# A Self-adaptive Heterogeneous PSO Inspired by Ants

Filipe V. Nepomuceno and Andries P. Engelbrecht

Department of Computer Science, University of Pretoria, Pretoria, South Africa
filinep@gmail.com, engel@cs.up.ac.za

**Abstract.** Heterogeneous particle swarm optimizers have been proposed where particles are allowed to implement different behaviors. A selected behavior may not be optimal for the duration of the search process. Since the optimality of a behavior depends on the fitness landscape it is necessary that particles be able to dynamically adapt their behaviors. This paper introduces two new self-adaptive heterogeneous particle swarm optimizers which are influenced by the ant colony optimization meta-heuristic. These self-adaptive strategies are compared with three other heterogeneous particle swarm optimizers. The results show that the proposed models outrank the existing models overall.

**Keywords:** particle swarm optimization, heterogeneous, self-adaptive, pheromone, ant colony optimization

## 1 Introduction

Particle swarm optimization (PSO) is a stochastic optimization technique introduced by Kennedy and Eberhart [6, 11]. PSOs contain a swarm of particles that move around in an $n$-dimensional search space trying to find an optimal solution to an optimization problem. Each particle's position represents a candidate solution and each particle has a velocity which determines the particle's next position. Homogeneous PSOs use the same position update and the same velocity update for all particles. This means that all the particles exhibit the same search patterns.

One of the problems that PSOs face is the exploration versus exploitation problem [8]. By making the PSO heterogeneous, the swarm can have multiple search patterns. This allows a better balance between exploration and exploitation. Previous work done on heterogeneous PSO (HPSO) includes experiments by Montes de Oca et al [15], the TRIBES PSO [3], the static HPSO (sHPSO) and dynamic HPSO (dHPSO) models [8], the adaptive learning PSO-II (ALPSO-II) [13], and the difference proportional probability PSO (DPP-PSO) [18]. Some of these approaches have shortcomings, e.g. the ALPSO-II is computationally expensive, and the DPP-PSO, sHPSO and dHPSO cannot adapt to the changing search landscape characteristics as the particles move through the search space.

This paper aims to overcome these shortcomings by proposing two self-adaptive heterogeneous models inspired by the foraging behavior of ants modeled with the ant colony optimization meta-heuristic (ACO-MH) [5]. The swarm

contains a behavior pool which contains different behaviors (update equations). The behaviors have an associated pheromone concentration which determines the probability of a particle choosing that behavior.

The proposed strategies are evaluated on a subset of the CEC 2005 benchmark functions and compared to the dHPSO, ALPSO-II and DPP-PSO. The results show that the proposed self-adaptive HPSO strategies perform well on unimodal and multimodal functions.

The rest of this paper is organized as follows: Section 2 provides background on PSOs and existing HPSO algorithms used for comparison in this paper. Section 3 introduces the proposed self-adaptive strategies. Section 4 details the experimental procedure followed and the results are presented and analyzed in Section 5. The paper is concluded in Section 6.

## 2 Background

This section provides details on the original PSO and heterogeneous PSO.

### 2.1 Particle Swarm Optimization

The original PSO is based on the papers by Kennedy and Eberhart [6, 11]. Each particle keeps track of the best position it has found called the particle's personal best or *pbest*. Particles are grouped together into neighborhoods and the best position of all the particles' *pbest*s, called the neighbourhood best or *nbest*, is recorded. Particles initially start in random positions in the search space and iteratively update their velocity and position. The following equations, with the addition of the inertia weight introduced by Shi and Eberhart [17], are used:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)\left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_{2j}(t)\left(\hat{y}_j(t) - x_{ij}(t)\right) \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

where $v_{ij}(t)$ represents the $i^{th}$ particle's velocity in the $j^{th}$ dimension at iteration $t$, $x_{ij}(t)$ is the particle's position, $y_{ij}(t)$ is the particle's *pbest*, $\hat{\mathbf{y}}(t)$ is the neighborhood's *nbest*, $w$ is the inertia weight to avoid sudden changes in direction, $r_{1j}, r2j \sim U(0,1)$ and $c_1$ and $c_2$ are constant acceleration coefficients. The velocity and position updates define a particle's search behavior.

### 2.2 Heterogeneous Particle Swarm Optimization

HPSO algorithms have a behavior pool and a behavior selection strategy. Behaviors which have been used in behavior pools are

- Original PSO with inertia weight (PSO) [6, 11, 17]
- Cognitive-only model (Cog-PSO) [9]
- Social-only model (Soc-PSO) [9]

- Bare-bones (BB-PSO) and modified bare-bones (BBMod-PSO) [10]
- Quantum PSO (QSO) [1]
- Time-varying inertia weight PSO (TVIW-PSO) [17]
- Time-varying acceleration coefficients (TVAC-PSO) [16]
- Fully informed particle swarm (FIPS) [14]

The dynamic HPSO (dHPSO), proposed by Engelbrecht [8], selects behaviors randomly from the behavior pool if the particle's *pbest* has not improved for ten iterations. A critique against the dHPSO is that the behavior selection is random and no information about the search process is used to guide the selection towards the most promising behaviors.

Spanvello and Montes de Oca's [18] difference proportional probability PSO (DPP-PSO) allows particles to change their behaviors to the *nbest*'s behavior based on a probability calculated by how much better the *nbest*'s fitness is compared to their own. Additionally, the DPP-PSO has a number of rigid particles for each behavior which do not change their behavior at all. This is to ensure that all behaviors occur in the swarm. One of the issues with the DPP-PSO is that it only contains two behaviors (FIPS and the original PSO with inertia weight) so there is not an adequate range of behavioral diversity to help in the swarm's search.

Li and Yang's [13] adaptive learning particle swarm optimizer-II (ALPSO-II) works as follows: progress values are kept for each particle and rewards are calculated based on the progress. A new behavior is then chosen probabilistically based on a selection ratio calculated using the rewards. The ALPSO-II also contains an archive position, *abest*, which stores the best solution found throughout the whole search. Every iteration each particle has a chance to update the *abest* position by replacing each element of the *abest* individually. If the replacement results in a better fitness, then the *abest* keeps that element. The main issue with ALPSO-II is that it is overly complex and computationally expensive. The *abest* update procedure can increase the number of function evaluations per iteration drastically for higher dimensional problems.

## 3 Pheromone Based Particle Swarm Optimizer

This section introduces the two proposed self-adaptive HPSO strategies.

Both the strategies' behavior pools consist of the original PSO with inertia weight, Cog-PSO, Soc-PSO, BB-PSO, BBMod-PSO, QSO, TVIW-PSO and TVAC-PSO.

The self-adaptive strategies are inspired by the foraging behavior of ants as modeled in the ant colony optimization meta-heuristic (ACO-MH) [5]. Ants are able to find the shortest route to a food source by secreting pheromone on the trails to the food source. Ants probabilistically follow the routes with the higher concentrations of pheromone. Over time, the pheromone evaporates making the routes less likely to be chosen if the ants do not continue using the route.

The pheromone heterogeneous particle swarm optimizer (pHPSO) uses these concepts to select a behavior for a particle. Each behavior, $b$, in the behavior pool

represents a candidate route, the particles represent ants and the pheromone concentration, $p_b$, represents the fitness of a behavior. Fitter behaviors have higher pheromone concentrations.

Each particle is initially assigned a random behavior from the behavior pool with the probability $\frac{1}{B}$ where $B$ is the total number of behaviors in the behavior pool. The initial pheromone concentration of each behavior is also $\frac{1}{B}$. Each iteration consists of four phases: behavior selection, particle update, pheromone update and pheromone evaporation.

During behavior selection, a new behavior is probabilistically chosen using probabilities computed using the pheromone concentrations when a particle stagnates. A particle is considered stagnant if its *pbest* does not improve for ten iterations as per [12]. The probability for choosing behavior $b$ is

$$prob_b(t) = \frac{p_b}{\sum_{i=1}^{B} p_i} \tag{3}$$

The particles then update their positions and velocities using their assigned behaviors during the particle update phase.

The next phase is updating the pheromone concentrations. Assuming minimization, the pheromone concentration is updated using one of the following strategies:

– Constant strategy (pHPSO-const)

$$p_b(t) = p_b(t) + \sum_{i=1}^{S_b} \begin{cases} 1.0 & \text{if } f\left(x_i(t)\right) < f\left(x_i(t-1)\right) \\ 0.5 & \text{if } f\left(x_i(t)\right) = f\left(x_i(t-1)\right) \\ 0.0 & \text{if } f\left(x_i(t)\right) > f\left(x_i(t-1)\right) \end{cases} \tag{4}$$

where $S_b$ is the number of particles using behavior $b$. This strategy rewards behaviors if they improve or maintain a particle's fitness regardless of the magnitude of the improvement. The update values were chosen as a starting point and other values will be investigated in the future.

– Performance based (linear) strategy (pHPSO-lin)

$$p_b(t) = p_b(t) + \sum_{i=1}^{S_b} \left(f\left(x_i(t-1)\right) - f\left(x_i(t)\right)\right) \tag{5}$$

Using this strategy behaviors are rewarded in proportion to the improvement of a particle's fitness over two iterations. A minimum concentration of 0.01 is used to prevent zero and negative concentrations.

Roulette wheel selection is biased to the more successful behaviors. To maintain diversity in behavior space and enhance its exploration, pheromone concentrations of all behaviors evaporate using

$$p_b(t+1) = \frac{\left(\sum_{i=1, i\neq b}^{B} p_i\right)}{\sum_{i=1}^{B} p_i} \times p_b \tag{6}$$

The amount of evaporation is proportional to a behavior's pheromone concentration, e.g. if $p_b(t)$ is 90% of the total pheromone concentrations then evaporation will decrease it by 90%.

In comparison to the heterogeneous strategies discussed in Section 3, the pheromone-based self-adaptive HPSO strategies have the following advantages:

- computationally less expensive than the ALPSO-II with less control parameters,
- behaviors are self-adapted based on the success of individual behaviors, and
- better exploration of the behavior space by using pheromone evaporation

## 4 Experimental Setup

This section provides information on the functions used to compare the different heterogeneous PSO strategies as well as the control parameters used for the algorithms. All functions and algorithms were implemented in CIlib (http://www.cilib.net). The test functions used to compare the different algorithms are functions $f_1 - f_{14}$ of the CEC 2005 benchmark functions [19]. The experiments were run 50 times on each function for 1000 iterations in dimensions 10 to 100 in increments of ten. Each swarm contained 30 particles.

The behaviors used an inertia weight, $w$, of 0.729844 and acceleration coefficients, $c_1$ and $c_2$, of 1.496180 [2]. The QSO used a cloud radius of 5, a value in between the values used in [12] and [1]. The TVIW-PSO decreased its inertia weight from 0.9 to 0.4 [7] and the TVAC-PSO increased its social acceleration coefficient from 0 to 2.5 and decreased its cognitive acceleration coefficient from 2.5 to 0 [16]. The value of 0 for the minimums was used for the behavior to mimic the Cog-PSO and Soc-PSO. The DPP-PSO used five rigid particles per behavior and $\beta = 5$ [18].

## 5 Results and Analysis

This section analyzes the results obtained for the experiments.

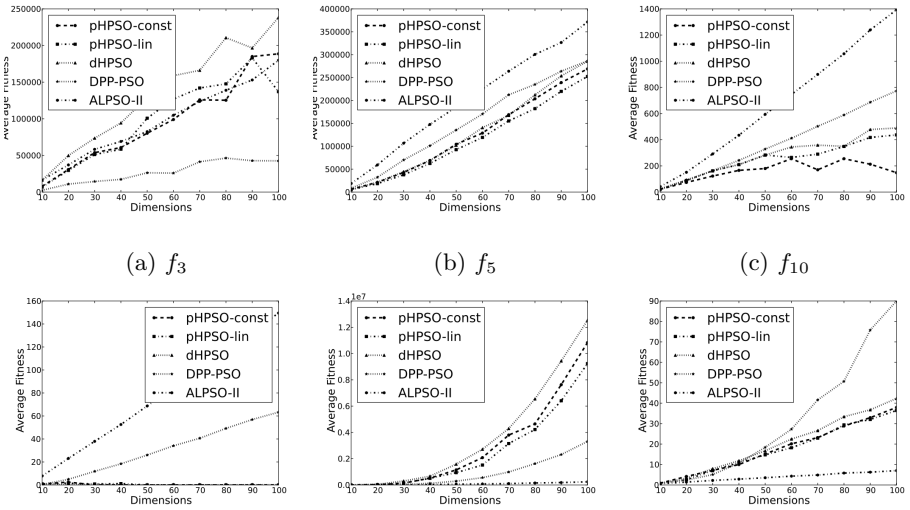
Table 1 summarizes the average rank of each algorithm over all the functions for each dimension. The numbers in bold indicate the highest rankings. In terms of scalability, the pHPSOs obtained the highest ranks for each dimension with the pHPSO-const obtaining the best overall rank. The linear pHPSO achieved better ranks in the lower dimensions and the constant pHPSO in the higher dimensions. The other algorithms ranked very similarly to each other overall with the DPP-PSO ranking the best of the three followed by the dHPSO then the ALPSO-II.

Figure 1 visualizes the scalability results for certain functions. The pHPSO strategies appeared to be unaffected by the increase in dimensionality for functions $f_1$, $f_2$, $f_4$, $f_6$, $f_7$ and $f_{11}$. For functions $f_3$, $f_5$, $f_{10}$, $f_{13}$ and $f_{14}$ the pHPSO strategies scaled linearly, and exponentially for functions $f_9$ and $f_{12}$. The $f_8$ function was the only one with a logistic trend. Compared to the ALPSO-II, the

**Table 1.** Average Ranks Over All Functions per Dimension

| Dimensions | ALPSO-II | DPP-PSO | dHPSO | pHPSO-const | pHPSO-lin |
|---|---|---|---|---|---|
| 10 | 4.15±1.64 | 3.07±0.88 | 3.14±1.55 | 2.43±0.72 | **2.21±1.15** |
| 20 | 3.86±1.68 | 3.36±1.11 | 3.07±1.33 | 2.75±1.24 | **1.96±0.77** |
| 30 | 3.57±1.68 | 3.5±1.05 | 3.43±1.45 | 2.36±0.97 | **2.14±1.06** |
| 40 | 3.57±1.68 | 3.64±0.97 | 3.29±1.39 | **2.07±0.88** | 2.43±1.24 |
| 50 | 3.43±1.63 | 3.43±0.98 | 3.64±1.44 | 2.36±1.29 | **2.14±0.83** |
| 60 | 3.43±1.64 | 3.71±0.88 | 3.43±1.45 | **2.0±0.93** | 2.43±1.18 |
| 70 | 3.36±1.67 | 3.29±1.03 | 3.64±1.34 | **2.21±1.32** | 2.5±1.05 |
| 80 | 3.5±1.68 | 3.29±0.88 | 3.71±1.39 | **2.0±0.85** | 2.5±1.3 |
| 90 | 3.36±1.67 | 3.5±0.91 | 3.79±1.42 | **2.0±1.07** | 2.35±0.89 |
| 100 | 3.5±1.68 | 3.43±0.82 | 3.57±1.45 | 2.29±1.28 | **2.21±0.94** |
| Mean | 3.57±0.23 | 3.42±0.18 | 3.47±0.23 | **2.24±0.23** | 2.3±0.17 |

pHPSO strategies scaled better on five of the functions, worse on three functions and the same on six of the functions. The DPP-PSO algorithm scaled better on nine of the functions, worse on two of the functions and the same on two of the functions. Compared to the dHPSO the pHPSO strategies scaled better on five of the functions and the same on the rest of the functions. The scalability patterns of the dHPSO were similar to the pheromone HPSO strategies.




(a) $f_3$ (b) $f_5$ (c) $f_{10}$

(d) $f_{11}$ (e) $f_{12}$ (f) $f_{13}$

**Fig. 1.** Scalability for Functions $f_3$, $f_5$ and $f_{10}$ to $f_{13}$

The Friedman test with the Bonferroni-Dunn post-hoc test [4] was used to show that, with 95% confidence, the pHPSO strategies perform significantly bet-

ter than the other strategies over all dimensions and functions. Figure 2 visualizes the comparison of the algorithms with the Bonferroni-Dunn test. Algorithms grouped with a bold line indicate that there is no significant difference between them.

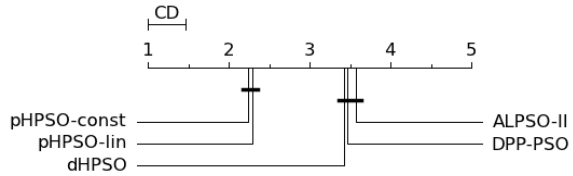



**Fig. 2.** Critical difference (CD) diagram comparing the algorithms with the Bonferroni-Dunn post-hoc test

## 6 Conclusion

This paper proposed two new strategies based on the pheromone updates of the ant colony optimization meta-heuristic (ACO-MH) to self-adapt behaviors in a heterogeneous particle swarm optimizer (HPSO). The two strategies select behaviors based on whether those behaviors are performing well in the current search landscape. Compared to existing HPSO algorithms the new strategies are computationally simpler and are self-adaptive.

Comparisons with the dynamic HPSO (dHPSO), difference proportional probability PSO (DPP-PSO) and adaptive learning PSO-II (ALPSO-II) showed that the new strategies outranked the others when evaluated on a number of unimodal and multimodal functions and that they scale relatively well with an increase in dimensionality.

Future studies include an analysis of the behavior pool, using different behavioral change triggers, a sensitivity analysis of the pheromone-based HPSO's parameters and comparing the behavior selection capabilities with other models using the same behavior pool.

## References

1. Blackwell, T., Branke, J.: Multi-swarm Optimization in Dynamic Environments. In: Raidl, G., Cagnoni, S., Branke, J., Corne, D., Drechsler, R., Jin, Y., Johnson, C., Machado, P., Marchiori, E., Rothlauf, F., Smith, G., Squillero, G. (eds.) Applications of Evolutionary Computing, Lecture Notes in Computer Science, vol. 3005, pp. 489–500. Springer Berlin / Heidelberg (2004)
2. Clerc, M., Kennedy, J.: The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation pp. 58–73 (2002)

3. Cooren, Y., Clerc, M., Siarry, P.: Performance Evaluation of TRIBES, an Adaptive Particle Swarm Optimization Algorithm. Swarm Intelligence 3, 149–178 (2009)
4. Demšar, J.: Statistical Comparisons of Classifiers Over Multiple Data Sets. The Journal of Machine Learning Research pp. 1–30 (2006)
5. Dorigo, M.: Optimization, Learning and Natural Algorithms (in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy (1992)
6. Eberhart, R., Kennedy, J.: A New Optimizer using Particle Swarm Theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. pp. 39–43 (1995)
7. Eberhart, R., Shi, Y.: Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 84–88 (2000)
8. Engelbrecht, A.: Heterogeneous Particle Swarm Optimization. In: Dorigo, M., Birattari, M., Di Caro, G., Doursat, R., Engelbrecht, A., Floreano, D., Gambardella, L., Groß, R., Sahin, E., Sayama, H., Stützle, T. (eds.) Swarm Intelligence, Lecture Notes in Computer Science, vol. 6234, pp. 191–202. Springer Berlin / Heidelberg (2010)
9. Kennedy, J.: The Particle Swarm: Social Adaptation of Knowledge. In: Proceedings of the IEEE International Congress on Evolutionary Computation. pp. 303–308 (1997)
10. Kennedy, J.: Bare Bones Particle Swarms. In: Proceedings of the IEEE Swarm Intelligence Symposium. pp. 80–87 (2003)
11. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks. pp. 1942–1948 (1995)
12. Leonard, B., Engelbrecht, A., van Wyk, A.: Heterogeneous Particle Swarms in Dynamic Environments. In: Proceedings of the IEEE Swarm Intelligence Symposium. pp. 1–8 (2011)
13. Li, C., Yang, S.: Adaptive Learning Particle Swarm Optimizer-II for Global Optimization. In: Proceedings of the Congress on Evolutionary Computation. pp. 1–8 (2010)
14. Mendes, R., Kennedy, J., Neves, J.: The Fully Informed Particle Swarm: Simpler, Maybe Better. IEEE Transactions on Evolutionary Computation pp. 204–210 (2004)
15. Montes de Oca, M., Peña, J., Stützle, T., Pinciroli, C., Dorigo, M.: Heterogeneous Particle Swarm Optimizers. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 698–705 (2009)
16. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients. IEEE Transactions on Evolutionary Computation pp. 240–255 (2004)
17. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 69–73 (1998)
18. Spanevello, P., Montes de Oca, M.: Experiments on Adaptive Heterogeneous PSO Algorithms. In: Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search Algorithms. pp. 36–40 (2009)
19. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real Parameter Optimization. Tech. rep., Nanyang Technological University (2005)