Swarm and Evolutionary Computation 58 (2020) 100718

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Population size in Particle Swarm Optimization

Adam P. Piotrowski [a,*], Jaroslaw J. Napiorkowski [a], Agnieszka E. Piotrowska [b]

[a] Institute of Geophysics, Polish Academy of Sciences, Ks. Janusza 64, 01-452, Warsaw, Poland
[b] Faculty of Polish Studies, University of Warsaw, Krakowskie Przedmiescie 26/28, 00-927, Warsaw, Poland

ARTICLE INFO

Keywords:
Particle swarm optimization
Swarm size
Population size
Swarm intelligence
Real-world problems
Metaheuristics

ABSTRACT

Particle Swarm Optimization (PSO) is among the most universally applied population-based metaheuristic optimization algorithms. PSO has been successfully used in various scientific fields, ranging from humanities, engineering, chemistry, medicine, to advanced physics. Since its introduction in 1995, the method has been widely investigated, which led to the development of hundreds of PSO versions and numerous theoretical and empirical findings on their convergence and parameterization. However, so far there is no detailed study on the proper choice of PSO swarm size, although it is widely known that population size crucially affects the performance of metaheuristics. In most applications, authors follow the initial suggestion from 1995 and restrict the population size to 20–50 particles. In this study, we relate the performance of eight PSO variants to swarm sizes that range from 3 up to 1000 particles. Tests are performed on sixty 10- to 100-dimensional scalable benchmarks and twenty-two 1- to 216-dimensional real-world problems. Although results do differ for the specific PSO variants, for the majority of considered PSO algorithms the best performance is obtained with swarms composed of 70–500 particles, indicating that the classical choice is often too small. Larger swarms frequently improve efficiency of the method for more difficult problems and practical applications. For unimodal problems slightly lower swarm sizes are recommended for the majority of PSO variants, but some would still perform best with hundreds of particles.

1. Introduction

Among versatile population-based swarm intelligence and evolutionary algorithms [17] that aim at solving global optimization of numerical problems, Particle Swarm Optimization (PSO) [19,28] is one of the most widely applied and esteemed. PSO applications may be found in almost every discipline of science [65], from language studies [49], to the development of nanotechnology [57].

The concept of PSO was based on the behaviour of flocks of birds [28]. PSO was introduced in 1995 [28], and since then a large number of its variants are proposed each year, with various levels of modifications. With rare exceptions, PSO variants are based on the movement of individual particles through the search space, which mimics the swarm of birds or insects. Discussing the history of PSO development, their importance among metaheuristics, or details of specific PSO variants, is well beyond the scope of this study; the interested reader is referred to numerous reviews [2,4,12,17,25,36,42,56,65].

As with other swarm intelligence or evolutionary computing methods, PSO has some control parameters that need to be set by the user. The number of these parameters varies for different PSO variants, but in the most widely used PSO versions with inertia weight [47] there are four control parameters. Three of them, namely two acceleration coefficients that govern the importance of cognitive and social impact (that originate from initial studies performed in 1995 [19]), and inertia weight (that has been added to PSO in 1998 [47]) are widely investigated in the literature [3,13,14,46]. Alongside, although not being a control parameter, the choice of specific network topology and its impact on the performance has also been widely studied [34,37,52]. The fourth control parameter in classical PSO is the swarm size (also called population size, or the number of particles).

The swarm size may be considered the most "basic" control parameter of PSO, as it simply defines the number of individuals in the swarm, and hence its setting can hardly be avoided. It is also one of the most difficult parameters to settle on most metaheuristics [17]. The effect of population size has been frequently studied in detail for various Evolutionary Algorithms [9,21], Genetic Algorithms [23,38] or Differential Evolution methods [35,40]. However, according to our knowledge, the impact of the swarm size on various variants of PSO has so far not been studied in detail.

In the first paper on PSO Kennedy and Eberhart [28] referred to

* Corresponding author.
E-mail address: adampp@igf.edu.pl (A.P. Piotrowski).

https://doi.org/10.1016/j.swevo.2020.100718
Received 11 March 2019; Received in revised form 1 April 2020; Accepted 21 May 2020
Available online 27 May 2020

Table 1
Particle Swarm Optimization variants tested. D means problem dimensionality.

Short name	Long name	Reference	Year	Population size suggested in source paper	Comments
ALCPSO	PSO with an aging leader and challengers	[10]	2013	20	In this variant the concept of the leader particle is used instead of **gBest** for the velocity update. The **gBest** is initially a leader, however the leader is ageing during the run and may be replaced by another competing particle. The modification intends to mitigate the possibility of trapping the swarm in local optima. The initial particle velocities are set to 0. Velocities are restricted to be no larger than 50% of the bounds span. Rebounding technique is used following [26]. Control parameters [10] are set as: $c_1 = c_2 = 2$; $\omega = 0.4$; the initial lifespan of the leader = 60; $T = 2$; $pro = 1/D$.
CLPSO	Comprehensive learning PSO	[29]	2006	10 for 10-dimensional problems; 40 for 30-dimensional problems; no suggestion for other cases	A state-of-the-art PSO variant, in which particles are not guided towards **gBest** at all, and instead of guiding towards their own **pBest**, each particle may be guided towards their own, or its neighbour's **pBest** according to some probability pc_i, defined separately for each particle. A lack of **gBest** guidance, and making the choice of **pBest** stochastic slows down convergence and makes moving of the particles more chaotic. Bounds handling technique proposed in the original CLPSO [29] is retained, although some doubts regarding its application has been raised [26]. The particle's maximum velocity has been limited to 20% of the bounds span for each coordinate; initial velocities are generated randomly within that range. Control parameters [29] (the MATLAB code available at http://www.ntu.edu.sg/home/epnsugan/] was used) are set as: $c_1 = 1.49445$; ω is diminishing linearly during run from 0.9 to 0.2; refreshing gap = 5; for definition of pc_i, see the source paper [29].
DEPSO	Dual-environmental PSO	[63]	2019	50	In DEPSO the center of weighted positions of k best particles is used for the velocity update, instead of **gBest** and **pBest**. DEPSO was developed not only for classical, but also noisy problems. The particle's maximum velocity has been limited to the bounds span for each coordinate; initial velocities are generated randomly within that range. Rebounding technique is used following [26]. Control parameters [63] are set as: $c_1 = 3.2$; ω is linearly reduced during run from 0.9 to 0.4; k is linearly reduced during run from 20 to 10; for details on setting weights needed to determine the position of the center, see source paper [63].
EPSO	Ensemble PSO	[33]	2017	20 for 10-dimensional problems and 40 for 30-dimensional problems; no suggestion for other cases	EPSO uses an ensemble of 4 PSO algorithms together. During the search the variants are chosen by means of a specified self-adaptive selection strategy. Control parameters differ for each variant used within EPSO, we used the values specified in the source paper [33] for each variant. The particle's maximum velocity has been limited to 50% of the bounds span for each coordinate; initial velocities are generated randomly within that range. Bounds handling technique proposed in CLPSO [29] is used. For more details we refer the reader to the source paper [33].
GLPSO	Genetic learning PSO	[24]	2016	50	GLPSO merges two parts. The first one based on PSO that is guided by composite exemplar (which uses information on personal and global best positions jointly). The second one is based on genetic operators (crossover and mutation). Both parts work in parallel. The particle's maximum velocity has been limited to 20% of the bounds span for each coordinate; initial velocities are generated randomly within that range. Rebounding technique is used following [26]. Control parameters [24] are set as: $c_1 = 1.49618$; $\omega = 0.7298$; mutation probability = 0.01; stopping gap (after which a tournament selection against 20% of population size is applied) = 7; for more details see the source paper [24].
HCLPSO	Heterogenous comprehensive learning PSO	[32]	2015	20 for 10-dimensional problems and 40 for 30-dimensional problems; no suggestion for other cases	HCLPSO divides the population into two sub-populations, one of which (composed of 5/8 of particles) follows the velocity update rules from classical PSO and the other one (composed of 3/8 of particles) from CLPSO. The particle's maximum velocity has been limited to 20% of the bounds span for each coordinate; initial velocities are generated randomly within that range. Bounds handling technique proposed in CLPSO [29] is used. Each sub-population uses its own variable values of control parameters. In the first sub-population c_1 is linearly decreasing from 2.5 to 0.5 during the search, and c_2 is linearly increasing from 2.5 to 0.5; in the second sub-population c_2 is linearly decreasing from 3 to 1.5 (c_2 is not used). ω is linearly decreasing during search from 0.99 to 0.2 for both strategies. For more details the reader is referred to the source paper [32].
IILPSO	Interswarm interactive learning strategy PSO	[43]	2016	50	IILPSO operate two distinct swarms of each size (if the number of the population size is odd, we set the first sub-swarm bigger) and determine the communication rules between them. The global version of IILPSO is implemented, as it performed better than the local one in tests performed in the original study [43]. The particle's maximum velocity has been limited to 12.5% [43] of the bounds span for each coordinate; initial velocities are generated randomly within that range. In IILPSO the velocity update may be done in different ways, depending on the

(*continued on next page*)

Table 1 (*continued*)

Short name	Long name	Reference	Year	Population size suggested in source paper	Comments
					state of the search, and may require two or three acceleration coefficients c, that are parameterized differently. Following the source paper [43], when two coefficients are used, c_1 is linearly decreasing from 2.5 to 0.5 during the search, and c_2 is linearly increasing from 2.5 to 0.5; when three coefficients are used, they are fixed to $c_1 = c_2 = 1$ and $c_3 = 2$. ω is linearly decreasing during search from 0.6 to 0.4. For other specific parameters of IILPSO the reader is referred to the source paper [43].
PSO-iw	PSO with inertia weight	[47]	1998	40	This is the basic and most popular variant of PSO nowadays. It differs from the initial idea [28] by introducing a weight factor ω attributed to former velocities during the velocity update. The initial particle velocities are randomly generated within 20% of the bounds span. During the run velocities are also restricted to be no higher than 20% of the bounds span. Parameter settings are as follows: $c_1 = c_2 = 1.49$, ω is linearly decreasing during search from 0.9 to 0.4.

zoological studies in which the movements of flocks composed of 15–30 birds were simulated. This seems to be a first guess as the authors used 20 particles in their first PSO tests [28]. Since then there is a general assumption, followed by the majority of current PSO variants [10,24,25,29,31,33,58,62,63] that PSO algorithms perform best with the population size set between 20 and 50 particles. This does not mean, of course, that the impact of the swarm size on PSO performance has never been considered; it was addressed a few times in early small-scale studies [7,48,51,54]. Over recent years the impact of swarm size was very rarely addressed, and even when it was, this was often done either 1. solely for the new variant that was introduced in a particular study [39,45], 2. in papers oriented at a single specific application [59,61], or 3. for very high-dimensional problems [11,60], for which the swarm size is often set much larger, depending on the complexity of the fitness landscape of the specific problem. Moreover, although the interest in variants with variable population size is rapidly increasing for various other metaheuristics (e.g. versions of Differential Evolution with linear population size reduction [50], which won a number of IEEE Competitions in Evolutionary Computation (IEEE CEC)), and although inertia weight and acceleration coefficients are frequently made adaptive in PSO (see a review in Ref. [25]), there are very few PSO variants with variable population sizes [5,6,8,18,27].

In this paper we perform a detailed study on the impact of swarm size on the performance of PSO algorithms for single-objective, non-dynamic numerical problems. We aim at giving an answer to the following questions: 1. is the historical guess that PSO performs best with 20–50 particles right?; 2. how much should the population size settings differ for various PSO variants?; 3. can we choose a specific population size that could serve as an initial guess for applications of different PSO variants in various fields of science? We select eight different PSO variants, including the earliest and the most recent versions (see Table 1 for details); then we perform tests on a large number of scalable artificial benchmarks and real-world problems. We follow the comparison criteria based on the algorithms' performance after the pre-defined number of function evaluations, as often used in IEEE Competitions on Evolutionary Computation. Very high-dimensional problems (of many hundreds or thousands of dimensions) are not considered in this study as they inevitably require different population size settings [11,60]. Due to their specific nature, multi-objective [15] or dynamic optimization problems [20] would require different comparison settings and criteria, hence are also outside the scope of the present paper. We are aware that the impact of swarm size is related to various other factors such as: the population topology of the specific algorithm [37], the fitness landscape of the specific problem [44], the number of function calls allowed in the particular comparison setting [41], or the bound handling techniques [26]. Analyzing all such factors together in a single study, however, would be virtually impossible.

2. Particle Swarm Optimization

In this section we define the basic PSO variant with inertia weight (PSO-iw) [47], to clarify the importance of population size on PSO algorithms. For definitions and parameterization of the remaining seven PSO variants considered in this study, readers are referred to the source papers [10,24,29,32,33,43,63] and Table 1.

Initial localizations $\mathbf{x}_{i0}$ of ps particles ($i = 1, \ldots, ps$) are randomly generated from the uniform distribution within the problem-specific bounds. Note that ps is the population size that needs to be pre-set by the user. The objective function values for the initial positions of particles are evaluated and stored as $\mathbf{pBest}_{i,g}$ (where g is the generation counter set initially to 0). Initial velocities $\mathbf{v}_{i,0}$ of particles are either generated from the pre-specified interval, or are set to 0. After initialization, in every generation g, the particles are moving through the search space according to the following equation:

$$\begin{aligned} v^d_{i,g+1} &= \omega_g \cdot v^d_{i,g} + c_1 \cdot rand1^d_{i,g}(0,1) \cdot \left(pBest^d_{i,g} - x^d_{i,g}\right) + c_2 rand2^d_{i,g}(0,1) \cdot \left(gBest^d_g - x^d_{i,g}\right) \\ x^d_{i,g+1} &= x^d_{i,g} + v^d_{i,g+1} \end{aligned} \tag{1}$$

where $d = 1, \ldots, D$ (D is problem dimensionality), $\mathbf{pBest}_{i,g}$ and $\mathbf{gBest}_g$ are the best position visited so-far by the i-th particle and the best position visited so far by any particle in the swarm, c_1 and c_2 are two acceleration coefficients set by the user, and ω_g is the inertia weight (that may depend on generation, hence index g, or be constant during the search). Two random numbers $rand1^d_{i,g}(0,1)$ and $rand2^d_{i,g}(0,1)$ are generated from [0,1] interval for each i-th particle and d-th dimension separately. During the run, for each particle three vectors are stored – its current position $\mathbf{x}_{i,g}$, its current velocity $\mathbf{v}_{i,g}$ and the best position $\mathbf{pBest}_{i,g}$ visited by that particle since the initialization of the search together with the associated objective function value (a scalar that we may call $pBesf_i$). The algorithm runs until the pre-defined maximum number of function calls is used.

Why then is swarm size important? In PSO the higher the swarm size,

the more scattered the search performed by the algorithm. With a higher population size each generation takes more function calls, and a larger part of the search space may be visited. When the fitness landscape has a complicated geometry and the number of allowed function calls is high enough (i.e., we have enough time), we may expect a higher swarm size to be beneficial. However, swarms that are too large may waste many function calls by flying in almost random directions. Hence, some "proper" setting of the swarm size is required.

3. Historical discussion on population size in Particle Swarm Optimization

In the first paper on PSO Kennedy and Eberhart [28] referred to the zoological studies in which the movements of flocks composed of 15–30 birds were simulated. This seems to be an initial guess as the authors of [28] use 20 particles in their first PSO tests. In a subsequent study [19] one may find information that the results discussed there were obtained with 20 particles, even though the authors used 10–50 particles for other applications. Also, 20 particles were used by Shi and Eberhart [47], who introduced an inertia weight into PSO, which is now considered a standard approach (PSO-iw). This initial guess seems to be reproduced in the vast majority of subsequent PSO studies, even though some empirical tests were eventually performed at the edge of the new millennium, as will be discussed below.

A year after introducing PSO with inertia weight [47], Shi and Eberhart [48] verified the setting of swarm size for their algorithm. They tested PSO-iw on four classical benchmark functions (10, 20 and 30-dimensional versions of Sphere, Rosenbrock, Rastrigin and Griewank) with population sizes set to 20, 40, 80 and 160. The authors presented their results graphically and in Tables. Unfortunately, they mainly discussed graphical representation, and concluded that "the performance of PSO is not sensitive to the population size" [48], even though from the published Tables it is clear that, apart from the Sphere function, the best results were often obtained when the highest population was used. The problem with this early attempt is that the authors set an equal number of generations for each test, not an equal number of function calls, hence the better performance of PSO with a larger swarm sizes may result from the larger number of function calls used. However, based on the detailed results published in their paper, one could at least wish to verify whether larger swarm sizes are needed in PSO.

Carlisle and Dozier [7] performed probably the most detailed test on population size so far. They tested PSO with inertia weight on five classical benchmark functions (30-dimensional Sphere, Rosenbrock, Rastrigin, Griewank, and 2-dimensional Schaffer F6) with population sizes set in intervals of 5, from 5 to 200. However, the authors did not compare performance after the specific number of function calls, but assumed that the particular problem is considered solved and the algorithm stops when a specific acceptance rate for each problem is achieved. The authors set the acceptance rate relatively high (for example equal to 100 for the Rosenbrock function), which significantly affected the results. They suggested that a population size of 30 particles optimally balances the performance with speed. However, the authors acknowledge that for some problems (e.g. 2-dimensional Shaffer's F6 function), the best results are obtained when at least 95 particles are used. This again could suggest that much larger population sizes than initially suggested may be needed for PSO if the problem to be solved is tough enough. This result also reminds us that the difficulty of the problem does not have to be directly related to dimensionality, and even 2-dimensional problems may require a higher population size.

Similar benchmark functions and acceptance error rates were used by Trelea [51], who tested the performance of PSO with inertia weight [47] with three population sizes, set to 15, 30 and 60 particles. In this test it was also found that the fastest convergence to acceptable results is obtained when the population size is set to 30, but the best final performance is achieved when the population size is set to 60 (highest tested value). Unfortunately, the author puts emphasis on the speed and suggested that the swarm size should be set to 30, skipping any discussion on the final performance which would favor 60.

In 2004 van der Bergh and Engelbrecht [54] tested a number of early PSO variants on five classical benchmark functions (also in the rotated framework) with the population size set to 10, 15 and 20 particles. The results depended on the specific algorithm, but the basic PSO performed best with 20 particles (the highest tested value). Similar tests, with population sizes set between 2 and 20 at 2-step intervals, were repeated by these authors for six benchmark functions and two PSO variants in Ref. [55], showing that better results are generally obtained with larger swarm sizes. However, as the population size settings were limited to low values, the conclusions from this study are limited too.

Zhang et al. [64] were probably the first to clearly express that larger population sizes are frequently needed for PSO. They tested PSO with inertia weight, with population sizes between 10 and 150 at 10-step intervals, on 9 benchmark functions of dimensionality between 2 and 30, and found that for more difficult problems population sizes higher than 50 are needed; for some higher-dimensional problems the best results were even achieved with the highest population sizes. However, the authors mitigated this finding in the discussion, and suggested using a swarm size between 20 and 50, depending on the difficulty of the problem.

Discussion on the population size of PSO in all relatively recent reviews seems to either be skipped, or based on suggestions given in the early studies highlighted above, without getting into detail about doubts that could easily come up when the results are focused on in depth. For example, in three review studies [12,25,65] no attention is given to swarm size at all. In the review by Banks et al. [2] there is no suggestion on population size setting, instead the reader is referred to some of the early studies discussed above. Also, in the review by Bonyadi and Michalewicz [4] no population size setting is recommended, instead attention is focused on three PSO variants with variable swarm sizes already proposed, and on the possible problems one may face when too small a swarm size is used. In a review by Poli et al. [42] it is recalled that the values of population size "in the range of 20–50 are quite common". Marini and Walczak [36] addressed the problem of population size in three sentences, claiming that it may be important, but "in most cases it has been demonstrated that when the number of individuals is larger than 50, PSO is not sensitive to the size of the population". Finally, Wang et al. [56] simply stated that the common selection is between 20 and 50 particles.

4. Algorithms, test problems and comparison criteria

In the present study we test the relationship between population size and performance for eight PSO algorithms proposed between 1998 and 2019, discussed in Table 1. With the exception of the population size, control parameters of the selected PSO variants are the same as suggested in the source papers. Tests are performed on 10-, 30-, 50- and 100-dimensional artificial benchmarks from IEEE CEC 2014 [30] and IEEE CEC 2017 [1] (each set is composed of 30 scalable problems), and on 22 real-world problems from IEEE CEC 2011 (1- to 216-dimensional [16]) that represent versatile scientific fields, from economics to physics. For details on all 82 considered problems the reader is referred to the source publications [1,16,30]. For each problem, every algorithm with a specific population size has been run 51 times [1]. The maximum number of function calls used during a run was the same as in the problems' source papers, namely 150000 for CEC 2011 [16] and $10000 \cdot D$ (where D is problem dimensionality) for the CEC 2014 [30] and CEC 2017 [1] sets.

For each PSO algorithm, we consider 15 different population size settings: $ps = 3$, 5, 10, 20, 30, 40, 50, 70, 100, 200, 300, 400, 500, 700 and 1000. The range of tested swarm size values is much wider than the often assumed 20–50 particles, and includes both very small and very large populations.

The goal of this study is not to compare different algorithms, but to specify the best population size settings for each of the considered PSO

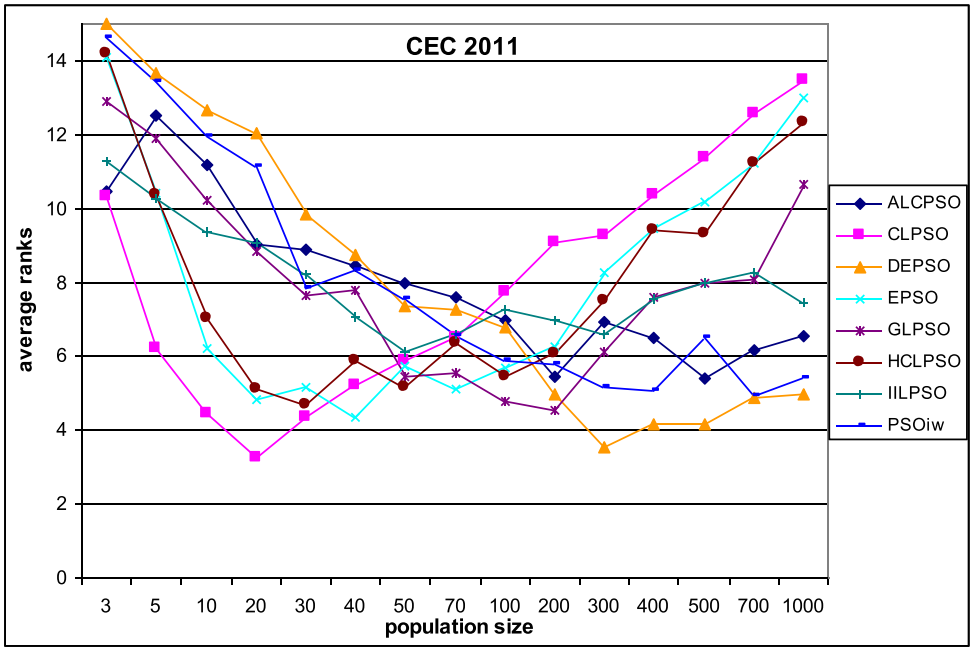

Fig. 1. Relationship between average ranks and population size for 8 different PSO algorithms on 22 real-world problems from CEC 2011. Each algorithm is run on every problem 51 times. The mean performance from these 51 runs is averaged over 22 problems. For each specific algorithm ranking is based on the performance obtained with different population sizes. No inter-comparison among various PSO algorithms is considered. The lower the rank, the better the results obtained with a particular population size by the specified algorithm.

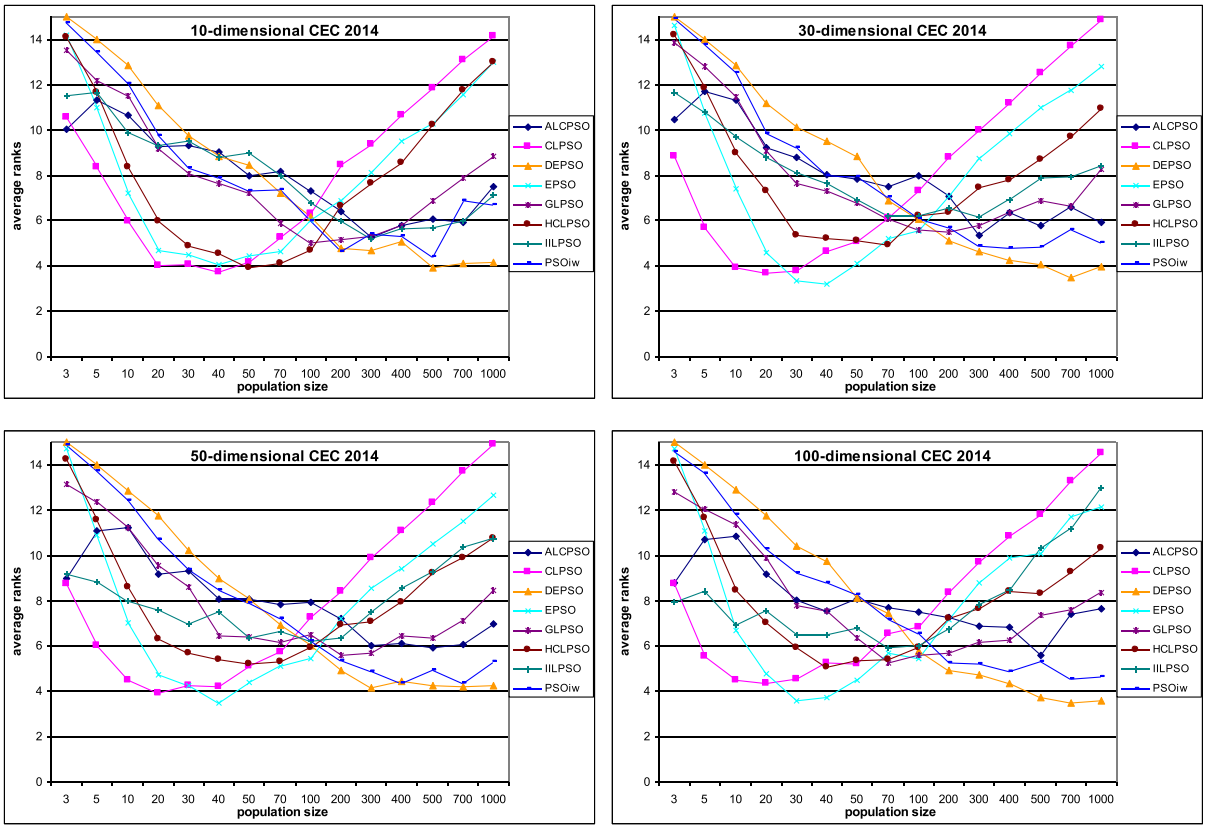

Fig. 2. Relationship between average ranks and population size for 8 different PSO algorithms on thirty 10-, 30-, 50- and 100-dimensional problems from CEC 2014. Each algorithm is run on every problem 51 times. The mean performance from these 51 runs is averaged over 30 problems. For each specific algorithm ranking is based on the performance obtained with different population sizes. No inter-comparison among various PSO algorithms is considered. The lower the rank, the better the results obtained with a particular population size by the specified algorithm.

variants, and to generalize the findings for PSO algorithms overall. We also seek to find a possible relationship between the best population size and the problem dimensionality (hence tests are performed on scalable CEC 2014 and CEC 2017 problems with different dimensionalities), and verify the relationships on a large number of real-world problems (CEC 2011).

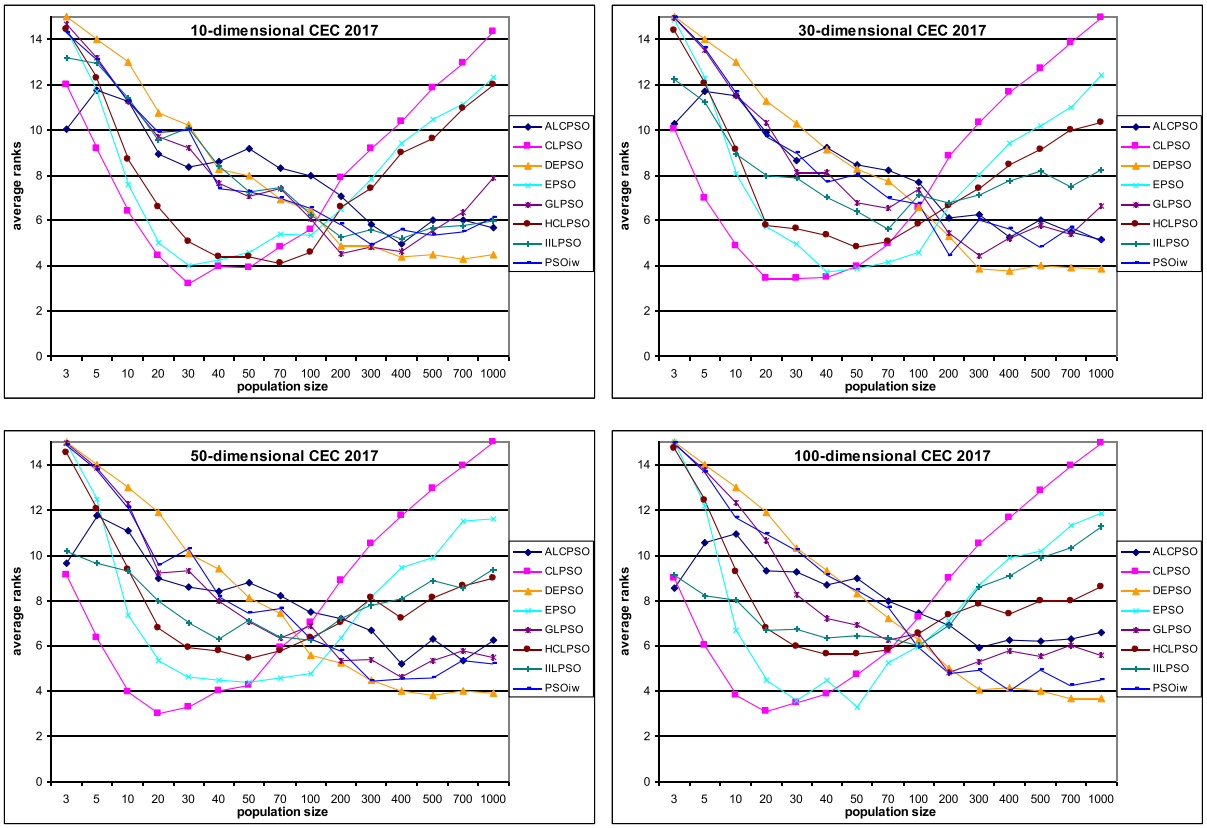

Fig. 3. Relationship between average ranks and population size for 8 different PSO algorithms on thirty 10-, 30-, 50-, and 100-dimensional problems from CEC 2017. Each algorithm is run on every problem 51 times. The mean performance from these 51 runs is averaged over 30 problems. For each specific algorithm ranking is based on the performance obtained with different population sizes. No inter-comparison among various PSO algorithms is considered. The lower the rank, the better the results obtained with a particular population size by the specified algorithm.

Considering our goals, the comparison criteria are designed as follows. Each of the eight algorithms is independently tested with 15 different population sizes on every problem (in the case of scalable benchmarks from the CEC 2014 and CEC 2017 sets – with each dimensionality). For every algorithm with each population size 51 runs are performed on every problem, as required in Ref. [1], and the mean (averaged over 51 runs) objective function values are obtained. Then, separately for each PSO variant and for each problem, population sizes are ranked from the best one, i.e., that with the lowest mean objective function value over 51 runs performed by the particular algorithm (rank 1), to the poorest (rank 15, as we compare 15 population sizes). After that, separately for each algorithm and each set of benchmark problems (i.e., CEC 2011, CEC 2014, CEC 2017), we compute the average ranks of every population size tested; the averaging is performed over all (30 or 22) problems within the particular set (CEC 2011, CEC 2014 or CEC 2017) and dimensionality (in the case of the CEC 2014 and CEC 2017 problems). Average ranks obtained are given graphically in Figs. 1–3. In addition, average rankings achieved on unimodal benchmark problems (problems 1–3 from the CEC 2014 and CEC 2017 sets) that are considered relatively simple, composition functions (problems 23–30 from the CEC 2014 and 21–30 from the CEC 2017 sets) that are considered especially hard to solve, and other "typical" multimodal and hybrid functions (problems 4–22 from CEC 2014 and 4–20 from CEC 2017) are presented in Figs. 4–6. The basic results (performances of each run of each algorithm with every swarm size on every problem) are given in the Supplementary Material.

We verified the statistical significance of the pair-wise comparisons among 15 population sizes for each algorithm, each set of problems, and each dimensionality, by means of the Friedman's test with the post-hoc Shaffer's static procedure at $\alpha = 0.05$, as suggested in Ref. [22,53]. However, as showing statistical pair-wise comparison for each considered algorithm and set of problems separately would require plenty of space, we summarize the statistical results in Tables 2–4. For each algorithm and problem set separately, we show for every population size the number of population sizes that are statistically significantly worse, and the number of population sizes that are statistically significantly better. In particular the statistical test shows only that results are different at $\alpha = 0.05$, but we interpret the results such that the particular population size is better (worse) if its mean average rank is lower (higher) than the competitors' average rank, and the test shows a statistically significant difference.

Sometimes a specific population size may be the best choice for some problems, but not useful for others. To find this out, for each algorithm, problem set and dimensionality, we count the number of wins achieved by each population size over all problems in the particular set (30 or 22). The particular population size is considered a winner for the specific algorithm and problem if it achieves the lowest mean (after 51 runs) objective function value among all 15 considered population sizes. Ties are counted as a win for each tied population size (as the user will be satisfied with each tied choice), hence there may be more than one population size winner for a specific problem and algorithm; as a result the number of wins summed over all population sizes may be higher than the number of problems. The number of wins for each population size is shown in Tables 5–7.

Finally, we relate the results obtained with different swarm sizes to those that could be achieved with the "classical" swarm sizes chosen within the [20,50] interval. In Tables 8–10 for each tested swarm size outside the reference interval [20,50] we show the percentage of problems for which the particular swarm size performs better than every considered swarm size from the reference interval (hence 20, 30, 40 and

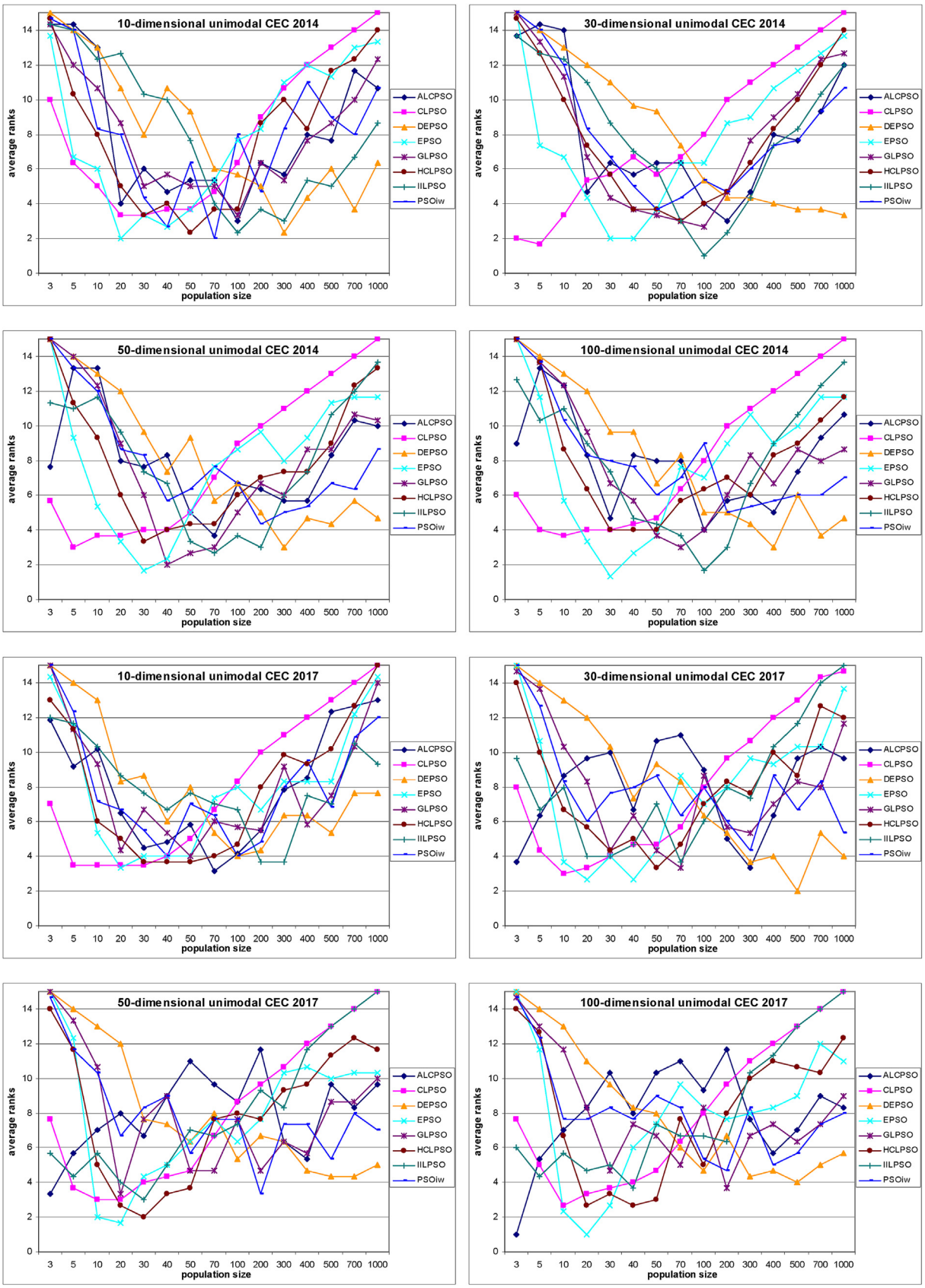

Fig. 4. Relationship between average ranks and population size for 8 different PSO algorithms on 10-, 30-, 50-, and 100-dimensional unimodal problems from CEC 2014 and CEC 2017. Only problems 1–3 from CEC 2014 and 1–3 from CEC 2017 are used to produce average ranks.

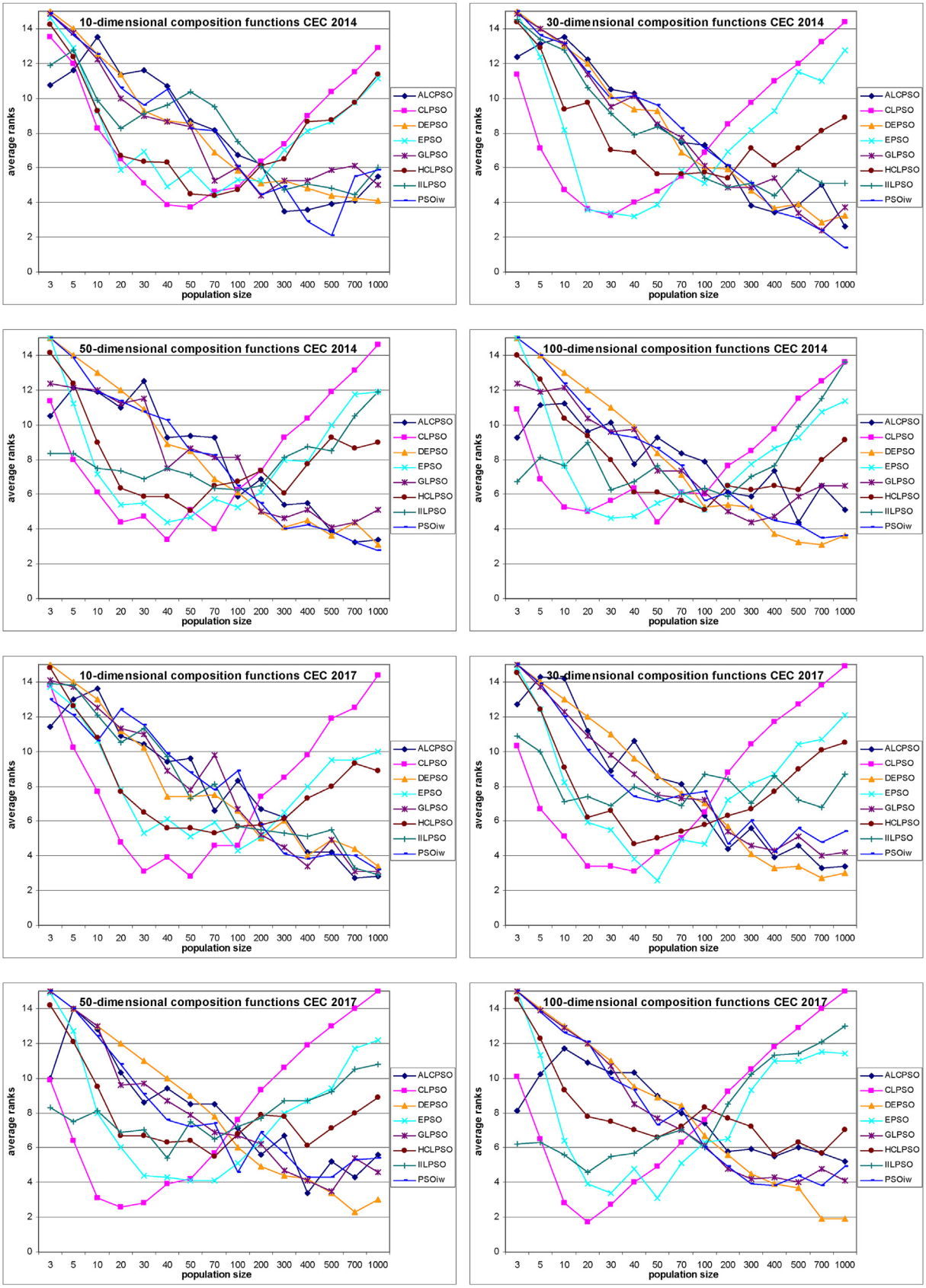

Fig. 5. Relationship between average ranks and population size for 8 different PSO algorithms on 10-, 30-, 50-, and 100-dimensional composition functions from CEC 2014 and CEC 2017. Only problems 23–30 from CEC 2014 and 21–30 from CEC 2017 are used to produce average ranks.

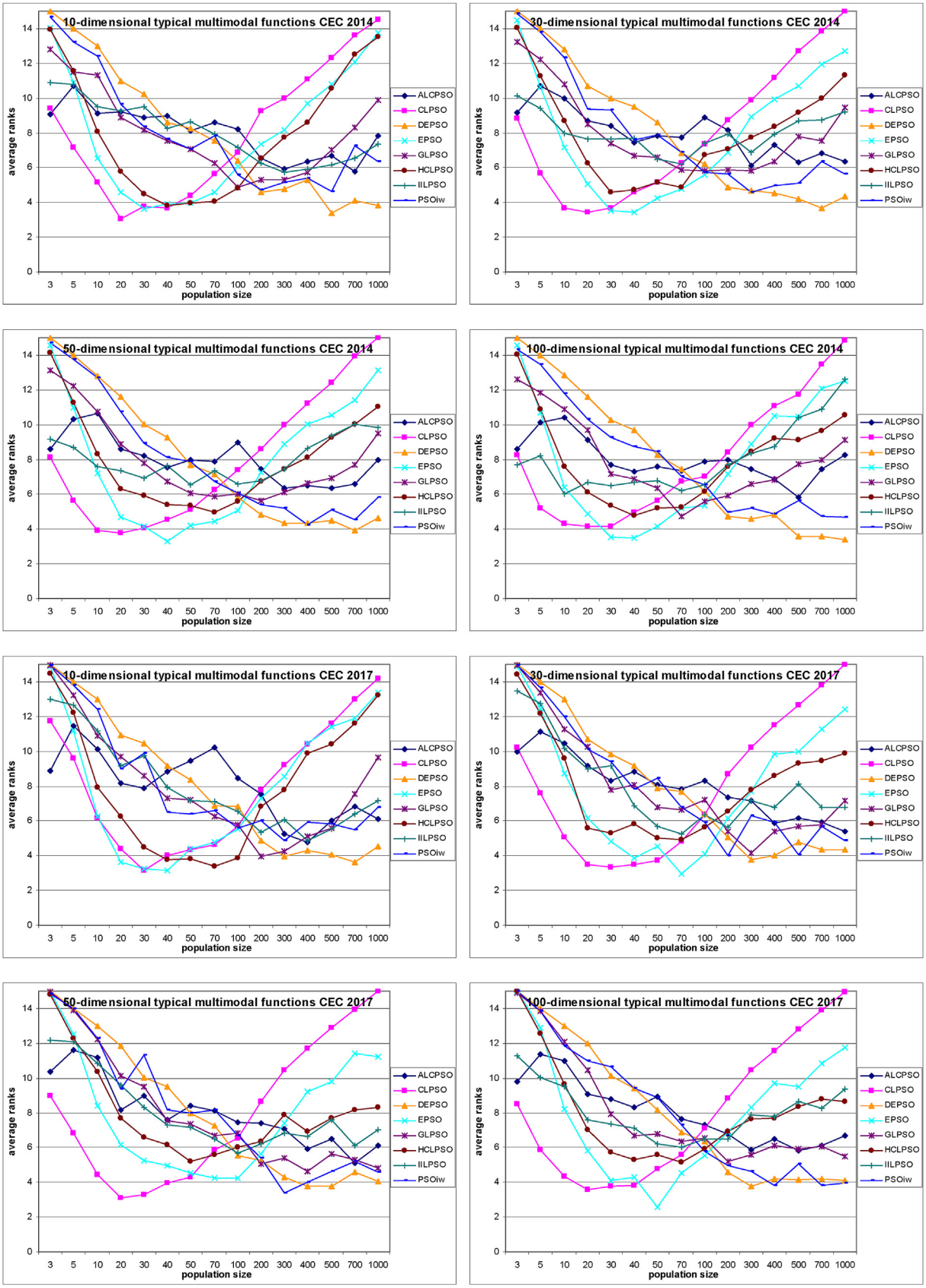

Fig. 6. Relationship between average ranks and population size for 8 different PSO algorithms on 10-, 30-, 50-, and 100-dimensional 'typical' multimodal and hybrid functions from CEC 2014 and CEC 2017. Only problems 4–22 from CEC 2014 and 4–20 from CEC 2017 are used to produce average ranks.

Table 2
CEC 2011 problems. Number of population sizes from which the particular population size of the specific algorithm performed statistically better (left half of the Table), or statistically worse (right half of the Table), with $\alpha = 0.05$. Results are based on ranks averaged over all 22 problems. Analysis is done separately for every algorithm.

	Number of population sizes from which particular population size is statistically significantly better													Number of population sizes from which particular population size is statistically significantly worse										
	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
ALCPSO	0	0	0	0	0	0	0	1	1	3	1	2	3	2	1	2	8	4	0	0	0	0	0	0	0	0	0	0	0	0
CLPSO	0	3	7	7	7	5	3	3	2	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	3	3	4	7	8	8
DEPSO	0	0	0	0	1	2	4	4	4	5	6	6	6	5	5	11	10	9	9	6	3	0	0	0	0	0	0	0	0	0
EPSO	0	0	3	6	5	6	4	5	4	3	2	1	0	0	0	10	6	0	0	0	0	0	0	0	0	0	2	4	8	9
GLPSO	0	0	0	0	1	1	4	4	4	4	2	1	1	1	0	10	5	4	0	0	0	0	0	0	0	0	0	0	0	4
HCLPSO	0	0	2	4	6	3	4	3	4	3	2	1	1	0	0	11	4	0	0	0	0	0	0	0	0	0	1	1	7	9
IILPSO	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PSO-iw	0	0	0	0	2	2	2	4	4	4	4	4	4	4	4	11	11	8	8	0	0	0	0	0	0	0	0	0	0	0

50). This measure would point out those PSO variants for which the classical choice of swarm size is generally applicable (even if it may not be the most effective) and those for which it is clearly flawed.

5. Results and discussion

In this study we address the problem of setting the PSO swarm size in detail, based on tests performed with eight PSO variants on 60 artificial benchmarks and 22 real-world applications. In Figs. 1–3 we show the average ranks for each of the 15 considered population sizes.

Tests performed on 22 real-world problems from different fields of science show (see Fig. 1) that the choice of the best population size depends on the PSO variant. For five PSO variants, namely ALCPSO, DEPSO, GLPSO, IILPSO and PSO-iw, the best performance is obtained with swarms composed of at least 50 particles. Considering that 20–50 particles are often suggested for PSO in the literature, the results obtained on real-world problems show that five out of eight PSO variants perform best with higher population sizes. In the case of four PSO variants, including the classical and most widely used PSO with inertia weight (PSO-iw) [47], the best results for real-world problems are obtained when the swarm is composed of 300–1000 particles (700 being the best pick). Choosing only 20 particles in the swarm, which is frequent in PSO applications, may be beneficial solely to CLPSO, and is clearly the wrong choice for five PSO variants. These observations are confirmed in the summary of the statistical analysis (Table 2). For the majority of PSO algorithms, on CEC 2011 real-world problems low population sizes never perform statistically significantly better than the larger swarms. On the contrary, swarms composed of 50–100 particles never perform statistically significantly poorer than other population size settings. The swarms composed of 200–300 particles are also never statistically significantly outperformed by other swarm sizes, if the CLPSO algorithm is excluded. Hence, the safest pick for real-world problems is to choose swarm size within the [50,300] range.

Average rankings of population sizes computed for real-world problems (Fig. 1) and artificial benchmarks with different dimensionalities (Figs. 2 and 3) are very similar. Irrespective of whether 10-, 30-, 50-, or 100-dimensional benchmarks are considered, the rankings are similar to those observed for real-world problems, and the results are confirmed by statistical analysis (Tables 3 and 4). However, for three algorithms (PSO-iw, ALCPSO and DEPSO) even larger swarms, composed of 400–1000 particles, perform best on benchmark problems CEC 2014 and CEC 2017 (see Tables 2 and 3). We may also find that the coefficient of linear relation between the natural logarithm of the best population size of the particular algorithm (ps) and the problem dimensionalities D for the CEC 2014 and CEC 2017 benchmarks is marginally negative, i.e., $\ln(ps) = 5.01651 - 0.00244 \cdot D$ (with very low $R^2 = 0.00477$), hence the choice of population size surprisingly does not depend on the problem dimensionality (as long as very high dimensional problems are not considered).

Five out of eight tested PSO variants perform best on scalable benchmarks when large swarms, often composed of 70–500 particles, are used. The best swarm size is, surprisingly, weakly related with problem dimensionality.

Among the remaining three PSO variants, CLPSO performs best with 20–40 particles, with 20 being the best choice for most cases. Out of 8 tested PSO algorithms, only CLPSO clearly requires such a low population size. EPSO often performs best with 30–50 particles, and HCLPSO performs similarly with a relatively wide range of swarm sizes, from 30 up to 100.

The picture given above leads to the conclusion that only two or three out of eight tested PSO variants perform best on advanced benchmarks or real-world problems with a classically assumed swarm size composed of 20–50 particles. For five PSO algorithms, including the most widely used PSO-iw [47], larger swarms, often composed of 70–500 particles, are advantageous. For some PSO variants even swarms composed of 1000 particles may lead to the best results on artificial benchmark problems

Table 3
CEC 2014 problems. Number of population sizes from which the particular population size of the specific algorithm performed statistically better (left half of the Table), or statistically worse (right half of the Table), with $\alpha = 0.05$. Results are based on ranks averaged over all 30 problems. Analysis is done separately for every algorithm.

	Number of population sizes from which particular population size is statistically significantly better															Number of population sizes from which particular population size is statistically significantly worse													
	3	**5**	**10**	**20**	**30**	**40**	**50**	**70**	**100**	**200**	**300**	**400**	**500**	**700**	**1000**	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
10-dimensional CEC 2014																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**5**	**3**	**3**	**3**	**0**	4	6	5	1	1	0	0	0	0	0	0	0	0	0	0
CLPSO	**0**	**2**	**5**	**8**	**8**	**8**	**8**	**6**	**5**	**2**	**1**	**0**	**0**	**0**	**0**	7	4	0	0	0	0	0	0	0	4	5	7	7	9	10
DEPSO	**0**	**0**	**0**	**1**	**2**	**3**	**3**	**4**	**4**	**6**	**6**	**5**	**7**	**7**	**7**	12	11	10	8	6	5	3	0	0	0	0	0	0	0	0
EPSO	**0**	**0**	**3**	**6**	**6**	**7**	**6**	**6**	**5**	**4**	**2**	**1**	**1**	**0**	**0**	11	7	0	0	0	0	0	0	0	0	1	5	6	8	9
GLPSO	**0**	**0**	**0**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**3**	**3**	**3**	**2**	**1**	12	10	7	2	0	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**2**	**5**	**5**	**7**	**7**	**7**	**6**	**4**	**4**	**2**	**1**	**0**	**0**	11	8	3	0	0	0	0	0	0	0	0	4	6	8	10
IILPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**2**	**3**	**5**	**3**	**3**	**2**	**2**	7	7	4	1	1	0	0	0	0	0	0	0	0	0	0
PSO-iw	**0**	**0**	**0**	**1**	**2**	**3**	**3**	**3**	**3**	**4**	**4**	**4**	**5**	**3**	**3**	12	11	10	4	1	0	0	0	0	0	0	0	0	0	0
30-dimensional CEC 2014																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**2**	**3**	**3**	**3**	**2**	**3**	4	7	6	0	0	0	0	0	0	0	0	0	0	0	0
CLPSO	**2**	**5**	**7**	**7**	**7**	**7**	**5**	**5**	**4**	**2**	**1**	**0**	**0**	**0**	**0**	4	0	0	0	0	0	0	0	0	4	7	8	8	10	11
DEPSO	**0**	**0**	**0**	**1**	**2**	**2**	**3**	**4**	**5**	**6**	**7**	**7**	**7**	**7**	**7**	12	11	9	8	7	6	5	0	0	0	0	0	0	0	0
EPSO	**0**	**1**	**3**	**7**	**8**	**9**	**7**	**6**	**6**	**4**	**2**	**1**	**0**	**0**	**0**	11	6	2	0	0	0	0	0	0	1	4	6	7	8	9
GLPSO	**0**	**0**	**0**	**1**	**2**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**2**	12	11	9	0	0	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**1**	**2**	**4**	**4**	**4**	**5**	**3**	**3**	**2**	**2**	**1**	**1**	**0**	12	9	1	0	0	0	0	0	0	0	0	0	0	4	6
IILPSO	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**1**	**0**	**0**	**0**	6	4	0	0	0	0	0	0	0	0	0	0	0	0	0
PSO-iw	**0**	**0**	**0**	**2**	**2**	**3**	**3**	**3**	**3**	**4**	**5**	**5**	**5**	**4**	**5**	12	12	10	6	4	0	0	0	0	0	0	0	0	0	0
50-dimensional CEC 2014																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**2**	0	5	6	0	0	0	0	0	0	0	0	0	0	0	0
CLPSO	**2**	**5**	**7**	**7**	**7**	**7**	**5**	**5**	**3**	**3**	**2**	**1**	**0**	**0**	**0**	4	0	0	0	0	0	0	0	0	4	7	7	9	11	12
DEPSO	**0**	**0**	**0**	**0**	**1**	**3**	**3**	**4**	**5**	**6**	**7**	**6**	**7**	**7**	**7**	11	10	10	8	7	6	4	0	0	0	0	0	0	0	0
EPSO	**0**	**1**	**4**	**7**	**7**	**7**	**7**	**6**	**6**	**3**	**2**	**1**	**1**	**0**	**0**	12	7	0	0	0	0	0	0	0	0	4	6	6	8	9
GLPSO	**0**	**0**	**0**	**0**	**1**	**3**	**3**	**3**	**3**	**4**	**3**	**3**	**3**	**3**	**2**	11	10	9	1	0	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**1**	**3**	**4**	**4**	**5**	**5**	**4**	**2**	**2**	**1**	**1**	**1**	**0**	12	8	0	0	0	0	0	0	0	0	0	0	2	5	6
IILPSO	**0**	**0**	**0**	**0**	**0**	**0**	**2**	**1**	**2**	**2**	**0**	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	3	4
PSO-iw	**0**	**0**	**0**	**1**	**2**	**3**	**3**	**3**	**4**	**5**	**5**	**6**	**5**	**6**	**5**	12	11	10	7	6	2	0	0	0	0	0	0	0	0	0
100-dimensional CEC 2014																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**0**	**0**	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0
CLPSO	**2**	**5**	**7**	**7**	**6**	**5**	**5**	**4**	**4**	**2**	**1**	**0**	**0**	**0**	**0**	3	0	0	0	0	0	0	0	0	2	6	8	8	10	11
DEPSO	**0**	**0**	**0**	**0**	**1**	**2**	**3**	**4**	**6**	**6**	**6**	**6**	**7**	**8**	**8**	11	10	9	8	7	7	3	2	0	0	0	0	0	0	0
EPSO	**0**	**0**	**4**	**7**	**7**	**7**	**7**	**6**	**6**	**4**	**1**	**1**	**1**	**0**	**0**	11	8	0	0	0	0	0	0	0	0	4	6	6	8	8
GLPSO	**0**	**0**	**0**	**0**	**2**	**2**	**3**	**4**	**4**	**4**	**3**	**3**	**3**	**2**	**1**	11	10	7	3	0	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**1**	**2**	**3**	**4**	**3**	**3**	**3**	**2**	**2**	**1**	**1**	**1**	**0**	12	8	0	0	0	0	0	0	0	0	0	0	0	1	5
IILPSO	**1**	**1**	**2**	**1**	**2**	**2**	**2**	**3**	**3**	**2**	**1**	**1**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	2	7	12
PSO-iw	**0**	**0**	**0**	**1**	**2**	**2**	**2**	**3**	**3**	**5**	**5**	**6**	**5**	**6**	**6**	12	11	8	6	6	3	0	0	0	0	0	0	0	0	0

Table 4
CEC 2017 problems. Number of population sizes from which the particular population size of the specific algorithm performed statistically better (left half of the Table), or statistically worse (right half of the Table), with $\alpha = 0.05$. Results are based on ranks averaged over all 30 problems. Analysis is done separately for every algorithm.

	Number of population sizes from which particular population size is statistically significantly better															Number of population sizes from which particular population size is statistically significantly worse													
	3	**5**	**10**	**20**	**30**	**40**	**50**	**70**	**100**	**200**	**300**	**400**	**500**	**700**	**1000**	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
10-dimensional CEC 2017																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**2**	**3**	**5**	**3**	**3**	**3**	5	6	6	1	0	0	1	0	0	0	0	0	0	0	0
CLPSO	**0**	**1**	**5**	**7**	**8**	**8**	**8**	**7**	**5**	**4**	**1**	**1**	**0**	**0**	**0**	8	5	0	0	0	0	0	0	0	3	5	7	8	8	11
DEPSO	**0**	**0**	**0**	**1**	**1**	**3**	**3**	**3**	**4**	**5**	**5**	**6**	**5**	**6**	**5**	12	10	10	7	6	2	0	0	0	0	0	0	0	0	0
EPSO	**0**	**0**	**3**	**6**	**6**	**6**	**6**	**6**	**6**	**5**	**3**	**1**	**1**	**0**	**0**	11	9	0	0	0	0	0	0	0	0	0	6	7	7	9
GLPSO	**0**	**0**	**0**	**1**	**2**	**2**	**3**	**2**	**3**	**5**	**5**	**5**	**4**	**3**	**2**	12	11	7	4	3	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**1**	**4**	**6**	**7**	**7**	**7**	**7**	**4**	**3**	**1**	**1**	**0**	**0**	11	8	4	0	0	0	0	0	0	0	0	5	5	7	8
IILPSO	**0**	**0**	**0**	**0**	**0**	**2**	**3**	**3**	**3**	**5**	**5**	**5**	**4**	**4**	**4**	10	10	9	3	6	0	0	0	0	0	0	0	0	0	0
PSO-iw	**0**	**0**	**0**	**1**	**1**	**3**	**3**	**3**	**3**	**5**	**5**	**5**	**5**	**5**	**4**	12	10	10	5	6	0	0	0	0	0	0	0	0	0	0
30-dimensional CEC 2017																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**3**	**3**	**4**	**3**	**4**	**5**	6	7	6	3	0	1	0	0	0	0	0	0	0	0	0
CLPSO	**2**	**4**	**7**	**7**	**7**	**7**	**7**	**7**	**4**	**3**	**1**	**0**	**0**	**0**	**0**	6	0	0	0	0	0	0	0	0	6	6	8	9	10	11
DEPSO	**0**	**0**	**0**	**0**	**1**	**3**	**3**	**3**	**4**	**6**	**8**	**8**	**7**	**8**	**8**	11	10	10	7	6	6	5	4	0	0	0	0	0	0	0
EPSO	**0**	**0**	**3**	**5**	**6**	**8**	**8**	**8**	**6**	**4**	**3**	**1**	**1**	**1**	**0**	12	9	3	0	0	0	0	0	0	0	3	5	6	7	9
GLPSO	**0**	**0**	**0**	**1**	**2**	**2**	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**4**	**3**	12	11	9	5	0	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**1**	**4**	**4**	**4**	**6**	**6**	**4**	**2**	**2**	**1**	**1**	**1**	**1**	13	8	2	0	0	0	0	0	0	0	0	0	2	6	6
IILPSO	**0**	**0**	**0**	**1**	**1**	**2**	**2**	**2**	**2**	**2**	**2**	**1**	**1**	**1**	**1**	12	6	0	0	0	0	0	0	0	0	0	0	0	0	0
PSO-iw	**0**	**0**	**0**	**2**	**2**	**3**	**2**	**3**	**3**	**5**	**3**	**4**	**5**	**4**	**5**	12	12	9	5	3	0	0	0	0	0	0	0	0	0	0
50-dimensional CEC 2017																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**2**	**3**	**2**	**3**	**2**	2	7	5	0	0	0	0	0	0	0	0	0	0	0	0
CLPSO	**3**	**5**	**7**	**8**	**7**	**7**	**7**	**5**	**4**	**3**	**1**	**0**	**0**	**0**	**0**	5	0	0	0	0	0	0	0	1	5	7	8	10	10	11
DEPSO	**0**	**0**	**0**	**0**	**2**	**2**	**4**	**4**	**6**	**6**	**6**	**7**	**7**	**7**	**7**	11	11	9	9	7	7	4	0	0	0	0	0	0	0	0
EPSO	**0**	**0**	**4**	**6**	**6**	**6**	**6**	**6**	**6**	**4**	**2**	**1**	**1**	**0**	**0**	11	9	0	0	0	0	0	0	0	0	0	6	6	8	8
GLPSO	**0**	**0**	**0**	**2**	**2**	**3**	**3**	**3**	**3**	**4**	**4**	**5**	**4**	**3**	**3**	12	12	10	1	4	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1**	**2**	**1**	**1**	**1**	13	8	0	0	0	0	0	0	0	0	0	0	0	0	0
IILPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PSO-iw	**0**	**0**	**0**	**2**	**1**	**3**	**3**	**3**	**4**	**4**	**5**	**5**	**5**	**5**	**5**	12	11	10	5	7	0	0	0	0	0	0	0	0	0	0
100-dimensional CEC 2017																														
ALCPSO	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**1**	0	4	6	0	0	0	0	0	0	0	0	0	0	0	0
CLPSO	**3**	**5**	**7**	**8**	**7**	**7**	**7**	**5**	**4**	**3**	**1**	**0**	**0**	**0**	**0**	5	0	0	0	0	0	0	0	1	5	7	8	10	10	11
DEPSO	**0**	**0**	**0**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**7**	**7**	**7**	**7**	11	10	9	8	7	6	5	0	0	0	0	0	0	0	0
EPSO	**0**	**0**	**4**	**7**	**7**	**7**	**7**	**6**	**6**	**4**	**1**	**1**	**1**	**0**	**0**	11	8	0	0	0	0	0	0	0	0	4	6	6	8	8
GLPSO	**0**	**0**	**0**	**1**	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	12	11	11	8	0	0	0	0	0	0	0	0	0	0	0
HCLPSO	**0**	**0**	**1**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1**	13	11	0	0	0	0	0	0	0	0	0	0	0	0	0
IILPSO	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**2**	**2**	**1**	**0**	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	2	7
PSO-iw	**0**	**0**	**0**	**1**	**1**	**2**	**2**	**3**	**5**	**6**	**6**	**7**	**6**	**7**	**7**	12	10	8	7	7	6	3	0	0	0	0	0	0	0	0

Table 5
Number of CEC 2011 problems on which the particular population size performed best for the specific algorithm. If different population sizes performed equally well on a particular problem, each is considered a winner for this problem, hence rows do not add up to 22 (number of problems).

Swarm size	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
ALCPSO	4	1	1	2	1	2	1	3	3	4	2	2	5	3	2
CLPSO	0	3	7	9	3	1	1	1	1	2	2	1	1	1	2
DEPSO	0	0	0	0	1	0	3	0	0	0	4	0	3	2	9
EPSO	1	2	4	6	4	2	2	3	4	3	1	1	1	1	1
GLPSO	1	0	1	2	1	3	3	3	4	4	3	1	2	4	2
HCLPSO	0	2	3	4	3	2	5	3	2	3	2	1	2	2	1
IILPSO	1	0	2	0	0	4	6	1	1	4	1	2	3	3	2
PSO-iw	0	0	0	0	0	3	2	2	1	3	3	2	1	5	9

Table 6
Number of 10-, 30-, 50-, and 100-dimensional CEC 2014 problems on which the particular population size performed best for the specific algorithm. If different population sizes performed equally well on a particular problem, each is considered a winner for this problem, hence rows do not add up to 30 (number of problems).

Swarm size	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
10-dimensional CEC 2014															
ALCPSO	5	0	0	3	1	0	2	1	0	0	4	4	3	3	4
CLPSO	1	2	6	7	4	4	5	1	2	0	1	0	0	0	0
DEPSO	0	0	0	1	0	0	2	1	0	3	5	1	5	3	11
EPSO	0	1	2	9	5	3	4	1	2	1	2	0	0	0	0
GLPSO	1	0	0	0	0	2	2	2	3	7	1	3	3	1	5
HCLPSO	0	0	1	1	4	7	4	5	4	1	2	0	1	0	0
IILPSO	1	1	1	1	1	1	1	0	4	2	2	3	3	6	3
PSO-iw	0	0	0	1	1	1	1	3	2	4	2	4	7	3	1
30-dimensional CEC 2014															
ALCPSO	4	1	1	4	1	0	0	0	0	4	1	2	3	2	8
CLPSO	2	4	8	7	3	2	1	2	0	0	1	0	0	0	0
DEPSO	0	0	0	0	0	0	0	1	2	5	1	1	1	7	12
EPSO	0	3	2	3	9	7	2	1	1	2	0	0	0	0	0
GLPSO	1	0	0	0	3	4	1	1	3	2	2	2	3	7	2
HCLPSO	0	1	2	1	6	2	4	4	2	1	1	4	0	2	0
IILPSO	2	0	4	2	0	1	1	1	5	1	2	1	1	3	6
PSO-iw	0	0	1	1	0	0	2	2	0	3	1	3	2	5	10
50-dimensional CEC 2014															
ALCPSO	6	1	1	3	0	2	1	1	1	1	2	1	3	2	5
CLPSO	3	6	5	4	2	5	1	3	0	2	0	0	0	0	0
DEPSO	0	0	0	0	0	0	1	1	2	4	3	3	3	4	11
EPSO	0	1	3	7	3	5	2	1	6	1	1	0	0	0	0
GLPSO	2	0	0	0	0	3	2	5	2	4	2	0	5	4	2
HCLPSO	0	1	2	5	2	4	3	2	2	2	2	3	0	2	1
IILPSO	5	2	2	1	1	1	2	1	6	4	1	0	0	1	3
PSO-iw	0	0	0	0	2	1	1	1	3	2	1	4	1	7	7
100-dimensional CEC 2014															
ALCPSO	7	1	1	2	1	3	0	1	1	1	2	2	2	2	4
CLPSO	3	5	7	3	1	2	3	0	4	0	0	2	0	0	0
DEPSO	0	0	0	0	0	0	0	0	2	5	0	3	1	4	15
EPSO	0	1	2	4	7	5	3	1	2	2	1	0	0	1	1
GLPSO	2	0	0	0	1	1	4	4	4	2	4	2	0	2	4
HCLPSO	0	0	4	3	2	4	2	3	4	1	1	0	2	4	0
IILPSO	5	3	5	1	1	3	1	4	2	2	1	0	0	1	1
PSO-iw	0	0	0	0	0	2	2	1	1	2	2	2	2	7	9

(see Figs. 2 and 3).

The different requirements of swarm size settings noted for different PSO variants may partly be clarified by their searching properties, or network topology used; here we briefly discuss the case for three variants: PSO-iw, DEPSO and CLPSO.

PSO-iw often performs best with relatively large swarms because all its particles are guided by the global best solution, which leads to premature convergence if the swarm size is small. If problems are simple and low-dimensional (as in the majority of initial tests made in early literature on PSO), the quick convergence is not a problem, and slow convergence may be considered a waste of time. This may explain early suggestions to set swarm sizes relatively small (see historical discussion in section 3). However, for higher-dimensional problems with multiple local optima, which are frequently addressed today, small PSO-iw swarms fail.

The DEPSO algorithm is guided by the weighted position of some of the best particles in the swarm, instead of the single best (global or local) particle. This weighted position may attract particles quickly if its location does not change during consecutive iterations. The more particles in the swarm, the higher the chance that different particles would be considered among the best ones in subsequent generations, and the weighted position of the best particles would move, preventing quick convergence.

CLPSO does not use any information on the global best position in the swarm. Instead, each particle is guided only by its own position, or by the best position of one of its neighbours. As a result, the CLPSO swarm converges slowly even if it is composed of a relatively small number of particles. With large swarms, the movement of particles in CLPSO become too chaotic, making convergence to any optima too slow.

Statistical analysis provides some more details on preferred swarm size settings. For each PSO variant tested, on almost none of the set problems did the swarm with 100 particles perform statistically significantly poorer than any other swarm size (see Tables 2–4). Only in the case of CLPSO, tested on 50- and 100-dimensional CEC 2017 problems, is

Table 7
Number of 10-, 30-, 50-, and 100-dimensional CEC 2017 problems on which the particular population size performed best for the specific algorithm. If different population sizes performed equally well on a particular problem, each is considered a winner for this problem, hence rows do not add up to 30 (number of problems).

Swarm size	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
10-dimensional CEC 2017															
ALCPSO	3	2	1	3	2	3	1	2	2	2	4	4	1	5	8
CLPSO	0	2	3	8	6	6	2	4	2	0	0	0	0	0	0
DEPSO	0	0	0	1	0	4	1	3	1	2	4	4	4	6	10
EPSO	0	0	3	9	3	5	5	3	5	4	1	2	1	0	0
GLPSO	0	0	1	3	3	3	3	3	3	4	7	4	3	5	3
HCLPSO	0	0	1	2	8	4	5	6	5	2	1	2	0	0	2
IILPSO	1	1	2	2	1	2	4	3	3	4	4	3	3	1	8
PSO-iw	1	0	2	1	1	4	2	3	5	3	3	4	4	6	5
30-dimensional CEC 2017															
ALCPSO	5	0	2	2	0	0	1	0	2	1	1	0	2	6	8
CLPSO	0	3	4	6	4	3	6	4	0	0	0	0	0	0	0
DEPSO	0	0	0	2	0	0	0	0	0	4	2	4	3	9	8
EPSO	0	0	2	4	4	5	4	7	4	0	0	0	0	0	0
GLPSO	0	0	0	2	4	0	3	0	0	2	5	1	1	4	8
HCLPSO	0	0	1	4	2	2	6	5	3	2	1	0	1	3	0
IILPSO	2	1	1	0	1	3	1	5	2	1	3	0	2	5	3
PSO-iw	0	0	0	0	2	2	2	1	2	4	2	2	3	4	6
50-dimensional CEC 2017															
ALCPSO	6	0	0	5	0	1	0	0	1	2	2	1	0	3	9
CLPSO	1	2	6	9	4	1	3	0	4	0	0	0	0	0	0
DEPSO	0	0	0	0	1	0	2	0	3	3	0	4	1	4	12
EPSO	0	0	4	6	2	3	4	2	6	3	0	0	0	0	0
GLPSO	0	0	0	2	0	1	2	4	0	1	3	5	2	2	8
HCLPSO	0	0	0	5	6	2	2	4	2	0	0	3	0	4	2
IILPSO	4	1	2	3	2	0	1	3	2	2	2	0	0	2	6
PSO-iw	0	0	0	1	1	0	2	0	1	3	7	3	1	5	6
100-dimensional CEC 2017															
ALCPSO	8	0	2	2	2	1	1	0	0	2	1	3	1	3	4
CLPSO	2	2	8	8	1	3	3	3	0	0	0	0	0	0	0
DEPSO	0	0	0	0	0	0	0	1	0	4	3	2	3	7	10
EPSO	0	1	3	7	4	3	9	0	1	1	0	0	1	0	0
GLPSO	0	0	0	1	0	4	1	2	2	4	0	2	1	3	10
HCLPSO	0	0	2	3	3	4	2	3	0	2	0	1	1	4	5
IILPSO	8	2	0	2	2	0	0	5	2	3	1	0	0	2	3
PSO-iw	0	0	0	1	0	0	2	1	2	3	4	3	1	4	9

Table 8
Competitive performance against classical swarm sizes 20–50. The percentage of CEC 2011 problems on which the particular population size performed better than each tested classical swarm size, e.g. swarm sizes set to 20, 30, 40 or 50 (for each PSO variant separately).

Swarm size	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
ALCPSO	14	9	14	reference swarm sizes				23	55	45	36	45	45	50	45
CLPSO	0	23	36					14	9	9	14	9	9	5	5
DEPSO	0	0	0					64	59	77	82	77	77	68	68
EPSO	0	5	14					23	23	14	9	9	9	5	5
GLPSO	5	0	5					32	55	64	45	36	41	41	27
HCLPSO	0	9	14					18	18	27	23	9	14	9	9
IILPSO	14	14	23					27	32	32	36	32	36	36	36
PSO-iw	0	0	0					45	55	50	55	64	45	55	59

the swarm with 100 particles statistically significantly outperformed by other swarm sizes. Swarms with 70 particles show, according to statistical tests, similarly robust performance for almost all algorithms, with the exception of DEPSO. Hence, swarms with 70–100 particles seems the "safest" option for any PSO version if there is no hint of whether the PSO variant prefers small (like CLPSO or EPSO) or large (like at least half of the tested PSO algorithms) sizes. For almost all PSO variants the choice of swarm size would statistically significantly affect the performance (IILPSO is the exception that seems remarkably unaffected by the population size as long as swarm sizes are not set to extreme values).

It is widely assumed that the optimal swarm settings may depend on the difficulty of the problem. Hence, in Figs. 4–6 we show how the performance of PSO variants is related to the population size for relatively simple unimodal problems (Fig. 4), for relatively difficult composition functions (Fig. 5), and for "typical" functions from the CEC 2014 and CEC 2017 sets. Of course, this part of the analysis does not include real-world problems.

The choice of the best population size for unimodal problems (see Fig. 4) is highly dependent on the specific PSO variant. In the case of CLPSO, swarms composed of 5–20 particles may be the best choice. In the case of ALCPSO – 3 particles may perform best on 50- or 100-dimensional unimodal CEC 2017 problems, but are among the worst choices for 10-dimensional variants (note that this finding is counter-intuitive), or for CEC 2014 benchmarks. However, the majority of PSO variants perform best on unimodal problems with swarm sizes set to 20–100; swarms composed of fewer than 20 particles often lead to poor performance. Finally, DEPSO and to some extent PSO-iw reach the best results with at least 300 particles. Hence, although generally to solve unimodal problems smaller swarms are sufficient and "classical" recommendations to use 20–50 particles do hold, some PSO variants would still require larger swarms.

Results obtained for composition functions (see Fig. 5) are similar to those averaged over all problems. However, for some PSO variants marginally larger swarms may perform better for composition functions.

Table 9
Competitive performance against classical swarm sizes 20–50. The percentage of 10-, 30-, 50- and 100-dimensional CEC 2014 problems on which the particular population size performed better than each tested classical swarm size, e.g. swarm sizes set to 20, 30, 40 or 50 (for each PSO variant separately).

Swarm size	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
10-dimensional CEC 2014															
ALCPSO	37	20	20	**reference swarm sizes**				30	53	57	73	67	60	60	50
CLPSO	7	17	30					10	13	7	7	3	3	3	3
DEPSO	0	0	0					63	73	77	80	73	77	80	70
EPSO	0	3	10					20	13	17	13	17	13	7	7
GLPSO	7	7	0					50	70	63	63	60	57	53	50
HCLPSO	0	0	3					30	30	20	20	10	10	7	7
IILPSO	13	13	20					43	63	60	60	63	60	60	60
PSO-iw	0	3	0					37	50	57	50	50	60	40	43
30-dimensional CEC 2014															
ALCPSO	27	10	13	**reference swarm sizes**				43	40	57	57	60	50	63	60
CLPSO	17	30	47					7	10	7	3	3	0	0	0
DEPSO	0	0	0					90	93	87	87	90	87	90	83
EPSO	0	10	13					10	10	10	7	7	3	0	0
GLPSO	3	3	3					53	63	60	60	53	50	50	47
HCLPSO	0	7	13					30	27	30	23	27	27	23	20
IILPSO	17	17	23					47	43	47	43	47	40	43	43
PSO-iw	0	0	3					53	57	63	70	70	73	60	67
50-dimensional CEC 2014															
ALCPSO	37	30	23	**reference swarm sizes**				37	37	53	53	60	60	60	53
CLPSO	17	30	47					13	13	13	3	3	0	0	0
DEPSO	0	0	0					90	90	87	87	80	83	80	70
EPSO	0	7	13					17	23	17	10	7	3	3	3
GLPSO	7	3	7					40	47	53	57	50	53	53	40
HCLPSO	0	3	10					43	37	37	30	30	23	20	23
IILPSO	27	27	30					23	43	37	33	27	20	20	20
PSO-iw	0	0	3					53	70	70	70	73	70	70	67
100-dimensional CEC 2014															
ALCPSO	33	30	27	**reference swarm sizes**				37	40	43	43	43	50	40	40
CLPSO	17	37	47					10	20	17	7	7	10	7	3
DEPSO	0	0	0					87	90	93	87	87	90	87	77
EPSO	0	10	13					13	23	20	10	7	7	3	7
GLPSO	7	7	3					57	57	57	47	53	47	50	47
HCLPSO	0	3	17					40	47	37	30	27	23	27	27
IILPSO	37	27	43					33	30	33	27	27	17	10	7
PSO-iw	7	3	7					60	73	80	77	77	73	73	80

For example, for 10-dimensional composition functions CLPSO achieved the best performance with 50 particles, instead of 20–40 that are recommended on 'typical' problems. ALCPSO frequently performs best with 500–1000 particles for higher dimensional composition functions. Irrespectively of problem dimensionality, for composition functions DEPSO performs best with 700–1000 particles. The impact of swarm size on the algorithm's performance is more severe for composition functions than for 'average' problems. For example, in the case of CLPSO the average rank of the swarm composed of 20 particles computed on all 100-dimensional CEC 2017 problems is above 3, but when only composition functions are considered (problems 21–30), it decreases below 2. In both cases swarms with 20 particles are recommended for CLPSO, but this recommendation is much stronger for difficult composition functions. We observe a similar situation for the basic PSO-iw on 10-dimensional CEC 2014 problems (where 500 particles are recommended anyway, but much more strongly for composition functions than for 'average' problems), for DEPSO on 100-dimensional CEC 2017, or EPSO on 30-dimensional CEC 2017 problems.

The choice of swarm sizes for 'typical' multimodal or hybrid functions (Fig. 6) barely differs from the findings based on all problems collected together (compare Fig. 6 with Figs. 2 and 3).

Why does choosing the swarm size according to difficult composition functions only and according to the whole set of problems lead to almost the same conclusions? In our opinion there are two main reasons: 1. A large part of the CEC 2014 and CEC 2017 problems are composition functions (roughly 25–30%), hence they have a large impact on the choice of the swarm size based on all problems. On the other hand, unimodal functions compose only 10% of the CEC 2014 and CEC 2017 problem sets. 2. Various composition functions may have very different characteristics, hence they do not necessarily require similar population sizes. It is simpler to fit control parameters to simple problems (e.g. unimodal functions) than to difficult problems with different properties.

Let us now go back to the results obtained for all considered problems. Another way of comparing the performance of population sizes is to count for how many problems (out of 30 in the case of the CEC 2014 or CEC 2017 benchmarks, or out of 22 in the case of real-world applications) the particular population size is the best choice, hence 'win' the competition (see Tables 5–7). Again, this counting is done separately for each algorithm and problem set. As we count ties as a win for each tied population size, the sum of wins over all tested population sizes for a particular algorithm may frequently be higher than the number of problems.

We find that the number of wins counted for each swarm size depends largely on the algorithm, as we could expect from previous discussions. However, this problem-wise approach reveals novel findings.

In the case of many algorithms (e.g. ALCPSO, EPSO, IILPSO) every population size may turn out to be the best choice for some problems. For example, for ALCPSO and EPSO each swarm size turns out to be the best choice (including ties) for at least one real-world problem. On the contrary, large swarms (composed of over 200 particles) are almost never the best choices for CLPSO (surprisingly, real-world problems are the exception), and small swarms (lower than 20) are almost never the best choice for classical PSO-iw or DEPSO. The fact that large swarms are never recommended for CLPSO on artificial benchmarks, but may perform best on some real-world problems (see Table 5) is intriguing. It may indicate that some features of real-world problems are simply missing in widely used benchmarks. Also, counting the number of wins again shows that the classical small swarms that are often applied for the popular PSO-iw algorithm may have often been inadequate. Our tests show that PSO-iw swarms composed of fewer than 40 particles do not

Table 10
Competitive performance against classical swarm sizes 20–50. The percentage of 10-, 30-, 50- and 100-dimensional CEC 2017 problems on which the particular population size performed better than each tested classical swarm size, e.g. swarm sizes set to 20, 30, 40 or 50 (for each PSO variant separately).

Swarm size	3	5	10	20	30	40	50	70	100	200	300	400	500	700	1000
10-dimensional CEC 2017															
ALCPSO	27	7	10	**reference swarm sizes**				47	40	43	57	60	57	50	53
CLPSO	3	3	10					20	17	10	7	0	0	0	0
DEPSO	0	0	0					63	63	77	73	77	77	77	67
EPSO	0	0	7					17	27	17	10	10	10	10	10
GLPSO	0	0	3					40	67	73	70	73	60	53	47
HCLPSO	0	0	3					40	30	20	20	17	13	10	13
IILPSO	3	3	3					43	53	67	57	60	57	63	53
PSO-iw	3	3	3					33	47	47	53	50	53	57	57
30-dimensional CEC 2017															
ALCPSO	27	20	20	**reference swarm sizes**				43	50	63	63	63	63	60	63
CLPSO	0	10	20					13	7	0	0	0	0	0	0
DEPSO	0	0	0					73	83	83	90	90	87	80	80
EPSO	0	0	7					37	27	17	13	3	0	0	0
GLPSO	0	0	0					50	50	63	63	63	60	60	50
HCLPSO	0	0	3					23	23	20	23	17	17	17	13
IILPSO	10	10	23					50	40	43	40	37	37	43	33
PSO-iw	0	0	3					40	40	63	57	60	63	60	60
50-dimensional CEC 2017															
ALCPSO	30	20	20	**reference swarm sizes**				43	43	53	53	57	57	57	50
CLPSO	7	13	30					3	13	0	0	0	0	0	0
DEPSO	0	0	0					83	90	90	87	90	90	87	87
EPSO	0	0	13					30	37	27	10	7	10	7	7
GLPSO	0	0	0					57	57	67	60	63	57	63	57
HCLPSO	0	0	0					30	40	37	30	33	33	27	30
IILPSO	23	20	13					40	40	37	37	37	37	37	33
PSO-iw	0	0	0					33	50	67	63	70	67	57	63
100-dimensional CEC 2017															
ALCPSO	40	30	30	**reference swarm sizes**				47	47	60	67	57	60	57	50
CLPSO	10	20	40					10	7	3	0	0	0	0	0
DEPSO	0	0	0					87	87	90	90	90	90	87	83
EPSO	0	3	13					3	7	7	3	3	3	3	0
GLPSO	0	0	0					60	53	63	57	57	53	53	57
HCLPSO	0	0	7					40	27	33	30	40	37	37	33
IILPSO	27	30	23					37	40	33	27	23	20	17	17
PSO-iw	0	0	10					43	77	77	70	77	70	77	77

perform best for any of the considered 22 real-world problems (see Table 5).

We also find that for each considered algorithm, swarms composed with 1000 particles turned out to be the best choice on some real-world problems. Moreover, for two PSO variants (DEPSO and PSO-iw) such large swarms lead to the best results on more than 40% of real-world problems. For artificial benchmarks large swarms are also the best choice for a respectable number of problems (see Tables 6 and 7), even in the case of algorithms like HCLPSO, which overall prefer moderately-sized swarms.

Very small swarms (composed of 3–5 particles) yield the best results on specific problems much more rarely. However, occasionally they do, even for high-dimensional benchmarks. For example, ALCPSO and IILPSO perform best with 3 particles on a large number of 100-dimensional problems (consult Tables 6 and 7). It is puzzling that CLPSO, the only algorithm that on average clearly prefers smaller swarms, so rarely performs best with only 3 particles. It is also interesting that two algorithms, namely ALCPSO and IILPSO, perform best either with the smallest (3), or with the largest (1000) swarm sizes for large share of problems.

The conclusion from counting the number of wins for each swarm size is hence somewhat frustrating. On the one hand, contrary to what one may expect, huge swarms may often be highly needed for some PSO algorithms on specific problems. On the other hand, very small swarms may also be required (even though more rarely) for similar PSO variants on some other problems. This should remind us that control parameter settings in metaheuristics is always problem-sensitive, and the suggested choices are almost never universal.

Finally in Tables 8–10 we verify for which percentage of problems the results may be improved if the swarm size is chosen outside of the classical [20,50] settings. We assume that the particular swarm size improves the performance on the specific problem only if the achieved results are better than the results obtained by a particular PSO variant with every considered swarm size from the reference interval (hence 20, 30, 40 and 50 – see the blank part of Tables 8–10). Hence, the fact that particular swarm size does not improve the performance over all four reference swarm sizes does not mean that it leads to poorer results. From Table 8 we see that in the case of ALCPSO, DEPSO, GLPSO and PSO-iw, the performance on over 50% of real-world problems may be improved if the swarm size is increased to at least 100 particles. IILPSO, as noted earlier in our discussion, shows an unusually even performance with various swarm sizes, hence although on average it performs best with larger swarms, improvements against all four reference swarm sizes are rarer. Results obtained by CLPSO and EPSO on real-world problems may only occasionally be improved when more than 50 particles are used. However, improving the performance by using smaller swarms is also rare for these two algorithms. In the case of artificial benchmarks (Tables 9 and 10) the percentage of problems on which all swarm sizes from reference intervals may be outperformed is often higher, and for some algorithms (DEPSO, PSO-iw) even reaches 75%–90%.

The above findings once again suggest that the classical PSO swarm size setting of 20–50, which came up twenty years ago from tests on simple low-dimensional problems, is an underestimation for many PSO variants, and does not hold for more difficult tasks.

6. Conclusions

In this study we have verified the relationship between the swarm size and the performance of Particle Swarm Optimization algorithms. We considered eight PSO variants, each tested with 15 different population sizes, ranging from 3 to 1000, on 60 scalable (10- to 100-dimensional)

artificial benchmarks and 22 real-world problems from different fields of science.

We found that the classical choice of 20–50 particles for PSO variants, which has roots in the historical papers on PSO published about two decades ago, is frequently inadequate. On average, for over half of tested PSO variants, including the classical PSO with inertia weight from the late 1990's [47], the best results are obtained when the swarm is composed of 70–500 particles. Only in the case of two PSO algorithms were the best results regularly obtained with the classical setting of 20–50 particles. If we have no hint which population size to use for a particular PSO variant, 70–100 particles is the safest choice. Whether we average results over a large number of artificial benchmarks, or over real-world problems, similar conclusions are obtained. There is, however, an exception: if unimodal problems are solely considered, the classical choice of 20–50 particles leads to the best results for most PSO variants, even though some algorithms still require much larger swarms. For difficult problems, like composition functions from the CEC 2014 and CEC 2017 benchmarks, often swarms composed of hundreds of particles are recommended, but this hint is not general and greatly depends on the specific PSO variant. Finally, problem scalability plays a limited role, as for a particular PSO variant on average almost the same population sizes yielded the best results for 10- and 100-dimensional artificial benchmarks.

However, if we look at each problem separately, we find that for some tasks very large, for others very small swarms may lead to the best results. We find that swarm sizes composed of 500–1000 particles lead to the best results on a larger number of problems than very small swarms. But still, in extreme cases some PSO algorithms perform best with swarms composed of 1000 particles on one problem, and with 3 particles on another, with the same dimensionality. This does not undermine the general conclusion from the study based on performance averaged over many problems, but highlights that in metaheuristics any control parameter setting is inevitably problem-dependent, and cannot possibly be assumed universal.

Author statement

Piotrowski AP: Conceptualization; Investigation; Methodology; Resources; Software; Validation; Writing - original draft; Writing - review & editing. Napiorkowski JJ: Software; Validation; Writing - original draft; Writing - review & editing. Piotrowska AE: Conceptualization; Investigation; Writing - original draft.

Declaration of competing interest

Authors of the manuscript entitled "Population size in Particle Swarm Optimization" (SWEVO_2019_167) declare that they have no conflict of interests.

Acknowledgments

Authors are very grateful to Dr Scott Jackson from the Institute of Molecular Biology and Biophysics, ETH Zurich for his help in language editing and corrections. This work was supported within statutory activities No 3841/E-41/S/2019 of the Ministry of Science and Higher Education of Poland.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.swevo.2020.100718.

References

[1] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Nanyang Technological University, Singapore, 2016. Technical Report.
[2] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part I: background and development, Nat. Comput. 6 (2007) 467–484.
[3] M.R. Bonyadi, Z. Michalewicz, Impacts of coefficients on movement patterns in the Particle Swarm Optimization algorithm, IEEE Trans. Evol. Comput. 21 (3) (2017) 378–390.
[4] M.R. Bonyadi, Z. Michalewicz, Particle Swarm Optimization for single objective continuous space problems: a review, Evol. Comput. 25 (1) (2017) 1–54.
[5] E.F. Campana, G. Fasano, A. Pinto, Dynamic analysis for the selection of parameters and initial population in Particle Swarm Optimization, J. Global Optim. 48 (2010) 347–397.
[6] E.F. Campana, M. Diez, G. Fasano, D. Peri, Initial particles position for PSO, in bound constrained optimization, advances in swarm intelligence, Part I, Springer Lecture Notes Comput. Sci. 7928 (2013) 112–119.
[7] A. Carlisle, G. Dozier, An off-the-shelf PSO, in: Workshop on Particle Swarm Optimization, 2001, pp. 1–6. Indianapolis, IN, USA.
[8] D.B. Chen, C.X. Zhao, Particle swarm optimization with adaptive population size and its application, Appl. Soft Comput. 9 (2009) 39–48.
[9] T.S. Chen, K. Tang, G.L. Chen, X. Yao, A large population size can be unhelpful in evolutionary algorithms, Theor. Comput. Sci. 436 (2012) 54–70.
[10] W.N. Chen, J. Zhang, Y. Lin, N. Chen, Z.H. Zhan, H.S.H. Chung, Y. Li, Y.H. Shi, Particle Swarm Optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.
[11] R. Cheng, Y.C. Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybern. 45 (2) (2015) 191–204.
[12] S. Cheng, H. Lu, X.J. Lai, Y.H. Shi, A quarter century of particle swarm optimization, Complex Intell. Syst. 4 (3) (2018) 227–239.
[13] C.W. Cleghorn, A.P. Engelbrecht, Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption, Swarm Intell. 12 (2018) 1–22.
[14] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[15] C.A. Coello Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 256–279.
[16] S. Das, P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems, Jadavpur Univ, Nanyang Technol. Univ, 2010. Technical Report.
[17] J. Del Ser, E. Osaba, D. Molina, X.S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A. Coello Coello, F. Herrera, Bio-inspired computation: where we stand and what's next, Swarm Evol. Comput. 48 (2019) 220–250.
[18] M.A.M. de Oca, T. Stutzle, K. van den Enden, M. Dorigo, Incremental social learning in particle swarms, IEEE Trans. Syst. Man Cybern. Part B Cybern. 41 (2) (2011) 368–384.
[19] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the 6th International Symposium on Micromachine Human Science, Nagoya, Japan, IEEE, Piscataway, NJ, USA, 1995, pp. 39–43.
[20] R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with Particle Swarms, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, Soul, Korea, IEEE, Piscataway, NJ, USA, 2001, pp. 94–100.
[21] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (2) (1999) 124–141.
[22] S. Garcia, F. Herrera, An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[23] M.S. Gibbs, H.R. Maier, G.C. Dandy, Using characteristics of the optimisation problem to determine the Genetic Algorithm population size when the number of evaluations is limited, Environ. Model. Software 69 (2015) 226–239.
[24] Y.J. Gong, J.J. Li, Y. Zhou, Y. Li, H.S.H. Chung, Y.H. Shi, J. Zhang, Genetic learning particle swarm optimization, IEEE Trans. Cybern. 46 (10) (2016) 2277–2290.
[25] K.R. Harrison, A.P. Engelbrecht, B.M. Ombuki-Berman, Self-adaptive particle swarm optimization: a review and analysis of convergence, Swarm Intell. 12 (2018) 187–226.
[26] S. Helwig, J. Branke, S. Mostaghim, Experimental analysis of bound handling techniques in Particle Swarm Optimization, IEEE Trans. Evol. Comput. 17 (2) (2013) 259–271.
[27] S.T. Hsieh, T.Y. Sun, C.C. Liu, S.J. Tsai, Efficient population utilization strategy for particle swarm optimizer, IEEE Trans. Syst. Man Cybern. Part B Cybern. 39 (2) (2009) 444–456.
[28] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Perth, Australia, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE, Piscataway, NJ, USA, 1995. IV 1942-1948.
[29] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[30] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Zhengzhou, China and Nanyang Technological University, Computational Intelligence Laboratory, Zhengzhou University, Singapore, 2013. Technical Report 201311.
[31] A.P. Lin, W. Sun, H.S. Yu, G.H. Wu, H.W. Tang, Global genetic learning Particle Swarm Optimization with diversity enhancement by ring topology, Swarm Evol. Comput. 44 (2019) 571–583.
[32] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning Particle Swarm Optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24.

[33] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Appl. Soft Comput. 55 (2017) 533–548.
[34] N. Lynn, M.Z. Ali, P.N. Suganthan, Population topologies for particle swarm optimization and differential evolution, Swarm Evol. Comput. 39 (2018) 24–35.
[35] R. Mallipeddi, P.N. Suganthan, Empirical study on the effect of population size on Differential Evolution algorithm, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, Piscataway, NJ, USA, 2008. Hong Kong, China.
[36] F. Marini, B. Walczak, Particle swarm optimization (PSO) A tutorial, Chemometr. Intell. Lab. Syst. 149 (2015) 153–165.
[37] R. Mendes, Population topologies and their infuence in Particle Swarm performance, PhD Thesis, Departamento de Informatica, Escola de Engenharia, Universidade de Minho, 2004.
[38] K.L. Mills, J.J. Filliben, A.L. Haines, Determining relative importance and effective settings for Genetic Algorithm control parameters, Evol. Comput. 23 (2) (2015) 309–342.
[39] Y.V. Pehlivanoglu, A new Particle Swarm Optimization method enhanced with a periodic mutation strategy and neural networks, IEEE Trans. Evol. Comput. 17 (3) (2013) 436–452.
[40] A.P. Piotrowski, Review of differential evolution population size, Swarm Evol. Comput. 32 (2017) 1–24.
[41] A.P. Piotrowski, M.J. Napiorkowski, J.J. Napiorkowski, P.M. Rowinski, Swarm intelligence and evolutionary algorithms: performance versus speed, Inf. Sci. 384 (2017) 34–85.
[42] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization an overview, Swarm Intell. 1 (2007) 33–57.
[43] Q. Qin, S. Cheng, Q.Y. Zhang, L. Li, Y.H. Shi, Particle Swarm Optimization with interswarm interactive learning strategy, IEEE Trans. Cybern. 46 (10) (2016) 2238–2251.
[44] H. Richter, A.P. Engelbrecht (Eds.), Recent Advances in the Theory and Application of Fitness Landscapes, Springer-Verlag, Berlin, Heidenberg, 2014.
[45] Z.H. Ruan, Y. Yuan, Q.X. Chen, C.X. Zhang, Y. Shuai, H.P. Tan, A new multi-function global particle swarm optimization, Appl. Soft Comput. 49 (2016) 279–291.
[46] A. Serani, C. Leotardi, U. Iemma, E.F. Campana, G. Fasano, M. Diez, Parameter selection in synchronous and asynchronous deterministic particle swarm optimization for ship hydro-dynamics problems, Appl. Soft Comput. 49 (2016) 313–334.
[47] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceeding of the IEEE Congress on Evolutionary Computation (CEC), IEEE, Piscataway, NJ, USA, 1998, pp. 69–73. Anchorange, AC, USA.
[48] Y. Shi, R. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, Washington, DC, USA, IEEE, Piscataway, NJ, USA, 1999, pp. 1945–1950.
[49] G. Tambouratzis, Using Particle Swarm Optimization to accurately identify syntactic phrases in free text, J. Artif. Intell. Soft Comput. Res. 8 (1) (2018) 63–77.
[50] R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, Bejing, China, IEEE, Piscataway, NJ, USA, 2014, pp. 1658–1665.
[51] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inf. Process. Lett. 85 (2003) 317–325.
[52] T. Tsujimoto, T. Shindo, T. Kimura, K. Jin'no, A relationship between network topology and search performance of PSO, in: WCCI 2012 IEEE World Congress on Computational Intelligence, 2012, pp. 1526–1531. Brisbane, Australia.
[53] A. Ulas, O.T. Yildiz, E. Alpaydin, Cost-conscious comparison of supervised learning algorithms over multiple data sets, Pattern Recogn. 45 (2012) 1772–1781.
[54] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[55] F. van der Bergh, A.P. Engelbrecht, A convergence proof for the particle swarm optimizer, Fundam. Inf. 105 (4) (2010) 341–374.
[56] D.S. Wang, D.P. Tan, L. Liu, Particle swarm optimization algorithms: an overview, Soft Comput. 22 (2018) 387–408.
[57] X. Wu, J. Dai, Y. Zhao, Z. Zhuo, J. Yang, X.C. Zheng, Two-dimensional boron monolayer sheets, ACS Nano 6 (8) (2012) 7443–7453.
[58] G.P. Xu, Q.L. Cui, X.H. Shi, H.W. Ge, Z.H. Zhan, H.P. Lee, Y.C. Liang, R. Tai, C.G. Wu, Particle Swarm Optimization based on dimensional learning strategy, Swarm Evol. Comput. 45 (2019) 33–51.
[59] L. Xueyan, X. Zheng, Swarm size and inertia weight selection of Particle Swarm Optimizer in system identification, in: 4th International Conference on Computer Science and Network Technology, ICCSNT, Harbin, China, 2015, pp. 1554–1556.
[60] Q. Yang, W.N. Chen, J.D. Deng, Y. Li, T.L. Gu, J. Zhang, A level-based learning swarm optimizer for large-scale optimization, IEEE Trans. Evol. Comput. 22 (4) (2018) 578–594.
[61] I.M. Yassin, M.N. Taib, R. Adnan, M.K.M. Salleh, M.K. Hamzah, Effect of swarm size parameter on binary particle swarm optimization-based NARX structure selection, in: IEEE Symposium on Industrial Electronics and Applications, Bandung, Indonesia, IEEE, Piscataway, NJ, USA, 2012.
[62] M. Zambrano-Bigiarini, M. Clerc, R. Rojas, Standard particle swarm optimization 2011 at CEC-2013: a baseline for future PSO improvements, in: Proceeding of the IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, IEEE, Piscataway, NJ, USA, 2013, pp. 2337–2344.
[63] J.Q. Zhang, X.X. Zhu, Y.H. Wang, M.C. Zhou, Dual-environmental particle swarm optimizer in noisy and noise-free environments, IEEE Trans. Cybern. 49 (6) (2019) 2011–2021.
[64] L.P. Zhang, H.J. Yu, S.X. Hu, Optimal choice of parameters for particle swarm optimization, J. Zhejiang Univ. - Sci. 6A (6) (2005) 528–534.
[65] Y.D. Zhang, S.H. Wang, G.L. Ji, A comprehensive survey on Particle Swarm Optimization algorithm and its applications, Math. Probl Eng. (2015) 931256.