Population Size Reduction in Particle Swarm Optimization Using Product Graphs

Iztok Fister Jr., Aleksandra Tepeh, Janez Brest and Iztok Fister

Abstract Purpose of this paper is to introduce a population size reduction in particle swarm optimization algorithm, where the reduction is performed by selecting two particles (also donor particles) randomly and replacing these by a new particle with elements determined from a set of pair values obtained by the Cartesian product of both donor particles for each particular element randomly. Average values of each pair values from the donor particles are calculated for corresponding elements of the new particle. The proposed PSOGP was applied on a benchmark function suite consisted of four well-known functions and compared with the original PSO algorithm. The results are very promising and show the potential of the proposed idea.

Keywords Product graphs · Optimization · Particle swarm optimization · Population size reduction

1 Introduction

In the past decades, nature-inspired algorithms gained immeasurable attention from scientific public. Nature-inspired algorithms are a very efficient tool for solving the different kinds of optimization tasks. In line with this, many various nature-inspired algorithms were proposed so far. Since the number of these algorithms became so huge, there is impossible to classify them all. Anyway, few years ago, one proposed taxonomy was developed in [5]. In this taxonomy, nature-inspired algorithms were

I. Fister Jr.(✉) · A. Tepeh · J. Brest · I. Fister
Faculty of Electrical Engineering and Computer Science,
University of Maribor, Smetanova Ul. 17, 2000 Maribor, Slovenia
e-mail: iztok.fister1@um.si

A. Tepeh
e-mail: aleksandra.tepeh@um.si

J. Brest
e-mail: janez.brest@um.si

I. Fister
e-mail: iztok.fister@um.si

R. Matoušek (ed.), *Mendel 2015*, Advances in Intelligent Systems
and Computing 378, DOI 10.1007/978-3-319-19824-8_7

divided into four groups based on their primary inspirations. These four groups can be summarized as follows:

- swarm intelligence (SI) based algorithms,
- bio-inspired algorithms (without SI-based algorithm),
- physics and chemistry based algorithms,
- other algorithms (e.g. inspiration of which have roots in sociology, history and some even in sport)

Currently, SI-based algorithms seem to be one of the more popular in research area. They were used in solving discrete and numerical optimization problems as well as practical applications, e.g., in [7, 11, 13, 17–19]. According to the NFL theorem [16], average performances of different algorithm families are the same when they are applied to all classes of problems. Therefore, a lot of special techniques were developed in order to improve the results by solving the different problems [4, 6, 14, 15].

This paper proposes a population size reduction borrowed from Brest and Maučec [2], where authors justified an advantage of this mechanism as follows. A diversity of population is high at the start of optimization. The diversity becomes lower when a search process progresses and stepwise directs itself in promising regions of the search space. Thus, the search process can continue with a smaller number of population members because of wasting the number of fitness function evaluations. Typically, the population size is reduced proportionally to the number of generations.

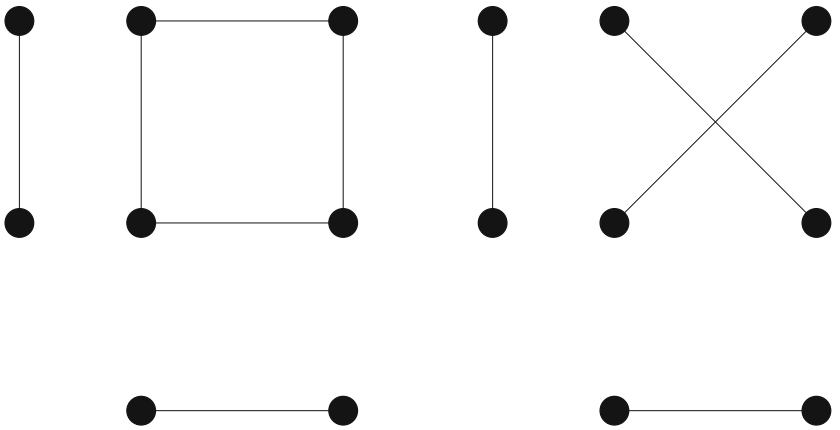
Primarily, a candidate for elimination must be determined by the reduction. This paper proposes the PSOPG algorithm that selects two donor particles from the current population randomly for replacing them with the new particle. Here, these two donor particles are treated as sets of nodes $V(G)$ and $V(H)$ that are combined using the Cartesian product of these sets, $V(G) \times V(H)$. The new particle is calculated with elements as an average of pairs of elements (u_i, v_j), where the node v_j is selected randomly. The PSOPG algorithm was tested on a benchmark of four well-known functions from the literature. Interestingly, the PSOPG outperformed the results of the original PSO algorithm and showed the potential for the future work, where different type of graph products can be applied.

The remainder of the paper is structured as follows: in Sect. 2 a basic overview of graph products is given, and Sect. 3 presents a short overview of a swarm intelligence and particle swarm optimization. In Sect. 4 the new algorithm is proposed. Section 5 presents experiments and results, while the concluding section shows directions for possible future work.

2 Product Graphs

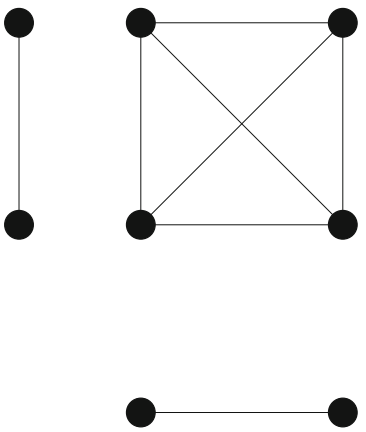
A graph product of two graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ is a new graph whose node set is the Cartesian product of sets $V(G) \times V(H)$ and where for any two nodes (g, h) and (g', h') in the product, the adjacency of those two nodes is determined entirely by the adjacency (or equality, or non-adjacency) of g and g',

and that of h and h'. It turns out that there are 256 different types of graph products that can be defined. Among products that have been widely investigated and have many significant applications are the Cartesian product, the direct product, the strong product and the lexicographic product (the terminology is not quite standardized, so these products may actually be referred to by different names, however, we adopt the terminology from [8], where a comprehensive overview of graph products can be found).



(a) Cartesian product

(b) Direct product



(c) Strong product

Fig. 1 Graph products

For the mentioned (so called standard) products, edges are defined as follows. In the *Cartesian product* $G \square H$ of graphs G and H two nodes (g, h) and (g', h') are adjacent when ($gg' \in E(G)$ and $h = h'$) or ($g = g'$ and $hh' \in E(H)$). In the *direct product* $G \times H$ nodes (g, h) and (g', h') are adjacent when $gg' \in E(G)$ and $hh' \in E(H)$. In the *strong product* $G \boxtimes H$ nodes (g, h) and (g', h') are adjacent whenever ($gg' \in E(G)$ and $h = h'$) or ($g = g'$ and $hh' \in E(H)$) or ($gg' \in E(G)$ and $hh' \in E(H)$). Note that $E(G \boxtimes H) = E(G \square H) \cup E(G \times H)$ and that the notation of these three products comes from multiplying two copies of K_2 (complete graph on two nodes), see Fig. 1. Finally, in the *lexicographic product* $G[H]$ nodes (g, h) and (g', h') are adjacent if either $gg' \in E(G)$ or ($g = g'$ and $hh' \in E(H)$).

It follows immediately from the definitions that the Cartesian, the direct and the strong product are commutative in the sense that the graph $G * H$ is isomorphic to $H * G$, where $*$ stands for any one of the three mentioned products. Note that the lexicographic product is not commutative (we want to emphasize this fact with the notation $G[H]$, although in the literature the notation $G \circ H$ is often used for this product). However, all four standard products are associative, i.e. for given graphs G_1, G_2, G_3 the graphs $(G_1 * G_2) * G_3$ and $G_1 * (G_2 * G_3)$ are isomorphic (here $*$ stands for any one of the four standard products). Associativity of all four products allows the easy extension of these products to arbitrarily many factors.

3 The Particle Swarm Optimization

The particle swarm optimization (PSO) is one of the older members of SI-based algorithm family. It was proposed by Kennedy and Eberhart [12] in 1995. The PSO algorithm mimics behavior of bird flocks. Therefore, it is a population-based algorithm, where the population consists of Np particles. Each particle represents a solution of the problem to be solved. Typically, the solution is represented as a vector $\mathbf{x}_i = \{x_{i,j}\}$ for $j = 1, \ldots, n$ with real-valued elements $x_{i,j} \in \mathbf{R}$, where n determines a dimensionality of the problem.

The pseudo-code of the original PSO algorithm is illustrated in Algorithm 1 that consists of the following components:

- initialization of population randomly (function init_particles),
- fitness function evaluation (function evaluate_the_new_solution),
- selection of the local best solution (lines 8–10),
- selection of the global best solution (lines 11–13),
- generation of the new solution (function generate_new_solution).

Additionally, the termination condition (function termination_condition_not_meet) needs to be defined in order to complete operating the PSO algorithm. Typically, the maximum number of fitness function evaluations *MAX_FE* is used as a termination condition. Interestingly, the PSO algorithm works with two sets of particles because it manages the local best solutions $\mathbf{p}_i^{(t)}$ for each particle $\mathbf{x}_i^{(t)}$. Moreover, the best solution in the population $\mathbf{g}^{(t)}$ is determined in each generation. The new particle position is generated as follows:

$$\begin{aligned} \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^{(t)} + C_1 U(0,1)(\mathbf{p}_i^{(t)} - \mathbf{x}_i^{(t)}) + C_2 U(0,1)(\mathbf{g}^{(t)} - \mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \end{aligned} \tag{1}$$

where $U(0,1)$ denotes a random value drawn from the uniform distribution, and C_1 and C_2 are learning factors.

4 The Proposed PSOPG Algorithm

The proposed algorithm PSOPG implements a population size reduction feature, where the population size Np is stepwise reduced by increasing the generation number t. This reduction is controlled by an additional parameter so-called *reductionRate* that determines when the reduction needs to be performed. The pseudo-code as presented in Algorithm 2 is added to a pseudo-code of Algorithm 1 after line 16 in order to implement this feature.

Algorithm 1 Tho original PSO algorithm

Input: PSO population of particles $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})^T$ for $i = 1, \ldots, Np$, MAX_FE.
Output: The best solution $\mathbf{g}$ and its corresponding value $f_{min} = \min(f(\mathbf{x}))$.

```
1: init_particles;
2: t = 0;   // generation counter
3: eval = 0;   // counter of the fitness function evaluations
4: while termination_condition_not_meet do
5:    for i = 1 to Np do
6:       f_i = evaluate_the_new_solution(x_i^(t));
7:       eval = eval + 1;
8:       if f_i ≤ pBest_i then
9:          p_i^(t) = x_i^(t); pBest_i^(t) = f_i; // save the local best solution
10:      end if
11:      if f_i ≤ f_min then
12:         g^(t) = x_i^(t); f_min = f_i; // save the global best solution
13:      end if
14:      x_i^(t) = generate_new_solution(x_i^(t));
15:   end for
16:   t = t + 1;
17: end while
```

Algorithm 2 Proposed algorithm PSOPG

```
1: if ((t mod reductionRate) == 0) then
2:    makeReduction;   // performing the population size reduction
3: end if
```

In fact, the additional code launches the process of population size reduction via the *makeReduction* function call. However, the question is which particle to eliminate from the population by this reduction. In this paper, an idea from product graphs [1] is implemented.

In the proposed PSOPG algorithm, each particle in the population is treated as an empty graph G (recall that an empty graph on n nodes consists of n isolated nodes and contains no edges), where elements of a particle are represented by a set of nodes $V(G)$. Two randomly selected donor particles, i.e. empty graphs G and H, enter in the population reduction process to generate the new particle with characteristics of the both donors. Two elements of donor particles are identified by the pair (u_i, v_j) for the i-th element of the new particle (also trial particle), where the element v_j is selected randomly. The value of this element is obtained by calculating the average value of elements u_i and v_j.

We remark that to describe our idea, the Cartesian product of sets would be enough. However, we used the notion of the Cartesian product of graphs since we believe that our idea could be improved using graphs. More precisely, we believe that representation of particles as paths (a path P_k is a graph with $V(P_k) = \{u_1, \ldots, u_k\}$ and $E(P_k) = \{u_i u_{i+1} | i \in \{1, \ldots, k-1\}\}$), or even other simple graphs, and usage of different graph products would be worth to study (Fig. 2).

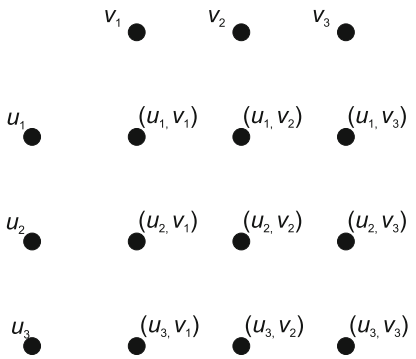


Fig. 2 The Cartesian product $G \square H$ of empty graphs G and H with $V(G) = \{u_1, u_2, u_3\}$ and $V(H) = \{v_1, v_2, v_3\}$. This product consists of isolated nodes, represented by the ordered pairs (u_i, v_j). In our case, the average value of corresponding nodes u_i and v_j in factors is calculated, that represents a potential value for the i-th element of the new particle

Motivation behind a selection of the candidate for elimination by the reduction is to preserve the particle in the next generation that transfers characteristics of both donors whereby both must be eliminated from the population. The selection operation acts as follows. The i-th element of the trial particle is selected randomly from the set $\{(u_i, v_j) | j \in \{1, \ldots, n\}\}$. As a result, the value for the i-th element of the trial particle is calculated as an average between a value of node u_i and v_j in the corresponding donor particles. In other words, this operation is expressed mathematically as

$$x_i^{(t+1)} = \frac{x_{u_i}^{(t)} + x_{v_j}^{(t)}}{2}. \tag{2}$$

As can be seen from Eq. 2, the trial particle becomes the new one i-th particle for the next generation. Thus, the j-th particle is eliminated by squeezing the current population according to the following equation

$$\mathbf{x}_k^{(t+1)} = \mathbf{x}_{k+1}^{(t)}, \quad \text{for } k = 1, \ldots, Np^{(t)} - 1. \tag{3}$$

Finally, the population size is decremented after this operation.

5 Experiments

The goal of our experimental work was to show that the population size reduction in the PSOPG algorithm using the product graphs outperforms the results of the original PSO by optimizing the benchmark test suite consisted of four well-known functions from literature. All algorithms were implemented in Python programming language, while the Cartesian products were obtained using NetworkX library. During the tests, the parameters of both PSO algorithms were set as follows: $C_1 = C_2 = 2.0$, $MAX\_FE = 50,000$ and the starting population size of the PSOPG $Np^{(0)} = 100$. The 25 independent runs were conducted, and the accumulated results were compared according to their best, median, worst and mean values, as well as standard deviation. In the remainder of the paper, the benchmark test suite is illustrated, the results of experiments are presented and discussed.

5.1 Benchmark Function Suite

Benchmark functions were downloaded from Neal Holtschulte website [10] and they are presented in Table 1.

Table 1 Summary of the benchmark functions

Tag	Function	Definition	Domain
F_1	Ackley's	$f(\mathbf{x}) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	$[-32, 32]$
F_2	Griewank	$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	$[-512, 512]$
F_3	Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$	$[-5.12, 5.12]$
F_4	Sphere	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$[-5.12, 5.12]$

The purpose of this preliminary research was to show that an idea of using the product graphs by reducing the population size could be successfully applied to the original PSO algorithm. Therefore, the original benchmark functions without shifting and rotating were considered in our tests. Note that functions of dimension $n = 10$ were employed during the tests.

5.2 Results

Four variants of the PSOPG algorithm were taken into consideration in order to show how the parameter *reductionRate* influences the results of the optimization. Thus, this parameter was varied as $reductionRate = \{80, 60, 40, 20\}$ that corresponds to a reduction of the population sizes for $\Delta_{Np} = \{6, 8, 13, 29\}$ when the initial population size of $Np^{(0)} = 100$ is considered. Additionally, two different population sizes were used for the original PSO, i.e., $Np = \{100, 50\}$.

The results of the mentioned tests are presented in Table 2, where the characteristic measures are displayed for each variant of the PSO and PSOPG algorithms for each specific function. Here, the particular variant of the algorithm is denoted either by the population size of the PSO or the parameter *reductionRate* of PSOPG that are enclosed by parentheses following the algorithm names. The best results according to mean values are represented bold in the table.

As can be seen from Table 2, the best results according to the mean values are obtained by the PSOPG using the smaller values of *reductionRate* (i.e., $reductionRate = \{20, 40\}$). Interestingly, the original PSO with population size $Np = 100$ outperformed the results of the PSO with smaller population size $Np = 50$ by the same number of the fitness function evaluations $MAX\_FE = 50,000$.

5.3 Discussion

Two facts can be concluded from the results of our experimental work. Firstly, the population size reduction using the product graphs in the PSOPG algorithm outperformed the results of the original PSO. Secondly, the parameter *reductionRate* has the important influence on the results of the PSOPG. The lower this value is, the better the results. However, in our tests, the allowed maximum reducing the population size was $\Delta_{Np} = 29$ for the starting population size $Np_0 = 100$. This means that the original population size can be maximally reduced for near 30 % that is still far from the reasonable minimum of 90 %. Therefore, more tests should be performed in the future in order to determine the correct behavior of this parameter.

It seems the used product graphs is similar to arithmetic crossover in evolutionary algorithms [3] or in PSO [9]. The purpose of our study was not to outperform

Table 2 Comparison of the experimental results

		F_1	F_2	F_3	F_4
PSO (100)	Best	3.21	0.29	1.90E-06	0.00033
	Median	7.38	1.10	1.39	0.46
	Worst	10.52	2.49	78.18	3.18
	Mean	7.51	1.10	16.28	0.65
	Std	1.67	0.53	23.06	0.76
PSO (50)	Best	4.50	0.23	9.58E-07	0.0063
	Median	7.24	0.88	2.33	0.27
	Worst	12.10	3.28	53.78	4.52
	Mean	7.40	1.02	11.62	0.70
	Std	1.89	0.59	17.04	0.97
PSOGP (80)	Best	1.92E-05	5.38E-07	3.37E-14	3.61E-18
	Median	3.17	0.53	5.68E-10	2.70E-10
	Worst	7.08	0.96	77.65	1.75E-06
	Mean	2.68	0.51	9.97	1.53E-07
	Std	2.04	0.29	22.18	3.92E-07
PSOGP (60)	Best	1.59E-06	1.72E-11	0.0	7.76E-18
	Median	3.57	0.44	3.22E-10	1.43E-12
	Worst	5.62	0.93	55.44	1.65E-06
	Mean	3.38	0.42	15.95	6.72E-08
	Std	1.55	0.31	21.38	3.24E-07
PSOGP (40)	Best	9.19E-07	3.77E-15	0.0	3.72E-25
	Median	2.80	0.54	1.77E-15	1.50E-17
	Worst	7.87	0.83	3.85	2.55E-11
	Mean	2.54	0.47	**3.85**	1.74E-12
	Std	2.16	0.25	13.07	5.99E-12
PSOGP (20)	Best	5.01E-14	0.0	0.0	7.69E-44
	Median	1.63E-07	0.35	0.0	7.68E-33
	Worst	7.10	0.83	58.07	3.14E-24
	Mean	**1.07**	**0.37**	11.14	**1.31E-25**
	Std	1.78	0.22	19.35	6.14E-25

operating this kind of crossover, but to show that the neighbourhood of candidate solution can be defined as product graphs. However, a potential of this idea needs to be fully elaborated in the future work.

6 Conclusion

In this paper we presented a new variant of the PSO algorithm. i.e., the PSOGP that proposes the use of graph products in order to help to reduce the population size. Experiments were conducted on a benchmark functions and results show the potential

of developed idea. However, this study leaves a lot of free space for further improvements. For instance, the other graph products like strong, lexicographic, direct could be applied in place of Cartesian product. Furthermore, in place of calculating the average values, the various operators between nodes should be also selected. Finally, there are also possibilities to test this idea in other nature-inspired algorithms.

Acknowledgments The second author is supported by ARRS Research Program P1-00383.

References

1. Bondy, J.-A., Murty, U.S.R.: Graph Theory, Graduate Texts in Mathematics. Springer, New York (2008)
2. Brest, J., Maučec, M.S.: Population size reduction for the differential evolution algorithm. Appl. Intell. **29**(3), 228–247 (2008)
3. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer Science & Business Media (2003)
4. Fister, I., Strnad, D., Yang, X.-S., Fister I.: Adaptation and hybridization in nature-inspired algorithms. In: Adaptation and Hybridization in Computational Intelligence, pp. 3–50. Springer (2015)
5. Fister I., Yang, X.-S., Fister, I., Brest, J., Fister, D.: A brief review of nature-inspired algorithms for optimization. Elektrotehniški vestnik **80**(3), 116–122 (2013)
6. Fister, I., Yang, X.-S., Ljubič, K., Fister, D., Brest, J., Fister, I.: Towards the novel reasoning among particles in PSO by the use of rdf and sparql. The Scientific World Journal **2014** (2014)
7. Gálvez, A., Iglesias, A.: Efficient particle swarm optimization approach for data fitting with free knot b-splines. Comput. Aided Des. **43**(12), 1683–1692 (2011)
8. Hammack, R.H., Imrich, W., Klavžar, S.: Handbook of Product Graphs. Discrete mathematics and its applications. CRC Press, Boca Raton (2011)
9. Hao, Z.-F., Wang, Z.-G., Huang, H.: A particle swarm optimization algorithm with crossover operator. In: 2007 International Conference on Machine Learning and Cybernetics, vol. 2, pp. 1036–1040. IEEE (2007)
10. Holtschulte, N., Moses, M.: Should every man be an island? (website). (2013)
11. Kaiwartya, O., Kumar, S., Lobiyal, D.K., Tiwari, P.K., Abdullah, A.H., Hassan, A.N.: Multiobjective dynamic vehicle routing problem and time seed based solution using particle swarm optimization. J. Sens. **2015** (2015)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings., IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
13. Olusanya, M.O., Arasomwan, M.A., Adewumi, A.O.: Particle swarm optimization algorithm for optimizing assignment of blood in blood banking system. Comput. Math. Methods Med. (2014)
14. Pluhacek, M., Senkerik, R., Zelinka, I., Davendra, D.: Gathering algorithm: A new concept of PSO based metaheuristic with dimensional mutation. In: 2014 IEEE Symposium on Swarm Intelligence (SIS), pp. 1–6. IEEE (2014)
15. Tvrdík, J., Poláková, R., Veselskỳ, J., Bujok, P.: Adaptive variants of differential evolution: Towards control-parameter-free optimizers. In: Handbook of Optimization, pp. 423–449. Springer (2013)
16. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
17. Ye, Z., Wang, M., Hu, Z., Liu, W.: An adaptive image enhancement technique by combining cuckoo search and particle swarm optimization algorithm. Comput. Intell. Neurosci. **2015** (2015)

18. Zhang, Y., Wu, L.: Crop classification by forward neural network with adaptive chaotic particle swarm optimization. Sensors **11**(5), 4721–4743 (2011)
19. Zhang, Y., Wu, L., Dong, Z., Wang, S., Zhou, Z.: Face orientation estimation by particle swarm optimization. In: 2009 Second International Symposium on Information Science and Engineering (ISISE), pp. 388–391. IEEE (2009)