens the way for reaching the breakthrough and finding the solutions of a very good quality (Watson et. al. 1998). On the base of the BB theory, the linkage learning techniques were identified and their classification was proposed

(Chen et al. 2007b). Linkage learning methods gather the information about possible dependencies between genes and use this information to improve their effectiveness. Linkage learning methods have proven to be effective against both: theoretical (Correa and Shapiro 2006; Kwasnicka and Przewozniczek, 2011; Laumanns and Ocenasek 2002; Pelikan et. al 2006;) and practical problems (Przewozniczek et al. 2015; Walkowiak et al. 2013).

The above description has lead us to conclusion that a method characterized by PSO's fast convergence speed (in which the PSO is used as a kind of clever local optimizer) and GA's capability of BBs exchange enhanced by linkage learning techniques has a potential to be an effective one. Therefore this paper proposes a Multi-Swarm Particle Optimization with Crossing (MSPOCk). In the proposed method, PSO is used as an intelligent semi-local optimization tool (thanks to its fast convergence capability) and the GA crossover is used to exchange BBs and is enhanced by the linkage learning mechanism. The proposed method may be especially useful for solving practical problems, where mixed (discrete and floating point) problem encoding is used (Walkowiak et al. 2013; Przewozniczek et al. 2015). In such problems one gene is represented by two values: discrete one and floating point one. The method that is both: PSO- and GA-based may turn out to be an effective tool for solving such problems.

This paper is organized as follows. In section 2 the related work is presented. Section 3 contains the detailed description of the proposed MSPOCk method. The experiment setup, results of the computational tests and detailed experiment results analysis are placed in section 4. Finally, the last section concludes this work.

2 RELATED WORK

Since the paper considers a number of different Evolutionary Computation fields this section is divided into the following subsections. In first, the description of linkage learning techniques is presented. Then, PSO propositions for solving the discrete problems, available in the literature, are presented. Third subsection presents the already known GA and PSO hybrids, Finally, the last subsection presents multi-population PSOs.

2.1 Linkage Learning Techniques

The classification of linkage learning methods was proposed in (Chen et al. 2007b). The classification concerns different method features. In order to distinguish a bad linkage from a good one, method may use only the fitness function (unimetric approach) (Kwasnicka and Przewozniczek 2011; Przewozniczek et al. 2015; Walkowiak et al. 2013). If in addition to fitness function, the method uses additional measures then such approach is called multimetric.

Linkage information may be represented in two different ways: virtual or physical. If the method uses matrices, graphs, gene patterns (Kwasnicka and Przewozniczek 2011) to represent linkage then the virtual way is used. If linkage is represented as the location of two or more genes in the chromosome (the closer the genes are in the chromosome the more dependent they are) then the linkage is represented in the physical way. The good example of physical linkage representation is the messy coding (Goldberg et. al 1993; Kwasnicka and Przewozniczek 2011; Przewozniczek et al. 2015; Walkowiak et al. 2013), where genes may have any position in the chromosome. Finally, linkage information may be stored in some central database or be distributed in a genetic-linkage model manner. Some methods, like Multi Population Pattern Searching Algorithm (MuPPetS) (Kwasnicka and Przewozniczek 2011), may be assigned more than one value to one linkage classification category. For example MuPPetS use both possible linkage representation approaches and both linkage storage ways.

Usually, the linkage information is used by the crossover operator. In this group the GA methods may be found like MuPPetS (Kwasnicka and Przewozniczek 2011, Przewozniczek et al. 2015, Walkowiak et al. 2013) or DSMGA (Fan et al.). Recently, an interesting study proposing the Hybrid Linkage Crossover (HLX) operator and incorporating it into the Differential Evolution was presented (Cai and Wang, 2015).

It is worth noting that linkage learning methods are successful in solving hard computational problems of both kinds: theoretical like CEC 2005 (Cai and Wang, 2015) or deceptive functions concatenations (Correa and Shapiro 2006; Kwasnicka and Przewozniczek, 2011; Pelikan et. al 2006;) and large-scale practical problems (Walkowiak et al. 2013; Przewozniczek et al. 2015).

2.2 Particle Swarm Optimization in Solving Discrete Problems

To our best knowledge, the first PSO for discrete problem optimization was proposed in (Kennedy and Eberhart 1997), this method will be called PSO_V1. Due to the problem nature, the particle speed is interpreted differently than in a classical PSO – it describes the probability that the particle position in each dimension will be assigned the value of 0 or 1. To calculate the particle position value on the base of particle speed, the sigmoid function presented in equation (1) is used.

$$sig(y) = \frac{1}{1+e^{-y}} \qquad (1)$$

The use of sigmoid function to translate the particle speed into the particle position allows for using the same particle speed computation as in classical PSO. To omit the problem of too high particle speeds, the speed limit is used. Here, as in (Bergh 2010), the speed limit is set on <-4; 4>. Finally, the *i*th particle position in *j*th dimension is calculated as follows.

$$x_{i,j} = \begin{cases} 0 & if \quad r \ge sig(v_{i,j}(t+1)) \\ 1 & if \quad r < sig(v_{i,j}(t+1)) \end{cases}, \qquad (2)$$

where r is a random number from [0;1] uniform distribution. It is randomly generated each time, the particle position at each dimension is generated.

The drawback of PSO_V1 is the unintuitive speed interpretation. Therefore the PSO_V2 (Khanesar et al. 2007) was proposed. PSO_V2 is another PSO-based method for discrete (binary) problem solving. In PSO_V2 two particle speeds are used: the first one describes the probability to change the position value from 0 to 1, while the second is opposite. The speeds are defined as follows.

$$v_{i,j}^{(0)}(t+1) = \lambda v_{i,j}^{(0)}(t) + d_{i,j,1}^{(0)} + d_{i,j,2}^{(0)}, \qquad (3)$$

where

$$d_{i,j,1}^{(0)} = \begin{cases} c_p r_p & if \quad x_{i,j}^{best} = 0 \\ -c_p r_p & if \quad x_{i,j}^{best} = 1 \end{cases}$$

$$d_{i,j,2}^{(0)} = \begin{cases} c_g r_g & if \quad g_j^{best} = 0 \\ -c_g r_g & if \quad g_j^{best} = 1 \end{cases}$$

$$v_{i,j}^{(1)}(t+1) = \lambda v_{i,j}^{(1)}(t) + d_{i,j,1}^{(1)} + d_{i,j,2}^{(1)}, \qquad (4)$$

where

$$d_{i,j,1}^{(1)} = \begin{cases} -c_p r_p & if \quad x_{i,j}^{best} = 0 \\ c_p r_p & if \quad x_{i,j}^{best} = 1 \end{cases}$$

$$d_{i,j,2}^{(1)} = \begin{cases} -c_g r_g & if \quad g_j^{best} = 0 \\ c_g r_g & if \quad g_j^{best} = 1 \end{cases}$$

where $v_{i,j}$ is the *i*th particle speed in *j*th dimension, t is the iteration number, λ is the inertia weight, c_p and c_g are positive constants, r_p and r_g are random number from [0;1] uniform distribution, $x_{i,j}$ is the *i*th particle position in *j*th dimension, $x_{i,j}^{best}$ and g_j^{best} are particle and global swarm optima found that far.

2.3 Genetic Algorithm and Particle Swarm Optimization Hybrids

The GA and PSO work in different ways and have different advantages and disadvantages. These differences make them good candidates for hybridization. Some PSO and GA hybrids were already proposed (Chen et al. 2007; Devicharan and Mohan 2004, Lovbjerg et al. 2001). In (Chen et al. 2007) the combination of crossover and selection operator, PSO mechanisms and the linkage learning are proposed. At the beginning of the method, the PSO proposed in (Mu et al. 2009) is used. Then, the groups of dependent genes are generated randomly. For example the randomly generated gene group may contain gene positions 2, 5 and 6 of 10-bit problem. As long as the average population fitness improves fast enough, the dependent gene groups remain, otherwise they are reinitialized. Note that such linkage learning mechanism is primitive and may occur ineffective.

2.4 Particle Swarm Optimization as a Multi-Population Method

Similarly to other evolutionary methods PSO may be modified to become a multi-population method. The good example may be found in (Chang, 2015), where PSO's original single swarm is divided into a number of subswarms. Such a multi-swarm PSO is used to solve multimodal problems. However, the proposed method does not incorporate any subswarm communication. A more evolved multi-swarm PSO was presented in (Dahzi et al. 2008). After a certain number of generations, the neighbouring subswarms exchange their globally best positions found that far. The Multi-Swarm Cooperative Particle Swarm Optimizer (MCPSO) was presented in (Niu et al. 2007). In MCPSO the subswarms are not equal. One of them is the master swarm, while the other are called slaves. The slave swarms are executed independently, their role is to preserve the particles diversity and support the master swarm with knowledge they are able to gather. Master swarm use its own knowledge and the information it is able to possess from slave swarms to improve the overall method effectiveness.

3 THE PROPOSED HYBRID

As stated in section 2.3, the GA and PSO hybrid proposed in (Chen et al. 2007) may be easily adjusted to solve binary problems. It is enough to exchange the standard PSO from (Mu et al. 2009) for PSO_V2. However, such a hybrid will have a significant drawback. Since all the particles know the best solution found by a method that far it is likely that many particles will converge to this solution. Therefore, in this section, a new GA and PSO hybrid improved by linkage learning mechanisms is proposed. In the proposed Multi-Swarm Particle Optimization with Crossing (MSPOCk), PSO_V2 is used only as fast, quasi local optimizer. From this point of view, the very fast PSO convergence is a positive feature (even if it leads directly to preconvergence). In order avoid the preconvergence, a number of PSO swarms are executed separately and in parallel. These swarms should converge to different solution space parts. In order to exchange the information between the best solutions found by the coevolving swarms, the dedicated crossover operator is used and enforced with linkage learning. If some of the subswarms start to explore the same or similar solution space regions then only one of these subswarms is left unchanged, the rest is reinitialized. Note that such a way of preserving the population diversity is not typical for PSO-based methods. The usual way is to change the population topology from *global best*, to *local best* (Kennedy and Mendes 2006). In the MSPOCk method the global best topology is used but only within single subswarm.

3.1 General Method Overview

The general MSPOCk method overview, for P particle subpopulations is presented in Figure 1.
The population initialization is done randomly for all the particles. Before execution of PSO for each subswarm, the check if subswarms are not too similar is done. If some subswarms explore the same or very similar solution space parts only one of them is left, the other are reinitialized in order to give them a chance to explore different solution space parts. This operation preserves population diversity.

In order to choose which subswarms should be reinitialized, the similarity measure of two subswarms is defined and presented in equation (5).

```
t=0;
InitializePopulation();
While(!StopCondition)
Begin
  ReinitializeSimilarSubswarms();
  for(int p = 0; p < P; p++)
    Run PSO_V2 for pth subswarm;
  UpdateLinkageInformation();
  CrossParticles();
  t = t+1;
End of while;
```

Figure 1: MSPOCk pseudocode.

$$d(G_x, G_y) = \sum_{j=1}^{n} \left| g_{x,j}^{best} - g_{y,j}^{best} \right|, \qquad (5)$$

where G_x, G_y are xth and yth subswarms respectively, and the $g_{x,j}^{best}$, $g_{y,j}^{best}$, are the jth dimension values of the best solution found by xth and yth subswarm respectively.

The similarity check is done for all possible subswarm pairs. If any two subswarms are more similar than a user defined value D then the subswarm with lower g_p^{best} value is reinitialized.

3.2 New Particle Population Creation on the Base of Crossover

In the particle creation phase, new particles, created by crossover operation, replace the old ones. The parents are chosen randomly with equal probability from k best individuals of each subswarm. Note, that except the incorporation of elitism, such a procedure is close to the tournament selection idea. Such method construction helps to preserve the population diversity. This feature is also enforced by the previously described subswarm reinitialization procedure.

Before crossing, the linkage information is updated. If this operation is done for the first time every gene is randomly assigned to one of the two gene groups: group A or group B. If the linkage information update is done later then the current gene groups are preserved, if the average population fitness has increased. Otherwise the gene groups are reinitialized.

When two parents are chosen the crossover operator is used to generate offspring. The crossover operator exchange genes marked by the current gene exchange pattern. To create one offspring genes from group A are taken from parent A and genes from group B are taken from parent B. For example, if the genotypes of parents A and B are: 111000110, 010101011, and the gene exchange pattern is AAABBBAAB then the offspring genotype will be: 111 101 11 1. After crossing, for each subswarm, the x_i^{best} and g_p^{best} are reinitialized with all the previous history of found solutions cleaned. The particles speeds are cleaned as well and are randomly reinitialized.

Note that such linkage generation procedure is similar to (Chen et al. 2007) and is still quite primitive. Nevertheless it removes the gene order dependency of single point crossover and allows for significant method performance improvement if good quality gene exchange pattern was created.

3.3 Proposed Method Summary

The proposed MSPOCk method is a multi-population method which uses one of the main weaknesses of PSO concept, namely the preconvergence, as an advantage. The crossover operator is used to transmit the genetic information between fully separated and coevolving subswarms. The mechanism of subswarm reinitialization introduces the idea of global mutation into the method.

The MSPOCk method uses simple, but still effective, linkage learning mechanisms to reinforce the quality of crossover operator. The linkage information is stored in a centralized way. All methods considered here use unimetric way to distinguish the good and bad linkage as it is based only on fitness value.

It is worth noting that MSPOCk is completely different to other multi-swarm PSOs presented in section 2.4. First of all it incorporates a direct data exchange between particles by the use of crossover operator. Second, it uses linkage learning to improve the crossover quality. Third, it reinitializes subswarms if they get too close to each other. Finally, the possible PSO preconvergence is used as advantage, not a drawback.

4 THE RESULTS

In this section the results of the performed experiments are presented. This section is organized as follows. First, the test problems based on deceptive functions concatenations are presented (Deb and Goldberg, 1992). In the second subsection the tuning procedure and its results for all the competing methods are presented. The third subsection describes and discusses the main results. The competing methods are: PSO_V1, PSO_V2, GA+PSO (GA and PSO hybrid from (Chen et al. 2007) with PSO_V2 used) and MSPOCk. All methods were coded in C++ in Qt 5.1.1 (MSCV 2010, 32-bit). The complete sourcecodes, results and summaries may be downloaded from: http://www.mp2.pl/download/ai/20150617_mspock.zip. All the test runs were executed on Intel Core i7-3632QM machine with 64-bit Windows 7 on board. Each method was executed 5 times for each test case. In order to make the results reliable, all the methods shared the parts of code whenever it was possible and all experiments were executed in clean system environment with no other resource consuming processes running.

4.1 The Deceptive Functions Concatenations

In the performed experiments the deceptive functions concatenations were used as a test problems. The deceptive functions were proposed in (Deb and Goldberg, 1992) and are a common test tool for methods that use the binary coding. The deceptive function value is based on its unitation. The unitation of a binary string is a number of '1's in the string. The function value increase as its unitation is decreasing, but the function value is optimal only for the maximal unitation. Therefore, the optimization methods are often misled to the suboptimal solutions characterized with low unitation.

The concatenations of deceptive functions are hard to solve for any method that does not have mechanisms based on the prior knowledge about deceptive function nature. Usually, the test cases are just concatenation of identical order-3 up to order-5 deceptive functions. Here, the test cases used, are more diverse in order to better imitate the real life problems. In other words – it seems doubtful that a real life problem will be built from the identical subproblems. It seems also reasonable that a real life problem may contain a part that is easy to solve for any GA-based method (Kwasnicka and Przewozniczek). Therefore the test cases are the concatenation of four different deceptive functions, presented in Table 1 and the *tail* function given in equation (6).

$$f_{tail}(x) = u / l, \qquad (6)$$

where u is the argument unitation, and l is a bit number.

Table 1: Deceptive functions used for tests.

u	3a	3b	5a	5b
0	0.900	9.000	0.900	9.000
1	0.45	4.500	0.675	6.750
2	0.000	0.000	0.450	4.500
3	1.000	10.000	0.225	2.250
4	N/A	N/A	0.000	0.000
5	N/A	N/A	1.000	10.000

The concatenations of the functions presented above were made in four different versions: *flat*, *flat+tail*, *rough*, *rough+tail*. The flat test cases were built from deceptive functions with the same maximum value (only 3a and 5a functions), while rough are the concatenation of functions with different maximum. In addition, to some of the test cases, the tail function was added to imitate the fact that some part of the real life problem may be fairly easy to solve. The list of used test cases is given in Table 2.

Table 2: The test cases used in the experiments.

Deceptive functions total len. (order of functions used)	Version	Deceptive function number				Tail length
		3a	3b	5a	5b	
30 (3 and 5)	Flat	5	0	3	0	0
	Flat+tail	5	0	3	0	30
	Rough	3	2	2	1	0
	Rough+Tail	3	2	2	1	30
30 (5)	Flat	0	0	6	0	0
	Flat+tail	0	0	6	0	30
	Rough	0	0	3	3	0
	Rough+Tail	0	0	3	3	30
50 (3 and 5)	Flat	10	0	4	0	0
	Flat+tail	10	0	4	0	50
	Rough	5	5	2	2	0
	Rough+Tail	5	5	2	2	50
50(5)	Flat	0	0	10	0	0
	Flat+tail	0	0	10	0	50
	Rough	0	0	5	5	0
	Rough+Tail	0	0	5	5	50
150 (3 and 5)	Flat	30	0	12	0	0
	Flat+tail	30	0	12	0	150
	Rough	15	15	6	6	0
	Rough+Tail	15	15	6	6	150
150 (5)	Flat	0	0	30	0	0
	Flat+tail	0	0	30	0	150
	Rough	0	0	15	15	0
	Rough+Tail	0	0	15	15	150

Note, that similar test case generation was proposed in (Kwasnicka and Przewozniczek, 2011). The genes in the concatenations were not shuffled since all competing methods are not gene order dependent, which is a desired feature for any method that uses genotype-based problem coding.

4.2 The Tuning Procedure

Finding the optimal parameter settings for the competing methods, does not seem possible due to practical reasons. Therefore the tuning procedure was as follows. First, the initial settings for each method were proposed on the base of authors experience and literature review. Second, each parameter was separately tuned in an order presented in Table 3. For each parameter, a range of values was checked and the best was chosen for further use. The final parameter settings, with their initial values are given in Table 3. The values λ, c_p and c_g for GA+PSO and MSPOCk methods were copied from the final values tuned for PSO_V2. The maximum computation time was 3000 seconds for all methods.

Table 3: Tuning results.

Method	Par.	Initial value	Final value	Par. description
PSO V1	N	1000	400	Population size
	λ	1	0.99	Inertia weight
	c_p	2	1.81	As in eq. (3), (4)
	c_g	2	1.50	As in eq. (3), (4)
PSO V2	N	1000	700	Population size
	λ	1	0.87	Inertia weight
	c_p	2	2.00	As in eq. (3), (4)
	c_g	2	1.86	As in eq. (3), (4)
GA + PSO	N	1000	100	Population size
	λ	0.87	0.87	Inertia weight
	c_p	2.00	2.00	As in eq. (3), (4)
	c_g	1.86	1.86	As in eq. (3), (4)
	l_{max}	200	60	PSO iteration number
	k	0.25	0.27	% of best particles used for crossover
	ε	0.01	0.05	Min. avr. fitness increase to stay with cur. linkage
MSPO Ck	N	1000	1000	Population size
	λ	0.87	0.87	Inertia weight
	c_p	2.00	2.00	As in eq. (3), (4)
	c_g	1.86	1.86	As in eq. (3), (4)
	l_{max}	200	160	PSO iter. num.
	k	0.25	0.05	% of best particles used for crossover
	P	25	25	Subswam number
	S	0.95	1.0	Min. subswarm similarity

4.3 Main Results

The main measure of the result quality in the performed tests is the function unitation – the higher number of '1's, the closer the solution is to the optimum. Note, that sometimes the function value difference may be relatively small, while the difference in unitation will be very large. The method capable of finding solutions with high unitation is expected to be more capable of leaving the local optima areas and less likely to stuck. Therefore, such a method should be more useful for solving hard computational problems (Kwasnicka and Przewozniczek, 2011). The average unitation for all the competing methods is given in Table 4.

Table 4: Average untiation for each experiment group.

Test case group	Ver.	PSO V1 [%]	PSO V2 [%]	GA+ PSO [%]	MSPOCk [%]
30 (3 and 5)	r	96.67	76.67	**100.00**	**100.00**
	r + t	98.33	88.33	95.00	**100.00**
	f	93.33	66.67	**100.00**	**100.00**
	f + t	96.67	86.67	93.33	**100.00**
30 (5)	r	93.33	60.00	**100.00**	**100.00**
	r + t	93.33	78.33	93.33	**100.00**
	f	90.00	70.00	**100.00**	**100.00**
	f + t	91.67	83.33	93.33	**100.00**
50 (3 and 5)	r	90.00	82.00	92.00	**100.00**
	r + t	97.00	89.00	90.00	**100.00**
	f	80.00	72.00	88.00	**100.00**
	f + t	94.00	87.00	83.00	**100.00**
50 (5)	r	82.00	52.00	74.00	**100.00**
	r + t	92.00	77.00	77.00	**100.00**
	f	74.00	40.00	54.00	**100.00**
	f + t	78.00	70.00	63.00	**100.00**
150 (3 and 5)	r	68.00	69.33	68.00	**74.00**
	r + t	**86.00**	85.00	83.33	84.33
	f	65.33	70.00	68.67	**71.33**
	f + t	84.33	**85.00**	83.67	82.67
150 (5)	r	26.00	23.33	26.00	**40.00**
	r + t	61.33	61.00	63.00	**69.67**
	f	14.00	22.00	18.67	**42.67**
	f + t	56.00	61.00	56.67	**69.67**

As shown in Table 4, the proposed MSPOCk method outperforms all the competing methods. It was the best for 22 of 24 test problems. For the shorter test cases the MSPOCk is able to report perfect results in every run, for the 150-bit problems the MSPOCk supremacy is still significant, but the unitation rate significantly drops down. Both binary PSO versions are able to propose reasonable results, but of significantly lower quality than MSPOCk. The MSPOCk advantage over the other methods is smaller for the longest problems built from mixed deceptive blocks of different size. It seems that these problems are hard enough to deceive all methods equalizing their quality. In all other problem groups MSPOCk has clear advantage over the competing methods (except the shortest problems, where GA+PSO is sometimes also able to gain optimal results in all runs). Note that the obtained results indicate the importance of building deceptive function concatenations also from blocks of mixed size, not only identical ones.

For both PSO and MSPOCk, the unitation increases if the problem is added a tail. This observation is an expected one – for any PSO- or GA-based method it is easy to optimize the tail-like functions. In the case GA+PSO the situation is sometimes opposite (for both 30-bit problems and one 50-bit). The reason of this phenomenon is that the more bits are necessary to encode the problem, the harder it is to randomly generate proper linkage information, therefore the overall effectiveness of GA+PSO drops down. Note, that GA+PSO generates the linkage information in the most primitive way – randomly. It seems clear that without any other performance improving mechanisms, such a method will not be able to effectively solve hard problems. On the other hand, although MSPOCk also use primitive linkage information generation procedure, it uses a number of different diversity preservation mechanisms and a kind of global mutation operator (subswarm reinitialization). Therefore, MSPOCk is able to remain effective even if the linkage information quality is, in general, low. Note, that thanks to the population diversity, MSPOCk is able to leave the local optima and continue the search for the global one.

Differently to results presented in (Kwasnicka and Przewozniczek, 2011), the competing method effectiveness was not always dependent on function type (rough/flat) and, if the dependency was occurring, then better results were proposed to the rough function type (eg. PSO_V1, PSO_V2 and GA+PSO results for both 50-bit problems). This last observation is partially opposite to the analysis presented in (Kwasnicka and Przewozniczek, 2011).

The explanation of this phenomenon may be that for rough functions it is easier for the method to concentrate its computation effort on the more valuable (measured in fitness function value) problem parts first, which improves the method effectiveness as it works like if it was solving a sequence of many shorter problems instead of single, but long one. In the flat problems case such computation effort concentration is impossible since all problem parts are equally valuable.

5 CONCLUSIONS AND FURTHER WORK

In this paper, the new PSO and GA hybrid with linkage learning mechanisms for solving the binary problems was presented. The proposed MSPOCk uses many coevolving subswarms in order to be able to leave local optima. The main paper purpose was to show that the combination of PSO fast convergence, GA capability of exchanging groups of genes via crossover operator reinforced with linkage learning is the promising way for discrete problems solving. The MSPOCk was significantly more effective than all other competing methods. In some of the test cases it was able to repetitively find the optimal solution.

As stated above, this work is only the presentation of the possible effectiveness potential behind the GA and PSO hybrids application to discrete problems. Therefore, the future work shall concern most of the MSPOCk mechanisms presented here:

- The linkage learning mechanisms, similar to (Kwasnicka and Przewozniczek, 2011; Pelikan et al. 2006) shall be introduced into the method. Such modification should improve the method effectiveness.
- The subswarm number should not be a method parameter. It should be designated by the method itself. The interesting idea is to use the same or similar mechanisms as presented in (Kwasnicka and Przewozniczek, 2011; Przewozniczek et al. 2015; Walkowiak et al. 2013) were a number of coevolving subpopulations change during the method run and is dependent on the method state.
- PSO_V2, used in MSPOCk, is unable to solve other than binary problems. The future work should concentrate on proposing PSO based methods capable of solving any discrete problem, not only binary one.
- The proposed MSPOCk should be compared on the base of wide experiment set with well known, effective GA-based methods like MuPPetS, BOA and hBOA.

The further study on MSPOCk should also consider its application to hard practical problems.

REFERENCES

Andrade, C., Toso, R., Resende, M., Miyazawa, F., 2015, Biased Random-Key Genetic Algorithms for the Winner Determination Problem in Combinatorial Auctions, In *Evolutionary Computation*, Vol. 23, No. 2: 279–307.

Baek,H., Ryu, J., Oh, J., Kim T., 2015, Optimal design of multi-storage network for combined sewer overflow management using a diversity-guided, cyclic-networking particle swarm optimizer – A case study in the Gunja subcatchment area, Korea, In *Expert Systems with Applications*, Vol. 42, Issue 20, pp. 6966-6975.

Bergh F., 2010, An Analysis of Particle Swarm Optimization. In Computer and Information Science, Vol.3, no. 1, pp.180-184.

Cai, Y, Wang, Y., 2015, Differential evolution with hybrid linkage crossover. In *Information Sciences*, Vol. 237, pp. 244-287.

Chang, W.D., 2015, A modified particle swarm optimization with multiple subpopulations for multimodal function optimization problems. In *Applied Soft Computing*, Vol. 33, pp. 170-182.

Chen, Y., Peng. W, Jian M., 2007, Particle Swarm Optimization With Recombination and Dynamic Linkage Discovery, In *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, Vol.37, Issue 6, pp.1460-1470.

Chen, Y., Sastry, K., Goldberg, D.E., 2007b, A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms, In *IlliGAL Report No. 2007014*, Illinois Genetic Algorithms Laboratory.

Correa, E.S., Shapiro, J.L., 2006, Model Complexity vs. Performance in the Bayesian Optimization Algorithm, In *Lecture Notes in Computer Science* , Vol. 4193, pp. 998-1007.

Dahzi, W.., Wu, CH., Ip, W.H., Wang, D., Yan, Y., 2008, Parallel multi-population Particle Swarm Optimization Algorithm for the Uncapacitated Facility Location problem using OpenMP. In IEEE Congress on Evolutionary Computation.

Deb, K., Goldberg, D. E., 1992, Sufficient Conditions for Deceptive and Easy Binary Functions, In *Annals of Mathematics and Artificial Intelligence*, Vol. 10, pp. 385-408.

Devicharan, D., Mohan, C.K., 2004, Particle Swarm Optimization with Adaptive Linkage Learning, In *Congress on Evolutionary Computation*, Vol.1, pp.530-535.

Eberhart, R. & Kennedy, J., 1995. A New Optimizer Using Particle Swarm Theory. In *Proceeding of, 6th International Symposium on Micro Machine and Human Science,* pp.530-535.

Fan, K., Yu, T. Lee, J., 2013, Linkage learning by number of function evaluations estimation: practical view of building blocks. In *Information Sciences*, Vol. 230, Issue 1, pp. 162–182.

Goldberg, D.E., Deb, K., Kargupta H., Harik, G., 1993, Rapid, accurate optimization of difficult problems using fast messy genetic algorithms, In *Proceedings of 5th International Conference on Genetic Algorithms.*

Kennedy, J. & Eberhart, R., 1997, A Discrete Binary Version of the Particle Swarm Algorithm. In *IEEE International Conference on Systems, Man and Cybernetics, Computational Cybernetics and Simulation*, Vol.5, pp.4104-4108.

Kennedy, J., Mendes, R., 2006, Neighborhood Topologies in Fully-InformedandBest-Of-Neighborhood ParticleSwarms, In *IEEE Transactions on Systems,Man,and Cybernetics, PartC: Applications and Reviews*, Vol. 36, Issue 4, pp.515-519.

Khanesar, M. A, Teshnehlab, M. & Shoorehdeli, M.A., 2007, A Novel Binary Particle Swarm Optimization, In *Proceedings of the 15th Mediterranean Conference on Control&Automation*, pp.1-6.

Kwasnicka, H., Przewozniczek, 2011, M., Multi Population Pattern Searching Algorithm: a new evolutionary method based on the idea of messy Genetic Algorithm, In *IEEE Transactions on Evolutionary Computation*, Vol. 15 Issue 5, pp.715-734.

Laumanns, M., Ocenasek, J., 2002, Bayesian Optimization Algorithms for multi-objective optimization, In *Lecture Notes in Computer Science* , Vol. 2439, pp. 298-307.

Lim, W.H., Isa, N., 2014, Bidirectional teaching and peer-learning particle swarm optimization, In *Information Sciences*, Vol. 280, pp. 111-134.

Liu, Q., 2015, Order-2 Stability Analysis of Particle Swarm Optimization, In *Evolutionary Computation*, Vol. 23, No. 2, pp. 187–216.

Lovbjerg, M., Rasmussen, T. K., Krink, T., 2001, Hybrid Particle Swarm Optimiser with Breeding and Subpopulations, In *Proceedings of the Genetic and Evolutionary Computation Conference*, Vol.24, pp.469-476.

Moubayed, N., Petrovski, A., McCall, J., 2014, D2MOPSO: MOPSO Based on Decomposition and Dominance with Archiving Using Crowding Distance in Objective and Solution Spaces, In *Evolutionary Computation*, Vol. 22, No. 1, pp. 47–77.

Mu, A.Q., Cao, D.X., Wang, X.H., 2009, A Modified Particle Swarm Optimization Algorithm, In *Natural Science*, Vol.1, No. 2, pp. 151-155.

Niu, B., Zhu, Y., He, X., Wu, H., 2007, MCPSO: A multi-swarm cooperative particle swarm optimizer, In Applied Mathematics and Computation, Vol. 185, pp. 1050-1062.

Pelikan, M., Sastry, K., Butz, M.V., Goldberg, D.E., 2006, Hierarchical BOA on Random Decomposable Problems, In *MEDAL Report No. 2006001*.

Przewozniczek, M., Goscien, R., Walkowiak, K., Klinkowski, M., 2015, Towards Solving Practical Problems of Large Solution Space Using a Novel Pattern Searching Hybrid Evolutionary Algorithm - An Elastic Optical Network Optimization Case Study, In *Expert Systems with Applications*, Vol. 42, pp. 7781-7796.

Rani K., Vikas K., 2014, Solving Travelling Salesman Problem Using Genetic Algorithm Based On Heuristic Crossover And Mutation Operator, In *International Journal of Research in Engineering & Technology*, Vol. 2, Issue 2, pp. 27-34.

Thierens, D., 1999, Scalability problems of simple genetic algorithms, In *Evolutionary Computation*, Vol. 7, Issue 4, pp. 331-352.

Walkowiak, K., Przewozniczek, M., Pajak, K., 2013, Heuristic Algorithms for Survivable P2P Multicasting, In *Applied Artificial Intelligence*, Vol. 27, Issue 4, pp. 278-303.

Watson, R.A., Hornby, G.S., Pollack, J.B., 1998, Hierarchical Building-Block Problems for GA Evaluation, In *Parallel problem solving from nature* , pp. 97-106.

Xu, L., Wang, J., Li, Y, Li, Q., Zhang, X., 2015, Resource allocation algorithm based on hybrid particle swarm optimization for multiuser cognitive OFDM network, In *Expert Systems with Applications*, Vol. 42, Issue 20, pp. 7186–7194.