A Self-adaptive Heterogeneous PSO for Real-Parameter Optimization

Filipe V. Nepomuceno
Department of Computer Science
University of Pretoria
Pretoria, South Africa, 0184
Email: filinep@gmail.com

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
Pretoria, South Africa, 0184
Email: engel@cs.up.ac.za

***Abstract*—Heterogeneous particle swarm optimizers (HPSO) allow particles to use different update equations, referred to as behaviors, within the swarm. Dynamic HPSOs allow the particles to change their behaviors during the search. These HPSOs alter the exploration/exploitation balance during the search which alters the search behavior of the swarm. This paper introduces a new self-adaptive HPSO and compares it with other HPSO algorithms on the CEC 2013 real-parameter optimization benchmark functions. The proposed algorithm keeps track of how successful each behavior has been over a number of iterations and uses that information to select the next behavior of a particle. The results show that the proposed algorithm outperforms existing HPSO algorithms on the benchmark functions.**

***Keywords*—Heterogeneous, particle swarm optimization, real parameter optimization, self-adaptive**

I. Introduction

Particle swarm optimization (PSO) is a popular stochastic search technique introduced by Kennedy and Eberhart in 1995 [6], [11] to model the swarming behavior of birds. PSO consists of a swarm of particles, each of which contain a position and a velocity. The particles then iteratively update their velocities and positions by taking into account their past experiences and the experience of the swarm.

In most PSOs every particle uses the same position and velocity update equations. The combination of the position and velocity update defines the behavior of the particles and collectively that of the swarm. One behavior may not always be optimal throughout the search process. Heterogeneous PSOs allow the particles to use different behaviors to each other [8], [16]. This is achieved by keeping a behavior pool from which the particles can choose their behaviors. Using multiple behaviors with different exploitation/exploration capabilities allows the algorithm to control when to explore and when to exploit [8].

The aim of this paper is to introduce a simple self-adaptive HPSO which automatically alters its exploration/exploitation balance. The proposed method keeps track of the frequency of success of the behaviors used by the particles for a number of iterations. This information is then used to select a new behavior when a particle has to change its behavior. The proposed method is called the frequency-based HPSO or f_k-HPSO where k refers to the number of iterations for which the success count is kept. The proposed method is compared with other HPSOs on the CEC 2013 benchmark function set for real parameter optimization [15].

The rest of this paper is organized as follows: Section II provides background information on PSOs and HPSOs. Section III introduces the proposed algorithm. The experimental setup including parameter selection and the benchmark function set is described in Section IV. Section V presents an analysis of the results and the paper is concluded in Section VI.

II. Background

This section provides background information on PSO, some of its variants and HPSO.

A. Particle Swarm Optimization

Particle swarm optimization is a popular stochastic optimization technique due to its simplicity and success in finding good solutions [4]. PSO consists of a swarm of particles which move around in an n-dimensional search space aiming to optimize a given function. Each particle has a position, which represents a candidate solution, and a velocity which determines the particle's next position. Each candidate solution has an associated fitness which is the value of the function being optimized at the point the candidate solution represents. The particles also keep track of the best position they have visited so far, called the personal best or $pbest$. The best particle in the swarm is called the global best or $gbest$ which is the particle with the best $pbest$. The swarm can also be separated into local neighborhoods in which case a neighborhood best or $nbest$. Every iteration each particle updates its velocity and position according to

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

where $v_{ij}(t)$ is particle i's velocity at time step t in dimension j, w is an inertia weight to scale the previous velocity, $x_{ij}(t)$ is the particle's position, $y_{ij}(t)$ is the particle's $pbest$, $\hat{y}_{ij}(t)$ is the $nbest$, c_1 and c_2 are acceleration coefficients used to scale the influence of the $pbest$ and $nbest$ and $r_{1j}, r_{2j} \sim U(0,1)$. Once the particles have updated their position their $pbest$s and $nbest$s are updated if the new positions improved the fitness of the particle.

Changing the values of c_1, the cognitive acceleration, and c_2, the social acceleration, alters the exploration and exploitation capabilities of the PSO. Larger values of c_1 compared to c_2 results in the particles moving more to their $pbest$ which facilitates more exploration. Larger values of c_2 compared to c_1 results in particles moving more to the $nbest$ which facilitates more exploitation [7].

Ratnaweera et al modified the PSO by changing the acceleration coefficients over time [18]. This variant is called the time-varying acceleration coefficient PSO (TVAC-PSO). The cognitive acceleration starts with a higher value than c_2 and linearly decreases while the social acceleration starts with a lower value and linearly increases. The linear change is done using

$$c_1(t+1) = (c_{1,final} - c_{1,initial}) \times \frac{t}{t_{max}} + c_{1,initial} \quad (3)$$

$$c_2(t+1) = (c_{2,final} - c_{2,initial}) \times \frac{t}{t_{max}} + c_{2,initial} \quad (4)$$

where c_{final} and $c_{initial}$ are the final and initial values of the acceleration coefficient respectively and t_{max} is the maximum number of iterations. Note that c_1 and c_2 are now functions over time.

Kennedy investigated different PSO models which removes each of the acceleration coefficients from the equation [10]. These models are called the social only model when there is no cognitive component, i.e. $c_1 = 0$, and the cognition only model when there is no social acceleration, i.e. $c_2 = 0$. The velocity update equation for the social-only model (sPSO) is

$$v_{ij}(t+1) = wv_{ij}(t) + c_2 r_{2j}(t)(x_{ij}(t) - \hat{y}_j(t)) \quad (5)$$

and for the cognition-only model (cPSO) is

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)(x_{ij}(t) - y_{ij}(t)) \quad (6)$$

Early on in the development of the PSO it was found that the velocity of the particles increases very quickly [6]. This causes the particles to search outside of the search area and the swarm can diverge [7]. To overcome the increase in velocity an upper bound, $\mathbf{v}_{max}$, was placed on the velocity [6]. The velocity update then becomes

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } -v_{max,j} < v'_{ij}(t+1) < v_{max,j} \\ -v_{max,j} & \text{if } v'_{ij}(t+1) \leq -v_{max,j} \\ v_{max,j} & \text{if } v'_{ij}(t+1) \geq v_{max,j} \end{cases} \quad (7)$$

where $v'_{ij}(t+1)$ is the velocity calculated using equation (1).

Shi and Eberhart added the inertia weight to make the particles gradually slow down which also affects exploration and exploitation [19]. High values of w prevents particles from slowing down more than lower values which is good for exploring the search space. Lower values of w allow particles to exploit a good region without overshooting a position too much. It was found that linearly decreasing the inertia weight from 0.9 to 0.4 produces good results [19]. The inertia weight is decreased according to

$$w(t+1) = (w_{final} - w_{initial}) \times \frac{t}{t_{max}} + w_{initial} \quad (8)$$

where w_{final} and $w_{initial}$ are the final and initial values of the inertia weight respectively.

Kennedy found that, during the search, the positions of the particles approximate a Gaussian distribution centered on the average of the $pbest$ and $nbest$ with a deviation of $\sigma = |y_{ij}(t) - \hat{y}_{ij}(t)|$. This PSO variant is called the bare bones PSO (BB-PSO) and consists of only a position update using

$$x_{ij}(t+1) \sim N\left(\frac{y_{ij}(t) + \hat{y}_j(t)}{2}, \sigma\right) \quad (9)$$

Another BB-PSO variant was developed to exploit the $pbest$, called the modified BB-PSO (modBB-PSO). The position update equation of the modBB-PSO is

$$x_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < ep \\ N\left(\frac{y_{ij}(t)+\hat{y}_j(t)}{2}, \sigma\right) & \text{otherwise} \end{cases} \quad (10)$$

where $ep = 0.5$ is the exploitation probability. The exploitation probability determines how often the $pbest$ is exploited.

The use of local topological neighborhoods slows down the propagation of information about good search areas [12]. This prevents the PSO from converging on a solution too quickly and maintains the diversity of the swarm for longer compared to a global topology. Some topological structures used for PSO include the ring topology [6], von Neumann topology, wheel topology and fully connected topology [12]. The von Neumann topology has been shown to produce good results for many problems [12].

The quantum PSO (QSO), proposed by Blackwell and Branke [3], was developed to maintain swarm diversity in dynamic environments. A percentage of the swarm updates its position randomly within the radius of a cloud around the $gbest$ while the rest of the swarm uses the standard update equations. The new position of a quantum particle becomes

$$x_{ij}(t+1) \sim N(\hat{y}_j, \sigma) \quad (11)$$

where σ is the radius of the quantum cloud.

B. Heterogeneous Particle Swarm Optimization

PSOs which use the same position and velocity update equations for all the particles in the swarm are homogeneous. This means that the particles all exhibit the same search behavior [8]. However, it might not always be optimal to use a single behavior for the whole search procedure. Different exploration and exploitation capabilities can be introduced into the swarm by allowing particles to use different behaviors from one another. These PSO variants are heterogeneous PSOs (HPSO). To allow HPSO to use different behaviors a behavior pool is maintained.

Three types of HPSO exist:

Static HPSO Particles are given random behaviors in the beginning of the search and maintain those behaviors for the whole search process

Dynamic HPSO Particles can change their behaviors during the search. The next behavior of a particle is selected deterministically or stochastically

Self-adaptive HPSO Particles select their next behavior based on the success or failure of the behaviors

Montes de Oca et al experimented with static HPSO using two behaviors: PSO with inertia weight and the fully-informed PSO (FIPS) [16]. The particles were given one of the two behaviors in different proportions. The outcome was that the HPSO performed better than the worst of the two individual behaviors.

Engelbrecht [8] introduced the dynamic HPSO (dHPSO) which assigns a random behavior to a particle if that particle has stagnated. A particle is considered stagnant if its $pbest$ does not improve for a number of iterations. The dHPSO uses five behaviors: TVAC-PSO, sPSO, cPSO, BB-PSO and modBB-PSO. Engelbrecht also showed that the dHPSO scales well with an increase in dimensionality [9]. Leonard and Engelbrecht applied the dHPSO to dynamic environments [13].

Spanvello and Montes de Oca [21] created an adaptive HPSO in which particles take on the behavior of the $nbest$ according to

$$p_{ij} = \frac{1}{1 + \exp\left(-\beta \frac{f(y_i(t)) - f(y_j(t))}{|f(y_j(t))|}\right)} \tag{12}$$

where p_{ij} is the probability that particle i will adopt particle j's behavior and β is a constant that determines how sensitive particle i is to the difference in fitness between particle j and itself. The probability is proportional to the difference between two particles' $pbest$ fitnesses. This variant is called the difference proportional probability PSO (DPP-PSO). The DPP-PSO uses the PSO with inertia weight and FIPS in its behavior pool.

Li and Yang developed the adaptive learning PSO II (ALPSO-II) [14] which contains four behaviors. Progress values are kept for each particle and rewards are calculated based on the progress values. A new behavior is then chosen probabilistically based on a selection ratio calculated using the rewards.

Nepomuceno and Engelbrecht created two self-adaptive HPSO variants inspired by the way ants communicate using pheromone [17]. Each behavior has a pheromone concentration which is updated every iteration according to how well the particles using those behaviors performed. Two pheromone update strategies were developed: a constant strategy (pHPSO-const) and a linear strategy (pHPSO-lin). Similar to the dHPSO, the particles change behaviors if they have stagnated. The new behavior is then probabilistically chosen using roulette wheel selection and the pheromone concentrations. The pheromone based HPSOs each use eight behaviors.

Engelbrecht [8] and Montes de Oca et al [16] provide a comprehensive list of other HPSO algorithms.

III. Frequency-based HPSO

When using the dynamic HPSO model, a number of considerations must be taken into account, namely: which behaviors to use in the behavior pool, when to change the behavior and which behavior to select. This section describes the algorithm proposed in this study, addressing each of the above considerations.

A. Behavior Selection

The main idea behind f_k-PSO is that behaviors have a higher probability of being chosen if they frequently perform well. Each behavior is assigned a success counter which keeps track of how many times that behavior has improved the fitness of the particle that uses it. The number of successes are only considered for the previous k iterations. This is because as the search progresses the particles may enter regions which require behaviors that performed badly in other regions.

Similar to the dHPSO and pHPSO, the particles in the f_k-PSO change behaviors when they stagnate, i.e. when their $pbest$ does not improve for a number of iterations. When a particle must change its behavior the next behavior is chosen using tournament selection. First a subset of the behavior pool is randomly chosen and the best behavior from the subset is selected as the particle's next behavior. Tournament selection is used to prevent a single behavior from dominating, while still giving the behaviors that perform better in the short term a higher probability of being chosen.

B. Behavior Pool

Determining which behaviors to place in the behavior pool is important because a particle can only change to a behavior within the behavior pool. Therefore, a combination of different types of behaviors is ideal. The f_k-PSO consists of the following behaviors:

- TVAC-PSO for early exploration and later exploitation
- TVIW-PSO for early exploration and later exploitation
- sPSO for its high exploitation
- cPSO for its hill-climbing ability
- modBB-PSO with the exploitation probability set to linearly increase from 0 to 1. This allows particles to jump out of local optima as well as exploit $pbest$ solutions more as the search progresses
- QSO with the cloud radius set to linearly decrease from the search bounds to 0 for better initial exploration. This also allows particles to jump out of local optima as well as exploit $nbest$ solutions more as the search progresses

C. Parameters

Disregarding the parameters of the individual behaviors, f_k-PSO has four parameters to be set: swarm size, stagnation threshold, k and the tournament size. Bratton and Kennedy [4] recommend a swarm size of between 20 and 100. As a result 50 was chosen for the f_k-PSO.

The other three parameters were tuned using iterated F-Race [1] which is described in Section IV. The stagnation threshold controls the number of iterations a behavior has to make an impact on the search. Higher stagnation thresholds allow the behaviors to get to a good region before they start improving while lower values allow the particles to change to a better behavior sooner.

The value of k determines how much of a behavior's previous success contributes to its current chances of being

selected. Higher values of k can result in sub-optimal behaviors being selected since they may be performing badly in the last few iterations. Lower values of k do not allow behaviors to gather momentum which means if a behavior only performs well for one or two iterations it has a better chance of being selected when other behaviors are more optimal.

The tournament size affects the probability of choosing badly performing behaviors. A value of one is the same as using random selection and a value equal to the behavior pool size is the same as elitist selection. Higher values means the behaviors with low success rates will more likely not be chosen which makes it difficult to change the swarm behavior when it is needed. Lower tournament sizes means it can be more difficult to exploit the behaviors that are performing well.

IV. Experimental Setup

This section details the methodology used for the experiments including parameter selection and a description of the benchmark functions.

A. Experimental Procedure

The proposed algorithm was compared to the ALPSO-II, DPP-PSO, pHPSO-const, pHPSO-lin and dHPSO on the CEC 2013 benchmark functions for real parameter optimization [15]. Each algorithm was executed on each function for 51 independent runs (as specified in the benchmark description [15]) in 10, 30 and 50 dimensions for $10000 \times D$ function evaluations where D is the dimension of the function. The algorithms were all implemented in CIlib (http://www.cilib.net).

The statistical analysis was done using the Friedman test with the Bonferroni-Dunn post-hoc test as described by Demšar [5].

B. Parameter Tuning

Parameter tuning was done using iterated F-Race [1] which is an iterated version of F-Race [2]. F-Race is a racing algorithm used for automatic algorithm configuration.

There are two types of parameters: behavior parameters, which are the parameters used by the individual behaviors, and algorithm parameters which are the parameters used by the algorithm which selects the behaviors. Tuning was done for all the algorithms on the parameters not related to the behaviors. The behavior parameters used were the same as in their original papers unless otherwise stated. The original behavior parameters were used because those are the parameters which result in the desired behavior for a particle, i.e. the reason that behavior was chosen for the behavior pool. The swarm size for all the algorithms was set to 50.

F-Race functions as follows: a number of initial parameter configurations are run on a series of problems which are representative of the problems for which the algorithms are being tuned. As the configurations are applied to the tuning problems the Friedman test is used to remove the configurations which are statistically worse than the current best configuration. This continues until only one configuration is left or a maximum tuning cost is reached.

Iterated F-Race performs the F-Race procedure a number of times. At each iteration the configurations are resampled taking the remaining configurations from the previous iteration as models.

Table I. Initial Parameter Ranges

DPP-PSO	
β	$[0.0001, 20]$
Number of rigid particles	$[1, 50]$
pHPSO-lin	
Gradient	$[0.0001, 2]$
Minimum pheromone	$[0.0001, 0.1]$
Stagnation threshold	$[1, 50]$
f_k-PSO	
k	$[1, 500]$
Tournament size	$[2, 6]$
Stagnation threshold	$[1, 50]$
dHPSO	
Stagnation threshold	$[1, 50]$
ALPSO-II	
q	$[1, 50]$
Minimum selection ratio	$[0.0001, 0.1]$
pHPSO-const	
Better score	$[0, 4]$
Same score	$[-2, 2]$
Worse score	$[-2, 2]$
Minimum pheromone	$[0.0001, 0.1]$
Stagnation threshold	$[1, 50]$

For this paper the initial parameters were sampled using a Sobol sequence [20] in the ranges shown in Table I. After an iteration is completed the minimum and the maximum of each parameter is calculated from the remaining configurations and the parameters are resampled within the new ranges, again using a Sobol sequence. Each algorithm tuned the parameters shown in Table I for 100 iterations using the CEC 2005 benchmark functions [22] as the tuning problems. The maximum tuning cost for each iteration was 25 problems i.e. the whole benchmark set.

The following settings were also used for tuning: the dimensions of each function were randomly selected in the range $[10, 20]$ to prevent the algorithm from being tuned to a single dimension. The Friedman test was only applied after two problems were completed for all the configurations. Each configuration was run five times and the mean was used for the Friedman test. Table II shows the final values obtained for the parameters.

Table II. Tuned Parameters

DPP-PSO		**dHPSO**	
β	0.65	Stagnation threshold	3
Number of rigid particles	2	**ALPSO-II**	
pHPSO-lin		q	8
Gradient	0.1	Minimum selection ratio	0.008
Minimum pheromone	0.004	**pHPSO-const**	
Stagnation threshold	5	Better score	0.25
f_k-PSO		Same score	-0.05
k	10	Worse score	-0.35
Tournament size	2	Minimum pheromone	0.005
Stagnation threshold	5	Stagnation threshold	8

C. Behavior Parameters

The behavior parameters used for the dHPSO, DPP-PSO, ALPSO-II and pHPSO algorithms were the same as in their original papers. The behavior parameters used by f_k-PSO is shown in Table III. The velocity was clamped for TVAC-PSO, TVIW-PSO, cPSO and sPSO to prevent the particles' velocities

from exploding. Those behaviors also use a decreasing inertia weight to allow more fine grained optimization towards the end of the search. For the same reason, the QSO cloud radius is decreased over time and modBB-PSO exploit probability is increased over time.

Table III. f_k-PSO BEHAVIOR PARAMETERS

	TVAC-PSO	TVIW-PSO	cPSO	sPSO
$w_{initial}$	0.9			
w_{final}	0.4			
c_1	2.5 to 0.0	1.0	2.0	0.0
c_2	0.0 to 2.5	1.0	0.0	2.0
v_{max}	Upper bound of the search space (100)			
	QSO			
Cloud radius	Upper bound of the search space to 0 (100 to 0)			
	modBB-PSO			
Exploit probability	0.0 to 1.0			

D. Benchmark Functions

The CEC 2013 benchmark set [15] consists of 28 functions: 5 unimodal functions, 15 basic multimodal functions and 8 composition functions. All the functions are scaled so that the search domain is $[-100, 100]^D$. Error values less than 10^{-8} are considered as 0 and all optimal error values are 0 i.e. $f_i(\mathbf{x}^*) - f_i^* = 0$ where $\mathbf{x}^*$ is the optimal solution and f_i^* is the optimal fitness.

V. RESULTS AND ANALYSIS

This section analyzes the results obtained for the simulations.

A. Fitness Comparison

Tables IV, V and VI summarize the results obtained for f_k-PSO in 10, 30 and 50 dimensions respectively. A comparison with the other algorithms is shown in Table VII which shows the average ranks over all dimensions for each function. The numbers in bold indicate the highest rankings. Overall, the f_k-PSO obtained the best rank followed by the pHPSO-lin, pHPSO-const, ALPSO-II, dHPSO and DPP-PSO. Figure 1 graphically shows the results of the statistical analysis indicating the average rank of each algorithm. Algorithms grouped with a bold line indicate that there is no significant difference between them.

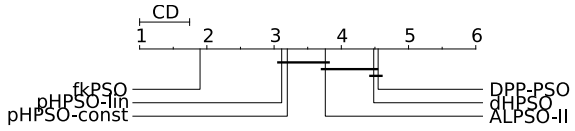


Figure 1. Critical-difference graph using the Bonferroni-Dunn post-hoc test for all functions

For the unimodal functions the f_k-PSO obtained the best rank for four of the five functions with two of those being shared with the other dHPSO and pHPSOs. The DPP-PSO got the highest rank for the fifth function, f_2. Overall for the unimodal functions the f_k-PSO obtained the highest rank followed by the two pheromone variants, the DPP-PSO and the ALPSO-II. Figure 2 shows the statistical differences between the algorithms for unimodal functions. The only statistical

Table IV. SUMMARY OF RESULTS IN 10D

	Best	Worst	Median	Mean	Std
f_1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_2	1.14e+04	5.33e+05	1.17e+05	1.44e+05	1.04e+05
f_3	1.16e-02	9.63e+06	2.60e+04	6.75e+05	1.96e+06
f_4	3.76e+01	1.43e+03	3.35e+02	4.16e+02	3.37e+02
f_5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_6	1.89e-01	9.11e+00	1.92e+00	2.64e+00	2.05e+00
f_7	2.62e-02	1.49e+01	7.62e-01	1.92e+00	2.70e+00
f_8	2.01e+01	2.05e+01	2.04e+01	2.03e+01	8.71e-02
f_9	2.07e-01	5.05e+00	2.66e+00	2.75e+00	1.09e+00
f_{10}	6.15e-02	1.35e+00	4.46e-01	5.13e-01	3.15e-01
f_{11}	0.00e+00	9.95e-01	0.00e+00	1.76e-01	3.79e-01
f_{12}	9.95e-01	1.49e+01	6.96e+00	7.04e+00	2.95e+00
f_{13}	3.15e+00	2.24e+01	1.22e+01	1.15e+01	4.77e+00
f_{14}	3.60e+00	1.75e+02	2.19e+01	3.78e+01	4.21e+01
f_{15}	1.39e+02	8.19e+02	4.70e+02	4.54e+02	1.67e+02
f_{16}	1.06e-01	6.34e-01	4.16e-01	4.07e-01	1.36e-01
f_{17}	3.15e-01	1.44e+01	1.18e+01	1.10e+01	3.28e+00
f_{18}	5.95e+00	2.00e+01	1.58e+01	1.56e+01	2.33e+00
f_{19}	2.74e-01	7.59e-01	5.01e-01	5.01e-01	1.38e-01
f_{20}	1.59e+00	3.52e+00	2.55e+00	2.52e+00	4.77e-01
f_{21}	1.00e+02	4.00e+02	4.00e+02	3.75e+02	7.10e+01
f_{22}	8.90e+00	3.25e+02	1.13e+02	1.22e+02	7.57e+01
f_{23}	1.06e+02	9.43e+02	5.48e+02	5.15e+02	1.97e+02
f_{24}	1.09e+02	2.11e+02	2.07e+02	2.03e+02	1.91e+01
f_{25}	1.05e+02	2.11e+02	2.07e+02	2.05e+02	1.44e+01
f_{26}	1.02e+02	3.11e+02	2.00e+02	1.89e+02	5.13e+01
f_{27}	3.02e+02	4.55e+02	3.64e+02	3.70e+02	3.22e+01
f_{28}	1.00e+02	6.28e+02	3.00e+02	3.26e+02	1.25e+02

Table V. SUMMARY OF RESULTS IN 30D

	Best	Worst	Median	Mean	Std
f_1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_2	3.19e+05	3.53e+06	1.46e+06	1.59e+06	8.03e+05
f_3	1.20e+05	1.99e+09	9.91e+07	2.40e+08	3.71e+08
f_4	1.95e+02	1.11e+03	4.43e+02	4.78e+02	1.96e+02
f_5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_6	3.55e+00	8.13e+01	2.45e+01	2.99e+01	1.76e+01
f_7	4.72e+00	1.55e+02	6.18e+01	6.39e+01	3.09e+01
f_8	2.08e+01	2.10e+01	2.09e+01	2.09e+01	6.28e-02
f_9	1.20e+01	2.42e+01	1.86e+01	1.85e+01	2.69e+00
f_{10}	5.18e-02	5.91e-01	2.04e-01	2.29e-01	1.32e-01
f_{11}	6.96e+00	4.28e+01	2.39e+01	2.36e+01	8.76e+00
f_{12}	2.89e+01	9.45e+01	5.47e+01	5.64e+01	1.51e+01
f_{13}	6.96e+01	1.61e+02	1.24e+02	1.23e+02	2.19e+01
f_{14}	2.44e+02	1.19e+03	6.79e+02	7.04e+02	2.38e+02
f_{15}	2.26e+03	4.36e+03	3.50e+03	3.42e+03	5.16e+02
f_{16}	2.43e-01	1.22e+00	8.56e-01	8.48e-01	2.20e-01
f_{17}	3.89e+01	7.25e+01	5.19e+01	5.26e+01	7.11e+00
f_{18}	5.00e+01	8.94e+01	6.79e+01	6.81e+01	9.68e+00
f_{19}	1.62e+00	6.69e+00	2.92e+00	3.12e+00	9.83e-01
f_{20}	1.02e+01	1.40e+01	1.21e+01	1.20e+01	9.26e-01
f_{21}	2.00e+02	4.44e+02	3.00e+02	3.11e+02	7.92e+01
f_{22}	2.45e+02	1.48e+03	8.56e+02	8.59e+02	3.10e+02
f_{23}	2.47e+03	4.98e+03	3.45e+03	3.57e+03	5.90e+02
f_{24}	2.31e+02	2.69e+02	2.48e+02	2.48e+02	8.11e+00
f_{25}	2.33e+02	2.65e+02	2.49e+02	2.49e+02	7.82e+00
f_{26}	2.00e+02	3.65e+02	3.41e+02	2.95e+02	7.06e+01
f_{27}	6.35e+02	9.99e+02	7.73e+02	7.76e+02	7.11e+01
f_{28}	1.00e+02	1.50e+03	3.00e+02	4.01e+02	3.48e+02

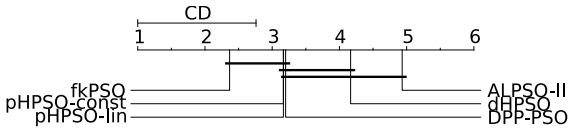


Figure 2. Critical-difference graph using the Bonferroni-Dunn post-hoc test for unimodal functions

Table VI. SUMMARY OF RESULTS IN 50D

	Best	Worst	Median	Mean	Std
f_1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_2	1.22e+06	5.11e+06	2.58e+06	2.76e+06	9.64e+05
f_3	5.05e+07	4.20e+09	6.16e+08	9.68e+08	9.65e+08
f_4	1.72e+02	1.16e+03	4.87e+02	5.25e+02	1.82e+02
f_5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
f_6	2.90e+01	1.01e+02	4.50e+01	5.51e+01	2.25e+01
f_7	3.35e+01	1.31e+02	7.46e+01	7.81e+01	2.21e+01
f_8	2.10e+01	2.12e+01	2.11e+01	2.11e+01	4.81e-02
f_9	2.56e+01	4.97e+01	3.95e+01	3.85e+01	5.31e+00
f_{10}	3.44e-02	6.26e-01	1.70e-01	2.13e-01	1.33e-01
f_{11}	5.07e+01	1.39e+02	8.26e+01	8.61e+01	1.99e+01
f_{12}	9.55e+01	2.16e+02	1.46e+02	1.45e+02	2.88e+01
f_{13}	1.81e+02	3.93e+02	2.73e+02	2.74e+02	4.53e+01
f_{14}	9.21e+02	2.91e+03	1.85e+03	1.96e+03	4.60e+02
f_{15}	4.86e+03	8.17e+03	6.61e+03	6.63e+03	8.09e+02
f_{16}	5.89e-01	1.87e+00	1.32e+00	1.30e+00	3.04e-01
f_{17}	8.28e+01	1.54e+02	1.14e+02	1.16e+02	1.58e+01
f_{18}	9.46e+01	1.63e+02	1.31e+02	1.32e+02	1.62e+01
f_{19}	4.12e+00	1.55e+01	7.15e+00	7.82e+00	2.34e+00
f_{20}	1.73e+01	2.35e+01	2.06e+01	2.06e+01	1.08e+00
f_{21}	2.00e+02	1.12e+03	8.36e+02	8.34e+02	3.55e+02
f_{22}	9.71e+02	3.68e+03	2.26e+03	2.22e+03	6.06e+02
f_{23}	5.62e+03	9.22e+03	7.35e+03	7.40e+03	8.61e+02
f_{24}	2.74e+02	3.33e+02	2.99e+02	3.00e+02	1.47e+01
f_{25}	2.75e+02	3.39e+02	3.00e+02	3.00e+02	1.27e+01
f_{26}	2.00e+02	4.33e+02	3.95e+02	3.90e+02	4.04e+01
f_{27}	1.02e+03	1.62e+03	1.32e+03	1.32e+03	1.23e+02
f_{28}	4.00e+02	3.61e+03	4.00e+02	1.63e+03	1.53e+03

Table VII. AVERAGE RANKS PER FUNCTION OVER ALL DIMENSIONS

		ALPSO-II	dHPSO	DPP-PSO	f_k-PSO	pHPSO-const	pHPSO-lin
Unimodal	f_1	4.00	**3.00**	5.00	**3.00**	**3.00**	**3.00**
	f_2	3.00	5.67	**1.33**	3.00	3.33	4.67
	f_3	6.00	4.33	1.67	**1.33**	4.33	3.33
	f_4	6.00	5.00	4.00	**1.67**	2.33	2.00
	f_5	5.67	**2.83**	4.00	**2.83**	**2.83**	**2.83**
	Mean	4.93	4.17	3.2	**2.37**	3.16	3.17
	Std	1.22	1.11	1.44	0.72	0.67	0.87
Basic Multimodal	f_6	**2.67**	4.00	**2.67**	3.33	3.67	4.67
	f_7	4.33	2.00	5.67	**1.00**	3.67	4.33
	f_8	3.00	5.00	2.33	**1.33**	4.33	5.00
	f_9	5.00	6.00	4.00	**1.00**	2.33	2.67
	f_{10}	3.33	4.00	4.33	**3.00**	**3.00**	3.33
	f_{11}	**1.00**	3.33	6.00	4.00	2.33	4.33
	f_{12}	6.00	2.67	5.00	**1.00**	3.00	3.33
	f_{13}	5.00	2.67	6.00	**1.00**	3.33	3.00
	f_{14}	**1.00**	5.00	6.00	2.67	4.00	2.33
	f_{15}	4.67	6.00	4.00	**1.00**	3.33	2.00
	f_{16}	2.00	6.00	4.33	**1.00**	3.33	4.33
	f_{17}	**1.00**	4.67	6.00	2.67	4.33	2.33
	f_{18}	5.67	4.33	4.33	**1.00**	2.33	3.33
	f_{19}	**1.00**	4.67	6.00	3.67	3.33	2.33
	f_{20}	3.33	4.33	6.00	**1.00**	2.33	4.00
	Mean	3.27	4.31	4.84	**1.91**	3.24	3.42
	Std	1.73	1.20	1.21	1.12	0.67	0.94
Composition	f_{21}	3.67	3.00	5.67	**2.33**	3.67	2.67
	f_{22}	**1.00**	5.00	6.00	2.00	4.00	3.00
	f_{23}	5.00	6.00	4.00	**1.00**	3.00	2.00
	f_{24}	5.00	6.00	4.00	**1.00**	2.67	2.33
	f_{25}	4.67	6.00	4.33	**1.00**	2.33	2.67
	f_{26}	4.00	5.00	5.33	**2.00**	2.33	2.33
	f_{27}	5.00	6.00	3.33	**1.00**	3.33	2.33
	f_{28}	3.33	3.00	6.00	**2.33**	3.67	2.67
	Mean	3.96	5.00	4.83	**1.58**	3.13	2.50
	Std	1.27	1.22	0.97	0.59	0.60	0.29
Mean		3.76	4.48	4.55	**1.90**	3.20	3.11
Std		1.87	1.42	1.63	**1.20**	1.00	1.14

differences are from the f_k-PSO to dHPSO and ALPSO-II and from the pHPSO-const to the ALPSO-II.

Considering the basic multimodal functions the f_k-PSO performed the best on two thirds of the functions (joint first with pHPSO-const on f_{10}) with the ALPSO-II outranking it on the remaining five. The DPP-PSO and the ALPSO-II ranked the best on f_6. Figure 3 shows the statistical differences between the algorithms for multimodal functions. The f_k-PSO is statistically different to all the other algorithms. The other statistical differences are from the pHPSO-const and ALPSO-II to the dHPSO and DPP-PSO and from the pHPSO-lin to the DPP-PSO.

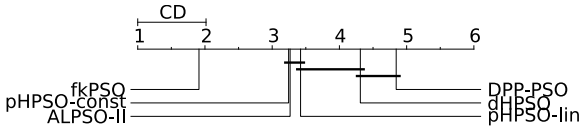


Figure 3. Critical-difference graph using the Bonferroni-Dunn post-hoc test for multimodal functions

For the composition functions the f_k-PSO outranked the other algorithms on all but one function, f_{22}, for which the ALPSO-II got the best rank. Figure 4 shows the statistical differences between the algorithms for the composition functions. Statistical differences were found from the f_k-PSO to pHPSO-const, ALPSO-II, DPP-PSO and dHPSO. There are also statistical differences from the the pHPSO-lin to the ALPSO-II, DPP-PSO and dHPSO and from the pHPSO-const to the DPP-PSO and dHPSO.

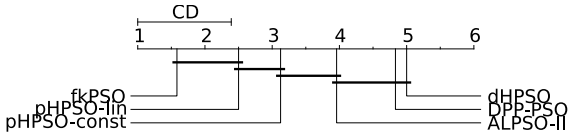


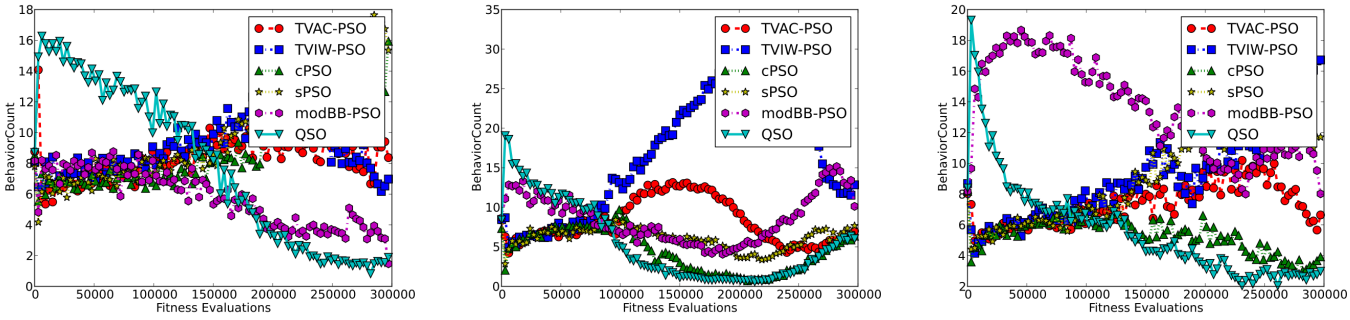
Figure 4. Critical-difference graph using the Bonferroni-Dunn post-hoc test for composition functions

B. Behavior Profiles

Figure 5 show the average behavior profiles of the f_k-PSO for f_4, f_{12} and f_{27} in 30 dimensions for all the runs. The figures show that all the behaviors perform better than other behaviors at some point during the search. The behavior profile plots also show that the exploration/exploitation balance is different for the different functions.

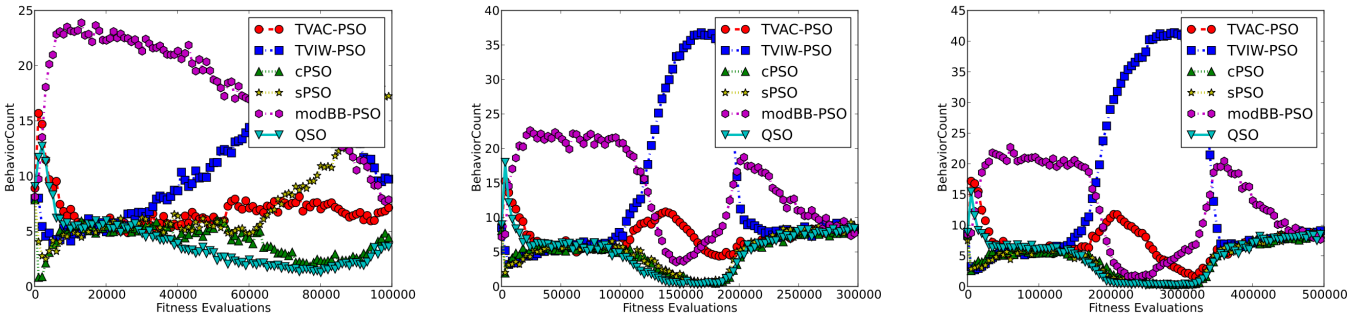
Figure 6 shows the average behavior profiles for f_{10} in all the dimensions. It shows that functions with similar fitness landscapes also have similar behavior profiles. As the dimension changes the behavior profiles roughly maintain the same concentration of each behavior.

In Figures 5(b), 6(b) and 6(c) the TVIW-PSO peaks near the middle of the search. At this point during the search the inertia weight is in the range $[0.6, 0.8]$ which is the optimal value for w for convergence [23] so the behavior's success score increases until the swarm has converged. The TVAC-PSO also gains a small surge of usage during the $[0.6, 0.8]$ interval and a smaller one near the end of the search.



(a) f_4 (b) f_{12} (c) f_{27}

Figure 5. Behavior profile plots for functions f_4, f_{12} and f_{27} in 30 dimensions



(a) 10D (b) 30D (c) 50D

Figure 6. Behavior profile plots for functions f_{10} in 10, 30 and 50 dimensions

The QSO's success will increase when it exploits the $nbest$ and the other behaviors fail to find better solutions, e.g. in Figure 5(a). Similarly, the modBB-PSO's success will improve when the particles' $pbest$s get exploited. In the beginning of the search the exploration done around the $pbest$s favours the modBB-PSO as shown in Figures 5(c) and 6.

Usage of the cPSO increases at the end of the search when the swarm refines the solution on which it has settled. The sPSO can cause the particles to pass through bad regions to get to the $nbest$. As a result it is not selected very often in the beginning of the search.

C. Convergence

Figure 7 shows the convergence plots for the functions in Figure 5 and Figure 6. Notice that the plots for the f_k-PSO contain a "dip" near the middle of the graphs. This suggests that up to that point the swarm was mostly exploring the search space and from that point on it is starting to exploit. This is consistent with the behaviors chosen for the behavior pool.

This behavior is good because, while it does not converge the quickest, in many cases it still obtains a better final fitness.

VI. Conclusions

This paper proposed a new self-adaptive HPSO to adaptively choose the exploration/exploitation point of the search. The proposed HPSO was compared to existing heterogeneous PSOs on the CEC 2013 real-parameter optimization benchmark functions. The results show that the proposed variant outperforms the existing HPSOs and that different exploration/exploitation balances are selected for the different functions.

Future research includes investigating different behavior changing triggers, analyzing the behavior pool and investigating the different behavior selection strategies on the same behavior pool.

References

[1] P. Balaprakash, M. Birattari, and T. Stützle, "Improvement strategies for the f-race algorithm: sampling design and iterative refinement," in *Proceedings of the 4th international conference on Hybrid metaheuristics*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 108–122.

[2] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 11–18.

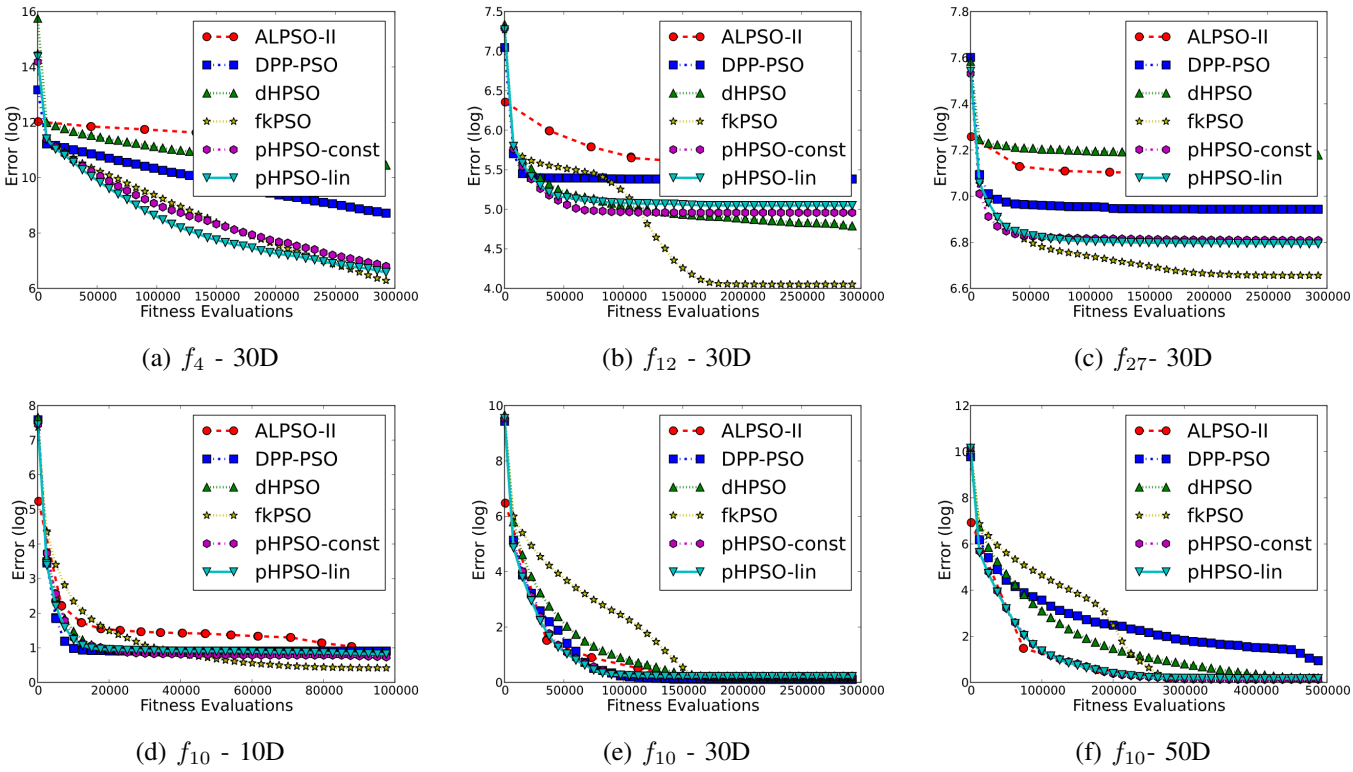
[3] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *EvoWorkshops*, 2004, pp. 489–500.

[4] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, 2007, pp. 120 –127.

[5] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[6] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[7] A. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. Wiley Publishing, 2007.



(a) f_4 - 30D (b) f_{12} - 30D (c) f_{27}- 30D

(d) f_{10} - 10D (e) f_{10} - 30D (f) f_{10}- 50D

Figure 7. Convergence plots for the functions in Figure 5 and Figure 6

[8] ——, "Heterogeneous particle swarm optimization," in *Proceedings of the 7th international conference on Swarm intelligence*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 191–202.

[9] ——, "Scalability of a heterogeneous particle swarm optimizer," in *Proceedings of the IEEE Symposium on Swarm Intelligence*, 2011, pp. 1–8.

[10] J. Kennedy, "The particle swarm: social adaptation of knowledge," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1997, pp. 303 –308.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[12] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2002, pp. 1671–1676.

[13] B. Leonard and A. Engelbrecht, "Scalability study of particle swarm optimizers in dynamic environments," in *Proceedings of the 8th international conference on Swarm Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 121–132.

[14] C. Li and S. Yang, "Adaptive learning particle swarm optimizer-II for global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.

[15] J. Liang, B.-Y. Qu, P. Suganthan, and A. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session and competition on real-parameter optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, Tech. Rep., 2013.

[16] M. Montes de Oca, J. Pena, T. Stutzle, C. Pinciroli, and M. Dorigo, "Heterogeneous particle swarm optimizers," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2009, pp. 698–705.

[17] F. Nepomuceno and A. Engelbrecht, "A self-adaptive heterogeneous pso inspired by ants," in *Proceedings of the 8th international conference on Swarm Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 188–195.

[18] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[19] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 1998, pp. 69–73.

[20] I. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," *Computational Mathematics and mathematical physics*, vol. 7, no. 4, pp. 86–112, 1967.

[21] P. Spanevello and M. Montes de Oca, "Experiments on adaptive heterogeneous PSO algorithms," in *Proceedings of Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, 2009, pp. 36–40.

[22] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2005.

[23] F. Van Den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, University of Pretoria, Pretoria, South Africa, 2002.