23

Stochastic Diffusion Search: A Tutorial

Mohammad Majid al-Rifaie

School of Computing and Mathematical Sciences
University of Greenwich, Old Royal Naval College, London, United Kingdom

J. Mark Bishop

Department of Computing
Goldsmiths, University of London, United Kingdom

CONTENTS

23.1	Introduction	307
23.2	Stochastic Diffusion Search	308
	23.2.1 The mining game	308
	23.2.2 Refinements in the metaphor	309
23.3	SDS architecture	310
23.4	Step by step example: text search	311
23.5	Source code	315
	23.5.1 Matlab	315
	23.5.2 C++	316
	23.5.3 Python	317
23.6	Conclusion	318
	References	318

23.1 Introduction

Noisy environments and incomplete data are often at the heart of hard, real-world search and optimisation-related problems, generating input that established search heuristics (e.g. tabu search [1], simulated annealing [2], etc.) sometimes have difficulty dealing with [3]. Conversely, ever since their inception, researchers have been attracted to the complex emergent behaviour, robustness and easy-to-understand architecture of nature-inspired swarm intelligence algorithms, and, particularly in challenging search environments. These algorithms have often proved more useful than conventional approaches [4]. SDS has been applied to various fields, including but not limited to: computational creativity [5, 6, 7, 8, 9] digital arts [10, 11, 12, 13] optimisation [14, 15, 16, 17] medical imaging [18, 19, 20, 21] machine learning [22, 23, 24, 25, 26] and other theoretical or practical domains [27, 28, 29, 30, 31, 32, 33].

This chapter explains the principles of Stochastic Diffusion Search, a multi-agent global search and optimisation swarm intelligence algorithm based upon simple iterated interactions between agents. First a high-level description of the algorithm is presented in the form of a search metaphor driven by social interactions. This is then followed by an example of a trivial 'text search' application to illustrate the core algorithmic processes by which standard SDS operates.

23.2 Stochastic Diffusion Search

Stochastic Diffusion Search (SDS) [34] has 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 [36]. Its computational roots stem from Geoff Hinton's interest in 3D object classification and mapping (see [37, 38] for Hinton's work and [34, 35] for the connection between Hinton mapping and SDS).

Unlike many nature inspired search algorithms, SDS has a strong mathematical framework, which describes the behaviour of the algorithm by investigating its resource allocation [39], convergence to global optimum [40], robustness and minimal convergence criteria [41] and linear time complexity [42]. In order to introduce SDS, a social metaphor, *the Mining Game*, is introduced.

23.2.1 The mining game

The mining game provides a simple metaphor outlining the high-level behaviour of agents in SDS:

> A group of friends (miners) learn that there is gold to be found on the hills of a mountain range but have no information regarding its distribution. On their maps, the mountain range is divided into a set of discrete hills and each hill contains a discrete set of seams to mine. Over time, on any day the probability of finding gold at a seam is proportional to its net wealth.
>
> To maximise their collective wealth, the miners need to identify the hill with the richest seams of gold so that the maximum number of miners can dig there (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 seam on his 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.
- At the end of the day, the miners congregate and over the evening each miner who is unhappy selects another miner at random to talk to. If the chosen miner is happy, he happily tells his colleague the identity of the hill he is mining (that is, he communicates his hill hypothesis, h, which thus both now maintain). Conversely, if the chosen miner is unhappy he says nothing and the original miner is once more reduced to selecting a new hypothesis – identifying the hill he is to mine the next day – at random.

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'. It can be shown that this process is isomorphic to SDS, and thus that the miners will naturally self-organise and rapidly congregate over hill(s) on the mountain range with a high concentration of gold.

Algorithm 21 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
```

23.2.2 Refinements in the metaphor

There are some refinements in the miners analogy, which will elaborate more on the correlation between the metaphor and different implementations of the algorithm. Whether an agent is active or not can be measured probabilistically or gold may be considered as a resource of discrete units. In both cases, the agents are either active or inactive at the end of each iteration[1]; this is isomorphic to standard SDS. The Mining Game can be further refined through either of the following two assumptions at each location:

1. Finite resources: the amount of gold is reduced each time a miner mines the area
2. Infinite resources: a conceptual situation with potentially infinite amount of gold

In the case of having finite resources, the analogy can be related to a real world experiment of robots looking for food to return to a notional nest site [43]. Hence the amount of food (or gold, in the mining analogy) is reduced after each discovery. In this case, the goal is identifying the location of the resources throughout the search space. This type of search is similar to conducting a search in a dynamically, agent-initiated changing environment where agents change their congregation from one area to another.

The second assumption has similarities with discrete function optimisation where values at certain points are evaluated. However further re-evaluation of the same points does not change their values and they remain constant.

23.3 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 21 for these phases in the mining game; for high-level SDS description see Algorithm 22):

- 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 and returning a domain independent boolean value. Later in the iteration, contingent on the strategy employed, successful hypotheses diffuse across the population and in this

[1] Whether an agent is active or not is defined using the following two methods:

- probabilistically: a function f takes a probability p as input and returns either true or false, $f(p) \Longrightarrow Active|Inactive$
- discretely: if there is gold, the agent will be active, otherwise it will be inactive.

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 the hypothesis. In the mining game metaphor, diffusion is performed by communicating a hill hypothesis.

Algorithm 22 SDS Algorithm.

```
Initialising agents()
While (stopping condition is not met)
   Testing hypotheses()
      Determining agents' activities (active/inactive)
   Diffusing hypotheses()
      Exchanging of information
End While
```

23.4 Step by step example: text search

In order to demonstrate the process through which SDS functions, an example is presented which shows how to find a set of letters within a larger string of letters. The goal is to find a 3-letter model (Table 23.1) in a 16-letter search space (Table 23.2). In this example, there are four agents. For simplicity of exposition, a perfect match of the model exists in the Search Space (SS).

In this example, a hypothesis, which is a potential problem solution, identifies three adjacent letters in the search space (e.g. hypothesis '1' refers to Z-A-V, hypothesis '10' refers to G-O-L).

TABLE 23.1
Model.

Index:	0	1	2
Model:	***S***	***I***	***B***

TABLE 23.2
Search Space.

Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Search Space	X	Z	A	V	M	Z	S	I	B	V	G	O	L	B	E	H

In the first step, each agent initially randomly picks a hypothesis from the search space (see Table 23.3). Assume that:

- the first agent points to the 12^{th} entry of the search space and in order to partially evaluate this entry, it randomly picks one of the letters (e.g. the first one, L):

L	B	E

- the second agent points to the 9^{th} entry and randomly picks the second letter (G):

V	**G**	O

- the third agent refers to the 2^{nd} entry in the search space and randomly picks the first letter (A):

A	V	M

- the fourth agent goes the 3^{rd} entry and randomly picks the third letter (Z):

V	M	**Z**

TABLE 23.3
Initialisation and Iteration 1.

Agent No:	1	2	3	4
Hypothesis position:	12 **L**-B-E	9 V-**G**-O	2 **A**-V-M	3 V-M-**Z**
Letter picked:	1^{st}	2^{nd}	1^{st}	3^{rd}
Status:	×	×	×	×

The letters picked are compared to the corresponding letters in the model, which is S-I-B (see Table 23.1).

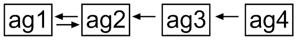
In this case:

- The 1^{st} letter from the first agent (L) is compared against the 1^{st} letter from the model (S) and because they are not the same, the agent is set inactive.
- For the 2^{nd} agent, the second letter (G) is compared with the second letter from the model (I) and again because they are not the same, the agent is set inactive.
- For the third and fourth agents, letters 'A' and 'Z' are compared against 'S' and 'B' from the model. Since none of the letters correspond to the letters in the model, the statuses of the agents are set inactive.

In the next step, as in the mining game, each inactive agent chooses another agent and adopts the same hypothesis if the selected agent is active. If the selected agent is inactive, the selecting agent generates a random hypothesis.

Assume that the first agent chooses the second one; since the second agent is inactive, the first agent must choose a new random hypothesis from the search space (e.g. 6). See Figure 23.1 for the communication between agents.

FIGURE 23.1
Agents Communication 1.



The process is repeated for the other three agents. As the agents are inactive, they all choose new random hypotheses (see Table 23.4).

TABLE 23.4
Iteration 2.

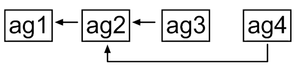
Agent No:	1	2	3	4
Hypothesis position:	6 S-**I**-B	10 G-O-**L**	0 **X**-Z-A	5 **Z**-S-I
Letter picked:	2^{nd}	3^{rd}	1^{st}	1^{st}
Status:	$\surd$	$\times$	$\times$	$\times$

In Table 23.4, the second, third and fourth agents do not refer to their corresponding letter in the model, therefore they become inactive. The first agent, with hypothesis '6', chooses the 2^{nd} letter (I) and compares it with the 2^{nd} letter of the model (I). Since the letters are the same, the agent becomes active.

At this stage, consider the following communication between the agents: (see Figure 23.2)

- the fourth agent chooses the second one
- the third agent chooses the second one
- the second agent chooses the first one

FIGURE 23.2
Agents Communication 2.



In this case, the third and fourth agents, which chose an inactive agent (the second agent), have to choose other random hypotheses each from the search space (e.g. agent three chooses hypothesis '1' which points to Z-A-V and agent four chooses hypothesis 4 which points to M-Z-S), but the second agent adopts the hypothesis of the first agent, which is active. As shown in Table 23.5:

- The first agent, with hypothesis '6', chooses the 3^{rd} letter (B) and compares it with the 3^{rd} letter of the model (B). Since the letters are the same, the agent remains active.
- The second agent, with hypothesis '6', chooses the 1^{st} letter (S) and compares it with the 1^{st} letter of the model (S). Since the letters are the same, the agent stays active.
- the third and fourth agents do not refer to their corresponding letter in the model, therefore they are set inactive.

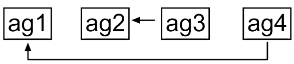
TABLE 23.5
Iteration 3.

Agent No:	1	2	3	4
Hypothesis position:	6 S-I-**B**	6 **S**-I-B	1 Z-**A**-V	4 M-Z-**S**
Letter picked:	3^{rd}	1^{st}	2^{nd}	3^{rd}
Status:	$\surd$	$\surd$	$\times$	$\times$

Because the third and fourth agents are inactive, they try to contact other agents randomly. For instance (see Figure 23.3):

- agent three chooses agent two
- agent four chooses agent one

FIGURE 23.3
Agents Communication 3.



Since agent three chose an active agent, it adopts its hypothesis (6). As for agent four, because it chose agent one, which is active too, it adopts its hypothesis (6). Table 23.6 shows:

- The first agent, with hypothesis '6', chooses the 1^{st} letter (S) and compares it with the 1^{st} letter of the model (S). Since the letters are the same, the agent remains active.
- The second agent, with hypothesis '6', chooses the 2^{nd} letter (I) and compares it with the 2^{nd} letter of the model (I). Since the letters are the same, the agent stays active.
- The third agent, with hypothesis '6', chooses the 3^{rd} letter (B) and compares it with the 3^{rd} letter of the model (B). Since the letters are the same, the agent becomes active.

- The fourth agent, with hypothesis '6', chooses the 1^{st} letter (S) and compares it with the 1^{st} letter of the model (S). Since the letters are the same, the agent is set active.

TABLE 23.6
Iteration 4.

Agent No:	1	2	3	4
Hypothesis position:	6 **S**-I-B	6 S-**I**-B	6 S-I-**B**	6 **S**-I-B
Letter picked:	1^{st}	2^{nd}	3^{rd}	1^{st}
Status:	$\surd$	$\surd$	$\surd$	$\surd$

At this stage, the entire agent populations are active pointing to the location of the model inside the search space.

23.5 Source code

The source code in this section provides the standard implementation of SDS in three programming languages, Matlab (listings 23.1), C++ (listing 23.2) and Python (listing 23.3).

23.5.1 Matlab

```
clear
% search space (ss); target text (model); population size (N)
ss = 'try to find sds in this sentence';
model = 'sds';
N = 10; maxIter = 30;

% INITIALISE AGENTS
hypo = randi([1 length(ss)-length(model)],1,N) ;
status = false(1,N);

for itr=1:maxIter
  activeAgents = 0;
  % TEST PHASE
  for i=1:N
    % PICK A MICROFEATURE TO PARTIALLY EVALUATE HYPOTHSIS
    microFeature = randi([1 length(model)]);
    if ss( hypo(i)+microFeature ) == model(microFeature)
      status(i) = true;
      activeAgents = activeAgents+1;
    else
      status(i) = false;
    end
  end

  % DIFFUSION PHASE
  for i=1:N
```

```
    if status(i) == false % INACTIVE AGENT
      rand = randi([1 N]); % PICK RANDOM AGENT TO COMMUNICATE
      if status(rand) == true % SHARE HYPOTHESIS
        hypo(i) = hypo(rand);
      else % PICK A RANDOM HYPOTHSIS
        hypo(i) = randi([1 length(ss)-length(model)]);
      end
    else % ACTIVE AGENT
      microFeature = randi([1 length(model)]); % PICK MICROFEATURE
      if ss( hypo(i)+microFeature ) == model(microFeature)
        status(i) = true;
      else
        status(i) = false;
      end
    end
  end
  activityPercentage = activeAgents * 100 / N;
  % DISPLAYING ACTIVITY PERCENTAGE AND THE FIRST AGENT'S HYPOTHESIS
  disp(['Active agents: ' num2str(activityPercentage) '% ... found: '
    ss(hypo(1):length(model))])
end
```

Listing 23.1
SDS code in Matlab.

23.5.2 C++

```
#include <iostream>
#include <string>
#include <stdlib.h>
#include <ctime>
using namespace std;

float r() { // GENERATE RANDOM NUMBER IN RANGE [0, 1)
  return (float) rand() / (RAND_MAX);
}

int main() {
  srand(time(NULL)); // TO GENERATE DIFFERENT RANDOM NUMBERS
  string ss = "try to find sds in this sentence"; // SEARCH SPACE
  string model = "sds"; // TARGET TEXT
  int N = 10; int maxIter = 30; int hypo[N]; bool status[N];

  // INITIALISE AGENTS
  for (int i=0; i<N; i++) {
    hypo[i] = r()*(ss.length() - model.length());
    status[i] = false;
  }

  // MAIN LOOP
  for (int itr=0; itr < maxIter; itr++) {
    int activeAgents = 0;
    // TEST PHASE
    for (int i=0; i<N; i++) {
      // PICK A MICROFEATURE TO PARTIALLY EVALUATE HYPOTHESIS
      int microFeature = r()*model.length();
      if (ss[ hypo[i]+microFeature ] == model[microFeature] ) {
        status[i] = true;
        activeAgents++;
      }
      else
        status[i] = false;
    }
    // DIFFUSION PHASE
    for (int i=0; i<N; i++) {
      if (status[i] == false){ // INACTIVE AGENT
```

```
        int rand = r()*N; // PICK RANDOM AGENT TO COMMUNICATE
        if (status[rand] == true) // SHARE HYPOTHESIS
          hypo[i] = hypo[rand];
        else // PICK A RANDOM HYPOTHESIS
          hypo[i] = r()*(ss.length() - model.length());
      }
      else { // ACTIVE AGENT
        int microFeature = r()*model.length(); // PICK MICROFEATURE
        if (ss[ hypo[i]+microFeature ] == model[microFeature] )
          status[i] = true;
        else
          status[i] = false;
      }
    }
    int activityPercentage = activeAgents * 100 / N;
    // DISPLAYING ACTIVITY PERCENTAGE AND THE FIRST AGENT'S HYPOTHESIS
    cout << "Active agents:" << activityPercentage << "%\t   ... found:
      " << ss.substr(hypo[0],model.length()) << endl;
  }
  return 0;
}
```

Listing 23.2
SDS code in C++.

23.5.3 Python

```
import numpy as np
# search space (ss); target text (model); population size (N)
ss = 'try to find sds in this sentence'
model = 'sds'
N = 10; maxIter = 30

# INITIALISE AGENTS
hypo = np.random.randint(len(ss)-len(model), size=(N));
status = np.zeros((N), dtype=bool)

for itr in range (maxIter):
  activeAgents = 0;
  # TEST PHASE
  for i in range (N):
    # PICK A MICROFEATURE TO PARTIALLY EVALUATE HYPOTHESIS
    microFeature = np.random.randint(len(model))
    if ss[ hypo[i]+microFeature ] == model[microFeature]:
      status[i] = True
      activeAgents += 1
    else:
      status[i] = False

  # DIFFUSION PHASE
  for i in range (N):
    if status[i] == False: # INACTIVE AGENT
      rand = np.random.randint(N) # PICK RANDOM AGENT TO COMMUNICATE
      if status[rand] == True: # SHARE HYPOTHESIS
        hypo[i] = hypo[rand];
      else: # PICK A RANDOM HYPOTHESIS
        hypo[i] = np.random.randint(len(ss)-len(model))
    else: # ACTIVE AGENT
      microFeature = np.random.randint(len(model)) # PICK MICROFEATURE
      if ss[ hypo[i]+microFeature ] == model[microFeature]:
        status[i] = True;
      else:
        status[i] = False;

  activityPercentage = activeAgents * 100 / N;
  # DISPLAYING ACTIVITY PERCENTAGE AND THE FIRST AGENT'S HYPOTHESIS
```

```
    print('Active agents:', activityPercentage, '%  ... found: ', ss[
      hypo[0]:hypo[0]+len(model) ] )
```

Listing 23.3
SDS code in Python.

23.6 Conclusion

This chapter aimed at providing an introduction to the main principles of Stochastic Diffusion Search (SDS). After providing a simple social metaphor, outlining the high-level behaviour of agents in SDS, the architecture of the algorithm is presented where the two main phases of SDS (test and diffusion) are detailed. This is then followed by a step by step example, which is demonstrated through simple text search using SDS. A complete code of SDS is then provided in Matlab, C++ and Python. Therefore, researchers and students alike are able to expand their understanding of SDS and apply the algorithms to the problems of their choice, especially where the concept of 'partial function evaluation' can be applied.

References

1. F. Glover et al. "Tabu search-part I". ORSA Journal on Computing, pp. 190-206, 1989.
2. S. Kirkpatric, C.D. Gelatt, M.P. Vecchi. "Optimization by simulated annealing". Science, vol. 220(4598), pp. 671-680, 1983.
3. Y. Jin, J. Branke. "Evolutionary optimization in uncertain environments-a survey". IEEE Transactions on Evolutionary Computation, vol. 9(3), pp. 303-317, 2005.
4. J.F. Kennedy, R.C. Eberhart, Y. Shi. "Swarm Intelligence". Morgan Kaufmann Publishers, 2001.
5. M.M. al-Rifaie, J.M. Bishop, S. Caines. "Creativity and Autonomy in Swarm Intelligence Systems". Cognitive Computation, vol. 4(3), pp. 320-331, 2012.
6. M.M. al-Rifaie, J.M. Bishop. "Weak and Strong Computational Creativity" in Computational Creativity Research: Towards Creative Machines. Atlantis Thinking Machines, vol. 7, pp. 37-49, Atlantis Press, 2015.
7. M.M. al-Rifaie, F.F. Leymarie, W. Latham, M. Bishop. "Swarmic autopoiesis and computational creativity". Connection Science, pp. 1-19, 2017.

8. J.M. Bishop, M.M. al-Rifaie. "Autopoiesis, creativity and dance". Connection Science, vol. 29(1), pp. 21-35, 2017.
9. M.M. al-Rifaie, A. Cropley, D. Cropley, M. Bishop. "On evil and computational creativity". Connection Science, vol. 28(1), pp. 171-193, 2016.
10. M.M. al-Rifaie, A. Aber, M. Bishop. "Cooperation of Nature and Physiologically Inspired Mechanisms in Visualisation" in Biologically-Inspired Computing for the Arts: Scientific Data through Graphics, IGI Global, United States, 2012.
11. M.M. al-Rifaie, M. Bishop. "Swarmic Paintings and Colour Attention". Lecture Notes in Computer Science, vol. 7834, pp. 97-108, Springer, 2013.
12. M.M. al-Rifaie, M. Bishop. "Swarmic Sketches and Attention Mechanism". Lecture Notes in Computer Science, vol. 7834, pp. 85-96, Springer, 2013.
13. A.M. al-Rifaie, M.M. al-Rifaie. "Generative Music with Stochastic Diffusion Search". Lecture Notes in Computer Science, vol. 9027, pp. 1-14, Springer, 2015.
14. M.M. al-Rifaie, M.J. Bishop, T. Blackwell. "An investigation into the merger of stochastic diffusion search and particle swarm optimisation" in *Proc. of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011*, pp. 37-44, 2011.
15. M.M. al-Rifaie, J.M. Bishop, T. Blackwell. "Information sharing impact of stochastic diffusion search on differential evolution algorithm". Memetic Computing, vol. 4(4), pp. 327-338, 2012.
16. M.M. al-Rifaie, M. Bishop, T. Blackwell. "Resource Allocation and Dispensation Impact of Stochastic Diffusion Search on Differential Evolution Algorithm" in Nature Inspired Cooperative Strategies for Optimization (NICSO 2011). Studies in Computational Intelligence, vol. 387, pp. 21-40, Springer, 2012.
17. M.G.H. Omran, A. Salman. "Probabilistic stochastic diffusion search" in Proc. of International Conference on Swarm Intelligence (ANTS 2012), Lecture Notes in Computer Science, vol. 7461, pp. 300-307, Springer, 2012.
18. M.M. al-Rifaie, A. Aber, D.J. Hemanth. "Deploying swarm intelligence in medical imaging identifying metastasis, micro-calcifications and brain image segmentation". Systems Biology, IET, vol. 9(6), pp. 234-244, 2015.
19. M.M. al-Rifaie, A. Aber. "Identifying Metastasis in Bone Scans with Stochastic Diffusion Search" in Proc. of International Symposium on Information Technologies in Medicine and Education, pp. 519-523, 2012.

20. M.M. al-Rifaie, A. Aber, A.M. Oudah. “Utilising Stochastic Diffusion Search to identify metastasis in bone scans and microcalcications on mammographs” in Proc. of IEEE International Conference on Bioinformatics and Biomedicine Workshops, pp. 280-287, 2012.
21. M.M. al-Rifaie, A. Aber, A.M. Oudah. “Ants intelligence framework; identifying traces of cancer” in The House of Commons, UK Parliment. SET for BRITAIN 2013. Poster exhibitions in Biological and Biomedical Science, 2013.
22. M.M. al-Rifaie, M. Y.-K. Matthew and M. d’Inverno. “Investigating swarm intelligence for performance prediction” in *Proc. of the 9th International Conference on Educational Data Mining*, pp. 264-269, 2016.
23. H. Alhakbani, M.M. al-Rifaie. “Feature selection using stochastic diffusion search” in *Proc. of the Genetic and Evolutionary Computation Conference*, pp. 385-392, 2017.
24. M.M. al-Rifaie, D. Joyce, S. Shergill, M. Bishop. “Investigating stochastic diffusion search in data clustering” in SAI Intelligent Systems Conference (IntelliSys), pp. 187-194, 2015.
25. H.A. Alhakbani, M.M. al-Rifaie. “Exploring Feature-Level Duplications on Imbalanced Data Using Stochastic Diffusion Search” in Multi-Agent Systems and Agreement Technologies, pp. 305-313, Springer, 2016.
26. I. Aleksander, T.J. Stonham. “Computers and Digital Techniques 2(1)” in Lecture Notes in Artificial Intelligence, vol. 1562, pp. 29-40, Springer, 1979.
27. M.A.J. Javid, M.M. al-Rifaie, R. Zimmer. “Detecting symmetry in cellular automata generated patterns using swarm intelligence” in *Proc. of 3rd International Conference on the Theory and Practice of Natural Computing (TPNC 2014)*, pp. 83-94, 2014.
28. F.M. al-Rifaie, M.M. al-Rifaie. “Investigating stochastic diffusion search in DNA sequence assembly problem” in *Proc. of SAI Intelligent Systems Conference (IntelliSys)*, pp. 625-631, 2015.
29. M.A.J. Javid, W. Alghamdi, A. Ursyn, R. Zimmer, M.M. al-Rifaie. “Swarmic approach for symmetry detection of cellular automata behaviour”. Soft Computing, vol. 21(19), pp. 5585-5599, 2017.
30. E. Grech-Cini. “Locating Facial Features”. PhD Thesis, University of Reading, Reading, UK, 1995.
31. P.D. Beattie, J.M. Bishop. “Self-localisation in the SENARIO autonomous wheelchair”. Journal of Intellingent and Robotic Systems, vol. 22, pp. 255-267, 1998.
32. A.K. Nircan. “Stochastic Diffusion Search and Voting Methods”. PhD Thesis, Bogaziki University, 2006.

33. K. de-Meyer, M. Bishop, S. Nasuto. "Small World Effects in Lattice Stochastic Diffusion Search". Lecture Notes in Computer Science, vol. 2415, pp. 147-152, 2002.

34. J.M. Bishop. "Stochastic searching networks" in *Proc. of 1st IEE Conf. on Artificial Neural Networks*, pp. 329-331, 1989.

35. J.M. Bishop, P. Torr. "The Stochastic Search Network" in Neural Networks for Images, Speech and Natural Language, pp. 370-387, Chapman & Hall, New York, 1992.

36. K. de-Meyer, J.M. Bishop, S.J. Nasuto. "Stochastic diffusion: Using recruitment for search" in Proc. of Symposium on Evolvability and Interaction, pp. 60-65, The University of London, UK, 2003.

37. G.F. Hinton. "A parallel computation that assigns canonical object-based frames of reference" in *Proc. of the 7th International Joint Conference on Artificial Intelligence*, vol. 2, pp. 683-685, 1981.

38. J.L. McClelland, D.E. Rumelhart, et al. "Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models", A Bradford Book, MIT Press, 1986.

39. S.J. Nasuto. "Resource Allocation Analysis of the Stochastic Diffusion Search". PhD Thesis, University of Reading, Reading, UK.

40. S.J. Nasuto, J.M. Bishop. "Convergence analysis of Stochastic Diffusion Search". Parallel Algorithms and Applications, vol. 14(2), pp. 89-107, 1999.

41. D.R. Myatt, J.M. Bishop, S.J. Nasuto. "Minimum stable convergence criteria for Stochastic Diffusion Search". Electronics Letters, vol. 40(2), pp. 112-113, 2004.

42. S.J. Nasuto, J.M. Bishop, S. Lauria. "Time Complexity Analysis of Stochastic Diffusion Search" in Proc. of International ICSC IFAC Symposium on Neural Computation, pp. 260-266, 1998.

43. M.J. Krieger, J.B. Billeter, L. Keller. "Ant-like task allocation and recruitment in cooperative robots". Nature, vol. 406, pp. 992-995, 2000.