An Investigation into the Merger of Stochastic Diffusion Search and Particle Swarm Optimisation

Mohammad Majid al-Rifaie
Department of Computing
Goldsmiths College, University of London
London, SE14 6NW, UK
m.majid@gold.ac.uk

Mark John Bishop
Department of Computing
Goldsmiths College, University of London
London, SE14 6NW, UK
m.bishop@gold.ac.uk

Tim Blackwell
Department of Computing
Goldsmiths College, University of London
London, SE14 6NW, UK
tim.blackwell@gold.ac.uk

ABSTRACT

This study reports early research aimed at applying the powerful resource allocation mechanism deployed in Stochastic Diffusion Search (SDS) [4] to the Particle Swarm Optimiser (PSO) metaheuristic [22], effectively merging the two swarm intelligence algorithms. The results reported herein suggest that the hybrid algorithm, exploiting information sharing between particles, has the potential to improve the optimisation capability of conventional PSOs.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization*

General Terms

Algorithms

Keywords

PSO, SDS, Metaheuristic

1. INTRODUCTION

In the literature swarm intelligence algorithms are typically evaluated using benchmarks that are often small in terms of their objective function computational costs [9, 35]; this is often not the case in real-world applications. This paper is an attempt to pave the way for more effectively optimising computationally expensive objective functions, by deploying the SDS diffusion mechanism to more efficiently allocate PSO resources via information-sharing between particles.

The use of SDS as an efficient resource allocation algorithm was first explored in [29, 7, 26] and these results provided motivation to investigate the application of the information diffusion mechanism originally deployed in SDS [1] with PSOs.

Communication – social interaction or information exchange – observed in social insects is important in all swarm intelligence algorithms, including SDS and PSOs. Although in real social interactions, not just the syntactical information is exchanged between the individuals but also semantic rules and beliefs about how to process this information [23], in typical swarm intelligence algorithms, only the syntactical exchange of information is considered.

In this paper, the two swarm intelligence algorithms are introduced, followed by the hybridisation strategy[2]. Afterwards, the results are reported and the performance of the hybrid algorithm is discussed.

2. STOCHASTIC DIFFUSION SEARCH

This section introduces Stochastic Diffusion Search (SDS) [4], a multi-agent global search and optimisation algorithm, which is based on simple interaction of agents. A high-level description of SDS is presented in the form of a social metaphor demonstrating the procedures through which SDS allocates resources.

SDS introduced a new probabilistic approach for solving best-fit pattern recognition and matching problems. SDS, as a multi-agent population-based global search and optimisation algorithm, is a distributed mode of computation utilising interaction between simple agents [8].

Unlike many nature inspired search algorithms, SDS has a strong mathematical framework, which describes the behaviour of the algorithm by investigating its resource allocation [26], convergence to global optimum [27], robustness and minimal convergence criteria [25] and linear time complexity [28]. In order to introduce SDS, a social metaphor *the Mining Game* [1] is used.

2.1 The Mining Game

This metaphor provides a simple high-level description of the behaviour of agents in SDS, where mountain range is divided into hills and each hill is divided into regions:

[1]The 'information diffusion' and 'randomised partial objective function evaluation' processes enable SDS to more efficiently optimise problems with costly [discrete] objective functions; see Stochastic Diffusion Search Section for an introduction to the SDS metaheuristic.

[2]In addition to the hybridised approach introduced here, there has been other works on merging PSO with other algorithms, such as Genetic Algorithm (GA), which are population based heuristic search technique [18, 13, 32, 30].

A group of miners learn that there is gold to be found on the hills of a mountain range but have no information regarding its distribution. To maximize their collective wealth, the maximum number of miners should dig at the hill which has the richest seams of gold (this information is not available a-priori). In order to solve this problem, the miners decide to employ a simple Stochastic Diffusion Search.

- At the start of the mining process each miner is randomly allocated a hill to mine (his hill hypothesis, h).
- Every day each miner is allocated a randomly selected region, on the hill to mine.

At the end of each day, the probability that a miner is happy is proportional to the amount of gold he has found. Every evening, the miners congregate and each miner who is not happy selects another miner at random for communication. If the chosen miner is happy, he shares the location of his hill and thus both now maintain it as their hypothesis, h; if not, the unhappy miner selects a new hill hypothesis to mine at random.

As this process is isomorphic to SDS, miners will naturally self-organise to congregate over hill(s) of the mountain with high concentration of gold.

In the context of SDS, agents take the role of miners; active agents being 'happy miners', inactive agents being 'unhappy miners and the agent's hypothesis being the miner's 'hill-hypothesis'.

Algorithm 1 The Mining Game

```
Initialisation phase
Allocate each miner (agent) to a random
  hill (hypothesis) to pick a region randomly

Until (all miners congregate over the highest
   concentration of gold)

   Test phase
      Each miner evaluates the amount of gold
        they have mined (hypotheses evaluation)
      Miners are classified into happy (active)
        and unhappy (inactive) groups

   Diffusion phase
      Unhappy miners consider a new hill by
        either communicating with another miner
        or, if the selected miner is also
        unhappy, there will be no information
        flow between the miners; instead the
        selecting miner must consider another
        hill (new hypothesis) at random
End
```

2.2 SDS Architecture

The SDS algorithm commences a search or optimisation by initialising its population (e.g. miners, in the mining game metaphor). In any SDS search, each agent maintains a hypothesis, h, defining a possible problem solution. In the mining game analogy, agent hypothesis identifies a hill. After initialisation two phases are followed (see Algorithm 1 for these phases in the mining game; for high-level SDS description see Algorithm 2):

- Test Phase (e.g. testing gold availability)
- Diffusion Phase (e.g. congregation and exchanging of information)

Algorithm 2 SDS Algorithm

```
Initialising agents()
While (stopping condition is not met)
   Testing hypotheses()
   Diffusion hypotheses()
End
```

In the test phase, SDS checks whether the agent hypothesis is successful or not by performing a partial hypothesis evaluation which returns a boolean value. Later in the iteration, contingent on the precise recruitment strategy employed, successful hypotheses diffuse across the population and in this way information on potentially good solutions spreads throughout the entire population of agents.

In the Test phase, each agent performs *partial function evaluation, pFE*, which is some function of the agent's hypothesis; $pFE = f(h)$. In the mining game the partial function evaluation entails mining a random selected region on the hill, which is defined by the agent's hypothesis (instead of mining all regions on that hill).

In the Diffusion phase, each agent recruits another agent for interaction and potential communication of hypothesis. In the mining game metaphor, diffusion is performed by communicating a hill hypothesis.

2.3 Standard SDS and Passive Recruitment

In standard SDS (which is used in this paper), *passive recruitment mode* is employed. In this mode, if the agent is inactive, a second agent is randomly selected for diffusion; if the second agent is active, its hypothesis is communicated (*diffused*) to the inactive one. Otherwise there is no flow of information between agents; instead a completely new hypothesis is generated for the first inactive agent at random (see Algorithm 3).

Algorithm 3 Passive Recruitment Mode

```
for ag = 1 to No_of_agents
   if (ag.activity() == false)
      r_ag = pick a random agent()
      if (r_ag.activity() == true)
         ag.setHypothesis(r_ag.getHypothesis())
      else
         ag.setHypothesis(randomHypothsis())
end
```

2.4 Partial Function Evaluation

One of the concerns associated with many optimisation algorithms (e.g. Genetic Algorithm [15], Particle Swarm Optimisation [22] and etc.) is the repetitive evaluation of a computationally expensive fitness functions. In some applications, such as tracking a rapidly moving object, the repetitive function evaluation significantly increases the computational cost of the algorithm. Therefore, in addition to reducing the number of function evaluations, other measures can be used in an attempt to reduce the computations carried out during the evaluation of each possible solution, as part of the overall optimisation (or search) processes.

The commonly used benchmarks for evaluating the performance of swarm intelligence algorithms are typically small in terms of their objective functions computational costs [9, 35], which is often not the case in real-world applications.

Examples of costly evaluation functions are seismic data interpretation [35], selection of sites for the transmission infrastructure of wireless communication networks and radio wave propagation calculations of one site [34] etc.

Costly objective function evaluations have been investigated under different conditions [19] and the following two broad approaches have been proposed to reduce the cost of function evaluations:

- The first is to estimate the fitness by taking into account the fitness of the neighbouring elements, the former generations or the fitness of the same element through statistical techniques introduced in [5, 11].
- In the second approach, the costly fitness function is substituted with a cheaper, approximate fitness function.

When agents are about to converge, the original fitness function can be used for evaluation to check the validity of the convergence [19].

Many fitness functions are decomposable to components that can be evaluated separately. In partial evaluation of the fitness function in SDS, the evaluation of one or more of the components may provide partial information to guide the subsequent optimisation process.

3. PARTICLE SWARM OPTIMISATION

Particle Swarm Optimisation (PSO) is population based optimization technique developed in 1995 by Kennedy and Eberhart [22, 10]. It came about as a result of an attempt to graphically simulate the choreography of fish schooling or birds flying (e.g. pigeons, starlings, and shorebirds) in coordinated flocks that show strong synchronisation in turning, initiation of flights and landing, despite the fact that experimental researches to find leaders in such flocks failed [16]. In particle swarms, although members of the swarm neither have knowledge about the global behaviour of the swarm nor a global information about the environment, the local interactions of the swarms result in complex collective behaviour, such as flocking, herding, schooling, exploration and foraging behaviour [31, 24, 3, 17].

3.1 Standard/Basic PSO

A swarm in PSO algorithm comprises of a number of particles and each particle represents a point in a multi-dimensional problem space. Particles in the swarm explore the problem space searching for the optimal position, which is defined by a fitness function. The position of each particle, x, is thus dependent on the particle's own experience and those of its neighbours. Each particle has a memory, containing the best position found so far during the course of the optimisation, which is called personal best (pbest or p). Whereas the best position so far found throughout the population, or the local neighbourhood, is called global best (p_g) and local best (p_l) respectively.

The standard PSO algorithm defines the position of each particle by adding a velocity to the current position. Here is the equation for updating the velocity of each particle:

$$v_{id}^{t} = w v_{id}^{t-1} + c_1 r_1 \left(p_{id} - x_{id}^{t-1}\right) + c_2 r_2 \left(p_{gd} - x_{id}^{t-1}\right) \quad (1)$$

$$x_{id}^{t} = v_{id}^{t} + x_{id}^{t-1} \quad (2)$$

where w is the inertia weight whose optimal value is problem dependent [33]; $\vec{v}_{id}^{t-1}$ is the velocity vector of particle i in dimension d at time step $t-1$; $c_{1,2}$ are the learning factors (also referred to as acceleration constants) for personal best and neighbourhood best respectively (they are generally constant and are usually set to 2); $\vec{r}_{1,2}$ are random numbers adding stochasticity to the algorithm and they are drawn from a uniform distribution on the unit interval $U(0,1)$; $\vec{p}_{id}$ is the personal best position of particle x_i in dimension d; and p_{gd} is global best (or neighbourhood best).
Therefore, PSO optimisation is based on particles' individual experience and their social interaction with the particle swarms.

After updating the velocities of the particles, their new positions are determined. Algorithm 4 summarises the behaviour of PSO algorithm when dealing with a minimisation problem.

Algorithm 4 PSO Pseudo Code

```
Initialise particles

While ( stopping condition is not met )
   For all particles
      Evaluate fitness value of each particle

      If (current fitness < pbest)
         pbest = current fitness
      If (pbest<global (or local) best)
         global (or local) best = pbest
      Update particle velocity
      Update particle position
   End
End
```

In this paper, Clerc-Kennedy PSO (PSO-CK) or constriction PSO is used:

$$v_{id}^{t} = \chi \left(v_{id}^{t-1} + c_1 r_1 \left(p_{id} - x_{id}^{t-1}\right) + c_2 r_2 \left(p_{gd} - x_{id}^{t-1}\right)\right) \quad (3)$$

where $\chi = 0.72984$ [6] is reported to be working well in general.

4. MERGING SDS AND PSO

The initial motivating thesis justifying the merging SDS and PSO is the partial function evaluation deployed in SDS, which may mitigate the high computational overheads entailed when deploying a PSO onto a problem with a costly fitness function. However before commenting on and exploring this area – which remains an ongoing research –, an initial set of experiments experiment aimed to investigate if the information diffusion mechanism deployed in SDS may on its own improve PSO behaviour. It is these results that are primarily reported in this paper.

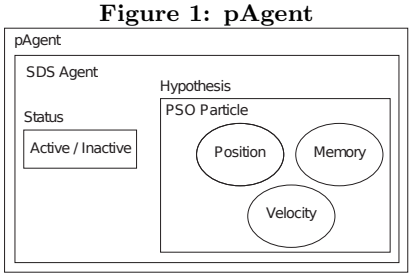
In this new architecture a standard set of benchmarks are used to evaluate the performance of the hybrid algorithm. The resource allocation (or recruitment) and partial function evaluation sides of SDS (see Section 2.4 are used to assist allocating resources (e.g. particles of the swarm) after partially evaluating the search space.

In the hybrid algorithm, each PSO particle has a current position, a memory (personal best position) and a velocity; each SDS agent, on the other hand, has hypothesis and status.

In the experiment reported here, every PSO particle is an SDS agent too – together termed *pAgents*. In the pA-

gent SDS hypotheses are defined by the PSO particle positions and an additional boolean variable (status) determines whether the pAgent is active or inactive (see Figure 1).

Figure 1: pAgent



The behaviour of the hybrid algorithm in its simplest form is presented in Algorithm 5.

Algorithm 5 Hybrid Algorithm

```
Initialise pAgents

While ( stopping condition is not met )
   For all pAgents
      Evaluate fitness value of each particle

      If ( evaluation counter MOD n == 0 )
      // START SDS
      // TEST PHASE
         for ag = 1 to No_of_pAgents
            r_ag = pick-random-pAgent()
            if ( ag.pbestFitness() <=
              r_ag.pbestFitness() )
               ag.setActivity (true)
            else
               ag.setActivity (false)
            end if
         end for

        // DIFFUSION PHASE
        for ag = 1 to No_of_pAgents
           if ( ag.activity() == false )
             r_ag = pick-random-pAgent()
             if ( r_ag.activity() == true )
                ag.setHypo( r_ag.getHypo() )*
             else
                ag.setHypo( randomHypo() )
             end if
        end for
      end if
      // END SDS

      If (current fitness < pbest)
         pbest = current fitness
      If (pbest<global (or local) best)
         global (or local) best = pbest
      Update particle velocity
      Update particle position
   End
End

* In setHypo() and getHypo(), Hypo refers to
the pAgent's hypothesis (position, memory
and velocity).
```

4.1 Test and Diffusion Phases in the Hybrid Algorithms

In the test-phase of a stochastic diffusion search, each agent has to partially evaluate its hypothesis. The guiding heuristic is that hypotheses that are promising are maintained and those that appear unpromising are discarded. In the context of the hybrid PSO-SDS algorithm, it is clear that there are many different tests that could be performed in order to determine the activity of each pAgent. A very simple test is illustrated in Algorithm 5. Here, the test-phase is simply conducted by comparing the fitness of each pAgent's particle's personal best against that of a random pAgent; if the selecting pAgent has a better fitness value, it will become active, otherwise it is flagged inactive. On average, this mechanism will ensure 50% of pAgents remain active form one iteration to another[3].

In the Diffusion Phase, each inactive pAgent picks another pAgent randomly, if the selected pAgent is active, the selected pAgent communicates its hypothesis to the inactive one; if the selected pAgent is inactive too, the selecting pAgent generates a new hypothesis at random from the search space.

As outlined in the pseudo-code of the hybrid algorithm (see Algorithm 5), after each n number of PSO function evaluations, one full SDS cycle[4] is executed. The hybrid algorithm is called $SDSnPSO$, where n refers to the number of PSO function evaluations before an SDS cycle should run.

In the next section, the experiment setup is reported and the results will follow.

5. RESULTS

In this section, a number of experiments are carried out and the performance of PSO is contrasted against the hybrid algorithm, $SDSnPSO$.

5.1 Experiment Setup

The algorithms are tested over a number of benchmarking functions from Jones et al [20] and De Jong [21] test suite, preserving different dimensionality and modality (see Table I and II in [6], where benchmark function equations, feasible bounds, the number of dimensions in which the benchmarks are used in the experiments, the optimum of each function which is known *a priori* and also the boundaries where particles are first initialised are presented).

The first two functions (Sphere/Parabola and Schwefel 1.2) have a single minimum and are unimodal functions; Generalised Rosenbrock for dimension D, where $D > 3$, is multimodal; Generalised Schwefel 2.6, Generalized Rastrigin, Ackley, Generalized Griewank, Penalised Function P8 and Penalised Function P16 are complex high-dimensional multi-modal problems with many local minima and a single global optimum; Six-hump Camel-back, Goldstein-Price, Shekel 5, 7 and 10 are lower-dimensional multi-modal problems with fewer local minima. Goldstein-Price, Shekel 5, 7 and 10 have one global optimum and Six-hump Camel-back has two global optima symmetric about the origin.

In order not to initialise the particles on or near a region in the search space known to have the global optimum, *region*

[3]NB. In standard SDS such high average activity would not be useful as it entails most agents will continue to exploit their current hypothesis rather than explore the search space, however in the hybrid algorithm the randomised subsequent behaviour of each pAgent offsets this effect.

[4]A full SDS cycle includes:

- one Test Phase which decides about the status of each pAgent, one after another
- one Diffusion Phase which shares information according to the algorithm presented

scaling technique is used [14], which makes sure particles are initialised at a corner of the search space where there are no optimal solution.

The experiments are conducted with the population of 50 particles in global neighbourhood. The halting criterion for this experiment is either to reach the optima (with distances less than 10^{-8}) or to exceed the 300, 000 function evaluations (FEs).

There are 30 independent runs for each benchmark function and the results are averaged over these independent trials. Three different performance measures[12] are used in this paper to compare different strategies used in the experiment. Accuracy, reliability and efficiency of each one of the algorithms are presented in separate tables.

Accuracy is defined by the quality of the global best position in terms of its closeness to the optimum position. If knowledge about the optimum position is known *a priori* (which is the case here), the following would define the accuracy:

$$\text{Accuracy}\,(S,t) = \left|f\left(p_g^t\right) - f\left(x_{opt}\right)\right| \tag{4}$$

where p_g^t is the global best position at time t and x_{opt} is the position of the known optimum solution.

Reliability is the percentage of trials where swarms converge with a specified accuracy and it is defined by:

$$\text{Reliability} = \frac{n^{'}}{n} \times 100 \tag{5}$$

where n is the number of trials in the experiment and $n^{'}$ is the number of successful trials.

Efficiency is the number of iterations or objective function evaluations to converge with a specified accuracy:

$$\text{Efficiency} = \frac{1}{n}\sum_{i=0}^{n} FEs \tag{6}$$

where n is the number of trials and FEs is the number of function evaluations before convergence.

In this paper, $SDSnPSO$, is presented with few variations of parameter, n, (the number of PSO evaluation before an SDS cycle is performed), $n = 1000$, 3000, and 30, 000. These values were selected merely to provide a brief initial exploration of the behaviour of the new hybrid algorithm over three relatively widely separated parameter values; no claim is made for their optimality.

5.2 Results

Table 1 shows the performance of the various hybrid algorithms alongside PSO-CK. For each benchmark and each algorithm, the table shows the accuracy, efficiency and reliability.

Although the focus of this paper is not finding the best n for $SDSnPSO$ (for this set of benchmarks), $n = 3000$ shows better results compared to other variants. The result table suggests that over-running the SDS cycle (e.g. when $n = 1000$) might move the swarm away from convergence. On the other hand, reducing the information sharing (e.g. when $n = 30,000$) appears to reduce the positive effect that SDS has on the overall behaviour of the swarm.

As Table 1 shows (for statistical details, see Tables 2 and 3), there is a trade-off between the reliability and the efficiency measures of $SDSnPSO$ and PSO. Adding SDS, decreases the efficiency, but increases the reliability. This can be viewed in f_{1-2} and f_{6-9}. In terms of the total number of convergences, $SDSnPSO$ ($n = 3000$), outperforms PSO (see the next section for detailed comparison and statistical analysis of the results).

6. DISCUSSION

The resource allocation process underlying SDS offers two closely coupled mechanisms to the algorithm's search component to speed its convergence to global optima. The first component is 'efficient, non-greedy information sharing' instantiated via positive feedback of potentially good hypotheses between agents; the second component is random 'partial hypothesis evaluation', whereby a complex, computationally expensive objective function is broken down into 'k independent partial-functions', each one of which, when evaluated, offers partial information on the absolute quality of current algorithm search parameters. It is this mechanism of iterated selection of a *random* partial function that ensures SDS does not prematurely converge on local minimum.

The resource allocation component of SDS in the hybrid algorithm is executed in the 'Diffusion Phase', when information is shared (diffused) among pAgents (see Algorithm 3). Analysis of the performance of the hybrid algorithm (see results above) demonstrates that adding the SDS resource allocation mechanism to the standard PSO architecture increases the probability of convergence (i.e. it increases algorithm 'reliability', as defined herein). In order to ensure that this improved robustness is due to the stochastic information sharing mechanism in SDS - and not merely an effect caused by randomising a selection of particle hypotheses after a number of PSO function evaluations (effectively instantiating a PSO with random-restarts) - a control experiment is run with a modified SDS Diffusion Phase (see Algorithm 6); in the control algorithm, after the test-phase the hypothesis of each inactive pAgent is merely randomised.

The performance of the control algorithm can be contrasted against PSO using three measures (accuracy, efficiency and reliability) defined in Section 5.1. TukeyHSD test is used for accuracy and efficiency measures, (see Tables 2 and 3).

In terms of accuracy, Tables 1 and 2, illustrate that no algorithm outperforms over all benchmarks. Similarly, Table 3 shows that in the case of successful convergence, whenever there is a significant difference between any pair of the algorithms, the efficiency of H3C is significantly worse than PSO-CK and the hybrid algorithm (H1, H3 & H30); and as the last row of Table 1 proves, the control algorithm is less reliable than PSO and the hybrid algorithm (H1, H3 & H30). As the efficiency and reliability of the control experiment (see the last column in Table 1) is worse than that of the hybrid algorithm, we can conclude that the SDS information sharing mechanism must play an essential role in improving the performance of the hybrid algorithm.

The second SDS component feature, which is currently only implicitly exploited by the hybrid algorithm, is 'randomised partial hypothesis evaluation'. In the Mining Game (see Section 2.1), "At the start of the mining process each miner maintains a [randomly allocated] hypothesis - their current belief of 'best hill' to mine"; and each miner mines one small randomly selected area of this hill rather than the entirety of it (i.e. revealing a partial estimate of the

Table 1: Accuracy and Efficiency Details

Accuracy (±standard error) is shown with two decimal places after 30 trials of 300,000 function evaluations. Mean FEs (±standard error) of successful trials are also shown in the second row of each benchmark alongside with the reliability of the algorithm. Total number of convergence of each algorithm over the benchmarks can be found in the last row.

	PSO-CK	**H1: SDSnPSO** ***n = 1,000*** ***generate Hypothesis***	**H3: SDSnPSO** ***n = 3,000*** ***generate Hypothesis***	**H30: SDSnPSO** ***n = 30,000*** ***generate Hypothesis***	**H3C: SDSnPSO** ***n = 3,000*** ***control algorithm***
f_1	**0.0 ±0.0** 23273 ±321 (100%)	**0.0 ±0.0** 40265 ±1006 (100%)	**0.0 ±0.0** 32842 ±736 (100%)	**0.0 ±0.0** 22386 ±265 (100%)	**0.0 ±0.0** 202963 ±4929 (100%)
f_2	**0.0 ±0.0** 183450 ±1655 (100%)	1.87E-08 ±8.90E-09 244969 ±4571 (93.33%)	**0.0 ±0.0** 212703 ±3172 (100%)	**0.0 ±0.0** 184962 ±2013 (100%)	2.30E-01 ±2.07E-02 -
f_3	5.53E+00 ±8.15E-01 -	1.23E+00 ±3.69E-01 -	1.63E+00 ±3.88E-01 -	1.43E+00 ±3.48E-01 -	6.45E+01 ±9.99E+00 -
f_4	3.83E+03 ±9.35E+01 -	3.30E+03 ±1.20E+02 -	2.59E+03 ±8.25E+01 -	2.96E+03 ±1.27E+02 -	1.06E+03 ±3.81E+01 -
f_5	6.04E+01 ±3.47E+00 -	3.84E+01 ±2.82E+00 -	9.55E+00 ±9.49E-01 -	1.56E+01 ±1.02E+00 -	3.76E+00 ±3.25E-01 -
f_6	6.78E-01 ±1.42E-01 46728 ±3408 (53.33%)	7.24E-01 ±1.57E-01 163539 ±9356 (53.33%)	1.04E-08 ±6.61E-10 67155 ±1989 (96.67%)	8.86E-02 ±6.21E-02 77059 ±5350 (93.33%)	3.61E-06 ±3.96E-07 -
f_7	1.70E-02 ±2.90E-03 23865 ±713 (26.67%)	5.92E-02 ±1.34E-02 41641 ±2924 (16.67%)	1.93E-02 ±3.59E-03 33131 ±1118 (33.33%)	2.21E-02 ±3.61E-03 22818 ±507 (26.67%)	1.35E-02 ±2.45E-03 236479 ±11177 (26.67%)
f_8	5.19E-02 ±3.06E-02 33934 ±1803 (83.33%)	1.38E-02 ±6.54E-03 58005 ±2436 (86.67%)	**0.0 ±0.0** 43019 ±1338 (100%)	1.73E-02 ±1.41E-02 37734 ±1626 (93.33%)	0.0 ±0.0 118090 ±2866 (100%)
f_9	1.32E-02 ±6.24E-03 29543 ±1495 (86.67%)	1.03E-02 ±5.72E-03 56071 ±2795 (90%)	3.30E-03 ±3.30E-03 38946 ±1319 (96.67%)	**0.0 ±0.0** 32684 ±2808 (100%)	**0.0 ±0.0** 152050 ±2951 (100%)
f_{10}	**0.0 ±0.0** 3607 ±61 (100%)	**0.0 ±0.0** 3470 ±81 (100%)	**0.0 ±0.0** 3498 ±65 (100%)	**0.0 ±0.0** 3661 ±74 (100%)	**0.0 ±0.0** 3551 ±79 (100%)
f_{11}	**0.0 ±0.0** 3880 ±63 (100%)	**0.0 ±0.0** 3534 ±57 (100%)	**0.0 ±0.0** 3832 ±78 (100%)	**0.0 ±0.0** 3984 ±79 (100%)	**0.0 ±0.0** 3921 ±94 (100%)
f_{12}	3.44E+00 ±5.44E-01 -	3.26E+00 ±6.25E-01 104323 (3.33%)	4.10E+00 ±6.06E-01 -	3.52E+00 ±5.96E-01 -	8.47E-01 ±3.52E-01 -
f_{13}	3.05E+00 ±6.18E-01 -	2.86E+00 ±6.22E-01 -	2.09E+00 ±5.61E-01 -	2.87E+00 ±6.21E-01 -	3.52E-01 ±2.44E-01 -
f_{14}	1.47E+00 ±5.52E-01 -	2.76E+00 ±6.42E-01 -	1.53E+00 ±5.70E-01 -	2.65E+00 ±6.57E-01 -	9.70E-01 ±4.09E-01 -
Σ	(195) 46.43%	(193) 45.95%	(218) 51.90%	(214) 50.95%	(158) 37.62%

Algorithm 6 Hybrid Algorithm Control

```
// DIFFUSION PHASE
for ag = 1 to No_of_agents
  if ( ag.activity() == false )
    ag.setHypo( randomHypo() )
  end if
end for
```

the gold content of the entire hill); following this approach, each miner forms a partial view of the gold content of their hill hypothesis (which is merely part of the overall mountain range: the entire search space).

In typical optimisation algorithms, the search process iterates the evaluation of one point in the n-dimensional search space (iterating an objective function evaluation). In a PSO, in addition to this information, each particle has implicit knowledge of a discrete sub-space (or $dSubS$) comprising the historical evidence implicit in the prior [m] objective-function evaluations it has performed. Thus, since the memory of each particle maintains the best point found so far, each particle, covering its $dSubS$, has partial knowledge of the full search space.

In the hybrid algorithm each pAgent maintains a fitness value which is the best objective function value it has currently found, based on its exploration of the search space so far. Thus constituted, each pAgent's personal best defines a 'partial view' of the entire search space (via the dSubS it has covered); hence, when the personal best values of two pAgents are compared in the test-phase of the hybrid algorithm, two partial views of the entire search space are contrasted. This is analogous to the 'test' process of the Mining Game as in both processes, agents become active or inactive contingent upon the agent's evaluation of a randomised partial view of the entire search space.

In both the Mining Game and the new hybrid SDSnPSO algorithm, the notion of partial-function evaluation differs importantly from that traditionally deployed in a simple discrete partial function SDS, where, for a given set of parameter values (the agent hypothesis) a complex objective function is broken into m components, only one randomly selected of which will be evaluated and the subsequent agent-activity is based on this. Clearly, as this process merely evaluates $1/m$ of the total number of computations required for the full hypothesis evaluation, it concomitantly offers a potentially significant performance increase. Whereas in the new hybrid SDSnPSO algorithm the objective function is evaluated in-toto, using a given set of parameter values (the agent's hypothesis) and the subsequent agent-activity is based on this. In the former case, the agent exploits knowledge of the partial objective function and in the process gains a potential partial-function performance dividend; in the latter the agent merely exploits partial knowledge of the search space - its discrete sub-space $dSubS$ - without the concomitant explicit partial-function performance increase. Ongoing work, on computationally more complex benchmark problems, seeks to exploit this 'partial-function dividend' with the hybrid SDSnPSO algorithm; if successful this offers further, potentially significant, performance improvements for the new hybrid algorithm.

Table 2: TukeyHSD Test Results for Accuracy

Based on TukeyHSD Test, if the difference between each pair of algorithms is significant, the pairs are marked and the accuracy difference between each pair is also reported.

	H1-CK	H3-CK	H30-CK	H3C-CK	H3-H1	H30-H1	H3C-H1	H30-H3	H3C-H3	H3C-H30
f_1	- 2.65E-10	- 2.09E-10	- -3.66E-11	x 4.14E-10	- -5.63E-11	- -3.02E-10	- 1.49E-10	- -2.45E-10	- 2.05E-10	- 4.50E-10
f_2	- 8.93E-09	- 6.70E-11	- 2.95E-11	x 2.30E-01	- -8.86E-09	- -8.90E-09	x 2.30E-01	- -3.75E-11	x 2.30E-01	x 2.30E-01
f_3	- -4.31E+00	- -3.90E+00	- -4.10E+00	x 5.89E+01	- 4.04E-01	- 2.04E-01	x 6.32E+01	- -2.00E-01	x 6.28E+01	x 6.30E+01
f_4	x -5.32E+02	x -1.24E+03	x -8.79E+02	x -2.78E+03	x -7.11E+02	- -3.47E+02	x -2.24E+03	- 3.64E+02	x -1.53E+03	x -1.90E+03
f_5	x -2.20E+01	x -5.09E+01	x -4.48E+01	x -5.67E+01	x -2.89E+01	x -2.28E+01	x -3.46E+01	- 6.04E+00	- -5.79E+00	x -1.18E+01
f_6	- 4.53E-02	x -6.78E-01	x -5.90E-01	x -6.78E-01	x -7.24E-01	x -6.35E-01	x -7.24E-01	- 8.86E-02	- 3.60E-06	- -8.86E-02
f_7	x 4.22E-02	- 2.29E-03	- 5.12E-03	- -3.43E-03	x -3.99E-02	x -3.71E-02	x -4.56E-02	- 2.84E-03	- -5.72E-03	- -8.56E-03
f_8	- -3.81E-02	- -5.19E-02	- -3.47E-02	- -5.19E-02	- -1.38E-02	- 3.46E-03	- -1.38E-02	- 1.73E-02	- -1.87E-10	- -1.73E-02
f_9	- -2.93E-03	- -9.89E-03	- -1.32E-02	- -1.32E-02	- -6.96E-03	- -1.03E-02	- -1.03E-02	- -3.30E-03	- -3.30E-03	- 2.92E-10
f_{10}	- 4.58E-10	- 1.42E-09	- 1.66E-10	- 1.52E-09	- 9.63E-10	- -2.93E-10	- 1.06E-09	- -1.26E-09	- 9.68E-11	- 1.35E-09
f_{11}	- 1.67E-10	- -4.13E-10	- 3.25E-10	- 6.85E-11	- -5.79E-10	- 1.58E-10	- -9.83E-11	- 7.38E-10	- 4.81E-10	- -2.57E-10
f_{12}	- -1.81E-01	- 6.62E-01	- 8.01E-02	x -2.59E+00	- 8.43E-01	- 2.61E-01	x -2.41E+00	- -5.82E-01	x -3.26E+00	x -2.67E+00
f_{13}	- -1.89E-01	- -9.53E-01	- -1.78E-01	x -2.69E+00	- -7.64E-01	- 1.17E-02	x -2.50E+00	- 7.75E-01	- -1.74E+00	x -2.52E+00
f_{14}	- 1.29E+00	- 5.73E-02	- 1.18E+00	- -5.01E-01	- -1.23E+00	- -1.03E-01	- -1.79E+00	- 1.13E+00	- -5.58E-01	- -1.68E+00

Table 3: TukeyHSD Test Results for Efficiency

Based on TukeyHSD Test, if the difference between each pair of algorithms is significant, the pairs are marked and the efficiency difference between each pair is also reported. Benchmarks with no convergence are removed.

	H1-CK	H3-CK	H30-CK	H3C-CK	H3-H1	H30-H1	H3C-H1	H30-H3	H3C-H3	H3C-H30
f_1	x 1.70E+04	x 9.57E+03	- -8.87E+02	x 1.80E+05	- -7.42E+03	x -1.79E+04	x 1.63E+05	x -1.05E+04	x 1.70E+05	x 1.81E+05
f_2	x 6.15E+04	x 2.93E+04	- 1.51E+03		x -3.23E+04	x -6.00E+04		x -2.77E+04		
f_6	x 1.17E+05	x 2.04E+04	x 3.03E+04		x -9.64E+04	x -8.65E+04		- 9.90E+03		
f_7	- 1.78E+04	- 9.27E+03	- -1.05E+03	x 2.13E+05	- -8.51E+03	- -1.88E+04	x 1.95E+05	- -1.03E+04	x 2.03E+05	x 2.14E+05
f_8	x 2.41E+04	x 9.08E+03	- 3.80E+03	x 8.42E+04	x -1.50E+04	x -2.03E+04	x 6.01E+04	- -5.29E+03	x 7.51E+04	x 8.04E+04
f_9	x 2.65E+04	- 9.40E+03	- 3.14E+03	x 1.23E+05	x -1.71E+04	x -2.34E+04	x 9.60E+04	- -6.26E+03	x 1.13E+05	x 1.19E+05
f_{10}	- -1.37E+02	- -1.09E+02	- 5.42E+01	- -5.63E+01	- 2.81E+01	- 1.91E+02	- 8.09E+01	- 1.63E+02	- 5.28E+01	- -1.11E+02
f_{11}	x -3.46E+02	- -4.74E+01	- 1.04E+02	- 4.12E+01	x 2.99E+02	x 4.50E+02	x 3.87E+02	- 1.51E+02	- 8.86E+01	- -6.24E+01

6.1 Conclusion

This paper presents a brief overview about the potential of integration of PSO with SDS. Here, SDS is primarily used as an efficient resource allocation mechanism responsible for facilitating communication between PSO particles. Additionally, an initial discussion of the similarity between the hypothesis test employed in the hybrid algorithm and the test-phase in SDS algorithm is presented, emphasising on the role of discrete sub-space evaluation (*dSubS* evaluation).

Results reported in this paper have demonstrated that initial explorations with the hybrid SDSnPSO algorithm outperform the performance of standard PSO architectures, even when applied to problems with low-cost fitness function evaluations (the benchmarks presented).

In ongoing research, in addition to investigating the performance of the hybrid algorithm in other sets of problems (e.g. CEC2005 or some real-world problems), further theoretical work seeks to develop the core ideas presented in this paper on problems with significantly more computationally expensive objective functions, where the performance improvement (relative to standard PSO) is anticipated to be much greater.

The artistic application of this integration is under further investigation and the early results are reported in [2].

7. REFERENCES

[1] M. M. al-Rifaie and M. Bishop. The mining game: a brief introduction to the stochastic diffusion search metaheuristic. *AISB Quarterly*, 2010.

[2] M. M. al-Rifaie, M. Bishop, and A. Aber. Creative or not? birds and ants draw with muscles. In *AISB 2011: Computing and Philosophy*, pages 23–30, University of York, York, U.K., 2011. ISBN: 978-1-908187-03-1.

[3] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Roadmap-based flocking for complex environments. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 104, Washington, DC, USA, 2002. IEEE Computer Society.

[4] J. Bishop. Stochastic searching networks. pages

329–331, London, UK, 1989. Proc. 1st IEE Conf. on Artificial Neural Networks.

[5] J. Branke, C. Schmidt, and H. Schmeck. Efficient fitness estimation in noisy environments. *In Spector, L., ed.: Genetic and Evolutionary Computation Conference, Morgan Kaufmann*, 2001.

[6] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Proc of the Swarm Intelligence Symposium*, pages 120–127, Honolulu, Hawaii, USA, 2007. IEEE.

[7] K. de Meyer. Explorations in stochastic diffusion search: Soft- and hardware implementations of biologically inspired spiking neuron stochastic diffusion networks. Technical Report KDM/JMB/2000/1, University of Reading, 2000.

[8] K. de Meyer, J. M. Bishop, and S. J. Nasuto. Stochastic diffusion: Using recruitment for search. *Evolvability and interaction: evolutionary substrates of communication, signalling, and perception in the dynamics of social complexity (ed. P. McOwan, K. Dautenhahn & CL Nehaniv) Technical Report*, 393:60–65, 2003.

[9] J. Digalakis and K. Margaritis. An experimental study of benchmarking functions for evolutionary algorithms. *International Journal*, 79:403–416, 2002.

[10] 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*, volume 43. New York, NY, USA: IEEE, 1995.

[11] M. A. el Beltagy and A. J. Keane. Evolutionary optimization for computationally expensive problems using gaussian processes. In *Proc. Int. Conf. on Artificial Intelligence'01*, pages 708–714. CSREA Press, 2001.

[12] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2006.

[13] A. A. A. Esmin, G. Lambert-Torres, and G. B. Alvarenga. Hybrid evolutionary algorithm based on PSO and GA mutation. In *Hybrid Intelligent Systems, 2006. HIS'06. Sixth International Conference on*, page 57, 2006.

[14] D. Gehlhaar and D. Fogel. Tuning evolutionary programming for conformationally flexible molecular docking. In *Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming*, pages 419–429, 1996.

[15] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[16] F. Heppner and U. Grenander. A stochastic nonlinear model for coordinated bird flocks. *American Association for the Advancement of Science, Washington, DC(USA).*, 1990.

[17] C. H. Janson. Experimental evidence for spatial memory in foraging wild capuchin monkeys,cebus apella. *Animal Behaviour*, 55:1229–1243, 1998.

[18] S. Jeong, S. Hasegawa, K. Shimoyama, and S. Obayashi. Development and investigation of efficient GA/PSO-hybrid algorithm applicable to real-world design optimization. *Computational Intelligence Magazine, IEEE*, 4(3):36–44, 2009.

[19] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *In: Soft Computing*, 9:3–12, 2005.

[20] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *J. Optim. Theory Appl.*, 79(1):157–181, 1993.

[21] K. A. D. Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[22] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[23] J. F. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm intelligence*. Morgan Kaufmann Publishers, San Francisco ; London, 2001.

[24] M. Mataric. *Interaction and Intelligent Behavior*. PhD thesis, Department of Electrical, Electronics and Computer Engineering, MIT, USA, 1994.

[25] D. R. Myatt, J. M. Bishop, and S. J. Nasuto. Minimum stable convergence criteria for stochastic diffusion search. *Electronics Letters*, 40(2):112–113, 2004.

[26] S. J. Nasuto. *Resource Allocation Analysis of the Stochastic Diffusion Search*. PhD thesis, University of Reading, Reading, UK, 1999.

[27] S. J. Nasuto and J. M. Bishop. Convergence analysis of stochastic diffusion search. *Parallel Algorithms and Applications*, 14(2), 1999.

[28] S. J. Nasuto, J. M. Bishop, and S. Lauria. Time complexity of stochastic diffusion search. *Neural Computation*, NC98, 1998.

[29] S. J. Nasuto and M. J. Bishop. Steady state resource allocation analysis of the stochastic diffusion search. *cs.AI/0202007*, 2002.

[30] K. Premalatha and A. M. Natarajan. Hybrid PSO and GA for global maximization. *Int. J. Open Problems Compt. Math*, 2(4), 2009.

[31] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[32] X. H. Shi, Y. H. Lu, C. G. Zhou, H. P. Lee, W. Z. Lin, and Y. C. Liang. Hybrid evolutionary algorithms based on PSO and GA. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 4, pages 2393–2399, 2004.

[33] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. *Lecture notes in computer science*, pages 591–600, 1998.

[34] R. Whitaker and S. Hurley. An agent based approach to site selection for wireless networks. In *1st IEE Conf. on Artificial Neural Networks*, Madrid Spain, 2002. ACM Press Proc ACM Symposium on Applied Computing.

[35] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias. Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2):245–276, 1996.