Generalized Self-adapting Particle Swarm Optimization Algorithm

Mateusz Uliński, Adam Żychowski, Michał Okulewicz(✉), Mateusz Zaborski, and Hubert Kordulewski

Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland
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 and proves to be better than any of the algorithms its incorporating.

Keywords: Particle Swarm Optimization
Self-adapting metaheuristics

1 Introduction

Since its introduction [9] and subsequent modifications [4,18] Particle Swarm Optimization (PSO) algorithm has attracted many researchers by its simplicity of implementation and easiness of parallelization [13]. PSO has currently a several standard approaches [4], multiple parameter settings considered to be optimal [7] and successful specialized approaches [3]. PSO have also been tried with various topologies [8,17], and unification [16] 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 [10] and our adaptive approach (GAPSO) seeks an emergence of such a behavior within a heterogeneous swarm.

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 hybridization with PSO. In Sect. 3 general overview of

A. Auger et al. (Eds.): PPSN 2018, LNCS 11101, pp. 29–40, 2018.
https://doi.org/10.1007/978-3-319-99253-2_3

the system's construction is provided. Section 4 describes adaptation scheme 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 Sect. 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 paragraphs 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. Please bear in mind, that in all methods we shall be considering the optimization problem to be a minimization problem.

Particle Swarm Optimization. PSO is an iterative global optimization metaheuristic method utilizing the ideas of swarm intelligence [9,18]. 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 [4,18]) 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 [19]. 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 results, easiness of implementation and parallelization, known parameters values ensuring theoretical convergence) it still needs to have its parameters tuned in order to balance its exploration vs. exploitation behavior [24]. In order to overcome those limitations a two–stage algorithm has been proposed [24]. 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 [23], 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 adapts its inertia (ω), cognitive (c_1) and social (c_2) coefficients accordingly. Finally, a heterogeneous self-adapting PSO has been proposed [14], but its pool of available behaviors has been limited only to 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 [1,20]. 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 [5,21]. Another approach [22], consists of maintaining both algorithms in parallel and introducing an information sharing scheme between them with additional random search procedure. PSO and DE can also be combined in a sequential way [6,11]. 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 [15], by generalizing PSO velocity update formula (Eq. (2)) into a following form (with

I being and indicator function and $N_k(i\text{th})$ being a kth neighborhood of ith particle):

$$\begin{aligned}\mathbf{v}_{t+1}^i = &\omega \cdot \mathbf{v}_t^i + c_1 \cdot (\mathbf{p}_{best}^i - \mathbf{x}_t^i) \\ &+ \sum_{k=1}^{|\mathcal{N}|} \sum_{j=1, j\neq i}^{|particles|} I(j\text{th} \in N_k(i\text{th})) c'_{j,k} \cdot (\mathbf{p}_{best}^j - \mathbf{x}_t^i) \\ &+ \sum_{k=1}^{\mathcal{N}} \sum_{j=1, j\neq i}^{|particles|} I(j\text{th} \in N_k(i\text{th})) c''_{j,k} \cdot (\mathbf{x}_t^j - \mathbf{x}_t^i)\end{aligned} \tag{6}$$

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 $c'_{j,k}$'s and $c''_{j,k}$'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 views $c'_{j,k}$ and $c''_{j,k}$ as functions. In most cases second index of c coefficients would be omitted, due to the fact that only a single neighborhood is considered.

In order to test the proposed generalized approach we have implemented five 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 Sect. 2 with a local neighborhood topology (with $size \in \mathbb{Z}_+$ being its parameter). Therefore, the I function defining the neighborhood takes a following form:

$$I_{SPSO}(j\text{th} \in N(i\text{th})) = \begin{cases} 1 & |i-j| \leq size \\ 1 & |i-j| \geq |particles| - size \\ 0 & \sim \end{cases} \tag{7}$$

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.

$$c'_j = \begin{cases} 0 & f(\mathbf{p}_{best}^j) > f(\mathbf{l}_{best}) \\ X \sim U(o, c_2) & f(\mathbf{p}_{best}^j) = f(\mathbf{l}_{best}) \end{cases} \tag{8}$$

Fully-Informed Particle Swarm Optimization. FIPSO particle [12] steers its velocity to the location designated by all of its neighbors. All the best solutions found so far by the individual particles are considered with weights $\mathcal{W}$ corresponding to the relative quality of those solutions. FIPSO particles utilize a complete neighborhood. Therefore, the indicator function I_{FIPSO} is equal to 1. The FIPSO particle is parametrized with a single value of an attraction coefficient c. Individual c'_j's (and c_1) follow the uniform distribution:

$$c'_j \sim U\left[0, \frac{c \cdot \mathcal{W}(f(\mathbf{p}^j_{best}))}{|particles|}\right] \tag{9}$$

Charged Particle Swarm Optimization. CPSO particle has been created for the dynamic optimization problems [3] 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. An ith charged particle has an additional parameter q controlling its repulse from other charged particles:

$$c''_{j,2} = -\frac{q^2}{||\mathbf{x}^i_t - \mathbf{x}^j_t||_2} \tag{10}$$

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 [16]. 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:

$$I_{UPSO}(j\text{th} \in N_k(i\text{th})) = \begin{cases} I_{SPSO} & k = 1 \\ 1 & \mathbf{p}^j_{best} \text{ is } \mathbf{g}_{best} \wedge k = 2 \\ 0 & \sim \end{cases} \tag{11}$$

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

Differential Evolution within the GPSO Framework. While Differential Evolution (DE) [19] 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/1/bin configurations 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:

$$\mathbf{x}_{test}^{(i,t+1)} = Bin(\omega\mathbf{v} + (\mathbf{p}_{best} - \mathbf{x}_{test}^{(i,t)}), \mathbf{g}_{best}) \tag{12}$$

where v follows a probability distribution based on random individuals' locations $\mathbf{p}_{best}^{rand1}$ and $\mathbf{p}_{best}^{rand2}$) 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 optimization algorithm's execution. Therefore, optimal swarm composition within GPSO framework should be designated in real-time. Swarm composition is modified by switching the behaviors of particles. Principle of work for adaptation scheme forming the Generalized Self-Adapting Particle Swarm Optimization (GAPSO) is presented below.

The main idea is to promote particle types that are performing better than others. Adaptation is based on the quality of success. The adaption utilizes roulette selection approach 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 chosen according to a list of probabilities (each corresponding to one of P particles' 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:

$$z_t^s = max(0, \frac{f(\mathbf{p}_{best}^s) - f(\mathbf{x}_t^s)}{f(\mathbf{p}_{best}^s)}) \tag{13}$$

Let $swarm_p$ be a set of p type particles from whole swarm. 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. On top of the described approach an additional special rule is applied: at least one particle for each behavior has to exists. This rule prevents behaviors for being excluded from further consideration, as they might be needed in a latter phase of optimization process.

5 Experiment Setup

The GAPSO algorithm has been implemented in Java[1]. The project 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 24 noiseless 5D and 20D test functions from BBOB 2017 benchmark[2].

Table 1. Individual algorithms parameters.

Algorithm	Parameters settings	Reference
SPSO	$\omega : 0.9; c1, c2 : 1.2$	[4]
CPSO	$\omega : 0.9; c1, c2 : 1.2$	[3]
FIPSO	$\omega : 0.9; c : 4.5$	[12]
UPSO	$\omega : 0.9; c1, c2 : 1.2, u : 0.5$	[16]
DE	$crossProb : 0.5; varF : 1.4$	[19]

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 GAPSO framework setup has been tuned on a small number of initial experiments, while the parameters of the individual optimization agents have been chosen according to the literature. The parameter values are presented in Tables 1 and 2.

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 (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 GAPSO 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.

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

6 Results

Results of the experiments are presented on the figures generated within BBOB 2017 test framework, showing ECDF plots of optimization targets achieved on a log scale of objective function evaluations.

Left subplot in Fig. 1 shows efficiency of 5 individual algorithms used in GAPSO tested independently for 5D functions. It can be observed that DE is coinciding to optimum faster than each of the PSO approaches. Advantage of the DE is even more evident for 20D functions (right subplot in Fig. 1).

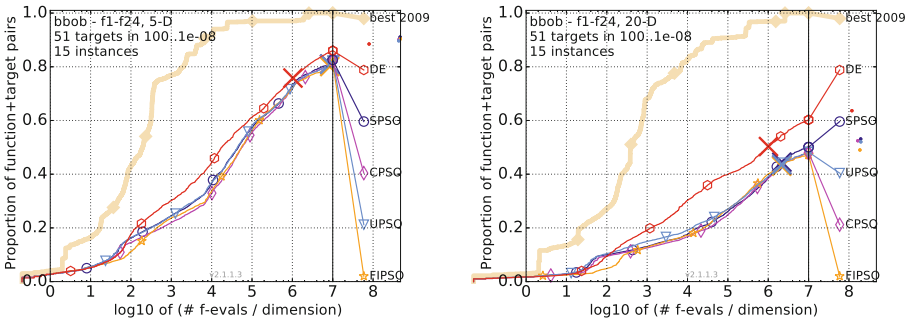

Fig. 1. Comparison of individual algorithms performance for all functions in 5 and 20 dimensions.

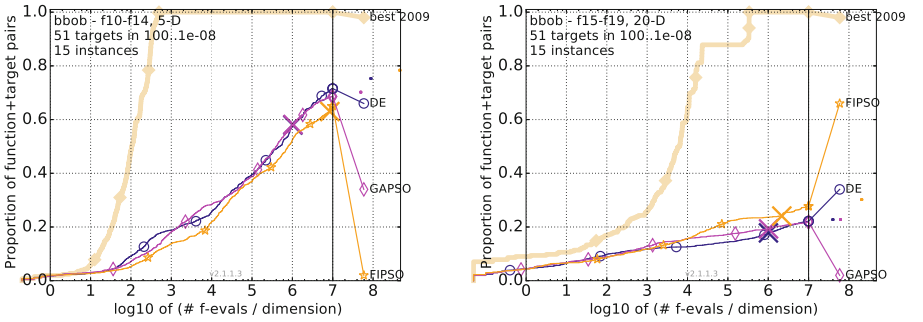

Fig. 2. Comparison of the best (DE) and the worst (FIPSO) individual algorithms with GAPSO for functions with high conditioning and unimodal in 5D (top) and multimodal functions with adequate global structure in 20D (right).

Subsequent charts (see Fig. 2) correspond to experiments carried out on selected algorithms with specified functions. In particular cases, differences in the effectiveness of algorithms can be observed. Left subplot in Fig. 2 shows advantage of DE algorithm in optimizing 5D unimodal functions with high conditioning. While another case, shown in right sublot in Fig. 2, presents FIPSO

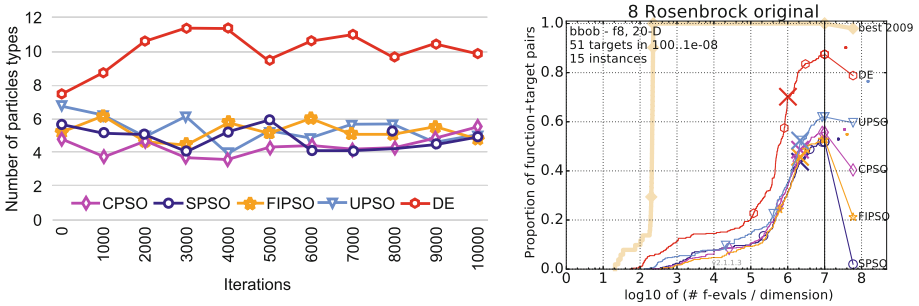

Fig. 3. Average number of particles types in swarm compared with ECDF plot of individual algorithms performance for 20D Rosenbrock function.

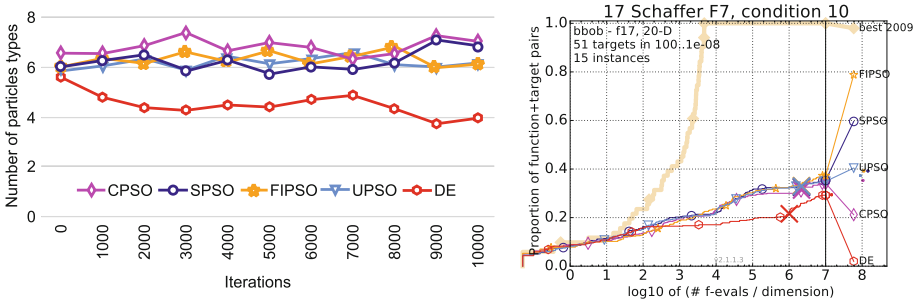

Fig. 4. Average number of particles types in swarm compared with ECDF plot of individual algorithms performance for 20D Schaffer function.

as an algorithm performing best for 20D multi-modal functions with adequate global structure. It can be observed that the proposed GAPSO algorithm remains competitive with both "clean" approaches.

Figures 3 and 4 present comparison of average number of particle's behaviors and efficiency of homogeneous swarms for two selected functions. For Rosenbrock's function (Fig. 3) DE swarm is significantly better than other kind of swarms and GAPSO algorithm adaptation method leads to greater number of DE particles in swarm. In the case when plain DE performance is worse than all the PSO-based approaches (see Fig. 4) GAPSO swarm contains significantly lower number of DE particles. It indicates that the proposed adaptation method controls the swarm composition according to the particular optimization function. It also can be observed that the performance of various PSO approaches is similar, and there is no noticeable difference between number of particles of particular kind.

Last experiment presents the overall effectiveness of the GAPSO performance on the whole set of 5D and 20D benchmark functions. Figure 5 presents the GAPSO results against the best (DE) and worst (FIPSO) performing algorithms. Results indicate that GAPSO has come out as a more effective approach, even though its adaptation has been performed during the optimization, and not beforehand.

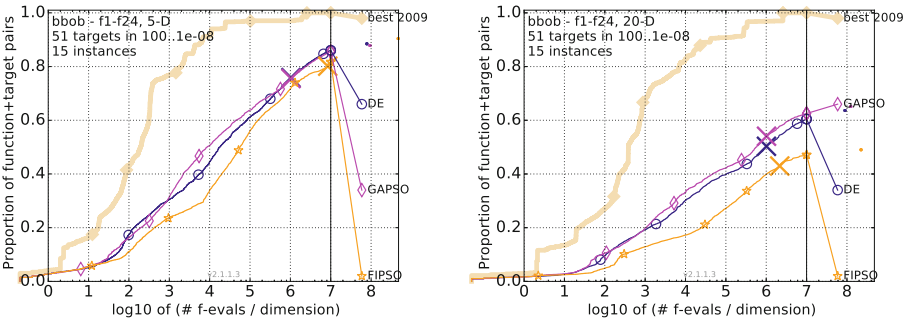

Fig. 5. GAPSO performance compared with the best (DE) and the worst (FIPSO) individual algorithms for all functions in 5D and 20D.

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

Due to space limitations, Table 3 provides only aggregated results[3]. GAPSO obtained best results (in terms of number of function evaluation) for 10 (5D) and 8 (20D) functions (out of 24), with 7 of those results being statistically significantly better than individual approaches. 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 the best individual particles' types, but also has the ability to outperform them.

Furthermore, GAPSO stability other a different initial behavior probabilities vectors 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

[3] Detailed outcomes are available at http://pages.mini.pw.edu.pl/~zychowskia/gapso.

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 were nearly the same. It shows, that the proposed method's ability to gaining equilibrium - optimal behaviors (from the algorithm's perspective) is independent of the initial state of behavior probabilities vector.

7 Conclusions and Future Work

The proposed generalized GPSO view on the Particle Swarm Optimization made it possible to introduce various types of predefined behaviors and neighborhood topologies within a single optimization algorithm. Including an adaptation scheme in GAPSO approach allowed to improve the overall performance over both DE individuals and PSO particles types on the test set of 24 quality functions. While the proposed approach remains inferior to algorithms such as CMA-ES [2], the adaptation scheme correctly promoted behaviors (particles) performing well on a given type of a function. It remains to be seen if other types of basic behaviors could be successfully brought into the GAPSO framework and compete with the state-of-the-art optimization algorithms.

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 interesting to 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. Araújo, T.D.F., Uturbey, W.: Performance assessment of PSO, DE and hybrid PSODE algorithms when applied to the dispatch of generation and demand. Int. J. Electrical Power Energy Syst. **47**(1), 205–217 (2013)
2. Beyer, H.G., Sendhoff, B.: Simplify your covariance matrix adaptation evolution strategy. IEEE Trans. Evol. Comput. **21**(5), 746–759 (2017)
3. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 29–49. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-49774-5_2
4. Clerc, M.: Standard particle swarm optimisation (2012)
5. Das, S., Abraham, A., Konar, A.: Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. Advances of Computational Intelligence in Industrial Systems. SCI, vol. 116, pp. 1–38. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78297-1_1
6. Epitropakis, M., Plagianakos, V., Vrahatis, M.: Evolving cognitive and social experience in particle swarm optimization through differential evolution: a hybrid approach. Inf. Sci. **216**, 50–92 (2012)

7. 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)
8. Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer and its adaptive variant. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **35**(6), 1272–1282 (2005)
9. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948 (1995)
10. Köppel, P., Sandner, D.: Synergy by Diversity: Real Life Examples of Cultural Diversity in Corporation. Bertelsmann-Stiftung, Gütersloh (2008)
11. Liu, H., Cai, Z., Wang, Y.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl. Soft Comput. **10**(2), 629–640 (2010)
12. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. IEEE Tran. Evol. Comput. **8**(3), 204–210 (2004)
13. Mussi, L., Daolio, F., Cagnoni, S.: Evaluation of parallel particle swarm optimization algorithms within the CUDA architecture. Inf. Sci. **181**(20), 4642–4657 (2011)
14. Nepomuceno, F.V., Engelbrecht, A.P.: A self-adaptive heterogeneous pso for real-parameter optimization. In: 2013 IEEE Congress on Evolutionary Computation, pp. 361–368. IEEE, June 2013
15. Okulewicz, M.: Finding an optimal team. In: FedCSIS Position Papers, pp. 205–210 (2016)
16. Parsopoulos, K.E., Vrahatis, M.N.: Unified particle swarm optimization for solving constrained engineering optimization problems. In: Wang, L., Chen, K., Ong, Y.S. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 582–591. Springer, Heidelberg (2005). https://doi.org/10.1007/11539902_71
17. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intell. **1**(1), 33–57 (2007)
18. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0040810
19. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
20. Thangaraj, R., Pant, M., Abraham, A., Bouvry, P.: Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl. Math. Comput. **217**(12), 5208–5226 (2011)
21. Zhang, W.J., Xie, X.F.: DEPSO: hybrid particle swarm with differential evolution operator. In: SMC 2003 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)
22. Zhang, C., Ning, J., Lu, S., Ouyang, D., Ding, T.: A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. Oper. Res. Lett. **37**(2), 117–122 (2009)
23. Zhan, Z.-H., Zhang, J., Li, Y., Chung, H.H.: Adaptive particle swarm optimization. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **39**(6), 1362–1381 (2009)
24. 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, June 2008