AN INVESTIGATION INTO THE USE OF SWARM INTELLIGENCE FOR AN EVOLUTIONARY ALGORITHM OPTIMISATION

The Optimisation Performance of Differential Evolution Algorithm Coupled with Stochastic Diffusion Search

Mohammad Majid al-Rifaie, John Mark Bishop and Tim Blackwell
Department of Computing, Goldsmiths College, University of London, London SE14 6NW, U.K.

Keywords: SDS, DE, EC, SI, Metaheuristics.

Abstract: The integration of Swarm Intelligence (SI) algorithms and Evolutionary algorithms (EAs) might be one of the future approaches in the Evolutionary Computation (EC). This work narrates the early research on using Stochastic Diffusion Search (SDS) – a swarm intelligence algorithm – to empower the Differential Evolution (DE) – an evolutionary algorithm – over a set of optimisation problems. The results reported herein suggest that the powerful resource allocation mechanism deployed in SDS has the potential to improve the optimisation capability of the classical evolutionary algorithm used in this experiment. Different performance measures and statistical analyses were utilised to monitor the behaviour of the final coupled algorithm.

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 (Whitley et al., 1996); 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 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 (Nasuto, 1999) and these results provided motivation to investigate the application of the information diffusion mechanism originally deployed in SDS[1] with DE.

In this paper, the swarm intelligence algorithm and the evolutionary algorithm are first introduced, followed by the coupling strategy. The results are reported afterwards and the performance of the coupled algorithm will be discussed.

[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 STOCHASTIC DIFFUSION SEARCH

This section introduces SDS (Bishop, 1989), 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). 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. Unlike many nature inspired search algorithms, SDS has a strong mathematical framework, which describes the behaviour of the algorithm by investigating its resource allocation, convergence to global optimum, robustness and minimal convergence criteria and linear time complexity. In order to introduce SDS, a social metaphor *the Mining Game* (al-Rifaie and Bishop, 2010) is used.

al-Rifaie M., Bishop J. and Blackwell T..
AN INVESTIGATION INTO THE USE OF SWARM INTELLIGENCE FOR AN EVOLUTIONARY ALGORITHM OPTIMISATION - The Optimisation Performance of Differential Evolution Algorithm Coupled with Stochastic Diffusion Search.
DOI: 10.5220/0003723005530558
In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications* (FEC-2011), pages 553-558
ISBN: 978-989-8425-83-6

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'.

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 (for high-level SDS description see Algorithm 1):

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

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.

Algorithm 1: SDS Algorithm.

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

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 Partial Function Evaluation

The commonly used benchmarks for evaluating the performance of swarm intelligence algorithms are typically small in terms of their objective functions computational costs, which is often not the case in real-world applications. Examples of costly evaluation functions are seismic data interpretation (Whitley et al., 1996), selection of sites for the transmission infrastructure of wireless communication networks and radio wave propagation calculations of one site (Whitaker and Hurley, 2002) etc.

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 numberical optimiser over continuous search spaces which was first introduced by Storn and Price (Storn and Price, 1995).

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:

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

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:

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

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, corssover 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^g_i (target vector) resulting in the corresponding vector v^g_i (mutant vector). Among the most frequently used mutation approaches are the following:

- DE/rand/1

$$v^g_i = x^g_{r_1} + F\left(x^g_{r_2} - x^g_{r_3}\right) \quad (3)$$

- DE/target-to-best/1

$$v^g_i = x^g_i + F\left(x^g_{best} - x^g_i\right) + F\left(x^g_{r_1} - x^g_{r_2}\right) \quad (4)$$

- DE/best/1

$$v^g_i = x^g_{best} + F\left(x^g_{r_1} - x^g_{r_2}\right) \quad (5)$$

- DE/best/2

$$v^g_i = x^g_{best} + F\left(x^g_{r_1} - x^g_{r_2}\right) + F\left(x^g_{r_2} - x^g_{r_3}\right) \quad (6)$$

- DE/rand/2

$$v^g_i = x^g_{r_1} + F\left(x^g_{r_2} - x^g_{r_3}\right) + F\left(x^g_{r_4} - x^g_{r_5}\right) \quad (7)$$

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^g_{best} and F is a positive control parameter for constricting the difference vectors.

3.2 Crossover

Crossover operation, improves population diversity through exchanging some components of v^g_i (mutant vector) with x^g_i (target vector) to generate u^g_i (trial vector). This process is led as follows:

$$u^g_{i,j} = \begin{cases} v^g_{i,j}, & \text{if } r \le CR \text{ or } j = r_d \\ x^g_{i,j}, & \text{otherwise} \end{cases} \quad (8)$$

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, which is another control parameter, specifes the level of inheritance from v^g_i (mutant vector).

3.3 Selection

The selection operation decides whether x^g_i (target vector) or u^g_i (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:

$$x^{g+1}_i = \begin{cases} u^g_i, & \text{if } f\left(u^g_i\right) \le f\left(x^g_i\right) \\ x^g_i, & \text{otherwise} \end{cases} \quad (9)$$

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

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 (Brest et al., 2009); providing extra knowledge about the problem space (Weber et al., 2010); information storage about previously explored areas (Zhang and Sanderson, 2009) and utilising adapting and control parameters to ensure population diversity (Zaharie, 2003).

4 COUPLING SDS AND DE

The initial motivating thesis justifying the coupling 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 the scenario where the optimisation process is initialised by n number of function evaluations (FEs) performed within the SDS test-diffusion cycle, in order to allocate the resources (agent) to the promising areas of the search space and then passing on the agents' positions to DE to resume the optimisation process, in most cases as a local search.

The goal of this process is to verify whether the information diffusion and dispensation mechanisms 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 coupled algorithm. The resource allocation (or recruitment) and partial function evaluation sides of SDS (see Section 2.3) are used to assist allocating resources 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 (coupled algorithm), as stated before, SDS test-diffusion cycle is run for n of FEs and then DE commences with the optimisation, taking its target vectors from SDS agents' positions.

The behaviour of the coupled algorithm in its simplest form is presented in Algorithm 2.

Algorithm 2: Coupled Algorithm.

```
Initialise Agents

x = initialInactiveErrorVector (e.g. 4)
y = initialActiveErrorVector   (e.g. 1)
// x > y

n = SDS_FE_Allowed

//SDS cycle
While ( FE <= n )
{
   // Decreasing the error vector over time
   If ( FE < stoppingErrV_DecreasePoint )
      iErrorV = x - (x*FE) / stoppingErrV_DecPoint
      aErrorV = y - (y*FE) / stoppingErrV_DecPoint
   End If
   // stoppingErrV_DecPoint < SDS_FE_Allowed

   // TEST PHASE
   For ag = 1 to NP
      r_ag = pick-random-agent()
      If ( F(ag) < F(r_ag) )
         ag.setActivity (true)
      Else
         ag.setActivity (false)
      End If
   End For

   // DIFFUSION PHASE
   For ag = 1 to NP
      If ( ag is not active )
         r_ag = pick-random-agent()
         If ( r_ag is active )
            ag.setHypo(
               Gaussian(r_ag.getHypo(),iErrorV))
         Else
            ag.setHypo( randomHypo() )
         End If
      End If
      Else
        ag.setHypo(Gaussian(ag.getHypo(),aErrorV))
   End for
}

// DE
While ( FE < FE_Allowed )
   For ( Agent = 1 to NP )
      Mutation : generate mutant vector
      Crossover: generate trial vector
      Selection: generate target vector
   End For
   Find Agent with best fitness value
End For
```

4.1 Test and Diffusion Phases in the Coupled Algorithm

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 coupled SDS-DE algorithm, it is clear that there are many different tests that could be performed in order to determine the activity of each agent.

A very simple test is illustrated in Algorithm 2. Here, the test-phase is simply conducted by comparing the fitness of each agent's fitness against that of a random agent; if the selecting agent has a better fitness value, it will become active, otherwise it will be flagged inactive. On average, this mechanism will ensure 50% of agents remain active from one iteration to another.

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

As outlined in the pseudo-code of the coupled algorithm (see Algorithm 2), after the initial n function evaluations during which SDS test-diffusion cylce iterates, DE algorithm should run.

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

5 EXPERIMENTAL SETUP

In this work, a number of experiments are carried out and the performance of one variation of DE algorithm (DE/best/1) is contrasted against the coupled SDS-DE algorithm (*sDE*). The algorithms are tested over a number of standard benchmarking functions, preserving different dimensionality and modality (see (al-Rifaie et al., 2011a; al-Rifaie et al., 2011b) for more information on the benchmarks used). The experiments are conducted with the population of 100 agents. The halting criterion for this experiment is when the function evaluations reaches $300,000$.

There are 30 independent runs for each benchmark function and the results are averaged over these independent trials.

The stopping condition for decreasing the error vectors is reaching $80,000$ FEs. DE is run after $100,000$ FEs until the temination criterion which is $300,000$ FEs. These values were selected merely to

Table 1: Accuracy Details.

	DE	**sDE** *SDS-DE*	**sDispDE** *SDS (Disp) DE*
f_1	2.80E-78±2.65E-78	1.35E-37±1.06E-37	3.36E-54±2.01E-54
f_2	**6.31E-02±1.55E-02**	**8.15E-01±2.00E-01**	7.58E+00±1.55E+00
f_3	3.45E+01±8.04E+00	3.45E+01±4.52E+00	2.65E+01±4.08E+00
f_4	4.59E+02±1.31E+02	8.55E+02±2.44E+02	6.17E+00±1.10E+00
f_5	1.75E+02±8.18E+00	5.69E+01±1.80E+00	**2.48E+01±1.26E+00**
f_6	1.87E+01±8.84E-01	**2.29E+00±6.48E-02**	**7.52E-01±1.30E-01**
f_7	**5.79E-02±1.77E-02**	1.02E+00±4.68E-01	**1.18E-02±2.99E-03**
f_8	1.34E+01±2.94E+00	**3.80E-02±2.20E-02**	**1.69E-01±8.07E-02**
f_9	1.62E+00±3.56E-01	**9.36E-02±2.50E-02**	**3.33E-02±1.48E-02**
f_{10}	4.90E-01±7.42E-02	**1.04E-16±2.06E-17**	**1.18E-16±2.06E-17**
f_{11}	1.57E+02±4.21E+01	**0.00E+00±0.00E+00**	**5.92E-17±2.80E-17**
f_{12}	5.05E+00±7.38E-17	**1.06E-08±2.37E-09**	2.28E+00±4.90E-01
f_{13}	5.27E+00±0.00E+00	**2.64E-07±4.22E-08**	1.76E+00±4.63E-01
f_{14}	5.36E+00±9.99E-17	**2.84E-07±5.17E-08**	2.85E+00±5.37E-01

Accuracy (±standard error) is shown with two decimal places after 30 trials of 300,000 FEs. For each benchmark, the best algorithm(s) which is **significantly** better (see Table 2) than the others is highlighted. In cases where more than one algorithm is highlighted in a row, the highlighted algorithms do not significantly outperform each other.

provide a brief initial exploration of the behaviour of the new coupled algorithm; no claim is made for their optimality.

6 EXPERIMENTAL RESULTS

Table 1 shows the performance of the coupled algorithm (sDE) alongside DE algorithm. For each benchmark and algorithm, the table shows the accuracy measure. The overal reliability of each algorithm is also reported.

The focus of this paper is not finding the best stopping point for decreasing the error vectors or the values of the initial error vectors (for this set of benchmarks), but rather investigate the effect of SDS algorithm on the performance of DE algorithm.

As Table 2 shows, over all benchmarks, other than f_7, DE algorithm does not significantly outperform the coupled algorithm. On the other hand, in most cases (f_{5-6} and f_{8-14}), the coupled algorithm significantly outperforms the classical DE algorithm.

More results and analyses are presented in the next section.

7 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:

- 'efficient, non-greedy information sharing' instantiated via positive feedback of potentially good hypotheses between agents;
- dispensation mechanism – SDS-led random-restarts – deployed as part of the diffusion phase;
- 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.

To further analyse the role of SDS in the coupled algorithm, the Diffusion Phase of SDS algorithm is modified (see Algorithm 3) to investigate the dispensation effect caused by randomising a selection of agent hypotheses (effectively instantiating the population with SDS-led random-restarts). In other words, after the SDS test-phase, the hypothesis of each inactive agent is randomised.

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

In some cases ($f_{4,5,7}$), solely the dispensation mechanism (sDispDE), which is facilitated by the test-phase of the SDS algorithm, demonstrates a significantly better performance compared to the coupled algorithm (see Table 1). However, in the several cases, the coupled algorithms outperform the modified algorithm: $f_{2,8}$ and f_{10-14}, out of which f_2 and f_{12-14} are performing significantly better (see Table 2).

Table 1 shows that among the highlighted algorithms, out of 14 bechmarks, *sDE* exhibits the best performance as it is among the most significant in 9 cases; *sDispDE* and *DE* are among the best in 7 and 2 cases, respectively.

The results show the importance of coupling the SDS-led restart mechanism (dispensation mechanism) and the information sharing which are both deployed in SDS algorithm.

The third SDS component feature, which is currently only implicitly exploited by the coupled algorithm, is 'randomised partial hypothesis evaluation' (see (al-Rifaie et al., 2011b) for a detailed explanation on the implicit deployment of this feature).

Table 2: TukeyHSD Test Results for Accuracy.

	DE - sDE	DE - sDispDE	sDE - sDispDE
f_1	–	–	–
f_2	–	X–o	X–o
f_3	–	–	–
f_4	–	–	o–X
f_5	o–X	o–X	o–X
f_6	o–X	o–X	–
f_7	X–o	–	o–X
f_8	o–X	o–X	–
f_9	o–X	o–X	–
f_{10}	o–X	o–X	–
f_{11}	o–X	o–X	–
f_{12}	o–X	o–X	X–o
f_{13}	o–X	o–X	X–o
f_{14}	o–X	o–X	X–o

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.

Algorithm 3: SDS Dispensation coupled with DE (*sDispDE*).

```
// DIFFUSION PHASE
For ag = 1 to No_of_agents
   If ( ag is not active )
      ag.setHypo( randomHypo() )
   Else
      ag.setHypo(Gaussian(ag.getHypo(),aErrorV))
   End If
End For
```

7.1 Conclusions

This paper presents a brief overview about the potential of coupling of DE with SDS. Here, SDS is primarily used as an efficient resource allocation and dispensation mechanism responsible for facilitating communication between the agents at the early stages of the optimisation. Results reported in this paper have demonstrated that initial explorations with the coupled sDE algorithm outperform the performance of (one variation of) classical DE architecture. We believe similar techniques (e.g. (Omran et al., 2011)) can be applied to other swarm intelligence and evolutionary algorithms. As reported in (al-Rifaie et al., 2011a; al-Rifaie et al., 2011b) SDS has been also successfully integrated (vs. coupled) into PSO and DE in a different framework. In ongoing research, further theoretical work seeks to develop the core ideas presented in this paper on problems with significantly more computationally expensive objective functions.

This reinforces the idea of the integration of SI algorithms with EAs as a potential future approach in Evolutionary Computation.

REFERENCES

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

al-Rifaie, M. M., Bishop, M., and Blackwell, T. (2011a). 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*, pages 37–44, New York, NY, USA. ACM.

al-Rifaie, M. M., Bishop, M., and Blackwell, T. (2011b). Resource allocation and dispensation impact of stochastic diffusion search on differential evolution algorithm; in nature inspired cooperative strategies for optimisation (NICSO 2011) proceedings. *Studies in Computational Intelligence. Springer.*

Bishop, J. (1989). Stochastic searching networks. pages 329–331, London, UK. Proc. 1st IEE Conf. on Artificial Neural Networks.

Brest, J., Zamuda, A., Boskovic, B., Maucec, M., and Zumer, V. (2009). Dynamic optimization using self-adaptive differential evolution. In *IEEE Congress on Evolutionary Computation, 2009. CEC'09.*, pages 415–422. IEEE.

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

Omran, M., Moukadem, I., al-Sharhan, S., and Kinawi, M. (2011). Stochastic diffusion search for continuous global optimization. *International Conference on Swarm Intelligence (ICSI 2011), Cergy, France.*

Storn, R. and Price, K. (1995). Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. TR-95-012, [online]. Available: http://www.icsi.berkeley.edu/ storn/litera.html.

Weber, M., Neri, F., and Tirronen, V. (2010). Parallel Random Injection Differential Evolution. *Applications of Evolutionary Computation*, pages 471–480.

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

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

Zaharie, D. (2003). Control of population diversity and adaptation in differential evolution algorithms. In *Proc. of 9th International Conference on Soft Computing, MENDEL*, pages 41–46.

Zhang, J. and Sanderson, A. (2009). JADE: adaptive differential evolution with optional external archive. *Evolutionary Computation, IEEE Transactions on*, 13(5):945–958.