Generalized Self-Adapting Particle Swarm Optimization algorithm

Mateusz Uliński, Adam Żychowski, Michał Okulewicz*,
Mateusz Zaborski, Hubert Kordulewski

Warsaw University of Technology, Faculty of Mathematics and Information Science,
M.Okulewicz@mini.pw.edu.pl

Abstract. This paper presents a generalized view on the family of swarm optimization algorithms. Paper focuses on a few distinct variants of the Particle Swarm Optimization and also incorporates one type of Differential Evolution algorithm as a particle's behavior. Each particle type is treated as an agent enclosed in a framework imposed by a basic PSO. Those agents vary on the velocity update procedure and utilized neighborhood. This way, a hybrid swarm optimization algorithm, consisting of a heterogeneous set of particles, is formed. That set of various optimization agents is governed by an adaptation scheme, which is based on the roulette selection used in evolutionary approaches. The proposed Generalized Self-Adapting Particle Swarm Optimization algorithm performance is assessed a well-established BBOB benchmark set.

Keywords: Particle Swarm Optimization, Self-adapting metaheuristics

1 Introduction

Since its introduction [10] and subsequent modifications [5, 20] Particle Swarm Optimization (PSO) algorithm has attracted many researchers by its simplicity of implementation and easiness of parallelization [11, 27]. PSO has currently a several standard approaches [5], multiple parameter settings considered to be optimal [8] and successful specialized approaches [3]. PSO have also been tried with various topologies [9, 19], and unification [18] and adaptation schemes.

This paper brings various population based approaches together, and puts them in a generalized swarm-based optimization framework (GPSO). The motivation for such an approach comes from the social sciences, where diversity is seen as a source of synergy [12] and our adaptive approach (GAPSO) seeks an emergence of such a behavior within a heterogeneous swarm [15].

The remainder of this paper is arranged as follows. Section 2 introduces PSO and its adaptive modifications, together with discussing Differential Evolution (DE) algorithm and its hybridizations with PSO. In Section 3 general overview of the system's construction is provided. Section 4 describes adaptation scheme

* Corresponding author.

and future system implementation details. Section 5 is devoted to a presentation of the experimental setup, in particular, the benchmark sets and parametrization of the methods used in the experiments. Experimental results are presented in Section 6. The last section concludes the paper.

2 Particle Swarm Optimization: modification and hybridization approaches

This section reviews optimization algorithms used as basic building blocks within our generalized approach: PSO and DE. Initial subsections introduce the basic forms of the PSO and DE algorithms, while the following summarize the research on hybridizing those approaches and creating the adaptive swarm optimizers.

Particle Swarm Optimization. PSO is an iterative global optimization metaheuristic method utilizing the ideas of swarm intelligence [10,20]. The underlying idea of the PSO algorithm consists in maintaining the swarm of particles moving in the search space. For each particle the set of neighboring particles which communicate their positions and function values to this particle is defined. Furthermore, each particle maintains its current position x and velocity v, as well as remembers its historically best (in terms of solution quality) visited location. In each iteration t, ith particle updates its position and velocity, according to formulas 1 and 2.

Position update. The position is updated according to the following equation:

$$\mathbf{x}_{t+1}^{i} = \mathbf{x}_{t}^{i} + \mathbf{v}_{t+1}^{i}. \tag{1}$$

Velocity update. In a basic implementation of PSO (as defined in [5,20]) velocity v_t^i of particle i is updated according to the following rule:

$$\mathbf{v}_{t+1}^{i} = \omega \cdot \mathbf{v}_{t}^{i} + c_1 \cdot (\mathbf{p}_{best}^{i} - \mathbf{x}_{t}^{i}) + c_2 \cdot (\mathbf{neighbors}_{best}^{i} - \mathbf{x}_{t}^{i}) \tag{2}$$

where ω is an inertia coefficient, c_1 is a local attraction factor (cognitive coefficient), $\mathbf{p}_{best}^{i}$ represents the best position (in terms of optimization) found so far by particle i, c_2 is a neighborhood attraction factor (social coefficient), $\mathbf{neighbors}_{best}^{i}$ represents the best position (in terms of optimization) found so far by the particles belonging to the neighborhood of the ith particle (usually referred to as $\mathbf{g}_{best}$ or $\mathbf{l}_{best}$).

Differential Evolution. DE is an iterative global optimization algorithm introduced in [21]. DE's population is moving in the search space of the objective function by testing the new locations for each of the specimen created by crossing over: (a) a selected $\mathbf{x}^j$ solution, (b) solution $\mathbf{y}_t^{(i)}$ created by summing up a scaled difference vector between two random specimen ($\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$) with a third solution ($\mathbf{x}^{(i)}$). One of the most successful DE configurations is DE/rand/1/bin,

where in each iteration t, each specimen $\mathbf{x}_t^i$ in the population is selected and mutated by a difference vector between random specimens $\mathbf{x}_t^{(i_1)}$ and $\mathbf{x}_t^{(i_2)}$ scaled by $F \in \mathbb{R}$:

$$\mathbf{y}_t^{(i)} = \mathbf{x}_t^{(i)} + F \times (\mathbf{x}_t^{(i_2)} - \mathbf{x}_t^{(i_1)}) \tag{3}$$

Subsequently, $y_t^{(3)}$ is crossed-over with x_t^{best} by binomial recombination:

$$\mathbf{u}_t^i = Bin_p(\mathbf{x}_t^{best}, \mathbf{y}_t^{(i)}) \tag{4}$$

Finally, the new location u_t^i replaces original x_t^i iff it provides a better solution in terms of the objective function f:

$$\mathbf{u}_t^i = \begin{cases} \mathbf{u}_t^i & \text{if } f(\mathbf{u}_t^i) < f(\mathbf{x}_t^i) \\ \mathbf{x}_t^i & \text{otherwise} \end{cases} \tag{5}$$

Adaptive PSO approaches. While a basic version of the PSO algorithm has many promising features (i.e. good quality of resutls, easiness of implementation and parallelization, known parameters values ensuring theoretical convergence) it still needs to have it parameters tuned in order to balance its exploration vs. exploitation behavior [28].

In order to overcome those limitations a two–stage algorithm has been proposed [28]. That algorithm switches from an exploration stage into an exploitation stage, after the first one seems to be "burned out" and stops bringing much improvement into the quality of the proposed solution. Another adaptive approach that has been proposed for the PSO [26], identifies 4 phases of the algorithm: *exploration*, *exploitation*, *convergence*, and *jumping out*. The algorithm applies fuzzy logic in order to assign algorithm into one of those 4 stages and adapt its inertia (ω), cognitive (c_1) and social (c_2) coefficients accordingly. Finally, a heterogeneous self-adapting PSO has been proposed [15], but it has been limited by its usage only of the swarm based approaches.

PSO and DE Hybridization. While DE usually outperforms PSO on the general benchmark tests, there are some quality functions for which the PSO is a better choice, making it worthwhile to create a hybrid approach [2, 22].

Initial approaches on hybridizing PSO and DE consisted of utilizing DE mutation vector as an alternative for modifying random particles coordinates, instead of applying a standard PSO velocity update [6, 23]. Another approach [16], consists of maintaining both algorithms in parallel and introducing an information sharing scheme between them. A similar approach can be found in [25] with additional random search procedure. PSO and DE can also be combined in a sequential way [7, 13]. In such an approach first the standard PSO velocity update is performed and subsequently various types of DE trials are performed on particle's p_{best} location in order to improve it further.

3 Generalized Particle Swarm Optimization

This article follows the approach set for a social simulation experiment [17], by generalizing PSO velocity update formula (eq. (2)) into a following form (with I being and indicator function):

$$\begin{aligned} v_{t+1}^i =& \omega \cdot v_t^i + c_1 \cdot (p_{best}^i - x_t^i) \\ &+ \sum_{i=1}^{|neighborhood|} \sum_{j=1, j\neq i}^{|particles|} I(j\text{th is } i\text{th neighbor}) c_j' \cdot (p_{best}^j - x_t^i) \\ &+ \sum_{i=1}^{|neighborhood|} \sum_{j=1, j\neq i}^{|particles|} I(j\text{th is } i\text{th neighbor}) c_j'' \cdot (x_t^j - x_t^i) \end{aligned} \tag{6}$$

In that way the social component extends into incorporating data from multiple neighbors and neighborhoods. The other part of generalization is not imposing an identical neighborhood structure over all particles, but letting each particle decide on the form of neighborhood. That way we take advantage of the agent-like behavior of swarm algorithms, were each individual is making its own decisions on the basis of simple rules and knowledge exchange (the other particles do not need to know behavior of a given particle, only its positions and sampled function values).

Proposed approach would be unfeasible if one would need to set up all c_j''s and c_j'''s to individual values, therefore we would rely on existing particles templates, where either all those coefficients would take the same value or most of them would be equal to zero. Our approach defines c_j' and c_j'' as functions.

In order to test the proposed generalized approach we have implemented six distinctive types of particles, coming from the following algorithms: Standard PSO (SPSO), Fully-Informed PSO (FIPSO), Charged PSO (CPSO), Unified PSO (UPSO), Differential Evolution (DE). Remainder of this section presents how each approach fits within the proposed GPSO framework.

Standard Particle Swarm Optimization. SPSO particle acts according to the rules of PSO described in Section 2 with a local neighborhood topology. Therefore, the I function defining the neighborhood takes a following form:

$$I(j\text{th is } i\text{th neighbor}) = \begin{cases} 1 & |i-j| \leq k \\ 1 & |i-j| \geq |particles| - k \\ 0 & \sim \end{cases} \tag{7}$$

Particle changes its direction using l_{best} location. Therefore, all values of $c'j$'s and $c''j$'s are equal to 0 except the one corresponding to the particle with the best p_{best} value in the neighborhood.

$$c_j' = \begin{cases} 0 & f(p_{best}^j) > f(l_{best}) \\ U(o, c_{c_2}) & f(p_{best}^j) = f(l_{best}) \end{cases} \tag{8}$$

Fully-Informed Particle Swarm Optimization. FIPSO particle updates its velocity in another way than SPSO [14]. FIPSO tends to the location designated by all of its neighbors. All best solutions found so far by individual particles in neighborhood are considered. Appropriate weighing of solutions is applied. FIPSO particles utilize a complete neighborhood. Therefore the indicator function I is always equal to 1. The particle is parametrized with c, a single attraction coefficient. Individual c'_j's (and c_1) take the following value:

$$c'_j = \frac{\sum_{i=1}^{|particles|} p^j_{best} \cdot U[O, \frac{c}{|particles|}]}{\sum_{i=1}^{|particles|} U[0, \frac{c}{|particles|}]} \tag{9}$$

Charged Particle Swarm Optimization. CPSO particle has been created for the dynamic optimization problems [4] and is inspired by the model of an atom. CPSO recognizes two particle types: neutral and charged. The neutral particles behave like SPSO particles. Charged particles, have a special component added to the velocity update equation. A charged particle has an additional parameter q controlling the repulse:

$$c''_j = -\frac{q^2}{|x^i_t - x^j_t|^2} \tag{10}$$

Charged particles repulse each other, so an individual sub-swarms are formed (as imposed by the neighborhood), which might explore areas corresponding to different local optima.

Unified Particle Swarm Optimization. UPSO particle is a fusion of the local SPSO and the global SPSO [18]. The velocity update formula includes both l_{best} and g_{best} solutions. In order to express that unification of global and local variants of SPSO the I indicator function takes the following form:

$$I(j\text{th is } i\text{th neighbor}) = \begin{cases} 1 & |i-j| \leq k \\ 1 & |i-j| \geq |particles| - k \\ 1 & p^{(j)}_{best} \text{ is } g_{best} \\ 0 & \sim \end{cases} \tag{11}$$

Thus, there are two co-existing topologies of the neighborhood, which justifies the choice of the general formula for the GAPSO (cf. eq. (6)).

Differential Evolution within the GPSO framework. While Differential Evolution (DE) [21] is not considered to be a swarm intelligence algorithm its behavior might be also fitted within the proposed framework GPSO. The reason for that is the fact that within the DE (unlike other evolutionary approaches) we might track a single individual as it evolves, instead of being replaced by its offspring.

DE/best/1/bin configuration and DE/rand/bin configuration are somewhat similar to the PSO with a g_{best} and l_{best} approaches, respectfully. The most important differences between DE and PSO behavior are the fact, that:

- DE individual always moves from the best found position (p_{best} in PSO), while PSO particle maintains current position, regardless of its quality,
- DE individual draws the 'velocity' (i.e. difference vector) from the global distribution based on other individuals location, while PSO particle maintains its own velocity.

Therefore, DE individual i movement might be expressed in the following way:

$$x_{test}^{(i,t+1)} = Bin(\omega v + (p_{best} - x_{test}^{(i,t)}), g_{best}) \tag{12}$$

where v follows a probability distribution based on random individuals' locations p_{best}^{rand1} and p_{best}^{rand2}) and Bin is a binomial cross-over operator.

4 Adaptation scheme

Different particle types perform differently on various functions. Moreover, different phases exists during optimization process. Some particle types perform better at the beginning, some perform better at the end of algorithm. Optimal swarm composition should be designated in real-time. Swarm composition is modified by changing behavior of each particle. Principle of work is presented below.

The main idea is to promote particle types (behaviors) that are performing better than others. Adaptation is based on quality of success. This approach can be described as roulette with probabilities proportional to success measure.

Let's assume that we have P particle types. Each particle changes its behavior every N_a iterations. Behavior is randomly chosen according to determined list of probabilities (P probabilities are given corresponding to P particle types). Each particle has the same vector of probabilities. At the beginning all probabilities are set to $\frac{1}{P}$. Each N_a iterations probabilities vector is changing (adapting) according to the following scheme.

The average value of successes per each particle's type from the last N_a observations is determined. Value of success z_t^s in iteration t for particle s is presented in the following equation:

$$z_t^s = max(0, \frac{f(p_{best}^s) - f(x_t^s)}{f(p_{best}^s)}) \tag{13}$$

Let $swarm_p$ be subswarm of p type particles from whole swarm. It is necessary to take into account all particles from $swarm_p$. So the average success $\hat{z}_p$ of given $swarm_p$ is obtained from $S_p * N_a$ values, where S_p is the size of $swarm_p$. See the following equation:

$$\hat{z}_t^p = \frac{1}{S_p * N_a} * \sum_{t=T}^{T-N_a} \sum_{s \in swarm_p} z_t^s \tag{14}$$

This procedure produces P success values. Let us label them as $z_1, z_2, \ldots, z_P$. Let Z be sum of given success values: $Z = \sum_p^P z_p$. So required vector of probabilities is $[\frac{z_1}{Z}, \frac{z_2}{Z}, \ldots, \frac{z_P}{Z}]$. Better average success induces grater probability of assigning given behavior to each particle.

Special rule has been proposed: at least one particle for each behavior has to exists. It prevents behaviors for being excluded.

5 Experiment setup

In order to test idea of the GAPSO algorithm an environment has been implemented in Java[1]. It consists of individual particles behaviors, an adaptation scheme, a restart mechanism, hill-climbing local optimization procedure for polishing the achieved results, and a port to the test benchmark functions. Tests have been performed on a 24 noiseless test functions from BBOB 2017 benchmark[2] with the use of COmparing Continuous Optimizers (COCO) dedicated platform. Algorithms have been tested on 5D and 20D functions.

Table 1. Individual algorithms parameters.

Algorithm	Parameters Settings	Reference
SPSO	$\omega : 0.9; c1, c2 : 1.2$	[5]
CPSO	$\omega : 0.9; c1, c2 : 1.2$	[4]
FIPSO	$\omega : 0.9; c : 4.5$	[14]
UPSO	$\omega : 0.9; c1, c2 : 1.2, u : 0.5$	[18]
DE	$crossProb : 0.5; varF : 1.4$	[21]

Table 2. Framework parameters

Parameter	Value
swarm size (S)	30
number of neighbors (k)	5
generations (G)	10^6
number of PSO types (P)	5
generations to adapt (N_a)	10
generations to restart particle (N_{rp})	15
generations to restart swarm (N_{rs})	200

Parameters. General framework setup has been tuned on a small number of initial experiments. While the parameters of individual optimization agents has been chosen on the state-of-the-art literature. All parameters values are presented in Tables 1 and 2. Five types of particle's behavior have been considered within the GAPSO framework, and as individual basic algorithms for comparison: SPSO, CPSO, UPSO, FIPSO, and DE.

Restarts. In order to fully utilize the algorithms' potential within each of the tested methods a particle is restarted if for N_{rp} iterations at least one of these 2 conditions persisted: (a) particle is its best neighbor, (b) particle has low velocity

[1] https://bitbucket.org/pl-edu-pw-mini-optimization/corpoalgorithm
[2] http://coco.gforge.inria.fr/

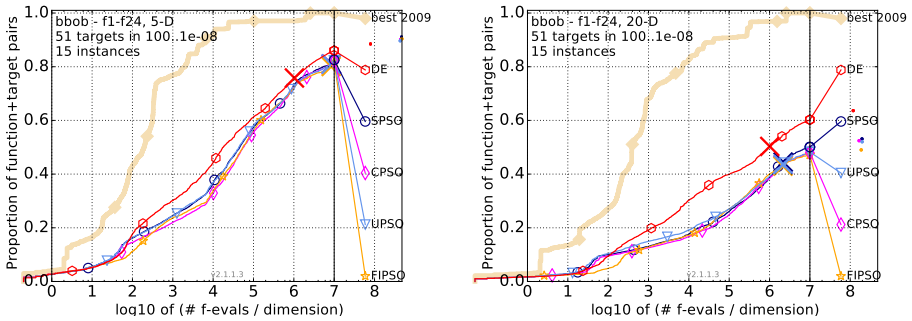

Fig. 1. Comparison of individual algorithms performance for all functions in 5 and 20 dimensions.

(sum of squares of velocities in each direction is smaller than 1). Additionally, the whole swarm is restarted (each particle that belongs to it is restarted), if value of best found solution has not changed since $N_{rs} \cdot D$, where D is dimension of function being optimized.

Local optimization. Finally (both in GAPS and individual approaches), before swarm restart and after the last iteration of the population based algorithms a local *hill-climbing* algorithm is used for $1000D$ evaluations, initialized with the best found solution.

6 Results

Results of the experiments are presented on the figures generated within BBOB test framework, showing percentage of optimization targets achieved on a log scale of objective function evaluations. Experiments were carried out on 24 5- and 20-dimensional benchmark functions, which are widely adopted in global optimization algorithms.

Left part of Figure 1 shows efficiency of 5 algorithms used in GAPSO but tested independently for 5-dimension functions. It can be observed that DE is coinciding to optimum faster than others. Efficiency of DE is more noticeable when 20-dimensions functions are evaluated (right part of Figure 1). Important observation for further experiments is performance of FIPSO which is the worst algorithm according to given measures. In next experiments DE, FIPSO and proposed by authors GAPSO will be compared.

Subsequent charts (see Figure 2) correspond to experiments carried out on selected algorithms with specified functions. In particular cases, differences in the assessment of the effectiveness of algorithms can be observed. Left part of Figure 2 shows advantage of DE algorithm in optimization 5-dimension functions f19-14 (functions with high conditioning and unimodal). Another case is

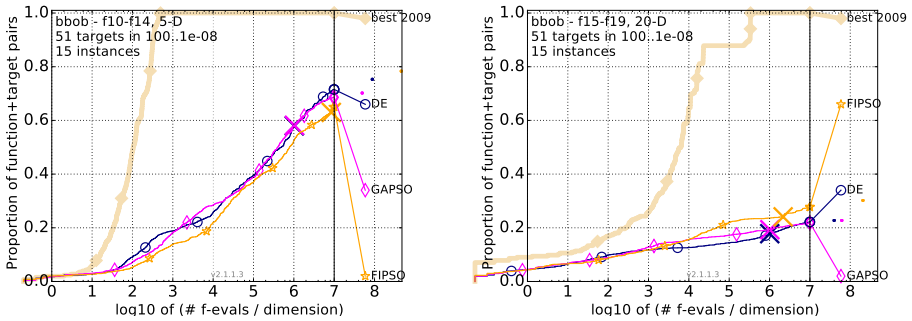

Fig. 2. Comparison of the best (DE) and the worst (FIPSO) individual algorithms with GAPSO for functions with high conditioning and unimodal in 5D (top) and multi-modal functions with adequate global structure in 20D (right).

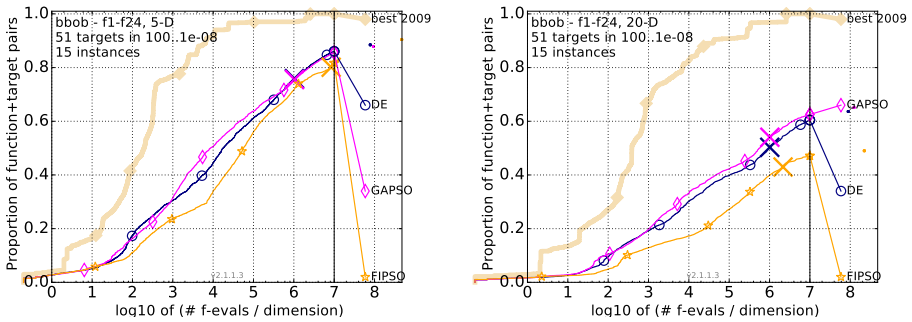

Fig. 3. GAPSO performance compared with the best (DE) and the worst (FIPSO) individual algorithms for all functions in 5D and 20D.

show in right part of Figure 2 where FIPSO algorithm for 20-dimension functions f15-f19 (multi-modal with adequate global structure) is performing best. Proposed GAPSO algorithm maintains level of performance to bests algorithms. The last experiment will be attempt of generalization of results associated with all functions for better evaluation of GAPSO performance. Results are shown in Figure 3. Experiments were conducted on all functions in 5 dimensions and 20 dimensions. Averaged results indicate that GAPSO is the most effective algorithm. Figures 4 and 5 present comparison of average number of particle's behaviors and efficiency of homogeneous swarms for two selected functions. For Rosenbrock's function (Figure 4) DE swarm is significantly better than other kind of swarms and GAPSO algorithm adaptation method leads to greater number of DE particles in swarm. On the other hand, in case when for instance homogeneous DE swarm performance is the worst from among all the other PSO kinds

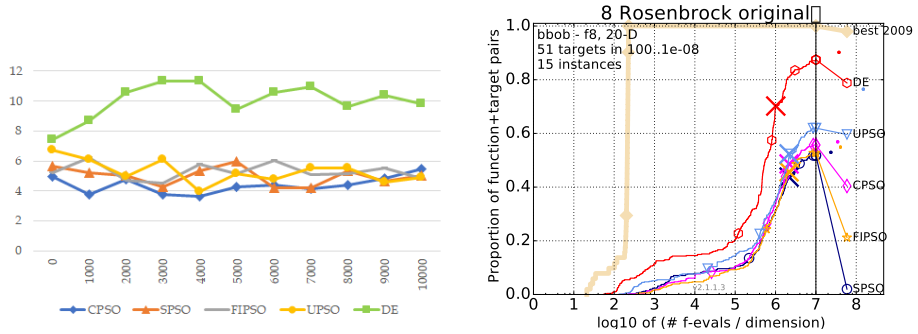

Fig. 4. Average number of particles kinds in swarm for Rosenbrock function compared with individual algorithms performance.

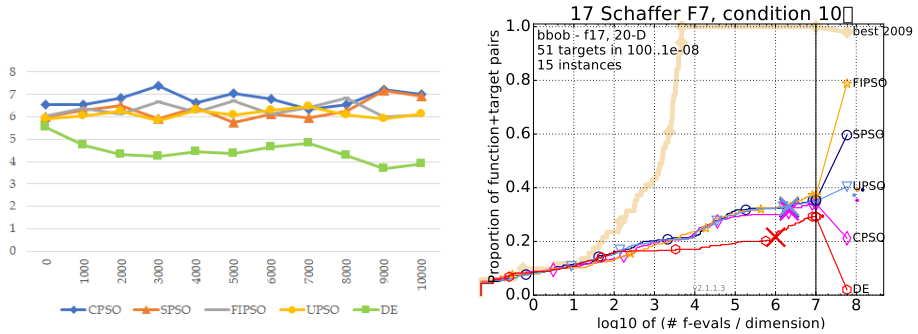

Fig. 5. Average number of particles kinds in swarm for Schaffer function compared with individual algorithms performance.

(see Figure 5) GAPSO swarm contains significantly lower number of DE particles. It indicates that proposed adaptation methods control swarm composition according to particular optimization function and it depends on individual PSO performance on this function. Results for all particles behaviors except DE are quite similar, so there is no noticeable difference between number of particles of particular kind.

For the sake of space limits we provide only aggregated results in Table 3. Detailed outcomes can be found in [1]. GAPSO obtained best results (in terms of number of function evaluation) for 10 (5D) and 8 (20D) functions (out of 24). 7 of those results are statistically significantly better. None of the other algorithms were statistically significantly better than GAPSO for any function. These results show that proposed algorithm not only adapted to reach results as good as best particle types, but also has ability to outperform its components.

Table 3. Aggregated results for 15 independent runs on 24 noiseless test functions from BBOB 2017 benchmark. Number of functions for which given algorithm yielded best results (in term of average number of function evaluations) is presented in *best* columns. Numbers in brackets show how many of results are statistically significantly better according to the rank-sum test when compared to all other algorithms of the table with $p = 0.05$. *Target reached* is the number of trials that reached the final target: $f_{opt} + 10^8$.

	5D		20D	
algorithm	best	target reached	best	target reached
CPSO	2 (0)	217	1 (0)	85
SPSO	1 (0)	221	2 (0)	91
FIPSO	2 (0)	211	4 (0)	83
UPSO	3 (0)	214	3 (0)	87
DE	6 (0)	173	4 (1)	117
GAPSO	10 (0)	172	8 (7)	120

Furthermore, algorithm stability depending on different initial behavior probabilities vector was examined. 7 types of vectors were considered: uniform (each behavior with the same probability), randomly generated vector and 5 vectors (one per each behavior) with probability equals 1 to one behavior and 0 for all other. Standard deviations obtained through all approaches on benchmark functions were not significantly different than standard deviations for each approach separately. For all above options just after about 100 generations (10 adaptation procedures) numbers of particles with particular behaviors are nearly the same. It shows proposed method's ability to gaining equilibrium - optimal (from the algorithm's perspective) behaviors numbers independently of the initial state (probabilities vector).

Initially, an experiment including also an Orthogonal Learning PSO (OLPSO) [24], as an behavior in GAPSO was also made. However, because of slow OLPSO convergence and chosen adaptation scheme, the obtained results were not satisfying [1].

7 Conclusions and future work

The proposed generalized view on the Particle Swarm Optimization made it possible to introduce various types of predefined behaviors and neighborhood topologies within a single algorithm. Including an adaptation scheme improved the overall performance over both DE individuals and PSO particles types on the test set of 24 quality functions. The adaptation scheme correctly promoted behaviors (particles) which performed well on a given type of a function. It remains to be seen what other types of behaviors could be successfully brought into the GAPSO framework.

Our future research activities shall concentrate on testing more types of particles and detailed analysis about their cooperation by observing interactions between different particles behaviors in each generation. It would be especially

evaluate a performance of some quasi-Newton method, brought into the framework of GPSO, as it could utilize the already gathered samples of the quality (fitness) function. Furthermore, other adaptation and evaluation schemes can be considered and compared with proposed method.

References

1. Gapso detailed results. http://pages.mini.pw.edu.pl/ zychowskia/gapso, accessed: 2018-03-30
2. Araújo, T.D.F., Uturbey, W.: Performance assessment of PSO, DE and hybrid PSODE algorithms when applied to the dispatch of generation and demand. International Journal of Electrical Power & Energy Systems 47(1), 205–217 (may 2013)
3. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary computation in dynamic and uncertain environments, pp. 29–49. Springer (2007)
4. Blackwell, T.M., Bentley, P.J.: Dynamic search with charged swarms. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. pp. 19–26 (2002)
5. Clerc, M.: Standard particle swarm optimisation (2012)
6. Das, S., Abraham, A., Konar, A.: Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. In: Advances of Computational Intelligence in Industrial Systems, pp. 1–38. Springer-Verlag (2008)
7. Epitropakis, M., Plagianakos, V., Vrahatis, M.: Evolving cognitive and social experience in Particle Swarm Optimization through Differential Evolution: A hybrid approach. Information Sciences 216, 50–92 (dec 2012)
8. Harrison, K.R., Ombuki-Berman, B.M., Engelbrecht, A.P.: Optimal parameter regions for particle swarm optimization algorithms. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 349–356. IEEE (2017)
9. Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer and its adaptive variant. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35(6), 1272–1282 (2005)
10. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. IV pp. 1942–1948 (1995)
11. Koh, B.I., George, A.D., Haftka, R.T., Fregly, B.J.: Parallel asynchronous particle swarm optimization. International Journal for Numerical Methods in Engineering 67(4), 578–595 (2006)
12. Köppel, P., Sandner, D.: Synergy by Diversity: Real Life Examples of Cultural Diversity in Corporation. Bertelsmann-Stiftung (2008)
13. Liu, H., Cai, Z., Wang, Y.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Applied Soft Computing 10(2), 629–640 (mar 2010)
14. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Tran. on Evolutionary Computation 8(3), 204–210 (2004)
15. Nepomuceno, F.V., Engelbrecht, A.P.: A self-adaptive heterogeneous pso inspired by ants. In: International Conference on Swarm Intelligence. pp. 188–195. Springer (2012)

16. Niu, B., Li, L.: A Novel PSO-DE-Based Hybrid Algorithm for Global Optimization. In: Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, pp. 156–163. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
17. Okulewicz, M.: Finding an optimal team. In: FedCSIS Position Papers. pp. 205–210 (2016)
18. Parsopoulos, K.E., Vrahatis, M.N.: Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems. In: International Conference on Natural Computation. pp. 582–591. Springer (2005)
19. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence 1(1), 33–57 (2007)
20. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. Proceedings of Evolutionary Programming VII (EP98) pp. 591–600 (1998)
21. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (1997)
22. Thangaraj, R., Pant, M., Abraham, A., Bouvry, P.: Particle swarm optimization: Hybridization perspectives and experimental illustrations. Applied Mathematics and Computation 217(12), 5208–5226 (feb 2011)
23. Wen-Jun Zhang, Xiao-Feng Xie: DEPSO: hybrid particle swarm with differential evolution operator. In: SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483). vol. 4, pp. 3816–3821. IEEE (2003)
24. Zhan, Z.H., Zhang, J., Li, Y., Shi, Y.H.: Orthogonal learning particle swarm optimization. IEEE transactions on evolutionary computation 15(6), 832–847 (2011)
25. Zhang, C., Ning, J., Lu, S., Ouyang, D., Ding, T.: A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. Operations Research Letters 37(2), 117–122 (mar 2009)
26. Zhi-Hui Zhan, Jun Zhang, Yun Li, Chung, H.H.: Adaptive Particle Swarm Optimization. IEEE Tran. on Systems, Man, and Cybernetics, Part B (Cybernetics) 39(6), 1362–1381 (2009)
27. Zhou, Y., Tan, Y.: GPU-based parallel particle swarm optimization. In: 2009 IEEE Congress on Evolutionary Computation. pp. 1493–1500. IEEE (2009)
28. Zhuang, T., Li, Q., Guo, Q., Wang, X.: A two-stage particle swarm optimizer. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). vol. 2, pp. 557–563. IEEE (2008)