Stochastic Diffusion Search Review

Mohammad Majid al-Rifaie, Mark, J. Bishop
Department of Computing
Goldsmiths College
University of London
London SE14 6NW, UK

***Abstract*—Stochastic Diffusion Search (SDS), first incepted in 1989, belongs to the extended family of Swarm Intelligence algorithms. In contrast to many nature-inspired algorithms, SDS has a strong mathematical framework describing its behaviour and convergence. In addition to concisely exploring the algorithm in the context of natural swarm intelligence systems, this paper reviews the developments of this algorithm, which are shown to perform well in a variety of application domains.**

I. Introduction

Noisy environments and incomplete data have often been at the heart of search and optimisation-related problems; something that conventional heuristics (e.g. Simulated Annealing [1], Tabu Search [2], etc.) often have difficulty dealing with. Additionaly, the easy-to-understand architecture of these algorithms and complex emergent behaviour has attracted many researchers.

This review paper surveys SDS, a multi-agent global search and optimisation algorithm, which is based on simple interaction of agents. After considering SDS in the broader context of natural swarms, a high-level description of SDS is presented in the form of a social metaphor followed by a simple search example demonstrating the procedures through which SDS conducts the search. The architecture and development of SDS are then discussed in greater detail. In addition to analysing the behaviour of SDS and the possibility of embedding different interaction strategies, the novel way it deals with computational costly objective funtions is investigated. Issues related to applications of SDS are then presented. We conclude this review by discussing potential research opportunities.

II. Swarm Intelligence

In recent years, studies of the behavour of social insects have suggested several new metaheuristics for use in collective intelligence. This has given rise to a concomitant increasing interest in distributed computation through the interaction of simple agents in nature-inspired algorithms (e.g. evolutionary algorithms [3], genetic algorithms [4], [5], ant algorithms [6], [7], particle swarm optimsation [8] and etc.).

Swarm intelligence which investigates collective intelligence, aims at modelling intelligence by looking at individuals in a social context and monitoring their interaction with one another as well as their interaction with the environment [9]. Natural examples of swarm intelligence that exhibit these forms of interaction are fish schooling, birds flocking, ant colonies in nesting and foraging, bacterial growth, animal herding, brood sorting and etc.

The story of the *blind men and the elephant* also suggests how social interaction can possibly lead to human intelligence. This famous tale set in verse by John Godfrey Saxe [10] in the 19th century, characterises six blind men approaching an elephant. They ended up having six different ideas about the elephant, as each person experienced one aspect of the elephant's body: wall (elephant's side), spear (tusk), snake (trunk), tree (knee), fan (ear) and rope (tail). The moral of the story is to show how people build their beliefs based on incomplete beliefs derived from incomplete knowledge about the world [11]. If the blind men had been communicating about what they were experiencing, they would have possibly come up with the conclusion that they were exploring the heterogeneous qualities that make up an elephant.

Although some writers (e.g. [12], [13]) blur the difference between adaptation and intelligence by claiming that intelligence is actually the ability to adapt, other writers of the field of swarm intelligence (e.g. [11]) emphasise that an individual is not an isolated information processing entity. Stochastic Diffusion Search, which functions by interaction between agents, adopts the second view and shares some characteristics and behaviours of swarms intelligence algorithms which can be best understood by observing the behaviours of social insects such as ants and bees in locating food sources and nest site location. In the next two parts of this section (II-A and II-B), SDS is investigated in this context.

A. Communication in Social Insects

Communication – social interaction or information exchange – observed in social insects is important in all swarm intelligence algorithms, including SDS. Although as stated in [11], in real social interations, not just the syntactical information is exchanged between the individuals but also semantic rules and beliefs about how to process this information; in swarm intelligence algorithms, only the syntactical exhange 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 [14]. Self-interation between individuals is carried out through recruitment and it has been demonstrated that there are various

recruitment strategies in ants [15] and honey bees [16], [17]. 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.

There are different forms of recruitment in social insects; it may take the form of local or global; one-to-one or one-to-many; and stochastic or deterministic mode. The nature of information exchange also varies in different environments and with different types of social insects. Sometimes the information exchange is more 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 in their community [18]. In the next part, different forms of information exchange in some social insects are discussed in further detail and their relation to SDS recruitment strategies are presented.

B. Methods of Communication

Chemical communication through pheromones forms the primary method of recruitment in ants. However in one species of ants, *Leptothorax acervorum*, where a 'tandem calling' mechanism (one-to-one communication) is used, the forager ant that finds the food location, recruits a single ant upon its return to the nest, and therefore the location of the food is physically publicised [19]. In group recruitment, an ant convenes a larger number of ants, leading them to the food location. Laying the pheromone trial from the food source to the nest is of more advanced nature, in which the leading ant is not physically in contact with other ants. However, the most advanced form of ant recruitment is mass recruitment [20] in which the worker ants follow the pheromone trail, but individual ants add an amount of pheromones alongside their journey towards the food location. Therefore, the amount of pheromone plays an important role in the outflow attraction of the ants.

In another primitive ant species where nest replacement is studied [21], an ant with a better nest location, summons an ant with a poorer choice. In this algorithm, API, ants are all called to the best nest found so far and subsequently they start exploring the area again for a better nest location. Different recruitment and communication algorithms induce different performances. Ants communicating through group recruitment are faster than tandem calling ants, and similarly, ants utilising mass recruitment are more efficient in their performances than the former recruitment strategies [20]. Ant algorithms were successfully applied to hard optimisation and search problems such as traveling salesman problem and the quadratic assignment problem [22].

However, as mentioned in [23], the success of the ants in reaching the food they have been recruited to obtain, varies from one species to another. In another form of communication, indirect or stigmergetic communication, the exchange of information is based on modifying the physical properties of the environment and its success lies in spatial and temporal attributes of mass recruitment and the positive feedback mechanism it employs. In this mode, which is based on using pheromone, short routes are loaded with more pheromone (because of the short time it takes the ants to travel [24]).

Although the recruitment behaviour of real ants recruitment is more complex than the behaviour in SDS, they are both population-based and find their optima via agents communicating with each other.

An ant-like task allocation has been investigated in [25] where robots were used to simulate different non-communication and communication strategies, concluding that ant-inspired techniques of decentralised control, namely tandom-calling recruitment mechanism [19] shows better results than single robots doing the same task. This technique of information exchange is an instance of a broader type of recruitment strategy utilised in stochastic diffusion search [26], which will be discussed in more detail, later in this paper.

In honey bees the group recruitment is performed by means of waggle dances, in which the direction of the dance shows the location of the food source and the speed of the dance represents the distance to the target area. Each bee chooses one of the dancing bees as a guide to the food source.

In SDS, direct one-to-one communication (which is similar to tandem calling recruitment) is utilised. The effect of different recruitment strategies are discussed later (see Section V).

C. Search and Optimisation

In the swarm intelligence literature, search and optimisation are often used interchangeably. Nevertheless, search has been catogorised in three broad types in [27]:

- In the first definition, search refers to finding a (target) model in a search space, and the goal of the algorithm is to find a match, or the closest match to the target in the search space. This is defined as *data search* and is considered as a classical meaning of search in computer science [28].
- In the second type, the goal is finding a path (*path search*) and the list of the steps leading to a certain solution is what the search algorithm tries to achieve. In this type of search, paths do not exist explicitly but are rather created during the course of the search.
- In the third definition, *solution search*, the goal is to find a solution among a large problem space of candidate solutions. Similar to the path search where paths do not exist explicitly, the search space consists of candidate solutions which are not stored explicitly but rather created and evaluated during the search process. However, on the contrary to the path search, the steps taken to find the solution are not the goal of the algorithm.

In optimisation, which is similar to the third search definition, the model of the first definition is replaced with an objective or fitness function which is used to evaluate possible solutions. In both search and optimisation, the positions of the optima are not known in advance (even though the optima itself might be known a-priori). The task of the fitness function is to measure the proximity of the candidate solutions to the optima based on the criteria provided by each optimisation problem. The

algorithm compares the output of the function with the output of the previously located candidate solutions and, for instance, in case of a minimisation problem, the smaller the output the better the solution. Data search can be seen as a caste of optimisation if the objective function tests the equality of the candidate solution with the model.

III. Stochastic Diffusion Search

Stochastic Diffusion Search (SDS) [29] 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 [30].

Unlike many natured inspired search algorithms, SDS has a strong mathematical framework, which describes the behaviour of the algorithm by investigating its resource allocation [31], convergence to global optimum [32], robustness and minimal convergence criteria [33] and linear time complexity [34].

In order to introduce SDS, a social metaphor, *the Mining Game*, is introduced.

A. 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

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

While (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
```

B. Refinements in the Metaphor

There are some refinements in the miners analogy, which will elaborate more on the correletion between the metaphor and different implementations of the algorithm.

Whether an agent is active or not can be measured probablistically or gold may be considered as resource of discrete units. In both cases all 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 [25]. Hence the amount of food (or gold, in the mining analogy) is reduced after each discovery. In that experiement, an ant-like algorithm is used to avoid robots interfering with one another; considering individual variation in performing the task; and also recruiting other robots when identifying a rich area is investigated. 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.

[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.

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.

IV. 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)

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 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.

Algorithm 2 SDS Algorithm

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

A. A Search Example

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 I) in a 16-letter search space (Table II). In this example, there are four agents. For simplicity of exposition, a perfect match of the model exists in the Search Space (SS).

Table I
MODEL

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

Table II
SEARCH SPACE

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

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

In the first step, each agent initially randomly picks a hypothesis from the search space (see Table III). 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 III
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 I).

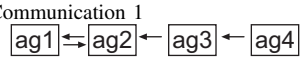
In this case:

- 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 status 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 1 for the communication between agents.

Figure 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 IV).

Table IV
ITERATION 2

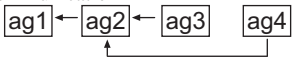
Agent No:	1	2	3	4
Hypothesis position:	6 S-**I**-B	10 G-O-**L**	0 **X**-Z-A	5 **Z**-S-I
Letter picked:	2^{nd}	3^{rd}	1^{st}	1^{st}
Status:	$\surd$	$\times$	$\times$	$\times$

In Table IV, 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 2)

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

Figure 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 V:

- 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.

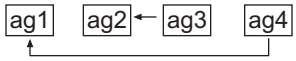
Table V
ITERATION 3

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 3):

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

Figure 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 VI 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 VI
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.

B. Initialisation and Termination

Although normally agents are uniformly distributed throughout the search space, if the search space is of a specific type, or knowledge exists about it *a priori*, it is possible to use a more intelligent (than uniform random distribution) startup by biasing the initialisation of the agents.

If there is a pre-defined pattern to find in the search space, the goal will be locating the best match or, if it does not exist, its best instantiation in the search space [32]. Similarly, in a situation which lacks a pre-defined pattern, the goal will be finding the best pattern in accord with the objective function.

In both cases, it is necessary to have a termination strategy. In one method[2], SDS terminates the process when a statistical equilibrium state is reached, which means that the threshold of the number of active agents is exceeded and the population maintained the same state for a specified number of iterations. In [35], four broad types of halting criteria are introduced:

1) No stopping criterion, where the user interrupts the course of action of the search or optimisation and is usually preferred when dealing with *dynamically changing* problem spaces or when there is no predefined pattern to look for

[2]Ibid

2) Time-based criterion, in which passing a pre-set duration of time is the termination point of the algorithm
3) Activity-based criterion, which is a problem-dependent halting criterion, and it is the most prevalent form in the SDS algorithm. The termination of the process is decided upon through monitoring the overall activity of the agents (e.g. reaching a certain user defined activity level, reaching a stable population state after a sudden increase in their activities)
4) Cluster-based criterion that keeps tracks of the formation of stable clusters.

Introducing stopping criteria adds extra computations to what would be a distributed algorithm otherwise. As an alternative to the full-model cluster-based criteria, just a small proportion of the population can be considered to check whether it points to the same hypothesis [35]. Increasing the size of the already monitored sample might be considered afterwards.

Additionally, in order to reduce the computational complexity of the search, it is also possible to run the termination procedure after every n iterations.

The two most common termination strategies in SDS (introduced in [32]) are the following:

- Weak halting criterion is the ratio of the active agents to the total number of agents. In this criterion, cluster sizes are not the main concern.
- Strong halting criterion investigates the number of active agents that forms the largest cluster of agents all adopting the same hypothesis.

Therefore, the choice of the halting mechanism is based on whether to favour the active agents in the whole of the agent populations (weak halting mechanism), which is similar to the activity-based criterion, or to consider the largest cluster of active agents (strong halting mechanism), which is similar to the cluster-based criterion.

C. Partial Function Evaluation

One of the concerns associated with many optimisation algorithms (e.g. Genetic Algorithm [5], Particle Swarm Optimisation [8] and etc.) is the repetitive evaluation of a computationally expensive fitness functions. In some applications, such as tracking a rapidly moving object, the repetitive function evaluation significantly increases the computational cost of the algorithm. Therefore, in addition to reducing the number of function evaluations, other measures should be taken in order to reduce the computations carried out during the evaluation of each possible solution as part of the 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 [36], [37], which is often not the case in real-world applications. Examples of costly evaluation functions are seismic data interpretation [37], selection of sites for the transmission infrastructure of wireless communication networks and radio wave propagation calculations of one site [38] and etc.

Costly functions have been investigated under different conditions [39] 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 [40], [41].
- 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 [39].

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 and means for guiding the optimisation.

The partial function evaluation of SDS allows the algorithm to absorb certain types of noise in the objective function without affecting the convergence time or the size and stability of the clusters.

Additionally, noise, which does not alter the averaged probabilities of the test score (probability of producing active agents during the test phase, averaged over all component functions) but increases the variance in the evaluation of component functions, has no effect on the resource allocation process of SDS [18]. However, if the value of test score changes as a result of noise presence, the resource allocation process may be influenced either:

- positively if the value of the test score increases
- or negatively if the value of the test score decreases

Dynamic Environments: The application of partial function evaluation is of more significance when the problem space is dynamically changing and the evaluation process is of more repetitive nature. Repeated (re)evalutions of fitness functions in many swarm intelligence algorithms necessiate having less costly fitness functions.

Diffusion or the selection mechanism tends to reduce the diversity in the population or the population homogeneity [18], which in turn leads to an inadequate subsequent reactions in a dynamically changing fitness function.

SDS aims at proposing a new solution (see Section IV-E) to the problem of population homogeneity by utilising an alternative method to balance the trade off between *wide exploration* of all possible solution in the problem space and the *detailed exploitation* of any possible smaller region which might be a candidate for holding the sought object.

D. Convergence

Convergence time is defined as the number of iterations needed before a stable population of active agents is formed.

The SDS algorithm allocates its resources by defining convergence as locating the best match in the search space.

An important factor in convergence is the ratio of the number of agents to the size of the solution space. In [42], it is proved that in a noiseless environment convergence always happens.

Additionally, in [32] it is proved that all agents become active when searching for a solution in a noiseless environment where a perfect match exists.

As mentioned before, the probability of an agent being active averaged over all component functions is the test score, which in turn determines the behaviour of SDS, and it is proved that the population size and the test score determine the average cluster size as well as convergence times.

The approximately linear time complexity of SDS is analysed in [32] and two extereme cases in the convergence time have been considered there:

- First, when, in the initial stages, some of the agents point to the correct position in the search space, which results in a shorter convergence time
- In the second case, there is no agent pointing to the correct position for some time after the initialisation, which may lead to a longer process before the first agent locates a potentially correct location.

It has also been shown that the accuracy and convergence time in SDS is proportionately robust to the amount of noise in the search space.

Convergence to a global optimal solution in SDS is discussed in [34].

E. Resource Allocation and Stability

In addition to convergence time, steady-state resource allocation is one of the important factors in the performance criteria of SDS [43]. In order to measure the robustness of the algorithm, in case of the presence of noise and imperfect matches, resource allocation is taken into account, which is determined as the average number of active states when the search shows steady-state behaviour. Although, resource allocation in standard SDS is *dynamic* and *self-regulatory*, there are certain issues to be investigated.

Local Exploitation and Global Exploration : In standard SDS, there is no explicit mechanism to shift the balance from local exploitation (*detailed exploitation*) to global exploration (*wide exploration*) of candidate solutions.

As observed in [44], a metaheuristic tries to exploit self-similarity and regularities of the fitness function, which indicates that neighbouring solutions in the problem space have alike properties. Adding this mechanism to SDS may be helpful; one way of embedding this into the algorithm is to add a small random offset to the hypotheses before copying them to other agents during the diffusion phase, which is similar to mutation in evolutionary algorithms [18], [35]. The effect of this minor change in the hypotheses is to investigate the nearby solutions, which generally serves as a hill-climbing mechanism improving the overall performance of the SDS and results in improved convergence time in solution spaces with self-similarity. Nevertheless, it also accelerates the finding of more optimal solutions in the vicinity of already found ones.

In dynamically changing environments, it is important to explore the solution space even after finding a suitable candidate solution, as once a good solution is detected, a large propertion of agents are attracted to it, thus limiting further exploration of the solution space. Therefore, the Context Sensitive and Context Free mechanisms (described in Section V-A) are proposed to shift the balance of the search back to exploration.

A full account of Markov chain based analysis of the stochastic nature of standard SDS for resource allocation and the steady state probability distribution of the whole swarm is extensively discussed in [31]. More information about search behaviour and resource allocation can also be found in [45], [46].

In heuristic multi-agent systems, the possibility of agents losing the best solution results in destabilising or even non-convergence of the algorithm. Conversely, it is shown that the solution found by SDS are exceptionally stable [47].

V. Variations in SDS

In SDS, similar to other optimisation algorithms, the goal is finding the best solution based on the criteria specified in the objective function. The collection of all candidate solutions (hypotheses) forms the search space and each point in the search space is represented by an objective value, from which the objective function is formed [18]. In the minimisation mode, for example, the lower the objective value is the better the result is.

Although there might not be a direct way of finding the best objective function for a problem, many optimisation problems can be transformed into the minimisation form [35].

One of the issues related to SDS is the mechanism behind allocating resources to ensure that while potential areas of the problem space are exploited, exploration is not ignored. For this purpose, different recruitments methods, where one agent recruits another one, are investaged:

A. Recruitment Strategies

Three recruitment strategies are proposed in [48]: active, passive and dual. These strategies are used in the Diffusion Phase of SDS. Each agent can be in either one of the following states: It is *active* if the agent is successful in the Test Phase; an agent is *inactive* if it is not successful; and it is *engaged* if it is invloved in a communication with another agent.

The standard SDS algorithm [29] uses the passive recruitment mode, which will be described next followed by other recruitment modes.

Passive Recruitment Mode: In the *passive recruitment mode*, if the agent is not active, another agent is randomly selected and if the randomly selected agent is active, the hypothesis of the active agent is communicated (or *diffused*) to the inactive one. Otherwise a new random hypothesis is generated for the inactive agent and there will be no flow of information (see Algorithm 3).

Active Recruitment Mode: In the *active recruitment mode*, active agents are in charge of communication with other agents. An active agent randomly selects another agent. If the randomly selected agent is neither active nor engaged in communication with another active agent, then the hypothesis of the active agent is communicated to the inactive one and the agent is flagged as engaged. The same process is repeated for the rest of the active agents. However if an agent is neither active nor engaged, a new random hypothesis is generated for it (see Algorithm 4).

Algorithm 3 Passive Recruitment Mode

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

Algorithm 4 Active Recruitment Mode

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

for ag = 1 to No_of_agents
   if ( ag.activity() == false and
        ag.getEngaged() == false )
      ag.setHypothesis( randomHypothesis() )
end
```

Dual Recruitment Mode: In *dual recruitment mode*, both active and inactive agents randomly select other agents. When an agent is active, another agent is randomly selected. If the randomly selected agent is neither active nor engaged, then the hypothesis of the active agent is shared with the inactive one and the inactive agent is flagged as engaged. Also, if there is an agent which is neither active nor engaged, it selects another agent randomly. If the newly selected agent is active, there will be a flow of information from the active agent to the inactive one and the inactive agent is flagged as engaged. Nevertheless, if there remains an agent that is neither active nor engaged, a new random hypothesis is chosen for it.

Context Sensitive Mechanism: Comparing the above-mentioned recruitment modes, it is theoretically determined in [48] that robustness and greediness decrease in the active recruitment mode. Conversely, these two properties are increased in dual recruitment strategy. Although, the greediness of dual recruitment mode results in decreasing the robustness of the algorithm, the use of *Context Sensitive Mechanism* limits this decrease [48], [31]. In other words, the use of context sensitive mechanism biases the search towards global exploration. In the context sensitive mechanism if an active agent randomly chooses another active agent that maintains the same hypothesis, the selecting agent is set inactive and adopts a random hypothesis. This mechanism frees up some of the resources in order to have a wider exploration throughout the search space as well preventing cluster size from overgrowing, while ensuring the formation of large clusters in case there exists a perfect match or good sub-optimal solutions (see

Algorithm 5 Dual Recruitment Mode

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

for ag = 1 to No_of_agents
if ( ag.activity()   == false and
     ag.getEngaged() == false )
   ag.setHypothesis( randomHypothesis() )
end
```

Algorithm 6).

Algorithm 6 Context Sensitive Mechanism

```
if ( ag.activity() == true )
   r_ag = pick a random agent()
   if ( r_ag.activity()   == true and
        ag.getHypothsis() == r_ag.getHypothsis() )
      ag.setActivity( false )
      ag.setHypotheis( randomHypothsis() )
```

Context Free Mechanism: In *Context Free Mechanism,* which is another recruitment mechanism, the performance is similar to context sensitive mechanism, where each active agent randomly chooses another agent. However, if the selected agent is active (irrespective of having the same hypothesis or not), the selecting agent becomes inactive and picks a new random hypothesis. By the same token, this mechanism ensures that even if one or more good solutions exist, about half of the agents explore the problem space and investigate other possible solutions (see Algorithm 7).

Algorithm 7 *Context Free Mechanism*

```
if ( ag.activity() == true )
   r_ag = pick a random agent()
   if ( r_ag.activity() == true )
      ag.setActivity( false )
      ag.setHypotheis( randomHypothsis() )
```

B. Synchronous and Asynchronous Update

Although, in the original SDS [29], synchronous mode is used, the diffusion of successful hypotheses can be accom-

plished synchronously or asynchronously.

In synchronous diffusion mode, the updates of all hypotheses occur simultaneously (all agents progress through the cycle of test-diffusion at the same time).

There are two methods for asynchronous mode; in the first method, the hypothesis of each agent is updated individually (agents, in turn, go through a test-diffusion cycle). In the second method, there is no explicit synchronisation between agents, which is the case in a true parallel implementation.

As mentioned in [35], in many variants, the behaviour of a asynchronous process is approximately the same as the synchronous one.

C. Composite Hypotheses

In standard SDS all hypotheses are homogeneous and thus have the same type. In this section, new variants of SDS are introduced where there are two *different* types of hypotheses working together. These SDS types are applied to solve parameter estimation problems, which is a more complicated search problem compared to pattern matching. In parameter estimation, outlier data (or random noise) is embeded in the data (or inlier data); and the goal is to find parameter values that best describe the inlier data [49]. Data Driven SDS [50] and coupled SDS [49], which have composite hypotheses, are both used to solve parameter estimation problems. An example of parameter estimation problem is locating a spoken word in an audio file which has some noise. In estimation problem, similar to other search problems, a cost function or objective funciton is required to measure how close the algorithm is to the inlier data or the model in the search space.

In parameter estimation, the objective function is optimised with respect to the estimated model parameters; that is why it is considered an optimisation problem [50].

Data Driven SDS: Data Driven SDS (DDSDS) is shown to outperform [50] Maximum Likelihood Estimator Sample Concensus (MLSESAC) which is a variant of RANdom SAmple Consensus (RANSAC), one of the most popular and robust estimators based on stochastic principles [51].

DDSDS contains a composite hypothesis: a manifold hypothesis, which maintains the minimum necessary dataset for describing a hypothesis; and a datum hypothesis, which represent the smallest building block of the hypothesis. If estimating a line is the problem, then the manifold hypothesis would consists of two points, which are sufficient to represent a line, and the datum hypothesis would be a single point that is randomly selected from the manifold hypothesis rather than the whole of the search space.

In the test phase, random datums are selected just from datum hypotheses that are associated with the agents. The probability of selecting a datum, which has no link with any agents is zero. This will dynamically constrain the selection to data generated by the inlier distribution [50]. Next, the distance of the agent's manifold hypothesis from the randomly selected datum is evaluated to see if the distance stays within the preset inlier threshold value. If this is the case, the agent's state becomes active.

In the diffusion phase, active agents diffuses its manifold and datum hypotheses to the inactive agent. When an inactive agent is not involved in any information exchange, similar to the initialisation phase, it chooses two random data from the entire search space for the manifold hypothesis and the datum hypothesis is selected from one of the two elements of the manifold hypothesis.

Coupled SDS: In Coupled SDS (CSDS) two independent populations of agents are formed each maintaining different type of hypothesis, namely the manifold hypotheses and datum hypotheses. On the contrary to DDSDS, datum hypotheses are selected randomly from the entire search space. The size of these two populations are not necessarily the same. They are randomly and independently initiated with data from the entire search space. In the test phase, the manifold hypothesis of one agent is compared with the datum hypothesis of another one. Based on the distance threshold, if the datum matches the manifold, both of the agents become active. This evaluation is called composite hypothesis evaluation, which is more complicated than the synchronous evaluation in standard SDS, where there is just one population of agents. Therefore, in addition to asynchronous test, two other synchronisation modes were proposed:

- Master/Slave Synchronisation, where one of the populations is master and the other is slave. The master hypothesis randomly select a hypothesis from the slave population for the test. In this mode, there will be m composite evaluation, where m is the size of the master population.
- Sequential Master Synchronisation is a variant of master/slave mode, where populations take turn to be master. Each iteration has n composite evaluations, which is the sum of all agents in both manifold and datum populations.

The diffusion phase in CSDS is similar to the standard SDS for each population independently, where the information flow is allowed *within* each population of agents and thus there is no information exchange between the manifold and datum population of agents [49].

It is empirically shown that DDSDS converges even when there are 50% more outliers and it also outperforms standard SDS in convergence time [50]. Both of these SDS variants have been proposed to improve the performance of the original SDS towards stable convergence in high noise estimation tasks.

VI. Applications

There are several applications associated with SDS which have been successfully applied to diverse problems.

SDS was first introduced by a simple text searching algorithm in 1989 [52] demonstrating the use of partial function evaluation technique, by partially evaulating the text to find the model or the best match.

Subsequently, in 1992, tracking eyes has been investigated in [42]. In this project, a hybrid stochastic search network is used to locate eye positions within grey scale images of human faces. It was shown that the network can accurately locate the eye features of all the subjects it has been trained with and it could reach over sixty percent success in locating eye features on subjects on which the system has not been explicitly trained with.

In another project in 1995, similar to the ones above, SDS is used in solving visual search tasks, such as object recognition (in this case, locating facial features[53]). The details of another visual related task for real time tracking of lips in video films is given in [53], where SDS uses a hybrid system of a set of n-tuple neurons [54].

Exploring a set of candidate positions to self-localise autonomous wheelchair or robot in a complex busy environment through a number of cells has been used in a method called Focused Stochastic Diffusion Network [55] in 1998; in this method, the space of possible positions is examined in parallel by a set of competive cooperative cells to find out the most likely position of the robot or wheelchair in the environment.

Later in 1999, emergent characteristics of neuron functionality has also been described using a new metaphor based on SDS utilising spiking neurons [56] and also it was argued that the metaphor of conventional computational description of brain operation is too restrictive.

SDS was also used in wireless transformation networks, where the location of transmission infrastructure is of significance to keep the cost minimum while preserving adequate area coverage [38]. In this application at 2002, given a set of candidate sites, a network should be designed so that at certain number of reception points, the signal from at least one transmitter can be received.

Sequence search application of SDS was tried in 2002, using Constrained Stochastic Diffusion Search (CSDS) [57]. It is an extension to best-fit string-maching SDS while allowing the identification of best-fit sequences (usally referred to as optimal alignment[58]), where there might be gaps between contiguous sub-strings of a model in the search space. CSDS is applied to the field of computational molecular biology (e.g. identifying regions of DNA that would code for an amino-acid sequence).

Another visual problem was attempted in 2005, using Group Stochastic Search (GSS) [59] with the goal of locating and tracking objects (e.g. head) in cluttered environment. In this application, each agent utilises SDS, an n-tuple weightless neural network [60] and a histogram intersection technique to score its location [61]. Since the application works when the speed is high, exhaustive and computationally expensive searchs for the head are not useful. GSS is an extension for video of SDS [62], which was introduced to located known patterns in images but is not able to deal with the changing search space of video.

In 2008 [63], SDS was used in feature tracking in the context of Atmospheric Motion Vectors derivation, as using template matching techniques, such as Euclidean distance or cross-correlation for tracking steps is very expensive computationally.

In an ongoing work by the authors, the potential of merging SDS with Particle Swarm Optimisation (PSO) [] is being investigated, where SDS's resource allocation is utilised.

A. Implementation on Hardware

SDS is inherently parallel in nature and the hardware implementation of the algorithm is feasible. Still, the fact that the original SDS model requires full inter-agent connectivity, where each agent is able to communicate directly with all others in the population, casues fundamental difficulty in the efficient implementation of the algorithm on parallel computer or dedicated hardware.

One of the solutions proposed in [43] was to limit the communication between the agents. Agents are considered spatially located in a lattice (Lattice SDS or LSDS) where each agent is only connected to the k-nearest neighbours.

As a second solution, the agent swarm can be divided into several sub-swarms. In this mode, each sub-swarm runs on a separate processor and they are fully connected while allowing just a low frequency of communication between swarms. This process is applied to the diffusion phase, during which agents communicate with each another.

Therefore, considering this form of SDS, agents just communicate with the ones they are connected to. It was shown that a network with randomly connected agents (random graph), with small number of long-range connections, performs similar to standard SDS or ordered lattice with roughly the same number of connections[3]. The following conclusion has been drawn that restricting the number of interconnectivity in random or small-world networks – which is a lattice with a few additional number of long-range connections – does not have huge effect on the performance of SDS algorithm. Also, the rate of information spread is higher in random graphs and small-world networks than ordered lattices.

Analysing the number of connections and the connection topology leads to the following conclusion: it has been argued that when a high-dimensional problem is considered, the time at which one of the agents becomes active (time to hit [52]), T_h, is bigger than the time required for the active agent to spread its successful hypothesis T_d [43]. Although random graphs have shorter T_d than regular lattices, but they are harder to implement on parallel hardware, because the connection are not necessarily local or short. In small-world lattice SDS topology, which shows the performance of a fully interconnected standard SDS, adding random links decrease T_d exponentially.

Therefore T_d is considered to be an important factor, which not only affect the convergence time, but is also seen as a parameter for resource allocation stability [64] as well as an indirect measure for robustness [43].

VII. Conclusions

This paper gives a brief account of the research carried out on stochastic diffusion search, a population-based, nature-inspired probabilistic approach, which solves best-match problems mainly by communication between agents. An important feature that makes SDS different from many other optimisation techniques is the mathematical framework that proves its convergence to optimal solution even in noisy search spaces. SDS is proposed to investigate dynamically changing environments and in contrast to many connectionist models that find the solution by approaching a specific point in the weight space which results in decreasing of their activity after convergence,

[3]Ibid

SDS is able to continue the exploration over the search space further on after locating the optimum.

The adaptability of SDS comes from its convergence, which ensures the diversity among its populations in the search space, which in turn makes SDS suitable for dynamically changing environments [32].

One of the on going researches on SDS is an attempt to propose another way of describing possible emergence of cognitive processing in a metaphor supported by SDS algorithm, where neurons are considered as communication devices rather than computational ones [65], [66].

On the computational side, in addition to continuous SDS optimisation, its resource allocation is currently being investigated in integration with other swarm intelligence algorithms.

REFERENCES

[1] S. Kirkpatric, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[2] F. Glover *et al.*, "Tabu search-part i," *ORSA journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[3] T. Back, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

[4] J. H. Holland, "Adaptation in natural and artificial systems," *Ann Arbor, MI, University of Michigan press*, 1975.

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

[6] M. Dorigo, "Optimization, learning and natural algorithms," *Milano: Politecnico di Italy, PhD thesis*, 1992.

[7] M. Dorigo, V. Maniezzo, A. Colorni, M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," *Dipartimento di Elettronica e Informatica, Politecnico di*, 1991.

[8] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. IV. Piscataway, NJ: IEEE Service Center, 1995, pp. 1942–1948.

[9] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford University Press, USA, 1999.

[10] J. G. Saxe, D. Lathen, and B. Chief, "The Blind Man and the Elephant," *The Poems of John Godfrey Saxe*, 1882.

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

[12] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press, 1995.

[13] W. Ashby, *Design for a Brain*. Chapman and Hall London, 1960.

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, p. 3942, 2000.

[15] B. Holldobler and E. O. Wilson, *The Ants*. Springer-Verlag, 1990.

[16] L. J. Goodman and R. C. Fisher, *The Behaviour and Physiology of Bees*. Oxon, UK: CAB International, 1991.

[17] T. D. Seeley, *The Wisdom of the Hive*. Harvard University Press, 1995.

[18] K. de Meyer, S. Nasuto, and J. Bishop, "Stochastic diffusion optimisation: the application of partial function evaluation and stochastic recruitment in swarm intelligence optimisation," *Springer Verlag*, vol. 2, Chapter 12 in Abraham, A. and Grosam, C. and Ramos, V. (eds), "Swarm intelligence and data mining", 2006.

[19] M. Moglich, U. Maschwitz, and B. Holldobler, "Tandem calling: A new kind of signal in ant communication," *Science*, vol. 186, no. 4168, pp. 1046–1047, 1974.

[20] R. Chadab and C. Rettenmeyer, "Mass recruitment by army ants," *Science*, vol. 188, pp. 1124–1125, 1975.

[21] N. MonmarchÃš, G. Venturini, and M. Slimane, "On how pachycondyla apicalis ants suggest a new search algorithm," *Future Generation Computer Systems*, vol. 16, no. 9, pp. 937–946, 2000.

[22] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.

[23] J. DENEUBOURG, J. PASTEELS, and J. VERHAEGHE, "Probabilistic behaviour in ants: a strategy of errors?" *Journal of theoretical biology*, vol. 105, no. 2, pp. 259–271, 1983.

[24] H. Fan, Z. Hua, J. Li, and D. Yuan, "Solving a shortest path problem by ant algorithm," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 5, 2004, pp. 3174–3177 vol.5.

[25] M. J. Krieger, J. B. Billeter, and L. Keller, "Ant-like task allocation and recruitment in cooperative robots." *Nature*, vol. 406, no. 6799, pp. 992–5, 2000.

[26] M. Bishop, K. de Meyer, and S. Nasuto, "Recruiting robots perform stochastic diffusion search," *SCRAP*, 2002.

[27] M. Mitchell, *An introduction to genetic algorithms, 1996*. MIT press.

[28] D. E. Knuth, *The art of computer programming. Vol. 3, Sorting and Searching*. Addison-Wesley Reading, MA, 1973.

[29] J. Bishop, "Stochastic searching networks." London, UK: Proc. 1st IEE Conf. on Artificial Neural Networks, 1989, pp. 329–331.

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

[31] S. J. Nasuto, "Resource allocation analysis of the stochastic diffusion search," Ph.D. dissertation, PhD Thesis, University of Reading, Reading, UK, 1999.

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

[33] D. R. Myatt, J. M. Bishop, and S. J. Nasuto, "Minimum stable convergence criteria for stochastic diffusion search," *Electronics Letters*, vol. 40, no. 2, pp. 112–113, 2004.

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

[35] K. de Meyer, "Foundations of stochastic diffusion search," Ph.D. dissertation, PhD thesis, University of Reading, Reading, UK, 2003.

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

[37] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245–276, 1996.

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

[39] Y. Jin, "A comprehensive survey of fittness approximation in evolutionary computation," *In: Soft Computing*, vol. 9, pp. 3–12, 2005.

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

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

[42] J. Bishop and P. Torr, "The stochastic search network," *Neural Networks for Images, Speech and Natural Language*, pp. 370–387, 1992.

[43] K. de Meyer, M. Bishop, and S. Nasuto, "Small world effects in lattice stochastic diffusion search," in *Proc. ICANN 2002*. Madrid, Spain: Lecture Notes in Computer Science, 2415, 2002, pp. 147–152.

[44] S. Christensen and F. Oppacher, "What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001, pp. 1219–1226.

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

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

[47] S. Nasuto and M. Bishop, "Stabilizing swarm intelligence search via positive feedback resource allocation," *Nature Inspired Cooperative Strategies for Optimization (Nicso 2007)*, 2008.

[48] D. Myatt, S. Nasuto, and J. Bishop, "Alternative recruitment strategies for stochastic diffusion search," *Artificial Life X, Bloomington USA*, 2006.

[49] J. Bishop, "Coupled stochastic diffusion processes," *Proc. SCARP, Reading, UK*, pp. 185–187, 2003.

[50] D. Myatt and J. Bishop, "Data driven stochastic diffusion networks for robust high-dimensionality manifold estimation - more fun than you can shake a hyperplane at," *Proc. SCARP, Reading, UK*, 2003.

[51] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[52] J. Bishop, "Anarchic techniques for pattern classification," Ph.D. dissertation, PhD Thesis, University of Reading, Reading, UK, 1989.
[53] E. Grech-Cini, "Locating facial features," Ph.D. dissertation, PhD Thesis, University of Reading, Reading, UK, 1995.
[54] I. Aleksander and T. Stonham, "Computers and digital techniques 2(1)," *Lect. Notes Art. Int 1562*, pp. 29–40, 1979.
[55] P. Beattie and J. Bishop, "Self-localisation in the senario autonomous wheelchair," *Journal of Intellingent and Robotic Systems*, vol. 22, pp. 255–267, 1998.
[56] S. J. Nasuto, K. Dautenhahn, and J. Bishop, "Communication as an emergent methaphor for neuronal operation," *Lect. Notes Art. Int 1562*, pp. 365–380, 1999.
[57] D. Jones, "Constrained stochastic diffusion search."
[58] M. Tompa, "Lecture notes on biological sequence analysis," *Dept. of Comp. Sci. and Eng., University of Washington, Seattle, Technical report*, 2000.
[59] M. Evans and J. Ferryman, "Group stochastic search for object detection and tracking."
[60] M. Morciniec and R. Rohwer, "The n-tuple classifier: Too good to ignore," Tech. Rep. Technical Report NCRG/95/013, 1995.
[61] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Citeseer, 1998, pp. 232–237.
[62] R. Summers, "Stochastic diffusion search: A basis for a model of visual attention?" 1998.
[63] A. Hernandez-Carrascal and S. Nasuto, "A SWARM INTELLIGENCE METHOD FOR FEATURE TRACKING IN AMV DERIVATION," *Ninth International Wind Workshop*, 2008.
[64] K. de Meyer, "Explorations in stochastic diffusion search: Soft-and hardware implementations of biologically inspired spiking neuron stochastic diffusion networks," Technical Report KDM/JMB/2000, Tech. Rep., 2000.
[65] S. Nasuto, J. Bishop, and K. de Meyer, "Communicating neurons: A connectionist spiking neuron implementation of stochastic diffusion search," *Neurocomputing*, vol. 72, no. 4-6, pp. 704–712, 2009.
[66] M. Bishop, "A Cognitive Computation Fallacy? Cognition, Computations and Panpsychism," *Cognitive Computation*, vol. 1, no. 3, pp. 221–233, 2009.

CONTENTS