Stochastic Diffusion Search: Partial Function Evaluation In Swarm Intelligence Dynamic Optimisation

Chapter *in* Studies in Computational Intelligence · October 2006

DOI: 10.1007/978-3-540-34690-6_8

CITATIONS
46

READS
397

3 authors:

Kris De Meyer
King's College London
37 PUBLICATIONS **537** CITATIONS
SEE PROFILE

Slawomir J Nasuto
University of Reading
211 PUBLICATIONS **4,173** CITATIONS
SEE PROFILE

John Mark Bishop
Goldsmiths University of London
129 PUBLICATIONS **2,374** CITATIONS
SEE PROFILE

Stochastic Diffusion Search: partial function evaluation in swarm intelligence dynamic optimisation

Kris De Meyer[1], Slawomir J. Nasuto[2], and Mark Bishop[3]

[1] King's College London, University of London, UK
kris_demeyer@kcl.ac.uk
[2] Department of Cybernetics, The University of Reading
Whiteknights, Reading, RG6 6AY, UK
s.j.nasuto@reading.ac.uk
[3] Department of Computing, Goldsmiths College
New Cross, London, SE14 6NW, UK
m.bishop@gold.ac.uk

1 Summary

The concept of partial evaluation of fitness functions, together with mechanisms manipulating the resource allocation of population based search methods, are presented in the context of Stochastic Diffusion Search, a novel swarm intelligence metaheuristic that has many similarities with ant and evolutionary algorithms. It is demonstrated that the stochastic process ensuing from these algorithmic concepts has properties that allow the algorithm to optimise noisy fitness functions, to track moving optima, and to redistribute the population after quantitative changes in the fitness function. Empirical results are used to validate theoretical arguments.

2 Introduction

In recent years there has been growing interest in a distributed mode of computation utilising interaction between simple agents, (e.g., evolutionary algorithms; particle swarm optimisation; ant algorithms etc.). Certain of these "swarm intelligence" systems have been directly inspired by observing interactions between social insects, such as ants and bees. For example, algorithms inspired by the behaviour of ants – ant algorithms – typically use the principle of communication via pheromone trails to successfully tackle hard search and optimisation problems (see [19] for a recent review). This indirect form of communication, based on modification of the physical properties of

the environment, has been termed stigmergetic communication. The problem solving ability of these algorithms emerges from the positive feedback mechanism and spatial and temporal characteristics of the pheromone mass recruitment system they employ. Other swarm intelligence methods explore mechanisms based on biological evolution, flocking behaviour, brood sorting and co-operative transport, [28].

Independently of the above mechanisms, Stochastic Diffusion Search (SDS) was proposed in 1989 as a population-based pattern-matching algorithm [3] [4]. Unlike stigmergetic communication employed in ant algorithms, SDS uses a form of direct communication between agents (similar to the tandem calling mechanism employed by one species of ants, Leptothorax Acervorum, [33]).

SDS uses a population of agents where each agent poses a hypothesis about the possible solution and evaluates it partially. Successful agents repeatedly test their hypothesis while recruiting unsuccessful agents by direct communication. This creates a positive feedback mechanism ensuring rapid convergence of agents onto promising solutions in the space of all solutions. Regions of the solution space labelled by the presence of agent clusters can be interpreted as good candidate solutions. A global solution is thus constructed from the interaction of many simple, locally operating agents forming the largest cluster. Such a cluster is dynamic in nature, yet stable, analogous to "a forest whose contours do not change but whose individual trees do" [1].

Optimisation problems with stochastic and dynamically changing objectives pose an interesting challenge to many swarm intelligence algorithms which require repeated (re)evaluations of the fitness function. For certain applications the computational cost of these evaluations can prove prohibitive: e.g., for online tracking of rapidly changing objectives, or for computationally expensive fitness functions. In addition, in the case of genetic optimisation of dynamically changing objectives, an additional complication comes from the tendency of selection mechanisms to reduce diversity in the population (population homogeneity), potentially resulting in inadequate responses to subsequent changes in the fitness function. The first issue has previously been addressed by methods that attempt to reduce the amount of evaluation work performed, e.g., by estimating fitness values or by evaluating cheap, approximative fitness functions instead. The second issue has typically been addressed by methods introducing or preserving diversity in the population.

SDS handles these two problems in a related, but slightly different manner: firstly, it utilises the radically different concept of *partial evaluation* of fitness functions to save on the computational cost of repeated evaluations, reminiscent of the partial information available to individual social insects as they engage in recruitment behaviour. Secondly, the variation and selection mechanisms employed by SDS offer a new solution to the population homogeneity problem providing an alternative mechanism to balance the tradeoff between a wide *exploration* of all feasible solutions and a detailed *exploitation* of a small number of them.

This chapter introduces SDS in the context of swarm intelligence algorithms and demonstrates its applications in the field of stochastic and dynamic optimisation. The chapter is structured as follows: Sect. 3 discusses interaction mechanisms in social insects. Section 4 introduces partial evaluation of the fitness function and the balance between exploration and exploitation in search (the allocation of resources). In Sect. 5, an in-depth account of the standard SDS algorithm is provided. Section 6 examines the similarities and differences between SDS and Swarm Intelligence algorithms. Alternative mechanisms for the manipulation of resource allocation in SDS are discussed in Sect. 7. Section 8 illustrates the use of SDS in a few simple stochastic and dynamic optimisation problems. Finally, discussion and conclusions are presented in Sect. 9 and Sect. 10 respectively.

3 Mechanisms of Interaction in Social Insects

Swarm intelligence views the behaviour of social insects – ants, bees, termites and wasps – as offering a powerful problem solving metaheuristic with sophisticated collective intelligence. Composed of simple interacting agents, this intelligence lies in a network of interactions among the individuals and between the individuals and the environment [6].

Social interaction in ants [24] and honey bees [21] [43] has evolved an abundance of different recruitment strategies with the purpose of assembling agents at some point in space for foraging or emigration to a new nest site.

Such recruitment forms can be local or global, one to one or one to many, deterministic or stochastic. The informational content of the interaction ranges from very simple to complex and can be partial or complete. However, all such recruitment mechanisms propagate useful information through the colony as a whole.

Often, the recruitment is based on exchange of very simple stimulative information to trigger a certain action. Although the stimulative effect of a recruitment signal is typically mixed with the directional function of the signal, they actually constitute different functions: the stimulative function is merely used to induce following behaviour in other individuals, whereas the directional function conveys the information of where exactly to go.

In ants, chemical communication through the use of pheromones constitutes the primary form of recruitment. From an evolutionary viewpoint, the most primitive strategy of recruitment seems to be tandem running: a successful foraging ant will, upon its return to the nest, attract a single ant (different strategies exist - chemical, tactile or through motor display) and physically lead this ant to the food source.

In so-called group recruitment, an ant summons several ants at a time, then leads them to the target area. In more advanced group recruitment strategies, successful scouts lay a pheromone trail from the food source to the nest. Although this trail in itself does not have a stimulative effect, ants that are

stimulated by a motor display in the nest can follow the trail to the food source without additional cues from the recruiter.

Finally, the most developed form of recruitment strategy is mass recruitment. Stimulation occurs indirectly: the pheromone trail from nest to food source has both a stimulative and directional effect. Worker ants encountering the trail will follow it without the need for additional stimulation. Individual ants deposit an amount of pheromones along the trail, dependent on the perceived quality or type of the food source. The outflow of foragers is dependent on the total amount of pheromone discharged. Recruitment strategies during emigration to new nest sites show a similar wide variety of physiology and behaviour.

In honeybees, both stimulation and orientation occur primarily via motor display. Bees that have successfully located a source of nectar or pollen will engage in so called *waggle dances*. The direction of the dance indicates the direction of the food source, whereas the velocity of the dance depends on the distance to the find. The perceived quality and accessibility of the food source influence the probabilities that a particular forager becomes a dancer, continues exploiting the food source without recruiting or abandons the food source and becomes a follower. A follower bee follows the dance of one randomly chosen dancing bee, then tries to find the food source indicated by that bees dance.

When compared to the stimulative function of recruitment strategies in ants, bees can be said to practice group recruitment: each bee directly recruits several other bees during its time on the dance floor. However, the directional function is very different. Whereas ants either have to lead the follower to the food source - which is time consuming - or leave signposts along the way; bees do neither. They have evolved a form of symbolic communication, more adapted to their specific conditions.

Different foraging and recruitment strategies induce different quantitative performances. For ants, it was demonstrated that tandem recruitment is slower than group recruitment, which in turn is slower than mass recruitment [12]. Also, the degree of accuracy - how many ants reach the food source for which they have been recruited - is dependent on the type of communication used and differs greatly from species to species [17].

Whatever the exact details of the recruitment behaviour, it leads to a dynamical balance between exploration of the environment and exploitation of the discovered food sources. Abstracting the social interaction and recruitment mechanisms observed in insect societies has inspired the design of many of the artificial swarm intelligence methods. The next section will concentrate on one such heuristic abstracted from natural systems – that of partial information exchange – and discuss its implications for search efficiency.

4 The Concept of Partial Evaluation

Many functions that have commonly been used as benchmark problems for swarm intelligence algorithms (e.g., evolutionary algorithms, particle swarm optimisation, etc.) typically have relatively small evaluation costs [18, 44]. This stands in stark contrast with real-world applications, which are not necessarily so well-behaved – for several possible reasons: the evaluation cost of a single candidate solution may be a rapidly-increasing function of the number of parameters, as e.g., for some problems in seismic data interpretation [44]; or, even an evaluation cost that is linear in the number of function parameters can be excessively high: for example, the selection of sites for the transmission infrastructure of wireless communication networks can be regarded as a *set-cover* problem [20] with an evaluation cost of candidate solutions that is linear in the number of sites; however, the evaluation of a single site involves costly radio wave propagation calculations [25]. Hence for swarm intelligence algorithms which explicitly evaluate costly fitness functions, it is not only important to limit the total number of fitness evaluations, but also the amount of computational work that is performed during the evaluation of a single candidate solution. This is exceedingly true for stochastic and dynamically changing problems, which may require multiple and continuing function evaluations.

The problem of costly function evaluations has been addressed many times independently for static and dynamic, noisy and noise-free problem settings (see [26] for a recent review). Two somewhat different approaches exist: firstly, the fitness of part of the individuals can be *estimated* – rather than calculated – from the fitness of other individuals or individuals from previous generations using tools from statistics [9, 27]. In the second line of approach, the costly fitness function is replaced with a cheaper, approximate fitness function, which is evaluated instead; when the search has started to converge, the computational process can switch to evaluating the original fitness function to ensure correct convergence [26].

In contrast, by analogy to the partial information about the environment available to individuals in insect societies, the approach advocated here capitalises on the fact that many fitness functions are *decomposable* into components that can be evaluated independently. An evaluation of only one or a few of these components – a *partial evaluation* of the fitness function – may still hold enough information for optimisation purposes. The next section will introduce a metaheuristic based on partial evaluation of fitness function.

5 Stochastic Diffusion Search

Stochastic Diffusion Search (SDS) is an efficient generic search method, originally developed as a population-based solution to the problem of best-fit pattern matching. SDS uses a one-to-one recruitment system akin to the

tandem-running behaviour found in certain species of ants. In this section we will introduce the SDS algorithm and subsequently demonstrate that efficient global decision making can emerge from interaction and communication in a population of individuals each forming hypotheses on the basis of partial evidence.

We start by providing a simple metaphor, the restaurant game, that encapsulates the principles of SDS behaviour.

5.1 The restaurant game

A group of delegates attends a long conference in an unfamiliar town. Each night they have to find somewhere to dine. There is a large choice of restaurants, each of which offers a large variety of meals. The problem the group faces is to find the best restaurant, that is the restaurant where the maximum number of delegates would enjoy dining. Even a parallel exhaustive search through the restaurant and meal combinations would take too long to accomplish. To solve the problem delegates decide to employ a Stochastic Diffusion Search.

Each delegate acts as an agent maintaining a hypothesis identifying the best restaurant in town. Each night each delegate tests his hypothesis by dining there and randomly selecting one of the meals on offer. The next morning at breakfast every delegate who did not enjoy his meal the previous night, asks one randomly selected colleague to share his dinner impressions. If the experience was good, he also adopts this restaurant as his choice. Otherwise he simply selects another restaurant at random from those listed in 'Yellow Pages'.

Using this strategy it is found that very rapidly significant number of delegates congregate around the best restaurant in town. Abstracting from this algorithmic process:

```
Initialisation phase
  whereby all agents (delegates) generate
  an initial hypothesis (restaurant)
loop
  Test phase
    Each agent evaluates evidence for its hypothesis
    (meal degustation). Agents divide into active
    (happy diners) and inactive (disgruntled diners).
  Diffusion phase
    Inactive agents adopt a new hypothesis by either
    communication with another agent (delegate) or, if the
    selected agent is also inactive, there is no information
    flow between the agents; instead the selecting agent
    must adopt a new hypothesis (restaurant) at random.
endloop
```

By iterating through test and diffusion phases agents stochastically explore the whole solution space. However, since tests succeed more often on good candidate solutions than in regions with irrelevant information, an individual agent will spend more time examining good regions, at the same time recruiting other agents, which in turn recruit even more agents. Candidate solutions are thus identified by concentrations of a substantial population of agents.

Central to the power of SDS is its ability to escape local minima. This is achieved by the probabilistic outcome of the partial hypothesis evaluation in combination with reallocation of resources (agents) via stochastic recruitment mechanisms. Partial hypothesis evaluation allows an agent to quickly form its opinion on the quality of the investigated solution without exhaustive testing (e.g. it can find the best restaurant in town without having to try all the meals available in each).

Terminology

In the original formulation of SDS a population of *agents* searches for the best solution to a given optimisation problem. The set of all *feasible* solutions to the problem forms the *solution space* $\mathcal{S}$. Each point in $\mathcal{S}$ has an associated *objective* value. The objective values taken over the entire solution space form an *objective function* f. For simplicity reasons, it is assumed that the objective is to minimise the sum of n {0,1}-valued *component functions* f_i:[4]

$$\min_{\forall s \in \mathcal{S}} f(s) = \min_{\forall s \in \mathcal{S}} \sum_{i=1}^{n} f_i(s) \qquad f_i : \mathcal{S} \rightarrow \{0,1\} \ . \tag{1}$$

Although this may seem as a serious restriction, many optimisation problems can actually be transformed into (1) – as explained in [31]. Section 8 will also give an example of such a transformation. During operation, each agent maintains a *hypothesis* about the best solution to the problem; a hypothesis is thus a candidate solution, or designates a point in the solution space. No a-priori assumptions are made about the representation of hypotheses: they can be binary strings, symbolic strings, integer numbers, or even (at least in theory) real numbers.

Algorithm

Agents in the original SDS algorithm operate synchronously. They undergo various stages of operation, which are summarised in the algorithm below

```
Initialise(Agents);
repeat
  Test(Agents);
```

[4]Component functions f_i can be deterministic or probabilistic.

```
  Diffuse(Agents);
until (Halting Criterion);
```

Initialise

As a first step, agents' hypothesis parameters need to be initialised. Different initialisation methods exist, but their specification is not needed for the basic understanding of the algorithm; a discussion can be found in [31].

Test

Each agent randomly selects a single component function f_i, $i \in \{1, \ldots, n\}$, and evaluates it for its particular hypothesis $s_h \in \mathcal{S}$. Based on the outcome of the evaluation, agents are divided into two groups: *active* and *inactive*. For active agents, $f_i(s_h) = 0$; for inactive agents, $f_i(s_h) = 1$. Please note that, by allowing f_i to be probabilistic, it is possible that different evaluations of $f_i(s_h)$ have a different outcome. The test phase is described in pseudo-code below.

```
for agent = 1 to (All Agents)
  cf = Pick-Random-Component-Function();
  if (cf(agent.hypothesis) == 0)
    agent.activity = TRUE;
  else
    agent.activity = FALSE;
  end
end
```

Diffuse

During the *diffusion phase*, each inactive agent chooses at random another agent for communication. If the selected agent is active, then the selecting agent copies its hypothesis: *diffusion* of information. If the selected agent is also inactive, then there is no flow of information between agents; instead, the selecting agent adopts a new random hypothesis. Active agents, from their side, do not start a communication session in standard SDS. The diffusion phase is summarised below.

```
for agent = 1 to (All Agents)
  if (agent.activity == FALSE)
    agent2 = Pick-Random-Agent(Agents);
    if (agent2.activity == TRUE)
      agent.hypothesis = agent2.hypothesis;
    else
      agent.hypothesis = Pick-Random-Hypothesis();
    end
  end
end
```

Halt

Several different types of halting criteria exist [31]; their specification is not needed for the understanding of the algorithm. The most simple halting criterion could be based on reaching a prescribed threshold of a total number of active agents.

From agent operation to population behaviour

The algorithmic description of agent operation is insufficient to understand how SDS solves optimisation problems. Therefore, it is necessary to consider what happens with the population as a whole: by iterating through test and diffusion phases individual agents continually explore the entire solution space. Since tests succeed more often in points in the solution space with good objective values, agents spend on average more time examining high-quality solutions, at the same time attracting other agents, which in turn attract even more agents – a mechanism that causes dynamic yet stable clusters of agents to form in certain points in the solution space. However, the limitedness of resources (the finite population size) ensures that only the *best* solution discovered so far is able to maintain a stable cluster of agents. It is this disproportionate allocation of resources that eventually allows the optimal solution to be identified from the largest cluster of agents, without any single agent ever evaluating the full objective function explicitly.

The stochastic process underlying the resource allocation in standard SDS – an ergodic Markov chain – has been thoroughly analysed [36]. The behaviour of the process is determined by probabilities of producing active agents during the test phase. For each candidate solution, these probabilities, averaged over all component functions, form the *test score* of the optimisation problem. The v test score does not only depend on the values of the objective function, but also on the particular test procedure used. Convergence times and average cluster size are functions of population size and the test score [36].

5.2 Previous Work on SDS

SDS was introduced in [3] [4] and subsequently applied to a variety of real-world problems: locating eyes in images of human faces [5]; lip tracking in video films [23]; self-localisation of an autonomous wheelchair [2] and site selection for wireless networks [25]. Furthermore, a neural network model of SDS using spiking neurons has been proposed [37]; [38]. Emergent synchronisation across a large population of neurons in this network can be interpreted as a mechanism of attentional amplification [16]. The analysis of SDS includes the characterisation of its steady state resource allocation [36], the proven convergence to the globally optimal solution [39] and linear time complexity [40].

6 Similarities and Differences between SDS and Social Insects Algorithms

6.1 Comparison with social insects

Contrary to the stigmergetic communication used in most ant algorithms, SDS uses a one-to-one recruitment system akin to the tandem-running behaviour found in certain species of ants. With reference to SDS it is claimed that efficient global decision making can emerge from interaction and communication in a population of individuals each forming hypotheses on the basis of partial evidence.

The recruitment process in real insects is much more complex than that used in SDS where the process of communicating a hypothesis has been completely abstracted. An agent does not have to go through a lengthy and possibly erroneous process of tandem running or waggle dancing to communicate its hypothesis parameters to another agent.

Although no ant or bee species matches exactly the recruitment behaviour of inactive or active agents in SDS, Pratt et al [42] describe the collective decision making strategy of a species of ants that use a similar tandem running recruitment strategy during nest migration. They come to the conclusion that these ants need higher individual cognitive abilities - such as the ability to compare the quality of two nest sites - to come to an optimal solution, as opposed to ants using stigmergetic forms of communication.

Nevertheless, the fundamental similarity between SDS and social insects suggests that global and robust decision making in both types of systems emerges quickly from the co-operation of constituent agents, each of which individually would not be able to solve the problem within the same time frame.

6.2 Comparison with Ant Algorithms

Both SDS and ant algorithms are population-based approaches to search and optimisation that use a form of communication reminiscent of communication in real ants. However, most ant algorithms, and especially the ones described by the ant colony optimisation metaheuristic [19], rely on the idea of stigmergetic communication. Good solutions emerge from temporal and spatial characteristics of the recruitment strategy: short routes receive more pheromones because it takes less time to travel them. In SDS, communication is direct, one-to-one and immediate; solutions do not emerge from temporal aspects of the recruitment system, but merely from the end result of recruitment - the spatial clustering of agents.

Non-stigmergetic ant algorithms have also been proposed. It was shown in [29] that a tandem running recruitment mechanism improves the foraging efficiency of a colony of robots. Further, an optimisation algorithm based on the foraging strategy of a primitive ant species has also been proposed, [34].

This algorithm - called API - alternates between evaluation phases and nest replacement phases. During evaluation, ants explore random points in a certain area around the nest site and remember the best sites. The evaluation phases allow for recruitment between ants: an ant with a better solution can summon an ant with a poorer solution to help it explore its area. However, recruitment on this level did not seem to improve significantly the results obtained. Nest replacement in API can also be considered as a form of recruitment: all the ants are summoned to the optimal point found so far, then start exploring anew. Although on a much slower time scale, the alternation between evaluation and nest replacement in API has similarities with the test and diffusion phases in SDS.

7 Variations on a Theme

Many variations of the standard SDS algorithm are possible: agent updates can occur synchronously for the whole population or asynchronously; the choice of another agent during diffusion can be restricted to agents in a certain neighbourhood or to the whole population; the activity of agents can be binary, integer or even real values, possibly reflecting the history of the agent; during testing, agents can vary the amount of evidence needed for a positive test of a hypothesis. During diffusion, agents can have different reactions to information from other agents, e.g. active agents could choose to communicate and modify their hypothesis according to the state of the contacted agent etc. Some of these modifications have been previously documented [2], [36], [14], [16]. Each of them has a distinct effect on the convergence and steady-state behaviour of the algorithm. However, it can be said that in all cases a dynamical balance between exploration of the solution space and exploitation of discovered solutions naturally emerges.

7.1 Manipulating the Resource Allocation Process

The resource allocation process of SDS can be manipulated in a number of ways by altering properties of the test and diffusion phase [31]. This section focusses on two modifications that are useful for application towards dynamic problems.

Shifting the balance towards local exploration

Standard SDS has no mechanism to exploit *self-similarity* in the objective function – a regularity exhibited by many real-world problems: namely the fact that nearby solutions in the solution space often have similar objective function values [13]. However, a mechanism introducing small variations on the diversity of hypotheses already present in the population can be easily

incorporated into the algorithm. One possibility is to perturb the copying of hypotheses parameters by adding a small random offset during replication of a hypothesis in the diffusion phase, much like mutation in evolutionary algorithms. The effect thereof is to smear out large clusters of agents over neighbouring locations in the solution space. It allows the SDS process to implicitly perform hill-climbing – resulting in improved convergence times in solution spaces with self-similarity [31] – as well as tracking of moving peaks. An example in Sect. 8 will demonstrate the latter point.

Shifting the balance towards global exploration

The conflicting demands of a continued wide exploration of the solution space (especially in dynamic environments), versus the need for a stable cluster exploiting the best solution discovered so far, are not necessarily satisfied in the most optimal way by standard SDS. Its allocation process is very greedy: once a good solution is detected, a large proportion of the population is allocated towards its exploitation, making these agents unavailable for further exploration. A mechanism that frees up part of these resources without severely disrupting the stability of clusters would increase the efficiency of SDS for many classes of problems, including dynamic ones. One such mechanism is *context-sensitive* SDS [36]. The sole difference with standard SDS resides in the diffusion phase for *active* agents: each active agent selects one agent at random; if the selected agent is active and supports the same hypothesis, then the selecting agent becomes inactive and picks a new random hypothesis. This self-regulatory mechanism counteracts the formation of large clusters: the probability that two active agents with the same hypothesis communicate during the diffusion phase increases with relative cluster size. This introduces a mechanism of negative selection or negative feedback to the original algorithm. For certain test scores, it also allows the formation of clusters on multiple similar, near-optimal solutions.

7.2 Standard SDS and stochastic objective functions

Certain types of noise in the objective function may be completely absorbed in the probabilistic nature of the partial evaluation process, and do not influence the search performance of SDS: i.e., they have no effect on convergence times and stability of clusters. More formally, noise that introduces or increases variance in the evaluation of component functions f_i – without altering the averaged probabilities of the *test score* – has no effect on the resource allocation process.

Only when noise changes the values of the test score can the resource allocation process be affected, with a potential for positive as well as negative consequences: a bias which pushes the best test score values more up than poor test score values is likely to accelerate convergence and increase the stability of clusters; conversely, a bias that increases lower test scores more than the

test score of the optimal solution will hamper search performance. In a worst case scenario, the bias could disturb the order of the test score and make SDS converge to a false optimum. However, without any knowledge about the probability distribution generating the noise, no optimisation method would be able to correct such noise. Section 8 presents an example demonstrating the robustness of SDS search performance to moderate amounts of noise.

7.3 Standard SDS and dynamic objective functions

In principle, standard SDS is immediately applicable to dynamically changing objective functions. The probabilistic outcome of the partial evaluation process, in combination with a continued random re-sampling of the solution space, means that the search process can reallocate its resources from a global optimum that has become sub-optimal to the new global optimum. Allocation of resources in standard SDS is *dynamic* and *self-regulatory*; however, it need not be *optimal*: for instance, no variational mechanism for the tracking of slowly moving peaks is present in the original formulation of SDS. However, as section 7.1 demonstrates, such a mechanism is easily included.

8 Examples

In general, synthetic dynamic benchmarks (as introduced in [7, 35]) make no assumptions about the computational costs of function evaluations. In other cases, objective functions that allow cheap function evaluations and that have often been used to benchmark optimisation algorithms in static, noise-free conditions – such as the DeJong test suite – have been adapted to reflect noisy [30] or dynamic [41] conditions. These two approaches do not allow to demonstrate the potential gain in algorithmic efficiency of partial function evaluation. It is therefore necessary to construct an alternative objective function that allows partial evaluation. Such a function can be constructed from the elementary image alignment problem depicted in Fig. 1. Please note that this example is meant as *proof of principle*, rather than an attempt to construct one optimised solution to a specific problem.

The problem consists of locating a small image within a larger image by finding the (x, y) transformation coordinates that produce the best match between the small image and a similar-sized part of the larger image. The small image is taken from another large image which was photographed from a slightly different angle. Sub-pixel sampling is not performed, meaning that the search space is discrete. The size of the solution space – all admissible combinations of x and y values – corresponds to the size of the large photograph, (300 by 860 pixels). The size of the small image is 30 by 40 pixels. The images are RGB colour images, meaning that 3 colour intensity values are available per pixel.

The measure to determine the degree of matching between the two images is the Manhattan distance over all colour intensity values R, G and B:

$$f(x,y) = \sum_{k,l}(|r_{kl} - R_{kl}(x,y)| + |g_{kl} - G_{kl}(x,y)| + |b_{kl} - B_{kl}(x,y)|) \quad (2)$$

Here r_{kl} stands for the red colour intensity value of pixel (k,l) in the small image, and $R_{kl}(x,y)$ for the red colour intensity value of pixel $(x+k, y+l)$ in the large image. The image alignment problem then consists of finding a solution to the problem:

$$\min_{x,y} f(x,y) \quad (3)$$

The motivation for choosing this particular problem is threefold: firstly, the solution space is small enough in size and number of dimensions so that it can be visualised (Fig. 2); secondly, the shape of the resulting landscape is more complex than is easily attainable with artificially constructed benchmark problems; thirdly, the objective function can be *decomposed* into *component functions* f_i (a single term of the summation in (2)) that can be evaluated independently.

Fig. 1. Image alignment problem. The task is to align a small image, taken from another image which was photographed from a slightly different angle, with this large image. The best alignment of the two images is indicated by the black rectangle

The solution space $\mathcal{S}$ is 2-dimensional and discrete, with $x \in \{1, \ldots, 860\}$ and $y \in \{1, \ldots, 300\}$. The size of the solution space is $860 * 300 = 258000$. The number of component functions f_i is determined by the number of terms in the summation of (2) and hence by the size of the small image and the

different colour intensity values: $30 * 40 * 3 = 3600$. Component functions f_i are discrete each with an integer range $[0, 255]$.

Minimisation problem (3) is easily transformed into problem (1); for component i and solution hypothesis (x, y), the test procedure should calculate the quantity:

$$t_i(x, y) = \frac{f_i(x, y)}{255} \tag{4}$$

The test procedure then needs to output 0 with probability $t_i(x, y)$, and 1 with probability $1 - t_i(x, y)$. This procedure ensures that the transformation of objective function values to test score values is strictly order-preserving, a sufficient condition for a correct optimisation of the objective function f [31].

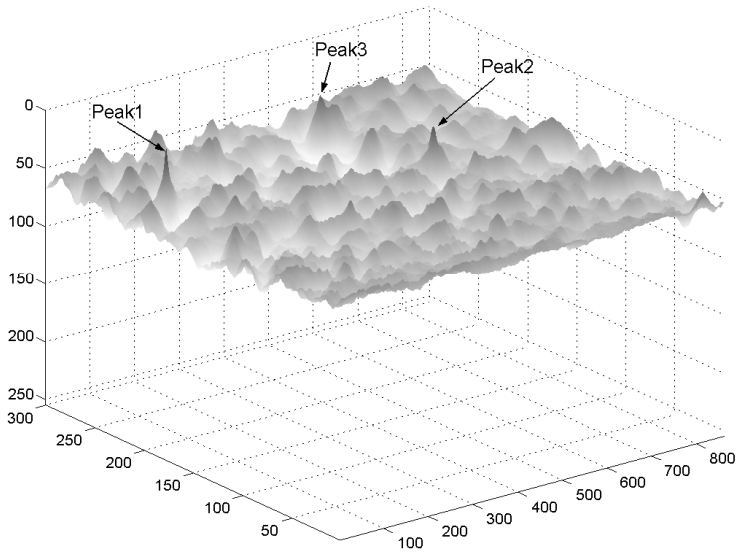

Fig. 2. Objective function generated from the image matching problem in Fig. 1. Peak 1 is the optimal solution. Peak 2 and 3 are of slightly lower quality. Peak 2 and 3 have been manually increased to make the problem more challenging

Characterisation of the search problem

Unlike well-known benchmark problems such as the DeJong test suite or Schaffer's F6 function the structure of this specific problem is not well characterised in terms of its search difficulty. This section provides an empirical assessment of search difficulty by comparing the behaviour of SDS with several common optimisation algorithms on a noise free problem: random search, a multi-start

best-improving hill climber, and a standard implementation of the particle swarm optimisation (PSO) algorithm[5]. The performance of SDS for noisy and dynamic perturbations of the objective function will be discussed in subsequent sections.

Random Search proceeds by choosing a solution at random and evaluating it until the optimal solution of Peak 1 in Fig. 2 is found.

Hill Climber A solution is chosen at random and evaluated. In subsequent iterations, the eight surrounding solutions are evaluated, and the search moves to the solution offering the greatest improvement in objective value. If no such improvement is possible (the search has arrived at a local optimum), then it is restarted in a new, randomly chosen location. These steps are performed until the optimal solution of Peak 1 is discovered.

PSO This algorithm follows the *local constriction* variant of PSO [28]. The algorithm runs until the optimal solution of Peak 1 has been discovered by at least 1 particle. Following parameters have been used: constriction coefficient $\chi = 0.729$; cognitive and social parameters $c1 = c2 = 2.05$; 200 particles with a neighbourhood radius of 1; and $\max V_x = 100$ and $\max V_y = 200$. These parameters have been chosen to give optimal search performance of the PSO algorithm. The reader is referred to [28] for details of the implementation.

SDS A standard SDS algorithm with a population size of 1000 agents has been used. A small mutational mechanism, as described in Sect. 7.1, has also been employed: during copying, the hypothesis is perturbed by adding a randomly generated offset to the (x, y) parameters of the active agent. The offset o_j is generated independently for x and y by:

$$o_j = \left[\frac{r}{s}\right] \tag{5}$$

where r is a normally-distributed random variable with zero mean and standard deviation 1, s is a parameter controlling the standard deviation of o_j, and $\left[\cdot\right]$ denotes rounding to the nearest integer. For this particular experiment, $s = 4$, resulting in an average copying accuracy of 92%, or in a mutation rate of 8%. The search is said to be converged when the optimal solution of Peak 1 has attracted a cluster of 1/3 of all agents. No other parameters need to be defined.

Figure 3 compares the search behaviour of these four algorithms: Figure 3a shows the cumulative distribution of total number of *partial* function evaluations for random search, hill climbing and PSO. Fig. 3b shows the cumulative distribution of total number of *partial* function evaluations

[5]Because noisy and dynamic conditions have led to several alternative PSO formulations that outperform the standard PSO algorithm under these specific conditions, this comparison is performed for noise-free and static conditions only. This will ensure that the characterisation of the search problem difficulty is not biased by the relatively poor performance of the standard PSO under these conditions.

for SDS. The efficiency of partial evaluation can be illustrated by comparing the evaluation cost of SDS with that of the three other algorithms. For example, random search needs around 191000 complete evaluations of (2) to attain a 50% success rate of locating the global optimum, this corresponds to $191000 * 3600 = 687600000$ evaluations of component functions f_i. In contrast, SDS has a median of around 683000 component evaluations, a difference of three orders of magnitude. For comparison, PSO needs 16000 full function evaluations and hill climbing 4000.

However, rather than just comparing these numbers, it is interesting to see from how many component functions f_i onwards SDS starts to outperform the other algorithms. The probabilistic, partial evaluation mechanisms in SDS transforms the search into a stochastic dynamical process that is independent of the number of component functions in the objective function, and only depends on the exact shape of the landscape. In other words, whether the landscape of Fig. 2 is generated by a function consisting of 100, 1000 or 10000 component functions, the averaged search behaviour of SDS is always the same. For this particular landscape, SDS would outperform random search for objective functions consisting of $683000/191000 \approx 4$ or more component functions f_i. For PSO this number becomes $683000/16000 \approx 43$, and for hill climbing $683000/4000 \approx 171$. The relatively poor performance of PSO compared to the hill climber can be explained by the fact that the swarm consisted of 200 particles each performing full function evaluations. It is likely that the performance of PSO relative to hill climber would improve if, for example, the dimensionality of the problem were increased.

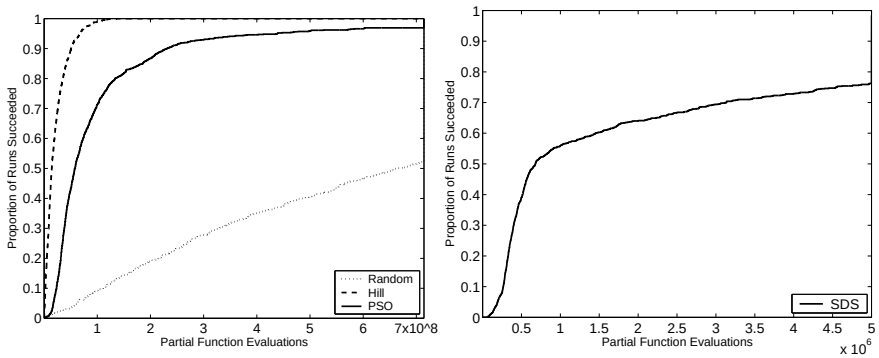

Fig. 3. Comparison of random search, hill climbing, PSO and SDS on the search problem of Fig. 2. Results are averaged over 1000 runs for each algorithm

The effect of noisy perturbations

To illustrate that SDS is relatively immune to certain types of noise, the following experiment was conducted: during every evaluation of a particular f_i, the outcome of the evaluation is perturbed with normally distributed noise with zero mean value and different levels of standard deviation: 5%, 10% and 20% of the actual function value of $f_i(x, y)$. The parameter settings for the standard SDS algorithm were the same as for the previous experiment. The cumulative distribution of convergence times is reported in Fig. 4. It can be seen that there is hardly any effect of the noise levels on the cluster formation process. Increasing the noise levels even further, beyond 20%, introduces a negative bias into the test score. Because of the non-linear properties of the SDS process, the effect on this particular landscape is to accelerate the search. However, this is not necessarily so for all objective functions. A detailed discussion of the ramifications of using such mechanisms to improve search performance is beyond the scope of this paper.

Moving peaks

To illustrate that a cluster of agents is able to track a moving peak, the following experiment was performed: the entire objective function of Fig. 2 was shifted one location to the left and one location to the front every 50 iterations of an SDS simulation. A population of 1000 context-sensitive agents with mutation parameter $s = 2$ (resulting in a copying accuracy of 47%, or a mutation rate of 53%) was run for 10000 iterations. Figure 5 summarises the results: the left graph depicts the total number of active agents (higher curve) and the size of the cluster at the location of the moving Peak 1 (lower curve). The right graph depicts the location of the *largest cluster* of agents every 50 iterations, just before a new shift of the objective function. The results show that the largest cluster of agents follows the movement of Peak 1 almost perfectly. 50 iterations of 1000 agents constitute 50000 evaluations of component functions, equivalent to only 14 evaluations of (2).

Changing peaks

To illustrate that a cluster of SDS agents can reallocate itself successfully when optimal solutions become sub-optimal, the following experiment was performed: 1000 context-sensitive SDS agents[6] were simulated for 5000 iterations, while peaks in the landscape were slowly decreased or increased. The results of this experiment can be seen in Fig. 6. After 1000 iterations, Peak 1 starts to deteriorate, with as consequence a gradually decreasing cluster size at that location. Peak 2 remains constant, while Peak 3 grows gradually in

[6]With the same perturbation of hypothesis-copying as in the previous experiment.

height and width. Shortly after Peak 1 becomes lower than Peak 2, there is a sudden shift of the dominant cluster towards the location of Peak 2. When Peak 3 grows larger than Peak 2, a similar shift occurs to the location of Peak 3.

9 Discussion

Pratt [42] suggests that Leptothorax Alpipennis require extra cognitive abilities in order to efficiently compare different nest sites. Although it could be that these ants need higher cognitive abilities because the exact dynamics of their recruitment process do not allow convergence on the best site in a fast enough time span, experience with SDS shows that these abilities are *in principle* not required. As long as one of the two nest sites has a higher probability of inducing recruitment, ants can come to a global decision about the best site without the ability of comparing the two sites directly.

Differences in the operation of SDS and the bulk of ant algorithms has resulted in their application in different types of search and optimisation problems. In Mitchell [32], a taxonomy of search problems has been proposed:

- Pattern matching problems, in which the goal is to locate a predefined target in a larger solution space.
- Optimisation problems, in which the goal is to select a solution from a set of candidates such that a given cost function is optimised.
- Path planning problems, in which the goal is to construct a path to reach a specified target.

Whereas SDS in its present form seems mostly applicable to the first type of search problems, ant algorithms have mostly been used for solving the second type. As such, both approaches seem complementary. However, the general principles behind SDS can clearly be applied to other problem classes. These are the principles of partial evaluation of candidate solutions and direct communication of information between agents. Using these principles, SDS can be defined as a new generic search method or *metaheuristic*, applicable to other types of problems outside the pattern-matching domain, such as model fitting; robust parameter estimation; and Inductive Logic Programming. Research in these areas is ongoing.

9.1 SDS and evolutionary algorithms

At first sight, the SDS algorithm, described in a language of agents, test and diffusion phases, may seem far removed from evolutionary algorithms. Indeed, it did not originate from metaphors about biological evolution, but from the field of neural networks [4]. However, SDS and algorithms inspired by Darwinian evolution fit both within a general framework of processes that are

governed by mechanisms of variation, selection and replication [11]. For SDS, this description applies to the perspective of the hypotheses: randomly picking new hypotheses and perturbing the copying process constitute mechanisms of variation, similar to random immigrants and mutation mechanisms in evolutionary algorithms; the rejection of hypotheses in the diffusion phase is a form of "death" for the hypotheses; hypothesis copying is a form of reproduction. Good hypotheses are more likely to survive test phases for longer, and are able to spread more to other agents. Finally, resources are limited, in that there is only a finite number of agents which hypotheses can occupy.

There are, of course, differences. Firstly, there is no explicit fitness-based selection: selection is the consequence of agent interaction, resulting in the most radical form of tournament selection. Secondly, because of the indirect and continual evaluation of individual hypotheses, SDS can be thought to simulate evolutionary processes *on a different timescale* than other types of evolutionary algorithms. Thirdly, because single agents lack the capacities to judge the quality of solutions on their own, good solutions need to be identified by clusters of agents. This means that SDS explicitly needs at least some level of convergence, whereas this is not necessarily true for other evolutionary algorithms.

10 Conclusions

It has been shown that SDS is in principle applicable to stochastic and dynamic optimisation problems. The algorithmic concepts of partial evaluation and mechanisms for altering the balance between exploration and exploitation – together with a well-developed understanding of how these influence the behaviour of the stochastic process underlying SDS – can be of potential interest to the swarm intelligence community at large. Although SDS has been applied to different types of optimisation problems, e.g., [2, 25], it has never before been applied explicitly to stochastic or dynamic optimisation problems. To this end, the present work draws on the expanded understanding of SDS developed in [31].

Future work should include a more precise characterisation of the influence of external noise on the search performance, in the context of the mathematical models of SDS developed in [36], as well as methods to estimate for specific types of problems how much – if anything – can be gained in computational efficiency from partial evaluation. Hybridisation with explicit hill-climbing strategies – already employed in [22] – and multi-population implementations – as described for evolutionary algorithms in [8] – may prove to be invaluable extensions to the simple methods presented here. Finally, a better understanding of the more complex, *focussed* SDS mechanisms, as employed in [2, 25], can render SDS useful for stochastic and dynamic problems of much larger scale than the ones described here.

References

1. Arthur, W B,(1994) Inductive Reasoning and Bounded Rationality (The El Farol Problem). Amer. Econ. Rev. Papers and Proceedings 84: 406
2. Beattie, P, Bishop, J (1998) Self-localisation in the SENARIO autonomous wheelchair. Journal of Intelligent and Robotic Systems 22: 255–267
3. Bishop, J M (1989) Anarchic Techniques for Pattern Classification. Chapter 5. PhD Thesis, University of Reading
4. Bishop, J (1989) Stochastic searching networks. In: 1st IEE Conf. ANNs, 329331 London
5. Bishop, J M, Torr, P (1992) The Stochastic Search Network. In: Lingard, R, Myers, D J, Nightingale, C Neural Networks for Images, Speech and Natural Language. Chapman and Hall, New York, 370387
6. Bonabeau, E, Dorigo, M, Theraulaz, G (2000) Inspiration for Optimization from Social Insect Behaviour. Nature 406: 3942
7. Branke, J (1999) Memory-enhanced evolutionary algorithms for dynamic optimization problems. In: Congress on Evolutionary Computation. Volume 3., IEEE 1875–1882
8. Branke, J, Kauler, T, Schmidt, C, Schmeck, H (2000) A multi-population approach to dynamic optimization problems. In Parmee, I., ed.: Adaptive Computing in Design and Manufacture, Springer 299–308
9. Branke, J, Schmidt, C, Schmeck, H (2001) Efficient fitness estimation in noisy environments. In Spector, L., ed.: Genetic and Evolutionary Computation Conference, Morgan Kaufmann 243–250
10. Branke, J (2003) Evolutionary approaches to dynamic optimization problems – introduction and recent trends. In: Branke, J, ed. Proceedings of EvoDOP
11. Campbell, D (1974) Evolutionary epistemology. In Schilpp, P, ed. The Philosophy of Karl Popper. Open Court 413–463
12. Chadab, R, Rettenmeyer, C (1975) Mass Recruitment by Army Ants. Science 188:11241125
13. Christensen, S, Oppacher, F (2001) What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function. In: Spector et al., L, ed. Genetic and Evolutionary Computation Conference, San Fransisco, Morgan Kaufmann 1219–1226
14. De Meyer, K (2000) Explorations in Stochastic Diffusion Search: Soft- and Hardware Implementations of Biologically Inspired Spiking Neuron Stochastic Diffusion Networks, *Technical Report KDM/JMB/2000/1*, University of Reading
15. De Meyer, K, Bishop, J M, Nasuto, S J (2002) Small-World Effects in Lattice Stochastic Diffusion Search, Proc ICANN2002 Madrid, Spain
16. De Meyer, K, Bishop, J M, Nasuto S J (2000) Attention through Self-Synchronisation in the Spiking Neuron Stochastic Diffusion Network. Consciousness and Cognition 9(2)
17. Deneuborg, J L, Pasteels, J M, Verhaeghe, J C (1983) Probabilistic Behaviour in Ants: a Strategy of Errors? Journal of Theoretical Biology 105:259271
18. Digalakis, J, Margaritis, K (2002) An experimental study of benchmarking functions for evolutionary algorithms. International Journal of Computer Mathemathics 79:403–416
19. Dorigo, M, Di Caro, G, Gambardella, L M (1999) Ant Algorithms for Discrete Optimization. Artificial Life 5(2):137172

20. Garey, M R, Johnson, D S (1979) Computers and Intractability: a guide to the theory of NP-completeness. W. H. Freeman
21. Goodman, L J, Fisher, R C (1991) The Behaviour and Physiology of Bees, CAB International, Oxon, UK
22. Grech-Cini, E, McKee, G (1993) Locating the mouth region in images of human faces. In: Schenker, P, ed. SPIE - The International Society for Optical Engineering, Sensor Fusion VI 2059, Massachusetts
23. Grech-Cini, E (1995) Locating Facial Features. PhD Thesis, University of Reading
24. Holldobler, B, Wilson, E O (1990) The Ants. Springer-Verlag
25. Hurley, S, Whitaker, R (2002) An agent based approach to site selection for wireless networks. In: ACM symposium on Applied Computing, Madrid, ACM Press
26. Jin, Y (2005) A comprehensive survey of fitness approximation in evolutionary computation. In: Soft Computing, 9:3–12.
27. El-Beltagy, M A, Keane, A J (2001) Evolutionary optimization for computationally expensive problems using Gaussian processes. In: Arabnia, H, ed. Proc. Int. Conf. on Artificial Intelligence'01, CSREA Press 708–714
28. Kennedy, J, Eberhart, R C (2001) Swarm Intelligence. Morgan Kaufmann
29. Krieger, M J B , Billeter, J-B, Keller, L (2000) Ant-like Task Allocation and Recruitment in Cooperative Robots. Nature 406:992995
30. Krink, T, Filipic, B, Fogel, G B, Thomsen, R (2004) Noisy Optimization Problems – A Particular Challenge for Differential Evolution? In: Proc. of 2004 Congress on Evolutionary Computation, IEEE Press 332–339
31. De Meyer, K (2003) Foundations of Stochastic Diffusion Search. PhD thesis, University of Reading
32. Mitchell, M (1998) An Introduction to Genetic Algorithms. The MIT Press
33. Moglich M, Maschwitz U, Holldobler B (1974) Tandem calling: a new kind of signal in ant communication. Science 186(4168):1046-7
34. Monmarch, N, Venturini, G, Slimane, M (2000) On How Pachycondyla Apicalis Ants Suggest a New Search Algorithm. Future Generation Computer Systems 16:937-946
35. Morrison, R W, DeJong, K A (1999) A test problem generator for non-stationary environments. In: Congress on Evolutionary Computation. Volume 3., IEEE 2047–2053
36. Nasuto, S J (1999) Resource Allocation Analysis of the Stochastic Diffusion Search. PhD Thesis, University of Reading
37. Nasuto, S J, Bishop, J M (1998) Neural Stochastic Diffusion Search Network - a Theoretical Solution to the Binding Problem. Proc. ASSC2, Bremen
38. Nasuto, S J, Dautenhahn, K, Bishop, J M (1999) Communication as an Emergent Methaphor for Neuronal Operation. Lect. Notes Art. Int. 1562:365380
39. Nasuto, S J, Bishop, J M (1999) Convergence Analysis of Stochastic Diffusion Search. Parallel Algorithms and Applications 14(2):89107
40. Nasuto, S J, Bishop, J M, Lauria, S (1998) Time Complexity of Stochastic Diffusion Search. Neural Computation (NC98), Vienna, Austria
41. Parsopoulos, K E, Vrahatis, M N, (2005) Unified Particle Swarm Optimization in Dynamic Environments, Lect. Notes Comp. Sci. 3449:590-599
42. Pratt, S C, Mallon, E B, Sumpter, D J T, Franks, N R (2000) Collective Decision- Making in a Small Society: How the Ant Leptothorax Alpipennis Chooses a Nest Site. Proc. of ANTS2000, Brussels, Belgium

43. Seeley, T D (1995) The Wisdom of the Hive. Harvard University Press
44. Whitley, D, Rana, S B, Dzubera, J, Mathias, K E (1996) Evaluating evolutionary algorithms. Artificial Intelligence 85:245–276

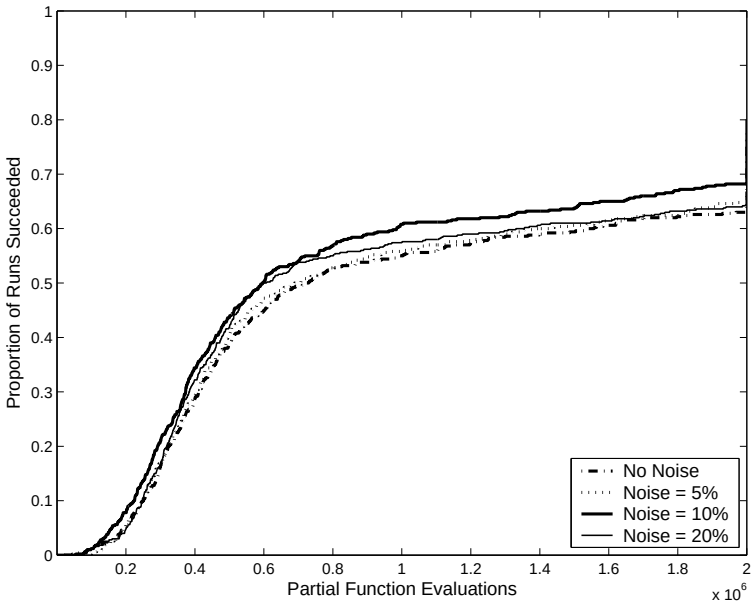

Fig. 4. Influence of noise on the cumulative distribution of convergence times. Results are averaged over 1000 runs for each of the noise levels

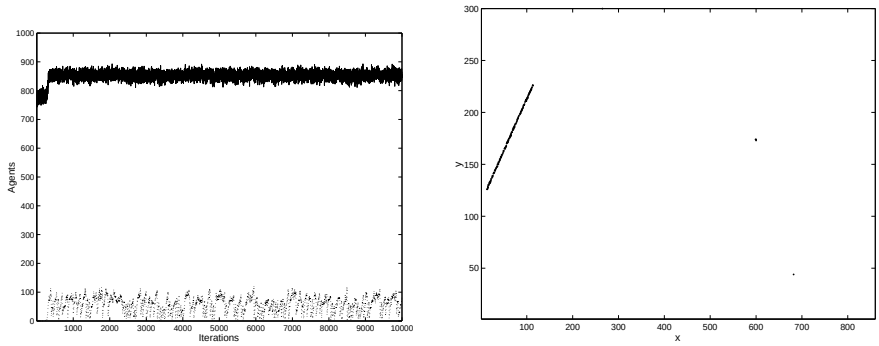

Fig. 5. Tracking of a moving peak. The left graph depicts overall activity and the size of the cluster at the moving location of Peak 1. The right graph depicts the (x, y) location of the largest agent cluster every 50 iterations

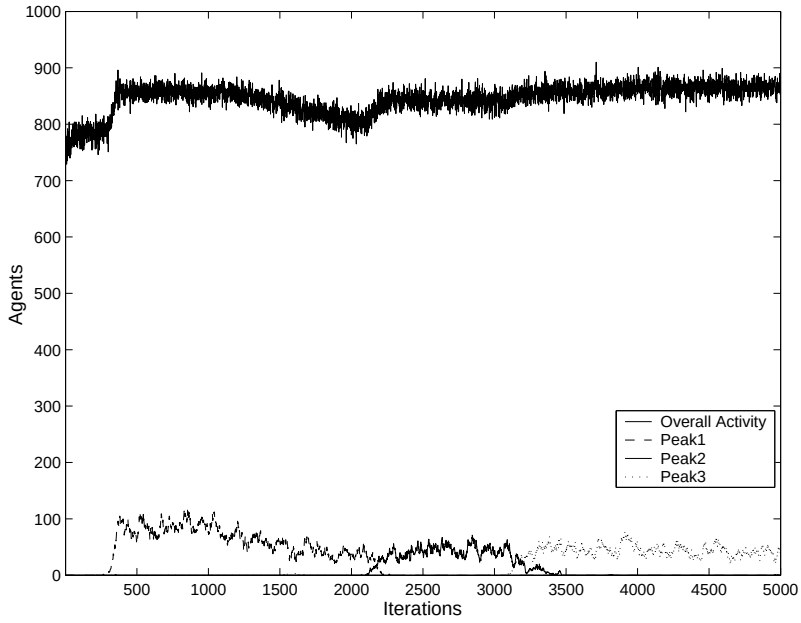

Fig. 6. Changing peaks. Depicted are the overall activity in the population and the cluster sizes at Peak 1, 2 and 3, for a population of 1000 agents run for 5000 iterations. After 1000 iterations, Peak 1 starts slowly decreasing in height and width, while Peak 3 starts slowly increasing. Peak 2 remains the same throughout the experiment. At iteration 1700, Peak 1 becomes lower than Peak 2. At iteration 2300, Peak 3 becomes higher than Peak 2, and keeps growing until iteration 4000. The height of the peaks changes very slowly: e.g. for Peak 1 only 0.3% of the function range every 100 iterations. However, even these subtle changes are reflected in the resource allocation of the agent population

Index