Hybrid Particle Swarm Optimisation Based on History Information Sharing

Wenlong Fu
School of Mathematics, Statistics and Operations Research
Victoria University of Wellington
New Zealand
Wenlong.Fu@msor.vuw.ac.nz

Mark Johnston
School of Mathematics, Statistics and Operations Research
Victoria University of Wellington
New Zealand
Mark.Johnston@msor.vuw.ac.nz

Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
New Zealand
Mengjie.Zhang@ecs.vuw.ac.nz

ABSTRACT

Particle Swarm Optimisation (PSO) is an intelligent search method based on swarm intelligence and has been widely used in many fields. However it is also easily trapped in local optima. In order to find a global optimum, some evolutionary search operators used in multi-agent genetic algorithms are integrated into a novel hybrid PSO, with the expectation of effectively escaping from local optima. Particles share their history information and then update their positions using the latest and best history information. Some benchmark high-dimensional functions (from 20 to 10000 dimensions) are used to test the performance of the hybrid algorithms. The results demonstrate that the algorithm can solve high-dimensional nonlinear optimisation problems and that the number of function evaluations required to do so increases with function dimension at a sublinear rate.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques Evolutionary prototyping; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

Particle Swarm Optimisation, Multi-Agent, Local Search, Function Optimisation

1. INTRODUCTION

Particle Swarm Optimisation (PSO) is a stochastic global optimisation method which originated from the simulation of the social behaviour of birds within a flock, as developed by Kennedy and Eberhart in 1995 [5]. As PSO is simple and fast, it is widely used in nonlinear function optimisation [3][4], object detection [10], coverage maximisation and energy conservation in wireless sensor networks [2], and many other applications [7][8].

The global optimisation of functions is an important topic in scientific and engineering research since many real situations can be modelled as nonlinear optimisation problems, and they are often required to be solved accurately within a limited number of function evaluations. Standard PSO [11] has a problem in that it rapidly converges to local optima, especially for high-dimensional functions. To protect particles from becoming trapped in local optima, many researchers have tried different strategies, integrating local search [1], differential evolution [16][18][4][9] and tabu restrictions [14] into PSO. All these hybrid PSO algorithms update particle velocities based on their best positions (including personal best positions and the global best position) and employ other operators to the particles directly. They can obtain improvement for some functions, but still cannot find a global optimum for some difficult functions such as the Generalized Rosenbrock function. The ability to find a global optimum is still not sufficiently good and the number of function evaluations increases sharply with function dimension.

1.1 Goals

The overall goal is to find optimal solutions for high-dimensional functions, i.e., up to 10000 dimensions. High-dimensional function optimisation is a challenge for many optimisation algorithms because function evaluations usually are increasing fast and more local optima exist when the dimension grows. In the literature, 30 dimensional functions are often employed to test algorithm performance. However, this is really not sufficient and more dimensions are required to test algorithm performance, such as the ability to actually find a global optimum and the complexity of function evaluations.

The goals of this paper are to develop and investigate a new hybrid PSO technique for finding global optima of high-dimensional functions, and to analyse the relationship between function evaluations and function dimension. Instead of using the standard PSO method, we develop a hybrid PSO technique in which a particle updates its velocity not only from its own history information, but also from the history information of other particles. Local-search opera-

GECCO'11, July 12–16, 2011, Dublin, Ireland.

tors are also applied. For evaluating the novel hybrid PSO technique performance, it is compared with some existing PSO-based methods and other algorithms on ten benchmark high-dimensional functions. We focus on whether the new approach *can* find the global optima of functions ranging from 20 to 10000 dimensions, and on *how many* function evaluations are used to find a global optimum. Specifically, we investigate the following objectives.

- Whether the proposed hybrid PSO can find a global optimum of the test functions as their dimension increases.
- Whether the proposed hybrid PSO can reduce function evaluations, compared with existing methods, if they can find a global optimum.
- What relationship exists between function dimension and number of function evaluations to find a global optimum, namely algorithm complexity, for the proposed hybrid PSO, compared to existing methods.
- Whether population size affects the number of function evaluations needed to find a global optimum.

1.2 Organisation

In the remainder of the paper, Section 2 briefly describes the background of PSO and strategies for sharing information. Section 3 develops the hybrid PSO algorithm. After the experimental design is presented in Section 4, Section 5 discusses the experimental results. Section 6 gives conclusions and future work directions.

2. BACKGROUND

This section briefly describes background on Particle Swarm Optimisation (PSO) and the evolutionary operators in Multi-Agent Genetic Algorithm (MAGA) [19].

2.1 Particle Swarm Optimisation

PSO is a stochastic global optimisation algorithm which does not require explicit knowledge of the gradient of the problem's objective function. PSO maintains a population of candidate solutions (called particles) and moves these particles around in a D-dimensional search space. Each particle moves according to the historical experiences of its own and those of its colleagues. Each particle has its own position and velocity, and updates its position using its current velocity.

The "standard PSO" (SPSO) is defined by (1) and (2) [11]:

$$\begin{aligned} v_{ik}^{t+1} &= \omega \times v_{ik}^{t} + \phi_1 \times rand() \times (p_{ik}^{t} - x_{ik}^{t}) \\ &\quad + \phi_2 \times rand() \times (g_{id}^{t} - x_{ik}^{t}) \qquad (1) \\ x_{ik}^{t+1} &= x_{ik}^{t} + v_{ik}^{t+1} \qquad (2) \end{aligned}$$

where ω is inertia weight, ϕ_1 and ϕ_2 are acceleration constants, $rand()$ are random values between 0 and 1, $v_i^t = [v_{i1}^t, v_{i2}^t, \ldots, v_{iD}^t]$ is the ith particle's velocity in generation t, $x_i^t = [x_{i1}^t, x_{i2}^t, \ldots, x_{iD}^t]$ is the ith particle's position in generation t, $p_i^t = [p_{i1}^t, p_{i2}^t, \ldots, p_{iD}^t]$ is the best position of the ith particle up to generation t, and $g_i^t = [g_{i1}^t, g_{i2}^t, \ldots, g_{iD}^t]$ is the best position found (the leader) in the neighbourhood of particle i up to generation t. The leader of a particle is specified by a connected neighbourhood topology, e.g., ring, lattice or complete graph. When termination criteria are satisfied, such as t being equal to the maximum generation, the global best position is taken as the solution to the problem.

2.2 Evolutionary Operators in Multi-Agent Genetic Algorithm

The Multi-Agent Genetic Algorithm (MAGA) [19] is an agent-based real-valued evolutionary algorithm for function optimisation. Agents are arranged in a square toroidal lattice topology, and work together in order to achieve the goals for a particular problem. Agents share information only with their four neighbours in the lattice through competition and crossover. It is different from PSO in that an agent can improve itself without interaction between agents, and it may be replaced by other better agents. MAGA has successfully solved 10 benchmark high-dimensional functions (from 20 to 10000 dimensions) [19]. But for the Generalized Rosenbrock function (formula (17) in Appendix A), it is hard to find a global optimum.

In MAGA, four evolutionary operators are employed (neighbourhood competition, neighbourhood orthogonal crossover, mutation and self-learning) which are briefly described here (see [19] for details). The *neighbourhood competition* operator replaces the position of an agent by an elementwise combination of its current position and the position of best agent in its neighbourhood. The *neighbourhood orthogonal crossover* operator replaces the position of an agent by the best candidate from a regular sample taken from the hyperrectangle with corners at the agent's current position and the position of the best agent in its neighbourhood. The *mutation* operator adds some Gaussian noise to elements of the agent's position with some probability which decreases with generation of evolution. The *self-learning* operator performs an intensive search in the local neighbourhood of the agent's current location using another population of agents arranged in a lattice.

2.3 Complexity of Function Evaluations With Respect to Function Dimension

For approximating the *complexity* of function evaluations, we use the function dimension as input and the average function evaluations as output, following [19]. We fit a power law curve ($y = kD^n$) for these points in one function optimisation problem, where y is the function evaluations, D is the function dimension, k and n are the estimated parameters. The complexity of function evaluations for one function is then $O(D^n)$.

For most genetic algorithms or particle swarm optimisation techniques, it is difficult to determine the complexity of function evaluations directly. An adaptive evolutionary algorithm (AEA) [6] has tested some functions with dimensions up to 10000. Also, MAGA [19] has optimised functions with dimensions from 20 to 10000.

3. A NEW HYBRID PSO ALGORITHM

Since particles often get trapped in a local optimum, maintaining particle diversity is a way to protect particles from all falling into local optima, at the potential expense of slower convergence. The aim here is to improve diversity of particle locations by encouraging particles to learn from the history of particles in their neighbourhood (with respect to predefined topology rather than current position), following [19],

Algorithm 1 *Inversion Operator*

1: Select the dimension(s).
2: Update these values by formula (3) and (4).
3: Select the best position to replace the particle position.

and by weakening the influence of the current best global position, following [3].

We propose a hybrid PSO approach using six different operators, each with a different purpose. Note that it is not our goal to determine the relative importance of each operator; this is left for future work.

Firstly, the neighbourhood competition operator (from MAGA) is slightly adapted to combine the particle's best position with the current position of its best neighbour. Secondly, the neighbourhood orthogonal crossover operator from MAGA is used as is. Thirdly, the self-learning operator from MAGA is adapted to use the slightly modified neighbourhood competition operator and the MAGA mutation operator on a lower-level population. Since the self-learning operator requires a large number of additional function evaluations, it is only triggered after a given number of generations during which the search progress is stagnating.

Fourthly, an *inversion operator* is adapted from opposition-based PSO [13]. This is a local-search operator that takes advantage of any natural symmetry in the problem. In [13], an exhaustive search is made of all "opposition" solutions in each dimension about the current position of a particle and the number of function evaluations used is the number of function dimensions. We propose to randomly select a small subset of dimensions of a particle and invert these element using (4), where $[x_{min}, x_{max}]$ is the domain of that dimension in the function search-space. We also introduce perturbation into the inversion operator (3). For the selected dimension, we use (4) and (3) to generate candidates and then select the best one, where δ is a small value for perturbation, r_{sign} is a random value either -1 or 1. The algorithm is described in Algorithm 1.

$$x_i \leftarrow x_i + \delta \text{ or } x_i - \delta \quad (3)$$

$$x_i \leftarrow x_{max} + x_{min} - x_i + r_{sign} \times \delta \text{ or } x_{max} + x_{min} - x_i \quad (4)$$

Fifthly, the *single dimension step* operator is another local-search operator, which we only apply to the global best particle. We search a random dimension by a small initial step. If the new position is better than the original, it will move to the new position; otherwise, the step size will be changed to search for a better position until a new better position is found. The algorithm is described in Algorithm 2, where r_c is a parameter to control the step, n_{stag} is the number of consecutive generations of stagnation. If there is no stagnation, $n_{stag} = 0$. T_{sys} is the tolerance value for the step. When the absolute value of *step* (*step* can be negative) is small enough, the operator is not worth using it to search for a better position because they are too close. As the operator requires additional function evaluations, we only apply it to the global best particle with a low probability.

Finally, the *velocity* operator applies the strategy of strengthening the influence from the neighbourhood and weakening the influence of the current best global position following [3]. The operator for updating particle velocity simplifies the parameters about the personal position and the global

Algorithm 2 *Single Dimension Step Operator*

1: Set $step = r_c/log(2.0 + n_{stag})$, randomly select one dimension.
2: If $|$ step $| < T_{sys}$, go to step 8.
3: Add value *step* to the selected dimension of position.
4: If the new position is better than the previous one, move to the new position and go to step 8.
5: Subtract $2step$ from the position in the selected dimension.
6: If the new position is better than the previous one, move to the new position and go to step 8.
7: Add value *step* to the selected dimension of position, and set $step = step/10$; go to step 3.
8: Finish the local search.

Algorithm 3 *Velocity Operator*

1: Update one particle's dummy velocity by formula (5)
2: Obtain the particle's velocity by formula (6).

position. The velocity is mainly affected by the neighbourhood history information but it may learn from itself or the global best position. The related operator is described as in Algorithm 3, where, in (5), $p^t_{r_{id}}$ and $p^t_{l_{id}}$ are the dth dimension of the particle i neighbourhood personal best position (the right and left particles in the *ring topology*) at generation t, and in (6), r_v is a random value between 0 and 1, p_v is the probability for selecting the personal best position or global best position.

$$v_{id}^{t+\frac{1}{2}} = \omega \times v_{id}^t + r_r \times p_{r_{id}}^t + (1 - r_r) \times p_{l_{id}}^t - x_{id}^t \quad (5)$$

$$v_{id}^{t+1} = \begin{cases} v_{id}^{t+\frac{1}{2}} + p_{id}^t - x_{id}^t, & r_v < p_v \\ v_{id}^{t+\frac{1}{2}} + g_{id}^t - x_{id}^t, & \text{otherwise} \end{cases} \quad (6)$$

The whole hybrid PSO is described in Algorithm 4, where p_{cross} and p_{invert} are the probabilities to select particles for the neighbourhood orthogonal crossover operator and the inversion operator, $n_{minstag}$ is the minimum value for the consecutive generations of stagnation, p_{step} is the probability for selecting the single dimension step operator, r_{step} is a random value from 0 to 1, and the global best particle is the best position known at the current generation.

4. EXPERIMENTAL DESIGN

This section describes the test functions and parameter settings for our experiments.

4.1 Multi-Modal and High-Dimensional Functions

Ten benchmark functions are drawn from [17] and we modify Generalized Schwefel's Problem 2.26 (formula (7) in Appendix A) so that the minimum value is 0. The modified function is f_1' (formula (8) in Appendix A) in Table 1. In this paper, we do not distinguish between the function f_1 and f_1' because they have the same solution and the only difference is the minimum value. Function f_1 to f_6 are multi-modal functions, f_7 to f_9 are unimodal functions, and f_{10} ($D \geq 4$) is a multi-modal function. We aim at minimum values for all functions and the minimum values are all 0. For calculating the number of function evaluations for each function optimisation problem, we used the termination criteria $|g - f| < \epsilon$,

Algorithm 4 *Hybrid PSO (HPSO)*

1: Initialise the particles, the local best positions and the global best position.
2: Apply the *neighbourhood competition operator* to all particles' history information.
3: Select particles with probability p_{cross} to use the *neighbourhood orthogonal crossover operator*.
4: Select particles with probability p_{invert} to use the *inversion operator*.
5: Update all particle velocities by the *velocity operator*.
6: Particles move to new positions by the latest velocities.
7: Check the progress, if it is stagnating, $n_{stag} = n_{stag} + 1$, otherwise $n_{stag} = 0$.
8: If $n_{stag} < n_{minstag}$, go to step 10.
9: Use the *self-learning operator* to the global best "particle".
10: If $r_{step} \geq p_{step}$, go to step 12.
11: The *single dimension step operator* is employed to the global best particle.
12: Check whether the current generation is less than maximum generation, if true go to step 2.
13: Output the global best particle as the solution.

where ϵ is a very small positive number given by the user, g is the minimum value found by the proposed hybrid PSO, and f is the minimum value of each function's solution. If this condition is satisfied, we consider the global minimum of a given function has been successfully found by HPSO.

4.2 Parameter Settings

For comparison with MAGA, we use population sizes which are squares, i.e., $\{25, 36, 49, 100\}$. Population size 100 is only used for a t-test to check whether the population size affects the number of function evaluations with low dimension, i.e., $D \in \{30, 60, 100, 500, 1000\}$ (see Section 5.3). Initial particle locations are uniformly randomly generated within the domain, and particle locations are subsequently restricted to the domain by clipping if necessary, i.e., a particle which flies out of the domain has its location set to the nearest point on the domain boundary. For calculating the function evaluations, $\epsilon = 0.0001$ is used in the proposed hybrid PSO. Each test is run 100 times independently. Since f_9 is very time consuming (for one function evaluation, the complexity is $O(D^2)$), we do not run all of high dimension experiments. For function f_{10}, it is extremely difficult to find a global optimum when its dimension is 10000; in fact, many existing algorithms cannot find the global optimum even at a dimension of 30. Therefore, we only analyse the complexities of function evaluations for functions f_1 to f_8 from dimension 20 to 10000 (namely 20, 30, 40, 60, 100, 300, 500, 1000, 1500, 2000, 2500, 3000, ..., 9500, 10000). The other parameters in the proposed PSO are $n_{minstag} = 5$, $p_v = 0.5$, $p_{cross} = 0.5$, $p_{invert} = 0.05$ and $p_{step} = 0.05$. These values are chosen based on common settings and initial experiments via empirical search.

5. EXPERIMENTAL RESULTS AND DISCUSSION

This section describes the experimental results and compares them with several typical existing algorithms.

Table 1: Test Functions

Test function	Space
$f_1' = D \times 418.983 - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^D$
$f_2 = \sum_{i=1}^{D} \{x_i^2 - 10\cos(2\pi x_i) + 10\}$	$[-5.12, 5.12]^D$
$f_3 = -20\exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right\}$ $-\exp\left\{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right\} + 20 + e$	$[-32, 32]^D$
$f_4 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]^D$
$f_5 = \frac{\pi}{D}\{10\sin^2(\pi y_1) +$ $\sum_{i=1}^{D-1}\{1 + 10\sin^2(\pi y_{i+1}\} + (y_d - 1)^2\}$ $+ \sum_{i=1}^{D}\mu(x_i, 10, 100, 4)$	$[-50, 50]^D$
$f_6 = 0.1\{10\sin^2(3\pi x_1)$ $+ \sum_{i=1}^{D-1}(x_i - 1)^2\{1 + 10\sin^2(3\pi x_{i+1})\} +$ $(x_d - 1)^2\{1 + \sin^2(2\pi x_D)\}\} +$ $\sum_{i=1}^{D}\mu(x_i, 5, 100, 4)$	$[-50, 50]^D$
$f_7 = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
$f_8 = \sum_{i=1}^{D} \mid x_i \mid + \prod_{i=1}^{D} \mid x_i \mid$	$[-10, 10]^D$
$f_9 = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$
$f_{10} = \sum_{i=1}^{D-1}\{100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\}$	$[-30, 30]^D$

Note: More details can be seen in Appendix A, f_1' is the modified Generalized Schwefel's Problem 2.26, D is the number of dimensions.

5.1 30-Dimensional Function Optimisation

Table 2 shows the comparison of average number of function evaluations among the hybrid PSO (HPSO), MAGA and other methods with ten 30-dimension function optimisation problems. The table is a compilation of results reported in the cited papers. Here both HPSO and MAGA uses population size 25; HPSO uses a ring topology and MAGA uses a square lattice. HPSO clearly takes the least number of function evaluations for functions f_1 to f_8 than the others. For function f_{10}, only HPSO can find a global optimum. For function f_9, HPSO is only slightly worse than MAGA, but better than other existing algorithms presented in Table 2.

HPSO vs MAGA: Compared with MAGA, the function evaluations of HPSO almost is only a quarter of MAGA for function f_1. For functions f_2, f_4, f_5, f_6 and f_7, they are about a half of function evaluations of MAGA. For functions f_3 and f_8, the function evaluations of HPSO are obviously less than MAGA. Except for function f_9 , HPSO evidently decreases the function evaluations for the others. MAGA cannot find the global solution of function f_{10}. This indicates that HPSO not only inherits good performance from MAGA, but also extends the particles' ability of search-

Table 2: Average Function Evaluations of HPSO, MAGA and Other Methods on 30 Dimensional Problems

Function	HPSO	MAGA [19]	SPSO [3]	PSO-DR M3 [3]	FORL [15]	ATM-PSO [14]
f_1	**2,871** $\pm$ 76	10,862	$\times$	$\times$	150,000	$\times$
f_2	**4,454** $\pm$ 42	11,427	$\times$	$\times$	150,000	15,187
f_3	**8,232** $\pm$ 122	9,656	$\times$	248,160	150,000	9,664
f_4	**4,955** $\pm$ 69	9,777	$\times$	70,348	$\times$	14,530
f_5	**5,528** $\pm$ 79	10,545	$\times$	95,810	130,000	14,030
f_6	**5,674** $\pm$ 104	11,269	$\times$	92,416	130,000	15,433
f_7	**5,492** $\pm$ 223	9,502	46,897	75,810	60,000	10,254
f_8	**7,380** $\pm$ 48	9,591	$-$	$-$	100,000	8,349
f_9	11,832 $\pm$ 550	**9,479**	297,800	$\times$	$\times$	69,029
f_{10}	141,726 $\pm$ 739	$\times$	$\times$	$\times$	$\times$	$\times$

ATM-PSO used the termination criteria $\epsilon = 100$ for f_{10} and $\epsilon = 10$ for f_2, $\epsilon = 2000$ for f_1, $\epsilon = 0.1$ for f_3, f_4, and $\epsilon = 0.01$ for the others; in this paper, $\epsilon = 0.0001$. Here '$\times$' means the related algorithm cannot find the solution and '$-$' means there is no experiment. For HPSO, average $\pm$ standard deviation is reported from 100 runs.

ing for a global optimum and conquers possible failure in MAGA.

HPSO vs FORL: Function Optimisation by Reinforcement Learning (FORL [15]), based on dimensional search in sequence, failed to find global solutions of function f_4, f_9, f_{10} for 30 dimensions. From Table 2, the number of function evaluations of FORL are more than HPSO (and in fact also MAGA), therefore, HPSO is better than FORL in these function optimisation problems.

HPSO vs Other PSOs: Compared with the new HPSO method, the other two PSO methods, ATM-PSO [14] and PSO-DR M3 [3], took more function evaluations, and ATM-PSO failed to find the global solutions for functions f_1 and f_{10}. PSO-DR M3 failed to find the global solutions of functions f_1, f_2, f_9 and f_{10}. There is no experiment on function f_8 for PSO-DR M3. The SPSO performed even worse; it did not find a global optimum for most of the functions except for f_7 and f_9, where an extremely large number of function evaluations were required. These results show that HPSO has evidently better performance than SPSO, ATM-PSO and PSO-DR M3 reported in [3] for the ten 30-dimensional function optimisation problems.

5.2 Function Evaluation Complexity

Table 3 shows the complexities ($O(D^n)$), estimated by log regression, for f_1 to f_8 of HPSO, MAGA and AEA. From Table 3, HPSO has lowest complexities of function evaluations for function f_1, f_2, f_4 and f_6 when the population size is 25, and the others belong to MAGA. However, for function f_2 in MAGA, the function evaluations are almost random and there are no suitable curves from which to estimate the complexity of function evaluations. Although the complexities of functions f_3, f_5, f_7 and f_8 for MAGA are lower than HPSO, from the Table 2, it shows that the function evaluations for these functions with 30-dimension by HPSO are fewer than MAGA. Therefore, in an appropriate dimension range, the function evaluation cost for these functions by HPSO is still less than that by MAGA. Figure 1 shows the details of the relationship between function evaluations and dimension for HPSO. The horizontal axis gives the function dimension and the vertical axis gives the function evaluations. Each vertical interval plots the average value of the number of function evaluations $\pm$ its standard deviation. The mid curve in each subfigure is the approximation of the average value in different dimensions. The boundary curves are the margins for all function evaluations based on these points' standard deviations. Figure 1 shows that the function evaluations are increasing when the dimension of functions grows, but the complexities are lower than linear.

Table 3: Complexity ($O(D^n)$) of HPSO, MAGA and AEA for Function f_1 to f_8

Function	$HPSO_{25}$	$HPSO_{36}$	$HPSO_{49}$	MAGA	AEA
f_1	**0.72**	0.57	0.61	0.78	1.03
f_2	**0.49**	0.47	0.47	$\times$	1.62
f_3	0.36	0.34	0.35	**0.06**	0.78
f_4	**0.18**	0.15	0.15	0.41	1.35
f_5	0.47	0.40	0.43	**0.39**	1.01
f_6	**0.63**	0.52	0.55	0.8	1.04
f_7	0.42	0.40	0.41	**0.11**	1.08
f_8	0.43	0.39	0.41	**0.15**	1.12

$HPSO_{25}$,$HPSO_{36}$,$HPSO_{49}$ mean the hybrid PSO uses population size 25, 36 and 49 separately. For fairness, only $HPSO_{25}$ is used to compare with the others methods because MAGA only used population size 25. The data of MAGA and AEA come from [19].

5.3 Analysis of the Relationship between Population Size and Function Evaluations

Is there a relationship between population size and function evaluations for the proposal hybrid PSO? Table 3 shows the complexities of function evaluations for HPSO with population sizes 25, 36 and 49. Except for functions f_1 and f_6, the complexity $O(D^n)$ is not changed very much. Figure 2 shows the function evaluations with different population sizes from dimensions 20 to 10000 for functions f_1 to f_8,

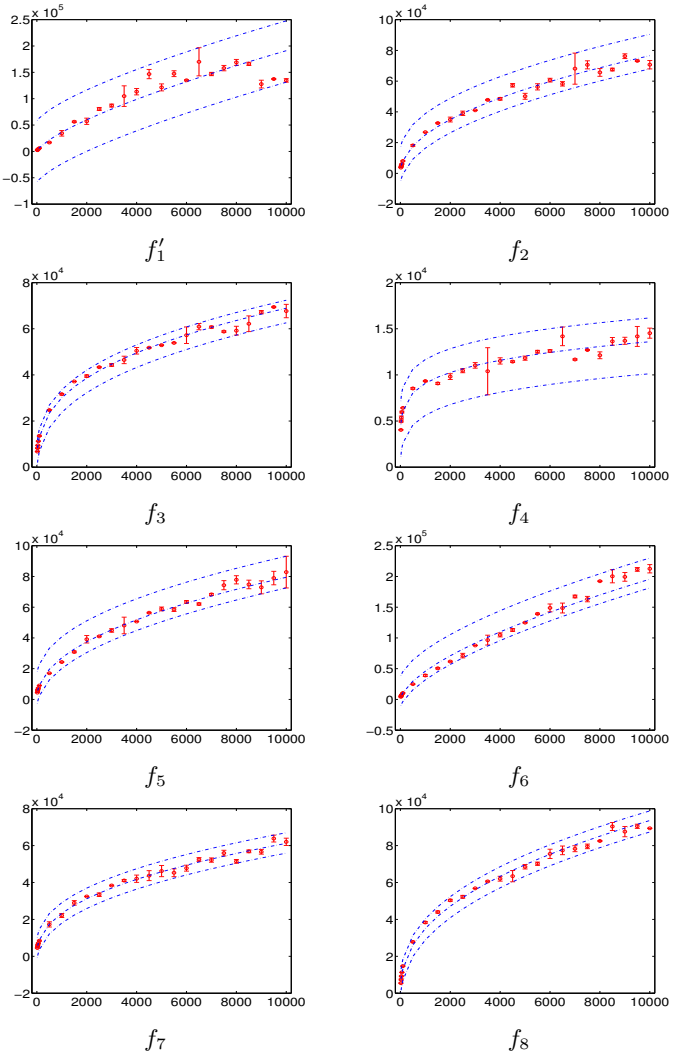

Figure 1: Function evaluations (vertical axis) for f_1' to f_8 with different dimensions (horizontal axis) by the hybrid PSO using population size 25.

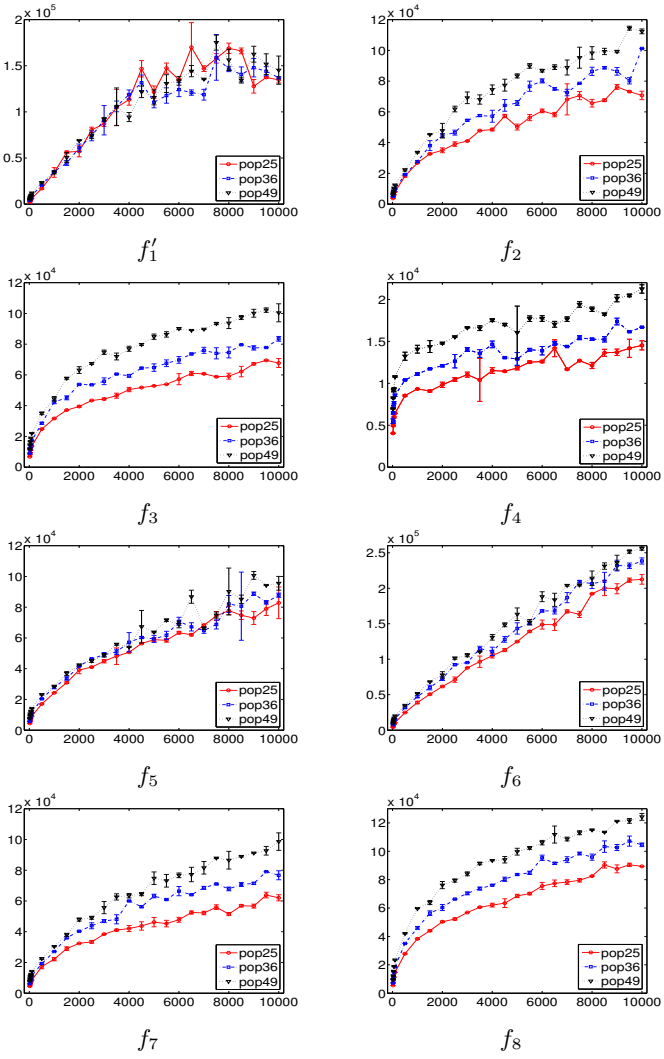

Figure 2: Function evaluations (vertical axis) for f_1' to f_8 with different dimensions (horizontal axis) by the hybrid PSO using population size 25, 36 and 49.

where the horizontal axis gives the dimension and the vertical axis gives function evaluations, and each curve shows the mean $\pm$ standard deviation of function evaluations at different dimensions for one population size. Function evaluations increase with population size, except possibly for function f_1 with high dimension. However, from these curves, the complexities of function evaluations are not markedly increasing.

Table 4 gives a comparison of function evaluations for different population sizes (36, 49 and 100) in low dimensions ($D \leq 1000$). In Table 4, we perform a t-test comparing population size 36, 49, 100 to population size 25 in order to find whether the same distribution exists in function evaluations for different populations size in HPSO. In Table 4, '$\times$' means the comparison from one pair is significantly different, '$\surd$' means that there is no significant difference between the pair. So the pair $(25, 36)$ means that comparison occurs for population sizes 25 and 36 for function evaluations with different dimensions. One pair $(25, 36)$ for dimension 1000 in function f_1 is "$\times$" in Table 4, which means that the number of function evaluations by HPSO with populations size 25 for 1000-dimensional function f_2 is significantly different from that by HPSO with population size 36. There are only five significantly different pairs, namely $(25, 36)$, $(25, 49)$ in 1000-dimensional function f_1, $(25, 36)$ in 500-dimensional and 1000-dimensional function f_2, and $(25, 36)$ in 30-dimensional function f_4. Since the number of function evaluations for the three functions in this dimension range is low, it is a potential reason for these differences. As can be seen, when dimension is not large than 1000 and when the population size is located in this range, function evaluations in HPSO belong to the same distribution with a high probability. Namely, the population size in a rational range does not affect the function evaluation very much, and HPSO is stable with the population size.

6. CONCLUSIONS

The goals of this paper were to investigate a novel hybrid PSO approach to solving high-dimensional function optimisation problems and to analyse the function evaluation

Table 4: Comparison of Function Evaluations in Population Size 36, 49, 100 with Population Size 25

	36					49					100			
Function	30	60	100	500	1000	30	60	100	500	1000	30	60	100	500
f_1	√	√	√	√	×	√	√	√	√	×	√	√	√	√
f_2	√	√	√	×	×	√	√	√	√	√	√	√	√	√
f_3	√	√	√	√	√	√	√	√	√	√	√	√	√	√
f_4	√	√	√	√	√	×	√	√	√	√	√	√	√	√
f_5	√	√	√	√	√	√	√	√	√	√	√	√	√	√
f_6	√	√	√	√	√	√	√	√	√	√	√	√	√	√
f_7	√	√	√	√	√	√	√	√	√	√	√	√	√	√
f_8	√	√	√	√	√	√	√	√	√	√	√	√	√	√

t-test is used and '×′ means significant difference with significance level 0.05, '√′ means insignificant difference. The first row is population size and the second is the function dimension.

complexity with respect to function dimension. By using the strategy of particles sharing history information, integrating three operators of MAGA together with an inversion operator and two local search operators, the proposed hybrid PSO approach was used to solve ten benchmark function optimisation problems. It successfully solved all ten optimisation functions with 30 dimensions, and also successfully found a global optimum for eight functions with dimension ranging from 20 to 10000. Compared with existing PSO-based and non-PSO algorithms examined in this paper, the new hybrid PSO approach can reduce function evaluations remarkably. We analysed the complexity of function evaluations against function dimension for eight function optimisation problems and determined that the relationships are sublinear; the average is lower than $O(D^{0.5})$. In low dimensions (less than 1000), small changes in population size do not have significant or obvious effect on function evaluations in the hybrid PSO method.

For future work we will analyse the interaction between the six operators integrated into the proposed approach. Relative importance and sensitivity of their parameters in the algorithm will be investigated. A potential framework for hybrid PSO techniques based on information sharing will be proposed, based on existing work and this paper, introducing different methods for updating velocity from standard PSO or its variants. We will try to apply the hybrid PSO method to large real-world applications to further examine its ability to find global solutions. It is likely that the inversion operator, designed to take advantage of symmetry, unfairly exploits the particular test problems considered in this paper. Hence we intend to apply HPSO to the CEC'2010 benchmark functions [12] which include shifted and rotated test problems.

7. REFERENCES

[1] R. Akbari and K. Ziarati. Combination of particle swarm optimization and stochastic local search for multimodal function optimization. In *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, pages 388–392, 2008.

[2] N. Aziz, A. Mohemmed, and M. Zhang. Particle swarm optimization for coverage maximization and energy conservation in wireless sensor networks. In *Proceedings of European Conference on the Applications of Evolutionary Computation*, volume 6025, pages 51–60. Springer, 2010.

[3] D. Bratton and T. Blackwell. A simplified recombinant PSO. *J. Artif. Evol. App.*, pages 1–10, January 2008.

[4] W. Fu, M. Johnston, and M. Zhang. Hybrid particle swarm optimisation algorithms based on differential evolution and local search. In *23rd Australasian Joint Conference on Artificial Intelligence*, pages 313–322, 2010.

[5] J. Kennedy and R. Eberhart. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks IV*, pages 1942–1948, 1995.

[6] X. Pan and L. Kang. An adaptive evolutionary algorithms for numerical optimization. In *Simulated Evolution and Learning First Asia-Pacific Conference, SEAL'96*, volume 1285, pages 27–34, 1997.

[7] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization: an overview. *Swarm Intelligence*, 1(1):33–57, 2007.

[8] R. Poli, J. Kennedy, T. Blackwell, and A. Freitas. Particle swarms: the second decade. *Journal of Artificial Evolution and Applications*, 2008, 2008.

[9] K. Price, R. Storn, and J. Lampinen. Differential evolution: a practical approach to global optimization. *Natural Computing Series*, 2005.

[10] M. Setayesh, M. Zhang, and M. Johnston. A new homogeneity-based approach to edge detection using PSO. In *Proceedings of the 24th International Conference on Image and Vision Computing New Zealand*, pages 231–236. IEEE Press, 2009.

[11] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 69–73, 1998.

[12] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2009. http://nical.ustc.edu.cn/cec10ss.php.

[13] H. Wang, H. Li, Y. Liu, S. Zeng, and C. Li. Opposition-based particle swarm algorithm with Cauchy mutation. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4750–4756, 2007.

[14] Y. Wang, Q. Xiang, and Z. Zhao. Particle swarm optimizer with adaptive tabu and mutation: a unified framework for efficient mutation operators. *ACM Transactions on Autonomous and Adaptive Systems*, 5(1):1–27, 2010.

[15] Q. H. Wu and H. L. Liao. High-dimensional function optimisation by reinforcement learning. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[16] B. Xin, J. Chen, Z. Peng, and F. Pan. An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization. *Science China and Springer-Verlag Berlin Heidelberg*, 53(5):980–989, 2010.

[17] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Trans. on Evolutionary Computation*, 3(2):82–102, 1999.

[18] W. Zhang and X. Xie. Depso: Hybrid particle swarm with differential evolution operator. In *IEEE International Conference on Systems, Man & Cybernetics (SMCC)*, pages 3816–3821, 2003.

[19] W. Zheng, J. Liu, M. Xue, and L. Jiao. A multiagent genetic algorithm for global numerical optimization. *IEEE Trans. on Systems. Man and Cybernetics—Part B: Cybernetics*, 34(2):1128–1141, 2004.

APPENDIX

A. BENCHMARK FUNCTIONS

1. *Generalised Schwefel 2.6*

$$f_1 = -\sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|}) \quad x \in [-500, 500]^D \tag{7}$$

$\min f_1 = f_1(420.9687, \ldots, 420.9687) = -D \times 418.983$.
The modified Generalised Schwefel 2.6

$$f_1' = D \times 418.983 - \sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|}) \quad x \in [-500, 500]^D \tag{8}$$

$\min f_1' = f_1'(420.9687, \ldots, 420.9687) = 0$.

2. *Generalised Rastrigin*

$$f_2 = \sum_{i=1}^{D} \{x_i^2 - 10\cos(2\pi x_i) + 10\} \quad x \in [-5.12, 5.12]^D \tag{9}$$

$\min f_2 = f_2(0, \ldots, 0) = 0$.

3. *Ackley*

$$\begin{aligned} f_3 \quad = \quad & -20\exp\{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\} \\ & -\exp\{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\} \\ & +20 + e \quad x \in [-32, 32]^D \end{aligned} \tag{10}$$

$\min f_3 = f_3(0, \ldots, 0) = 0$.

4. *Generalized Griewank*

$$f_4 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1 \quad x \in [-600, 600]^D \tag{11}$$

$\min f_4 = f_4(0, \ldots, 0) = 0$.

5. *Penalized function P8*

$$\begin{aligned} f_5 \quad = \quad & \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}\{1 + 10\sin^2(\pi y_{i+1}\} \\ & +(y_d - 1)^2\} + \sum_{i=1}^{D}\mu(x_i, 10, 100, 4) \\ \text{where} \quad & y_i = 1 + \frac{1}{4}(x_i + 1) \quad x \in [-50, 50]^D \end{aligned} \tag{12}$$

$\min f_5 = f_5(0, \ldots, 0) = 0$.

6. *Penalized function P16*

$$\begin{aligned} f_6 \quad = \quad & 0.1\{10\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2\{1 + \\ & 10\sin^2(3\pi x_{i+1})\} + (x_d - 1)^2\{1 + \sin^2(2\pi x_D)\}\} \\ & +\sum_{i=1}^{D}\mu(x_i, 5, 100, 4) \quad x \in [-50, 50]^D \end{aligned} \tag{13}$$

where

$$\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < a \end{cases}$$

$\min f_6 = f_6(1, \ldots, 1) = 0$.

7. *Sphere/parabola*

$$f_7 = \sum_{i=1}^{D} x_i^2 \quad x \in [-100, 100]^D \tag{14}$$

$\min f_7 = f_7(0, \ldots, 0) = 0$.

8. *Schwefel's Problem 1.2*

$$f_8 = \sum_{i=1}^{D} | x_i | + \prod_{i=1}^{D} | x_i | \quad x \in [-10, 10]^D \tag{15}$$

$\min f_8 = f_8(0, \ldots, 0) = 0$.

9. *Schwefel 1.2*

$$f_9 = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2 \quad x \in [-100, 100]^D \tag{16}$$

$\min f_9 = f_9(0, \ldots, 0) = 0$.

10. *Generalized Rosenbrock*

$$f_{10} = \sum_{i=1}^{D-1}\{100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\} \quad x \in [-30, 30] \tag{17}$$

$\min f_{10} = f_{10}(1, \ldots, 1) = 0$.