On the Performance of Particle Swarm Optimization Algorithms in Solving Cheap Problems

Abdullah Al-Dujaili
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: aldujail001@e.ntu.edu.sg

M. R. Tanweer
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: muhammad170@e.ntu.edu.sg

S. Suresh
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: ssundaram@ntu.edu.sg

Abstract—Eight variants of the Particle Swarm Optimization (PSO) algorithm are discussed and experimentally compared among each other. The chosen PSO variants reflect recent research directions on PSO, namely parameter tuning, neighborhood topology, and learning strategies. The Comparing Continuous Optimizers (COCO) methodology was adopted in comparing these variants on the noiseless BBOB testbed. Based on the results, we provide useful insights regarding PSO variants’ relative efficiency and effectiveness under a cheap budget of function evaluations; and draw suggestions about which variant should be used depending on what we know about our optimization problem in terms of evaluation budget, dimensionality, and function structure. Furthermore, we propose possible future research directions addressing the limitations of latest PSO variants. We hope this paper would mark a milestone in assessing the state-of-the-art PSO algorithms, and become a reference for swarm intelligence community regarding this matter.

I. INTRODUCTION

Particle Swarm Optimization (PSO) [1] is a search-based optimization algorithm, inspired from the social behavior of bird swarm and fish schooling in search of food. It has been widely used for solving numerous optimization problem [2], [3] and poised as an efficient tool for complex real world optimization problems [4]. PSO has always been attracting researchers towards development of more efficient and robust variant because of its simplicity and lower computational cost [5], [6], [7], [8]. Recently [9], the performance of the basic PSO algorithm on BBOB testbed [10], [11] has been evaluated where it exhibited promising characteristics. The current state-of-the-art research in PSO has significantly enhanced its performance, and introduced a diverse collection of modified PSO algorithms (PSO variants), with one variant performing better than the other on a class of optimization problems rather than the set of all remaining problems. This leads to a growing gap among PSO variants, and although we can learn a lot from these variants jointly, it is not often the case.

This paper aims to bridge the gap among the recent advances by the PSO community by comparing eight different PSO variants carefully selected to reflect various aspects of the community’s recent works. Through experimental evaluation, this paper marks the differences in them, investigates the suitability of the algorithms for various optimization problems. Furthermore, it draws several concluding points that may be fruitful for the PSO community.

To achieve the paper’s goals, the experimental evaluation must be able to discover and distinguish the good features of the variants over others, and show their differences over different stages of the search for various goals and situations. We have adopted the Comparing Continuous Optimizer (COCO) methodology [12] as it meets our requirements. It comes with a testbed of 24 scalable noiseless functions [11] addressing such real-world difficulties as ill-conditioning, multi-modality, and dimensionality. It has always been argued that PSO provides better solutions if it is given a fair amount of time [13]. Therefore, all the PSO variants are tested under cheap-budget settings to test their capabilities in locating the optimum solution.

The rest of the paper is organized as follows: Section II provides a brief description of the selected PSO variants. In Section III, the numerical assessment of the algorithms is presented, starting with the experimental setup (Section III-A), and the algorithms’ empirical computational complexity (Section III-B); afterwards, the procedure for evaluating the algorithms’ performance is elaborated (Section III-C); followed by a discussion of the results (Section III-D). Section IV summarizes the main conclusions from this study, and suggests possible directions to further improve the state-of-the-art PSO.

II. SELECTED ALGORITHMS

As mentioned in the previous section, there is a large body of research on PSO involved in enhancing its performance. In the past two decades, there exist several research directions including parameter tuning, neighborhood topology, and unique learning strategies. In this paper, the following PSO variants under the mentioned research directions have been selected.

Parameter Tuning: There are numerous PSO variants with different parameter setting as summarized in [4]. Among them, the constriction factor PSO ($\chi$PSO)[5] has shown better performance on selected problems by controlling the magnitude of velocities and enhancing the convergence characteristics.

Neighbourhood Topology: The three selected PSO variants in this category are the Fully Informed Particle Swarms (FIPS) [14], the Unified PSO (UPSO) [7] and the Dynamic Multi-Swarm PSO (DMPSPO) [6]. FIPS [14] introduced a new concept of using a weighted sum of the neighboring particles for a particle’s update. UPSO [7] has intelligently utilized both the global and local search topologies within a single algorithm to benefit from both search directions. The DMPSPO algorithm

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.188
provided an entirely new neighbourhood selection strategy by introducing a dynamically changing neighborhood topology for search.

**Unique Learning Strategies:** The Comprehensive learning PSO (CLPSO) [8], Self Regulating PSO (SRPSO) [4] and improved SRPSO (iSRPSO) [15] are the variants selected from this group. CLPSO [8] has been proven to be an efficient and effective algorithm in diverging the particles towards better solutions. Recently, inspired from human learning principles, a PSO variant mimicking human self-cognitive learning strategy has been proposed as SRPSO. The SRPSO algorithm exhibits promising behavior by providing optimal/near-optimal solution with faster convergence characteristics [4]. In iSRPSO [15], more human learning principles has been incorporated to utilize human self learning and social interaction capabilities.

Furthermore, the basic PSO algorithm [1] has been selected as a baseline of performance. The detailed experimental procedures are given in the next section.

### III. Numerical Assessment

#### A. Setup

We benchmarked the selected PSO variants on 24 functions (15 instances per function) of the BBOB tested under cheap-budget settings of $10^4 \times D$ function evaluations. We used a MATLAB implementation of the algorithms retrieved from http://www.ntu.edu.sg/home/epnsugan/, with a modification on the terminating criteria of the variants to stop if the target function value $f_i$ is achieved. The parameters of all the PSO variants are set to their standard values provided in the codes. The choice of the swarm size is not trivial; for DMSPSO, we used a swarm size of 60 as it was used by its authors [6]; for the rest of the variants, a swarm size of 50 is used as a popular choice for many numerical assessments of PSO algorithms. Investigating the effect of the swarm size is beyond the scope of this paper.

#### B. Computational Complexity

In order to evaluate the complexity of the algorithms (measured in time per function evaluation), we have run the PSO variants on the function $f_8$ for $1000 \times D$ function evaluations for dimensions 2, 3, 5, 10, 20, and 40. The complexity is then measured as the total time taken by all these runs divided by the total count of function evaluations (i.e., the sum of #FEs over all the algorithm runs). The code was run on a 64-bit Windows PC with E5-1650 @ 3.20GHz CPU with 16GB of memory. As shown from Fig. 1, UPSO and DMSPSO are computationally complex as they are using neighbourhood search which requires higher computational time. On the other hand, the complexity of SRPSO is almost similar to that of the basic PSO algorithm indicating its computational efficiency. FIPS and CLPSO are utilizing personal best information of all the other particles for update process that increases the computational requirements. The other algorithms, namely $\chi$PSO and iSRPSO are more computationally complex than the basic PSO algorithm.

While this evaluation may provide an insight on the complexity of the algorithms, the structure of the function to be optimized may considerably affect their complexity. For instance, DMSPSO failed to solve (stuck for hours) $f_5$ for dimensions greater than 10. Nevertheless, we still believe that the reported timing results give a relative measure of compared algorithms’ computational complexity.

#### C. Performance Evaluation Procedure

The performance evaluation procedure is set up according to [12], where each algorithm is run on the functions given in [10], [11] with multiple trials per function. A set of target function values is specified per function. The algorithms are evaluated based on the number of function evaluations required to reach a target. The Expected Running Time (ERT) used in the coming figures and tables, depends on a given target function value, $f_i = f_{opt} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach $f_i$, summed over all trials and divided by the number of trials that actually reached $f_i$ [12], [16]. **Statistical significance** is tested with the rank-sum test for a given target $\Delta f$, using, for each trial, either the number of needed function evaluations to reach $\Delta f$ (inverted and multiplied by $-1$), or, if the target was not reached, the best $\Delta f$-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

#### D. Performance Evaluation Discussion

Results from the performance experiments are reported in Figures 2, 3, 4, 5 and 6; and in Tables I and II. Overall, Fig. 2 shows that the ERT of all variants grows on an order worse than a quadratic one with respect to the problem dimensionality. From the rest of figures and tables, the following can be stated about PSO variants.

PSO: despite of its being a baseline of performance, PSO outperforms other variants (CLPSO most of the time) in several problems, especially in lower-dimension multi-modal problems, solving around 60% of the 5–D problems, and 20% of the 40–D problems. The linear slope function $f_5$, and the Lunacek bi-Rastrigin are the most challenging problems for PSO.

![Fig. 1: Timing Performance of PSO variants. The y-axis represents the time (in millisecond) per a function evaluation (FE). All the algorithms were run on the function $f_8$ with a budget of 1000D function evaluations on a PC with: 64-bit Windows, E5-1650 @ 3.20GHz CPU, 16GB of memory.](image-url)
PSO-cf (\(\gamma\)-PSO): performs well in general over separable functions, and is overall better than PSO. One can notice that PSO-cf curves enjoy a steep increase for \(\#FEs \leq 100D\) favoring a multi-restart strategy for PSO-cf.

CLPSO: shows a remarkable behavior on separable functions with increasing dimensions surpassing the artificial “best 2009” algorithm. However, the performance degrades on the other function categories, with a pronounced gap in performance with respect to the nearest better performing algorithm.

UPSO: suffers on multi-modal functions. Similar to PSO-cf, its curves rise sharply with low number of function evaluations, favoring a multi-restart strategy.

FIPS: suffers on ill-conditioned functions, with a moderate performance over all other function categories. It appears that FIPS is suitable for intermediate-dimensionality problems (e.g., \(10 – D\)).

DMSPSO: addresses moderate and ill-conditioned functions far better than the rest of the algorithms, although the performance gap diminishes with dimensionality. Moreover, it also shows a robust top performance over other function categories. However, we have noticed two main drawbacks: its computational complexity, and that it was not able to handle (stuck for hours) the linear slope function \((f_5)\) in higher dimensions \((D > 10)\).

SRPSO & iSRPSO: their performances are coupled most of the time, with a pronounced improvement by iSRPSO on multi-modal functions over SRPSO and other variants in higher dimensions. However, this is not the case for separable functions.

IV. CONCLUSION

This paper provides an extensive comparison of eight PSO variants on the noiseless BBOB testbed under cheap-budget settings. Based on the results, DMSPSO should be able to produce a good-quality solution if a large evaluation budget is given. Furthermore, the following remarks can be made:

Algorithms Suitability: PSO-cf, CLPSO, DMSPSO, and iSRPSO are suitable for small number of function evaluations, separable functions, ill-conditioned problems, and multi-modal functions, respectively. For the rest of the problems, on the other hand, DMSPSO is suitable.

Algorithms Rectifications: In general, weakly structured multi-modal functions (e.g., \(f_{24}\)) impose a challenge on all PSO algorithms. DMSPSO, being one of the top performers, should be redesigned to consider functions of linear slope (e.g., \(f_5\)). The strikingly different behavior of CLPSO on separable functions from its behavior on the rest of all problems is an interesting investigation to pursue. Multi-restart strategies may be of a great advantage to low-complexity algorithms such as PSO-cf.

Data, Code, and Future Benchmarking: The data of these experiments will be made available on the BBOB webpage [12]. Furthermore, a repository for PSO variant codes and their data will be hosted online at https://sites.google.com/site/psobenchmark/. The authors invite researchers to compare their PSO variants against the discussed algorithms, providing the community with a prime reference on the relative performance of past and future PSO variants.

ACKNOWLEDGMENT

The authors wish to extend their thanks to the ATMRI:2014-R8, Singapore, for providing financial support to conduct this study. Thanks extended to Prof. P. N. Suganthan, EEE-NTU, Singapore for sharing the codes.

REFERENCES

Fig. 2: Expected running time (ERT in number of \( f \)-evaluations as \( \log_{10} \) value), divided by dimension for target function value \( 10^{-8} \) versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of \( f_1 \) and \( f_{24} \). Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with \( p < 0.01 \) and Bonferroni correction number of dimensions (six). Legend: ○:PSO, ▽:PSO-cf, ⋆:SRPSO, ◻:iSRPSO, △:FIPS, ♢:UPSO, ◇:CLPSO, ◊:DMPSO
Fig. 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.

Fig. 4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 10-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.
Fig. 5: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in $10^{[−8..2]}$ for all functions and subgroups in 20-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.

Fig. 6: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 50 targets in $10^{[−8..2]}$ for all functions and subgroups in 40-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.
<table>
<thead>
<tr>
<th>( \Delta f )</th>
<th>( \Delta \tau )</th>
<th>SRPSO</th>
<th>DMSPSO</th>
<th>CLPSO</th>
<th>PSO-cf</th>
<th>FIPS</th>
<th>ERM</th>
</tr>
</thead>
<tbody>
<tr>
<td>( = 0 )</td>
<td>( = 0 )</td>
<td>( &lt; 0.1 )</td>
<td>( &lt; 0.1 )</td>
<td>( &lt; 0.1 )</td>
<td>( &lt; 0.1 )</td>
<td>( &lt; 0.1 )</td>
<td>( &lt; 0.1 )</td>
</tr>
<tr>
<td>( \neq 0 )</td>
<td>( \neq 0 )</td>
<td>( \neq 0 )</td>
<td>( \neq 0 )</td>
<td>( \neq 0 )</td>
<td>( \neq 0 )</td>
<td>( \neq 0 )</td>
<td>( \neq 0 )</td>
</tr>
</tbody>
</table>

### TABLE 1: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 5. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and target, the corresponding best ERT in the first row. The different target \( \Delta f \)-values are shown in the top row. \#suc is the number of trials that reached the (final) target \( f_{\text{opt}} + 10^{-8} \). The median number of conducted function evaluations is additionally given in italics, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with \( p = 0.05 \) or \( p = 10^{-8} \) when the number \( k \) following the star is larger than 1, with Bonferroni correction by the following the star is larger than 1.
| Entry | PSO | PSO-cf | DMSPSO | CLPSO | iSRPSO | PSO-cf | PSO | DMSPSO | CLPSO | iSRPSO | PSO-cf | PSO | DMSPSO | CLPSO | iSRPSO | PSO-cf | PSO | DMSPSO | CLPSO | iSRPSO | PSO-cf | PSO | DMSPSO | CLPSO | iSRPSO | PSO-cf | PSO | DMSPSO | CLPSO | iSRPSO | PSO-cf | PSO | DMSPSO | CLPSO | iSRPSO |
|-------|-----|--------|--------|-------|--------|--------|-----|--------|-------|--------|--------|-----|--------|-------|--------|--------|-----|--------|-------|--------|--------|-----|--------|-------|--------|--------|-----|--------|-------|--------|--------|-----|--------|-------|--------|--------|-----|--------|-------|--------|--------|-----|--------|-------|--------|
| $\Delta f$ | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 |
| succ | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 |
| $f_{opt}$ | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 |
| $n$ | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 |
| $t_{run}$ | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 | 1e0 |

** TABLE II:** Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 in dimension 20. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm according to ERT in the first row. The different target $\Delta f$-values are shown in the top row. #succ is the number of trials that reached the (final) target $f_{opt} + 10^{-8}$. The median number of conducted function evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-6}$ when the number $k$ following the star is larger than 1, with Bonferroni correction by the number of instances.