Noname manuscript No.
(will be inserted by the editor)

Information Sharing Impact of Stochastic Diffusion Search on Differential Evolution Algorithm

Mohammad Majid al-Rifaie and John Mark Bishop and Tim Blackwell

Received: date / Accepted: date

Abstract This work details the research aimed at applying the powerful resource allocation mechanism deployed in Stochastic Diffusion Search to the Differential Evolution, effectively merging a nature inspired swarm intelligence algorithm with a biologically inspired evolutionary algorithm. The results reported herein suggest that the hybrid algorithm, exploiting information sharing between the population elements, has the potential to improve the optimisation capability of classical Differential Evolution algorithms. This claim is verified by running several experiments using state-of-the-art benchmarks. Additionally, the significance of the frequency within which Stochastic Diffusion Search introduces communication and information exchange is also investigated.

Keywords Stochastic Diffusion Search · Differential Evolution · Metaheuristic · Global Optimisation · Information Sharing

1 Introduction

In the literature, nature inspired swarm intelligence algorithms and biologically inspired evolutionary algorithms are typically evaluated using benchmarks that are often small in terms of their objective function computational costs [14, 41]; this is often not the case in real-world applications. This paper

Mohammad Majid al-Rifaie
Goldsmiths, University of London, New Cross, London SE14 6NW, United Kingdom; E-mail: m.majid@gold.ac.uk

John Mark Bishop
Goldsmiths, University of London, New Cross, London SE14 6NW, United Kingdom; E-mail: m.bishop@gold.ac.uk

Tim Blackwell
Goldsmiths, University of London, New Cross, London SE14 6NW, United Kingdom; E-mail: tim.blackwell@gold.ac.uk

is an attempt to pave the way for more effectively optimising computationally expensive objective functions, by deploying the Stochastic Diffusion Search (SDS) diffusion mechanism to more efficiently allocate Differential Evolution (DE) resources via information-sharing between the members of the population.

The use of SDS as an efficient resource allocation algorithm was first explored in [31,22,28] and these results provided motivation to investigate the application of the information diffusion mechanism originally deployed in SDS[1] with DE.

Communication – social interaction or information exchange – observed in social insects is important in all swarm intelligence and evolutionary algorithms, including SDS and DE algorithms.

This work investigates the communication between the members of the population as the mean to maintain population diversity, which is facilitated by using the resource allocation and resource dispensation of SDS algorithm.

In a former work [6], SDS is merged with Particle Swarm Optimisation (PSO) algorithm and the promising results of this hybridisation alongside some statistical analysis of its performance are reported.

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 [19], in typical swarm intelligence algorithms, only the syntactical exchange of information is considered.

In the study of the interaction of social insects, two important elements are the individuals and the environment, which will result in two integration schemes: the first one is the way in which individuals self-interact and the second one is the interaction of the individuals with the environment [11]. Self-interaction between individuals is carried out through recruitment strategies and it has been demonstrated that, typically, differing recruitment strategies are used by ants [16] and honey bees. These recruitment strategies are used to attract other members of the society to gather around one or more desired areas, either for foraging purposes or for moving to a new nest site.

In general, there are many different forms of recruitment strategies used by social insects; these may take the form of local or global strategies; one-to-one or one-to-many communication; and deploy stochastic or deterministic mechanisms. The nature of information exchange also varies in different environments and with different types of social insects. Sometimes the information exchange is quite complex where, for example it might carry data about the direction, suitability of the target and the distance; or sometimes the information sharing is simply a stimulation forcing a certain triggered action. What all these recruitment and information exchange strategies have in common is distributing useful information throughout their community [24].

In this paper, the swarm intelligence algorithm and the evolutionary algorithm are first introduced, followed by the hybridisation strategy. Afterwards,

[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.

the results are reported and the performance of the hybrid algorithm is discussed.

This work is an extension of ideas first presented at the Nature Inspired Cooperative Strategies for Optimization Conference (NICSO 2011) [7]. In the work discussed herein, after detailing the results of several experiments using CEC'05 benchmarks, the impact of SDS-led commununication and information exchange among the agents is discussed in greater depth (e.g. running a control algorithm to measure the effect of hybridisation, demonstrating the quality of the active and inactive populations, etc.) and future research topics are presented afterwards.

2 Stochastic Diffusion Search

This section introduces SDS [10], a multi-agent global search and optimisation algorithm, which is based on simple interaction of agents (inspired by one species of ants, *Leptothorax acervorum*, where a 'tandem calling' mechanism (one-to-one communication) is used, where the forager ant which finds the food location, recruits a single ant upon its return to the nest, and therefore the location of the food is physically publicised [26]). 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 [23].

Unlike many nature inspired search algorithms, SDS has a strong mathematical framework, which describes the behaviour of the algorithm by investigating its resource allocation [28], convergence to global optimum [29], robustness and minimal convergence criteria [27] and linear time complexity [30]. In order to introduce SDS, a social metaphor *the Mining Game* [3] 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:

> 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

```
01: Initialisation phase
02: Allocate each miner (agent) to a random
03:   hill (hypothesis) to pick a region randomly
04:
05: Until (all miners congregate over the highest
06:    concentration of gold)
07:
08:    Test phase
09:       Each miner evaluates the amount of gold
10:         they have mined (hypotheses evaluation)
11:       Miners are classified into happy (active)
12:         and unhappy (inactive) groups
13:
14:    Diffusion phase
15:       Unhappy miners consider a new hill by
16:         either communicating with another miner
17:         or,if the selected miner is also
18:         unhappy, there will be no information
19:         flow between the miners; instead the
20:         selecting miner must consider another
21:         hill (new hypothesis) at random
22: 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

```
01: Initialising agents()
02: While (stopping condition is not met)
03:    Testing hypotheses()
04:    Diffusion hypotheses()
05: End While
```

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).

2.4 Partial Function Evaluation

One of the concerns associated with many optimisation algorithms (e.g. Genetic Algorithm [15], Particle Swarm Optimisation [18] and etc.) is the repeti-

Algorithm 3 Passive Recruitment Mode

```
01: For ag = 1 to No_of_agents
02:    If ( !ag.activity() )
03:       r_ag = pick a random agent()
04:       If ( r_ag.activity() )
05:          ag.setHypothesis( r_ag.getHypothesis() )
06:       Else
07:          ag.setHypothesis( randomHypothesis() )
08:       End If/Else
09:    End If
10: End For
```

tive 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 [14,41], which is often not the case in real-world applications. Examples of costly evaluation functions are seismic data interpretation [41], selection of sites for the transmission infrastructure of wireless communication networks and radio wave propagation calculations of one site [40] etc.

Costly objective function evaluations have been investigated under different conditions [17] 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 [12,9].
- 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 [17].

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 Differential Evolution

DE, one of the most successful Evolutionary Algorithms (EAs), is a simple global numerical optimiser over continuous search spaces which was first introduced by Storn and Price [34,35].

DE is a population based stochastic algorithm, proposed to search for an optimum value in the feasible solution space. The parameter vectors of the population are defined as follows:

$$x_i^g = \left[x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g\right], i = 1, 2, ..., NP \tag{1}$$

where g is the current generation, D is the dimension of the problem space and NP is the population size. In the first generation, (when $g = 0$), the i^{th} vector's j^{th} component could be initialised as:

$$x_{i,j}^0 = x_{min,j} + r\left(x_{max,j} - x_{min,j}\right) \tag{2}$$

where r is a random number drawn from a uniform distribution on the unit interval $U(0, 1)$, and x_{min}, x_{max} are the lower and upper bounds of the j^{th} dimension, respectively. The evolutionary process (mutation, crossover and selection) starts after the initialisation of the population.

3.1 Mutation

At each generation g, the mutation operation is applied to each member of the population x_i^g (target vector) resulting in the corresponding vector v_i^g (mutant vector). Among the most frequently used mutation approaches are the following:

- DE/rand/1

$$v_i^g = x_{r_1}^g + F\left(x_{r_2}^g - x_{r_3}^g\right) \tag{3}$$

- DE/target-to-best/1

$$v_i^g = x_i^g + F\left(x_{best}^g - x_i^g\right) + F\left(x_{r_1}^g - x_{r_2}^g\right) \tag{4}$$

- DE/best/1

$$v_i^g = x_{best}^g + F\left(x_{r_1}^g - x_{r_2}^g\right) \tag{5}$$

- DE/best/2

$$v_i^g = x_{best}^g + F\left(x_{r_1}^g - x_{r_2}^g\right) + F\left(x_{r_2}^g - x_{r_3}^g\right) \tag{6}$$

- DE/rand/2

$$v_i^g = x_{r_1}^g + F\left(x_{r_2}^g - x_{r_3}^g\right) + F\left(x_{r_4}^g - x_{r_5}^g\right) \tag{7}$$

where r_1, r_2, r_3, r_4 are different from i and are distinct random integers drawn from the range $[1, NP]$; In generation g, the vector with the best fitness value is x_{best}^g and F (which is set to 0.5) is a positive control parameter for constricting the difference vectors.

3.2 Crossover

Crossover operation, improves population diversity through exchanging some components of v_i^g (mutant vector) with x_i^g (target vector) to generate u_i^g (trial vector). This process is led as follows:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } \ r \leq CR \ \text{ or } j = r_d \\ \\ x_{i,j}^g, & \text{otherwise} \end{cases} \tag{8}$$

where r is a uniformly distributed random number drawn from the unit interval $U(0,1)$, r_d is randomly generated integer from the range $[1, D]$; this value guarantees that at least one component of the trial vector is different from the target vector. The value of CR (set to 0.5), which is another control parameter, specifies the level of inheritance from v_i^g (mutant vector).

3.3 Selection

The selection operation decides whether x_i^g (target vector) or u_i^g (trial vector) would be able to pass to the next generation $(g+1)$. In case of a minimisation problem, the vector with a smaller fitness value is admitted to the next generation:

$$x_i^{g+1} = \begin{cases} u_i^g, & \text{if } \ f(u_i^g) \leq f(x_i^g) \\ \\ x_i^g, & \text{otherwise} \end{cases} \tag{9}$$

where $f(x)$ is the fitness function.

DE, like other evolutionary algorithms, suffers from premature convergence where the population lose their diversity too early and get trapped in local optima, therefore performing poorly on problems with high dimension and many local optima.

DE is known to be relatively good in comparison with other EAs and PSOs at avoiding premature convergence. However, in order to reduce the risk of premature convergence in DE and to preserve population diversity, several methods have been proposed, among which are: multi-population approaches [20,37,21,13]; providing extra knowledge about the problem space [32,39]; information storage about previously explored areas [43]; utilising adapting and control parameters to ensure population diversity [42]; using CrowdingDE for tracking and maintaining multiple optima [38,33].

This paper proposes information exchange and agent dispensation (SDS-led random restart) as methods to avoid premature convergence and preserve population diversity.

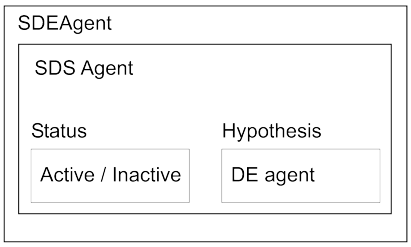
4 Merging SDS and DE Algorithms

The initial motivating thesis justifying the hybridisation of SDS and DE is the partial function evaluation deployed in SDS, which may mitigate the high computational overheads entailed when deploying a DE algorithm 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 aimed to investigate if the information diffusion mechanism deployed in SDS may on its own improve DE behaviour. These are the 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 and dispensing resources (e.g. members of the DE population) after partially evaluating the search space.

Each DE agent has three vectors (target, mutant and trial vectors); and each SDS agent has one hypothesis and one status. In the experiment reported here (hybrid algorithm), every member of DE population is an SDS agent too – together termed *SDEAgents*. In SDEAgents, SDS hypotheses are defined by the DE target vector, and an additional boolean variable (status) determining whether the SDEAgent is active or inactive (see Figure 1). The behaviour of the hybrid algorithm in its simplest form is presented in Algorithm 4.

Fig. 1 Encapsulating SDS agent and DE agent as SDE-Agent



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 DE-SDS algorithm, it is clear that there are many different tests that could be performed in order to determine the activity of each SDEAgent. A very simple test is illustrated in Algorithm 4. Here, the test-phase is simply conducted by comparing the fitness of each SDEAgent's target vector against that of a random SDEAgent; if the selecting SDEAgent

Algorithm 4 Hybrid Algorithm

```
01: Initialise SDEAgents
02:
03: For ( FE_Counter = 1 to FE_Allowed )
04:
05:    For ( SDEAgent = 1 to NP )
06:       Mutation : generate mutant vector
07:       Crossover: generate trial  vector
08:       Selection: generate target vector for next generation
09:    End For
10:
11:    If ( generation counter % n == 0 )
12:    // START SDS
13:    // TEST PHASE
14:       For ag = 1 to NP
15:          r_ag = pick-random-SDEAgent()
16:          If (ag.targetVecFitness() < r_ag.targetVecFitness())
17:             ag.setActivity (true)
18:          Else
19:             ag.setActivity (false)
20:          End If
21:       End For
22:
23:       // DIFFUSION PHASE
24:       For ag = 1 to No_of_SDEAgents
25:          If ( best ag ) continue
26:          Else If ( !ag.activity() )
27:             r_ag = pick-random-SDEAgent()
28:             If ( r_ag.activity() )
29:                ag.setHypo( r_ag.getHypo() )*
30:             Else
31:                ag.setHypo( randomHypo() ) **
32:             End If
33:          End If
34:       End For
35:    End If
36:    // END SDS
37:
38:    Find SDEAgent with best fitness value
39:
40: End For

*  In setHypo() and getHypo(), 'Hypo' refers to
   the SDEAgent's hypothesis (see Fig. 1). This
   indicates that a clone of r_ag is generated.
** 'randomHypo()' uses the entire search space
   to  reinitialise the agent.
```

has a better fitness value, it will become active, otherwise it will be flagged inactive. On average, this mechanism will ensure 50% of SDEAgents remain active from one iteration to another.

In the Diffusion Phase, each inactive SDEAgent picks another SDEAgent randomly, if the selected SDEAgent is active, the selected SDEAgent com-

municates its hypothesis to the inactive one; if the selected SDEAgent is also inactive, the selecting SDEAgent generates a new hypothesis at random from the search space.

As outlined in the pseudo-code of the hybrid algorithm (see Algorithm 4), after each n generations, one full SDS cycle[2] is executed. The hybrid algorithm is called $SDSnDE$, where n refers to the number of generations 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 one variation of DE algorithm (DE/best/1) is contrasted against the hybrid algorithm, $SDSnDE$.

5.1 Experiment Setup

The algorithms are tested over a number of benchmarking functions designed for the Special Session on Real Parameter Optimization organised in the 2005 IEEE Congress on Evolutionary Computation (CEC 2005), reported in [36], where a complete description of these benchmarks are provided:

- Unimodal Functions (5):
  - F_1: Shifted Sphere Function
  - F_2: Shifted Schwefel's Problem 1.2
  - F_3: Shifted Rotated High Conditioned Elliptic Function
  - F_4: Shifted Schwefel's Problem 1.2 with Noise in Fitness
  - F_5: Schwefel's Problem 2.6 with Global Optimum on Bounds
- Multimodal Functions[3] (9):
  - Basic Functions (7):
    - F_6: Shifted Rosenbrock's Function
    - F_7: Shifted Rotated Griewank's Function without Bounds
    - F_8: Shifted Rotated Ackley's Function with Global Optimum on Bounds
    - F_9: Shifted Rastrigin's Function
    - F_{10}: Shifted Rotated Rastrigin's Function
    - F_{11}: Shifted Rotated Weierstrass Function
    - F_{12}: Schwefel's Problem 2.13
  - Expanded Functions (2):
    - F_{13}: Expanded Extended Griewank's plus Rosenbrock's Function (F_8F_2)
    - F_{14}: Shifted Rotated Expanded Scaffer's F_6

All benchmarks have been shifted in order to ensure there are no optima in the centre of the search space.

[2] Test Phase: decides about the status of each SDEAgent, one after another; Diffusion Phase: shares information according to the algorithm presented

[3] Hybrid Composition Functions are not used in this work.

The experiments are conducted with the population of 100 agents; the halting criterion for this experiment is exceeding 300, 000 function evaluations (FEs). There are 30 independent runs for each benchmark function and the results are averaged over these independent trials.

Accuracy, which is used as performance measure, is defined by the quality of the best agent 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 accuracy:

$$\text{Accuracy}\,(S,t) = |f\,(x^{g}_{best}) - f\,(x_{opt})| \tag{10}$$

where x^{g}_{best} is the best agent at generation g and x_{opt} is the position of the known optimum solution.

In this paper, $SDSnDE$, is presented with few variations of parameter, n (the number of generations before an SDS cycle is performed): $n = 5$, 50, and 200. 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 DE algorithm. For each benchmark and algorithm, the table shows the accuracy measure.

Table 2 shows if there is any significant difference between any pair of algorithms. The results reported next are based on the data presented in this table. In order to highlight the presense of any significant difference between the algorithms, TukeyHSD Test [25] with 95% family-wise confidence level is utilised.

It is shown that out of 24 cases where there exist a significant difference between DE and the hybrid algorithms, the hybrid algorithms outperform DE significantly in 21 cases (%88 significant outperformance). Although H5 outperforms DE in the majority of cases, DE shows outperformance in three cases. However in terms of the other hybrid algorithms (H50 and H200), DE is outperformed significantly in all cases.

The focus of this paper is not finding the best n for $SDSnDE$ (for this set of benchmarks), but rather investigate the effect of SDS algorithm on the performance of DE algorithm. However, among the hybrid algorithms, H50 outperforms H5 and H200 in all cases.

As demonstrated in Table 2, in f_2, f_{3-4}, the performance of H5, which has the highest rate of information exchange, is weaker than the other hybrid algorithms with lower information sharing. This implies that the performance of some problems might be negatively affected by excessive information exchange (e.g. in f_5, $F_{H5} > F_{H50} > F_{H200}$, where F is the fitness value).

Additionally, excessive reduction in information exchange reduces the positive effect that SDS has on the overal behaviour of the agents in DE. For

instance, in $f_{9,12,13}$, the more the information exchange the better the performance ($F_{H5} < F_{H50} < F_{H200}$).

These results confirms the importance of SDS-led information sharing between DE agents. Using this hybridisation technique, the only problem dependent parameter to tune is the frequency at which SDS cycle is run (e.g. In this set of experiments, the hybrid algorithm with the middle ground frequency, H50, shows more promising results than frequencies 5 and 200, used in H5 and H200 respectively).

This demonstrates the importance of deploying the right frequency of communication to maximise the positive effect of the hybridisation.

6 Discussion

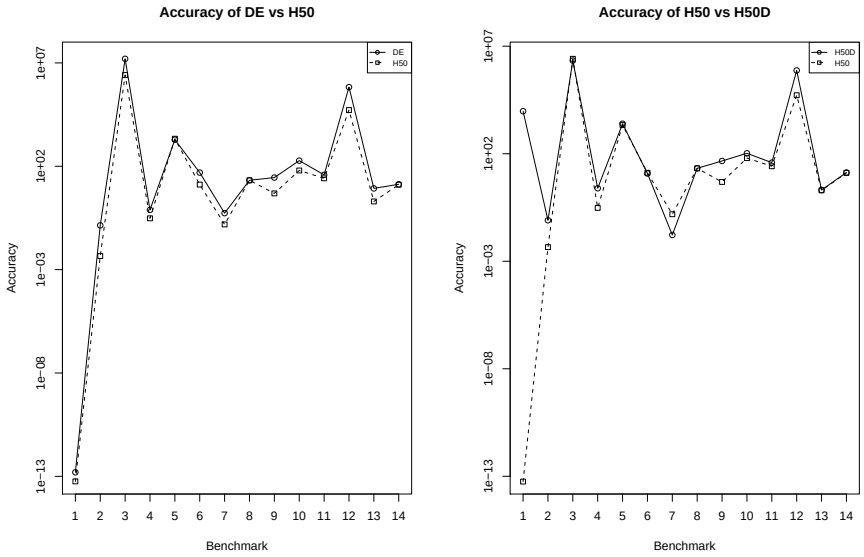
The resource allocation process underlying SDS offers three closely coupled mechanisms to the algorithm's search component to speed its convergence to global optima. The first component is 'efficient, non-greedy information sharing' instantiated via positive feedback of potentially good hypotheses between agents; the second component is the dispensation mechanism – SDS-led random-restarts – deployed as part of the diffusion phase; the third 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 and dispensation components of SDS in the hybrid algorithm are executed in the 'Diffusion Phase', where information is shared (diffused) among SDEAgents (see Algorithm 3). Analysis of the performance of the hybrid algorithm (see results above) demonstrates that adding the SDS resource allocation and dispensation mechanisms to the classical DE architecture improves the overall performance of the algorithm (i.e. it enhances algorithm accuracy, as defined herein).

The graphs in Fig. 3 illustrate the difference between the accuracy of active and inactive populations of three different benchmarks (f_1: unimodal, f_9: basic multimodal, and f_{13}: expanded multimodal) in H50 before and after the diffusion phase. Since the active populations are not affected during the diffusion phase, they are represented by one line on each graph.

As shown in the graphs, compared to the difference between the active and inactive populations prior to the diffusion phase, the accuracy of the inactive populations after the diffusion phase is much worse than the active populations. In f_1 and f_9, there are occasions when all members of the population coverge to one solution (optimum so far); when this occurs, based on line 16 of Algorithm 4 (where having equal fitness results in the selecting agent becoming inactive), there would be no active agents in that cycle; this explains the absence of the active line in the graphs. Theoretical analysis is undergoing to

Fig. 2 Accuracy of DE vs H50 and H50 vs H50D



understand the core interaction of active populations with the inactive ones in this context.

To further analyse the role of SDS in the hybrid algorithms, the Diffusion Phase of SDS algorithm is modified (see Algorithm 5) to investigate the dispensation effect caused by randomising a selection of agent hypotheses after a number of DE function evaluations (effectively instantiating a DE with SDS-led random-restarts). In other words, after the SDS test-phase, the hypothesis of each inactive SDEAgent is randomised.

As detailed in Table 1, although, information sharing plays an important role in the performance of hybrid DE algorithm, the significance of dispensation mechanism (in randomly restarting some of the agents) in improving the performance of DE algorithm cannot be discarded.

Whenever there is a significant difference between the performance of the hybrid algorithm and its corresponding algorithm with SDS-led restart-only mechanism, which is facilitated by the test phase of the SDS algorithm, H50 exhibits significant outperformance over H50D (see Table 2).

As demonstrated, the lack of information exchange mechanism in the hybrid algorithm resulted in worse performance. This shows the important impact of the information sharing strategy on the overall performance of the hybrid algorithm. The outperformance of H50 over DE and H50D is demonstrated in Fig. 2.

The third 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

Algorithm 5 SDS-led Restart-only Hybrid Algorithm

```
01: // DIFFUSION PHASE
02: For ag = 1 to No_of_agents
03:   If ( !ag.activity() )
04:     ag.setHypo( randomHypo() )
05:   End If
06: End For
```

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 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 DE population, in addition to this information, each agent has implicit partial knowledge from other agents (derived from the mutation, crossover and selection mechanisms) comprising the historical evidence implicit in the prior [m] objective-function evaluations the population has performed. Thus, since each agent finds its current position by using this implicit knowledge, it has partial knowledge of the full search space.

In the hybrid algorithm each SDEAgent 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 SDEAgent's target vector defines a 'partial view' of the entire search space (via the partial interaction it has with the rest of the population through mutation, crossover and selection). Hence, when the fitness values of two SDEAgents' target vector 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 SDSnDE 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 SDSnDE 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 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 SDSnDE algorithm; if successful, this offers further, potentially significant, performance improvements for the new hybrid algorithm.

7 Conclusion

This paper presents an overview of the integration of DE with SDS. Here, SDS is primarily used as an efficient resource allocation and dispensation mechanism responsible for facilitating communication between DE agents. 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. The performance of the hybrid algorithm is also tested against a control algorithm, demonstrating the importance of information sharing in the SDS algorithm. Additionally, the behaviour of the active and inactive populations is illustrated, showing the accuracy of each population before and after the SDS algorithm. Results reported in this paper have demonstrated that the hybrid SDSnDE algorithm outperforms the performance of (one variation of) classical DE architecture, 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 (including 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 classical DE) is anticipated to be much greater. Additionally, since this research establishes that the performance of the hybrid algorithms is influenced by value of n (in SDSnDE), the possibility of activating SDS based on how DE population evolves, remains an ongoing research topic.

Fig. 3 Accuracy of the active and inactive populations before and after diffusion in H50.

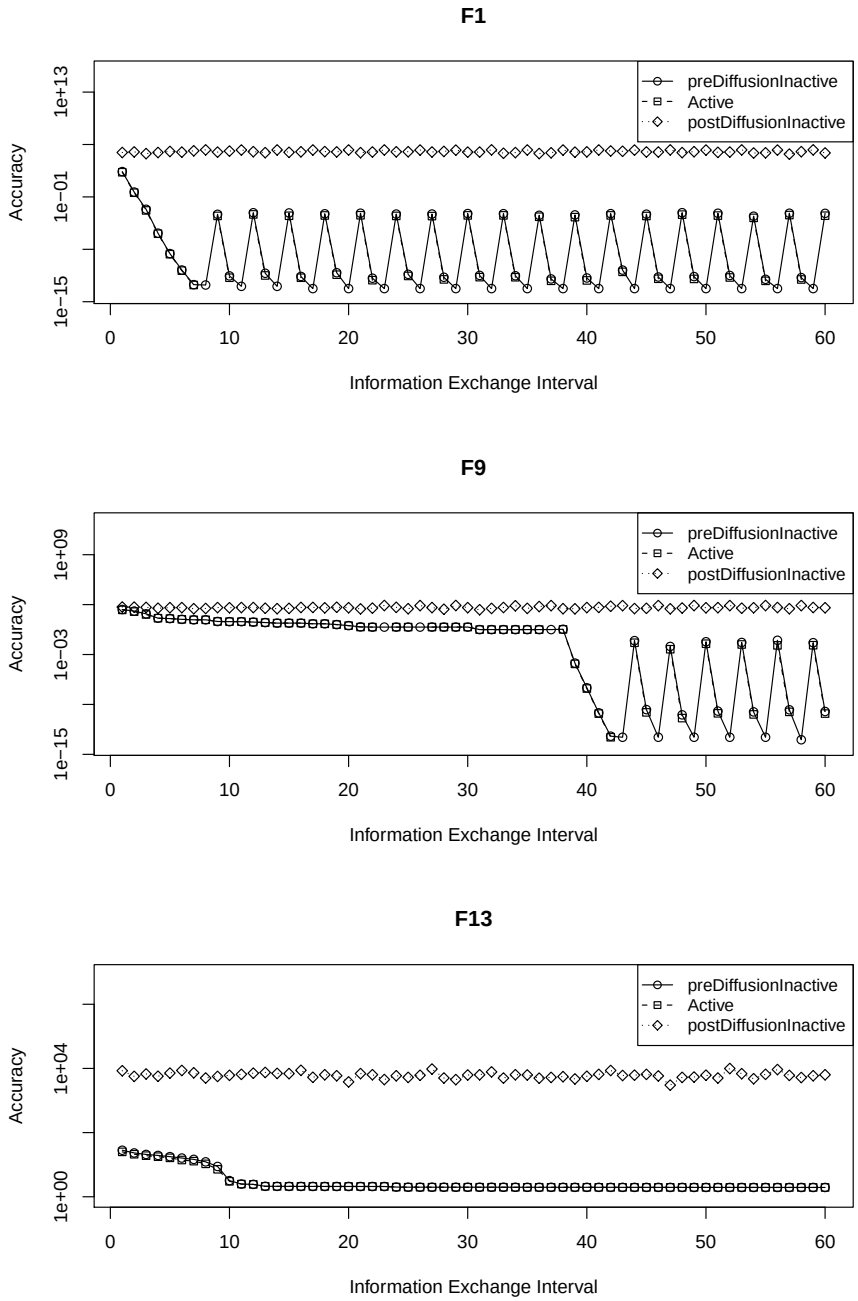


Table 1 Accuracy Details

Accuracy (±standard error) is shown with two decimal places after 30 trials of 300,000 function evaluations (FEs). For each benchmark, algorithms which are **significantly** better (see Table 2) than the others are highlighted. Note that the highlighted algorithms do not significantly outperform each another.

	DE	**H5: SDSnDE** ***n = 5***	**H50: SDSnDE** ***n = 50***	**H200: SDSnDE** ***n = 200***	**H50D: Dispenser** ***n = 50***
f_1	1.57E-13±2.09E-14	1.91E-13±1.76E-14	5.68E-14±0.00E+00	6.06E-14±2.63E-15	9.50E+03±5.36E+03
f_2	1.39E-01±4.12E-02	5.41E-01±1.08E-01	4.55E-03±1.04E-03	2.76E-02±4.17E-03	8.09E-02±2.13E-02
f_3	1.58E+07±1.78E+06	3.23E+06±2.97E+05	2.59E+06±2.06E+05	6.16E+06±6.30E+05	2.15E+06±2.08E+05
f_4	7.78E-01±1.33E-01	2.57E+01±3.93E+00	3.04E-01±5.30E-02	7.91E-01±1.69E-01	2.48E+00±4.04E-01
f_5	1.94E+03±1.62E+02	3.90E+03±2.22E+02	2.15E+03±1.37E+02	2.13E+03±1.72E+02	2.48E+03±1.40E+02
f_6	4.96E+01±2.01E+01	3.36E+01±7.21E+00	1.30E+01±3.72E+00	1.92E+01±4.36E+00	1.21E+01±3.00E+00
f_7	5.40E-01±8.11E-02	2.14E-02±3.36E-03	1.54E-01±3.49E-02	4.30E-01±6.77E-02	1.67E-02±2.79E-03
f_8	2.10E+01±9.01E-03	2.10E+01±1.28E-02	2.10E+01±6.45E-03	2.10E+01±9.41E-03	2.10E+01±9.36E-03
f_9	2.84E+01±1.34E+00	1.61E-01±7.78E-02	4.88E+00±4.68E-01	2.23E+01±1.43E+00	4.61E+01±9.35E+00
f_{10}	1.88E+02±3.79E+00	7.76E+01±4.40E+00	6.25E+01±3.59E+00	5.49E+01±2.97E+00	1.05E+02±1.23E+01
f_{11}	3.85E+01±1.16E+00	2.95E+01±8.35E-01	2.62E+01±1.28E+00	3.34E+01±1.30E+00	3.85E+01±1.04E+00
f_{12}	6.74E+05±1.30E+04	4.06E+04±3.06E+03	5.25E+04±4.20E+03	1.06E+05±9.19E+03	7.46E+05±9.66E+03
f_{13}	8.52E+00±5.33E-01	1.46E+00±4.78E-02	1.98E+00±8.00E-02	2.64E+00±1.02E-01	2.06E+00±7.09E-02
f_{14}	1.35E+01±3.45E-02	1.32E+01±5.33E-02	1.28E+01±7.22E-02	1.34E+01±5.41E-02	1.34E+01±4.78E-02

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. X–o shows that the left algorithm is significantly better than the right one; and o–X shows that the right algorithm is significantly better than the one, on the left.

	DE-H5	**DE-H50**	**DE-H200**	**DE-H50D**	**H5-H50**	**H5-H200**	**H5-H50D**	**H50-H200**	**H50-H50D**	**H200-H50D**
f_1	–	–	–	x – o	–	–	x – o	–	x – o	x – o
f_2	x – o	–	–	–	o – x	o – x	o – x	–	–	–
f_3	o – x	o – x	o – x	o – x	–	–	–	x – o	–	o – x
f_4	x – o	–	–	–	o – x	o – x	o – x	–	–	–
f_5	x – o	–	–	–	o – x	o – x	o – x	–	–	–
f_6	–	–	–	–	–	–	–	–	–	–
f_7	o – x	o – x	–	o – x	–	x – o	–	x – o	–	o – x
f_8	–	–	–	–	–	–	–	–	–	–
f_9	o – x	o – x	–	x – o	–	x – o	x – o	x – o	x – o	x – o
f_{10}	o – x	o – x	o – x	o – x	–	–	x – o	–	x – o	x – o
f_{11}	o – x	o – x	o – x	–	–	–	x – o	x – o	x – o	x – o
f_{12}	o – x	o – x	o – x	x – o	–	x – o	x – o	x – o	x – o	x – o
f_{13}	o – x	o – x	o – x	o – x	–	x – o	–	–	–	–
f_{14}	o – x	o – x	–	–	o – x	–	x – o	x – o	x – o	–

References

1. al-Rifaie, M.M.: When birds and ants set off to draw. http://www.ursyn.com/20111/Mohammad.html (2011). IV2011 (London) & cgiv2011 (Singapore) - DIGITAL ART GALLERY
2. al-Rifaie, M.M., Aber, A., Bishop, M.: Cooperation of nature and physiologically inspired mechanisms in visualisation. In: A. Ursyn (ed.) Biologically-Inspired Computing for the Arts: Scientific Data through Graphics. IGI Global, United States (2012). ISBN13: 9781466609426, ISBN10: 1466609427
3. al-Rifaie, M.M., Bishop, M.: The mining game: a brief introduction to the stochastic diffusion search metaheuristic. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour Quarterly (AISBQ) **130** (2010)
4. al-Rifaie, M.M., Bishop, M.: Weak vs. strong computational creativity. In: AISB 2012: Computing and Philosophy. University of Birmingham, Birmingham, U.K. (2012)
5. al-Rifaie, M.M., Bishop, M., Aber, A.: Creative or not? birds and ants draw with muscles. In: AISB 2011: Computing and Philosophy, pp. 23–30. University of York, York, U.K. (2011). ISBN: 978-1-908187-03-1
6. al-Rifaie, M.M., Bishop, M., Blackwell, T.: An investigation into the merger of stochastic diffusion search and particle swarm optimisation. In: GECCO '11: Proceedings of the 2011 GECCO conference companion on Genetic and evolutionary computation, pp. 37–44. ACM, Dublin, Ireland (2011)
7. al-Rifaie, M.M., Bishop, M., Blackwell, T.: Resource allocation and dispensation impact of stochastic diffusion search on differential evolution algorithm; in. In: Nature Inspired Cooperative Strategies for Optimisation (NICSO 2011), Springer (2011)
8. al-Rifaie, M.M., Bishop, M., Caines, S.: Creativity and autonomy in swarm intelligence systems. In: M. Bishop, Y. Erden (eds.) Cognitive Computation: Computational Creativity, Intelligence and Autonomy. Springer (2012). DOI: 10.1007_s12559-012-9130-y
9. el Beltagy, M.A., Keane, A.J.: Evolutionary optimization for computationally expensive problems using gaussian processes. In: Proc. Int. Conf. on Artificial Intelligence'01, pp. 708–714. CSREA Press (2001)
10. Bishop, J.: Stochastic searching networks. pp. 329–331. Proc. 1st IEE Conf. on Artificial Neural Networks, London, UK (1989)
11. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature **406**, 3942 (2000)
12. Branke, J., Schmidt, C., Schmeck, H.: Efficient fitness estimation in noisy environments. In Spector, L., ed.: Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2001)
13. Brest, J., Zamuda, A., Boskovic, B., Maucec, M., Zumer, V.: Dynamic optimization using self-adaptive differential evolution. In: IEEE Congress on Evolutionary Computation, 2009. CEC'09., pp. 415–422. IEEE (2009)
14. Digalakis, J., Margaritis, K.: An experimental study of benchmarking functions for evolutionary algorithms. International Journal **79**, 403–416 (2002)
15. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA (1989)
16. Holldobler, B., Wilson, E.O.: The Ants. Springer-Verlag (1990)
17. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. In: Soft Computing **9**, 3–12 (2005)
18. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948. IEEE Service Center, Piscataway, NJ (1995)
19. Kennedy, J.F., Eberhart, R.C., Shi, Y.: Swarm intelligence. Morgan Kaufmann Publishers, San Francisco ; London (2001)
20. Kozlov, K., Samsonov, A.: New migration scheme for parallel differential evolution. In: Proceedings of the international conference on bioinformatics of genome regulation and structure, pp. 141–144 (2006)
21. Mendes, R., Mohais, A.: DynDE: a differential evolution for dynamic optimization problems. In: The 2005 IEEE Congress on Evolutionary Computation CEC2005., vol. 3, pp. 2808–2815. IEEE (2005)

22. de Meyer, K.: Explorations in stochastic diffusion search: Soft- and hardware implementations of biologically inspired spiking neuron stochastic diffusion networks. Tech. Rep. KDM/JMB/2000/1, University of Reading (2000)
23. de Meyer, K., Bishop, J.M., Nasuto, S.J.: 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)
24. de Meyer, K., Nasuto, S., Bishop, J.: Stochastic diffusion optimisation: the application of partial function evaluation and stochastic recruitment in swarm intelligence optimisation. Springer Verlag **2, Chapter 12 in Abraham, A. and Grosam, C. and Ramos, V. (eds), "Swarm intelligence and data mining"** (2006)
25. Miller, R.: Simultaneous statistical inference. SPRINGER-VERLAG INC., 175 FIFTH AVE., NEW YORK, NY, 1981, 300 (1981)
26. Moglich, M., Maschwitz, U., Holldobler, B.: Tandem calling: A new kind of signal in ant communication. Science **186**(4168), 1046–1047 (1974)
27. Myatt, D.R., Bishop, J.M., Nasuto, S.J.: Minimum stable convergence criteria for stochastic diffusion search. Electronics Letters **40**(2), 112–113 (2004)
28. Nasuto, S.J.: Resource allocation analysis of the stochastic diffusion search. Ph.D. thesis, University of Reading, Reading, UK (1999)
29. Nasuto, S.J., Bishop, J.M.: Convergence analysis of stochastic diffusion search. Parallel Algorithms and Applications **14(2)** (1999)
30. Nasuto, S.J., Bishop, J.M., Lauria, S.: Time complexity of stochastic diffusion search. Neural Computation **NC98** (1998)
31. Nasuto, S.J., Bishop, M.J.: Steady state resource allocation analysis of the stochastic diffusion search. Arxiv preprint cs/0202007 (2002)
32. Smuc, T.: Improving convergence properties of the differential evolution algorithm. In: Proceedings of the MENDEL 2002 - 8th International Conference on Soft Computing, pp. 80–86 (2002)
33. Stoean, C., Preuss, M., Stoean, R., Dumitrescu, D.: Multimodal optimization by means of a topological species conservation algorithm. Transactions on Evolutionary Computation, IEEE **14**(6), 842–864 (2010)
34. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces (1995). TR-95-012, [online]. Available: http://www.icsi.berkeley.edu/ storn/litera.html
35. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
36. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Tech. rep., Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, IIT Kanpur (2005)
37. Tasgetiren, M., Suganthan, P.: A multi-populated differential evolution algorithm for solving constrained optimization problem. In: IEEE Congress on Evolutionary Computation CEC2006., pp. 33–40. IEEE (2006)
38. Thomsen, R.: Multimodal optimization using crowding-based differential evolution. In: Congress on Evolutionary Computation, 2004. CEC2004., vol. 2, pp. 1382–1389. IEEE (2004)
39. Weber, M., Neri, F., Tirronen, V.: Parallel Random Injection Differential Evolution. Applications of Evolutionary Computation pp. 471–480 (2010)
40. Whitaker, R., Hurley, S.: An agent based approach to site selection for wireless networks. In: 1st IEE Conf. on Artificial Neural Networks. ACM Press Proc ACM Symposium on Applied Computing, Madrid Spain (2002)
41. Whitley, D., Rana, S., Dzubera, J., Mathias, K.E.: Evaluating evolutionary algorithms. Artificial Intelligence **85**(1-2), 245–276 (1996)
42. Zaharie, D.: Control of population diversity and adaptation in differential evolution algorithms. In: Proceedings of the MENDEL 2003 - 9th International Conference on Soft Computing, pp. 41–46 (2003)
43. Zhang, J., Sanderson, A.: JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on **13**(5), 945–958 (2009)