Stochastic Diffusion Search for Continuous Global Optimization

Mahamed G. H. Omran[1], Imad Moukadem[1], Salah al-Sharhan[1], Mariam Kinawi[1]

[1] Department of Computer Science
Gulf University for Science & Technology,
P.O. Box:7207, Hawally 32093, Kuwait
Phone: +965-2530-7433
Fax: +965-2530-7030
Email: {omran.m, moukadem.I, alsharhans, kinawi.m}@gust.edu.kw

Abstract

Stochastic Diffusion Search (SDS) is a multi-agent, naturally inspired search and optimization algorithm that is based on direct (one-to-one) communication between agents. SDS has been successfully applied to a wide range of optimization problems. In this paper, SDS is used to tackle the continuous nonlinear function optimization problem. The proposed method is tested on a mini-benchmark of four problems with promising results. The results show that SDS can also be used as an intelligent way to choose a starting point for a local search method.

Keywords:

Stochastic diffusion search, swarm intelligence, continuous nonlinear function optimization, metaheuristics, local search.

Stochastic Diffusion Search for Continuous Global Optimization

Mahamed G. H. Omran[1], Imad Moukadem[1], Salah al-Sharhan[1], Mariam Kinawi[1]

[1] Department of Computer Science
Gulf University for Science & Technology,
P.O. Box:7207, Hawally 32093, Kuwait
Email: {omran.m, moukadem.I, alsharhans, kinawi.m}@gust.edu.kw

Abstract

Stochastic Diffusion Search (SDS) is a multi-agent, naturally inspired search and optimization algorithm that is based on direct (one-to-one) communication between agents. SDS has been successfully applied to a wide range of optimization problems. In this paper, SDS is used to tackle the continuous nonlinear function optimization problem. The proposed method is tested on a mini-benchmark of four problems with promising results. The results show that SDS can also be used as an intelligent way to choose a starting point for a local search method.

Keywords:

Stochastic diffusion search, swarm intelligence, continuous nonlinear function optimization, metaheuristics, local search.

1 Introduction

Swarm Intelligence (SI) is the collective behavior of decentralized and self-organized systems. SI systems are typically made up of a population of agents or particles interacting with each other and with their environment. Interaction between agents yields collective intelligent behavior. Typical SI algorithms are ant colony optimization (ACO), particle swarm optimization (PSO) and stochastic diffusion search (SDS).

Stochastic Diffusion Search [1] is an efficient resource allocation algorithm that has been applied to a wide range of applications [3] However, there is little or no work on using SDS to solve continuous nonlinear function optimization defined as follows,

$$\text{(P) Minimize } f(\mathbf{x}) : \mathbf{L} \leq \mathbf{x} \leq \mathbf{U} \qquad \mathbf{(1)}$$

The vector $\mathbf{x} = (x_1, \ldots, x_D)$ is composed of real-valued variables, and the vectors $\mathbf{L}$ and $\mathbf{U}$ are assumed finite and to satisfy $\mathbf{L} < \mathbf{U}$. The function $f(\mathbf{x})$ is typically multimodal, so that local optima do not in general correspond to global optima. In this paper, we investigate the use of SDS in this relatively unexplored area.

The remainder of the paper is organized as follows: Section 2 provides an overview of SDS. The proposed approach is presented in Section 3. Results of the experiments are presented and discussed in Section 4. Finally, Section 5 concludes the paper.

2 Stochastic Diffusion Search (SDS)

Stochastic Diffusion Search (SDS) [1] is a population-based, naturally inspired optimization algorithm that is based on direct (one-to-one) communication between agents.

The SDS algorithm consists of two phases, test phase and diffusion phase. In the test phase, each agent tests its hypothesis (a potential solution to the problem). In the diffusion phase, agents share information about hypotheses via one-to-one communication.

SDS can be described using the Restaurant Game [3]:

"A group of delegates attends a long conference in an unfamiliar town. Every night each delegate must 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."

SDS convergence to global optimum has already been studied by [8] and its time complexity has been analyzed by [7]. SDS has been successfully applied to a wide range of optimization problems (e.g. object recognition[2], site selection for wireless networks [9], amongst others [3]).

3 The Proposed Method

In this paper, we propose a new variant of SDS that can be used as a continuous global optimizer. In the proposed approach, each agent has hypothesis and status. Hypotheses are defined as candidate solutions to a problem while status is a Boolean variable used to distinguish between active and inactive agents.

The general algorithm of the proposed method is shown in Alg. 1. The algorithm starts by initializing a population of agents. The fitness of each agent is then determined. In the test phase, each agent is compared against another randomly chosen agent. If it has a better fitness function than the selected agent, it is set as *active*, otherwise it is set as *inactive*.

In the diffusion phase, if an agent, $\mathbf{x}_i$, is inactive, a random agent, $\mathbf{x}_j$, is chosen. If $\mathbf{x}_j$ is active, it communicates its hypothesis to $\mathbf{x}_i$. This could be done in many ways. In this paper, $\mathbf{x}_i$ is set to a solution in the neighborhood of $\mathbf{x}_j$. The procedure is described below.

A region around $\mathbf{x}_j$ is defined such that $\mathbf{L}_o \leq \mathbf{x}_j \leq \mathbf{U}_0$, where $\mathbf{L}_0$ and $\mathbf{U}_0$ are defined as follows:
$\mathbf{L}_0 = \mathbf{x}_j - 0.5w(\mathbf{U} - \mathbf{L})$ and $\mathbf{U}_0 = \mathbf{x}_j + 0.5w(\mathbf{U} - \mathbf{L})$
Where w is a user-specified parameter between 0 and 1 (a value of 0.1 is used in this study). The following correction is used if any component of $\mathbf{L}_0$ and $\mathbf{U}_0$ lies outside its bounds.

```
if $L_{od} < L_d$ then
        $U_{od} = U_{od} + (L_d - L_{od})$

        $L_{od} = L_d$

elseif $U_{od} > U_d$ then
        $L_{od} = L_{od} - (U_{od} - U_d)$
        $U_{od} = U_d$
endif
```

Where $d \in D$.

On the other hand, if the selected agent, $\mathbf{x}_j$, is inactive, $\mathbf{x}_i$ is re-initialized. Re-initialization can be done at random or in a more intelligent way. In this paper, we used the method proposed by [4]. The method finds the largest uncharted area and put $\mathbf{x}_i$ on the middle of that area. This is done by sorting the coordinates of the population for each dimension. The biggest interval is then chosen. The new coordinate is set as the middle of this interval.

In short, during the diffusion phase, an *intensification* process is applied around an active (i.e. good) agent while a *diversification* process is used if both agents are inactive. A good balance between intensification and diversification is needed for any search method and this balance is achieved by our SDS.

After a fixed number of function evaluations (FEs), SDS terminates and the best solution is improved using a local search method. In this paper, the Enhanced Unidimensional Search (EUS) algorithm [6] is used as our local search method (other methods could also be used). Based on preliminary tests, the EUS' *ratio* parameter is to 0.9 in our experiments.

```
Initialize agents and calculate their fitness

Repeat

  /* Test-phase */
    for each agent, $\mathbf{x}_i$
    randomly choose an agent, $\mathbf{x}_j$, from the population
    if $f(\mathbf{x}_i) \leq f(\mathbf{x}_j)$ then
      $active_i$ = true
    else
      $active_i$ = false
    endif
    endfor

  /* Diffusion-phase */
  for each agent, $\mathbf{x}_i$
    if $active_i$ == false then
      randomly choose an agent, $\mathbf{x}_j$, from the population
      if $active_j$ == true then
        set $\mathbf{x}_i$ to a solution randomly chosen from the neighborhood of $\mathbf{x}_j$
      else
        reinitialize $\mathbf{x}_i$ to a solution in an uncharted region
      endif
    endif
  endfor

Until a stopping criterion is met

Let $\mathbf{x}_g$ be the agent with the best (i.e. minimum) fitness function
Improve $\mathbf{x}_g$ using a local search method
```

Alg. 1: The proposed SDS algorithm

4 Experimental Results

To illustrate the difference between SDS and EUS, we consider the results on a mini-benchmark of four problems (see Table 1) used by [4]. Although this test bed is small, it is very representative. The first function, Tripod, is very deceptive. The second and third functions, Compression spring and Gear train, are quasi real-world problems that are moderately multimodal. The fourth function, CEC2005 Rosenbrock F6, is a difficult multimodal function. Experiments were conducted with an initial population of 50 agents. EUS was launched starting from the best of these agents. The rationale behind this comparison is to show that SDS improves the performance of EUS and to show that SDS can be used with any local search method as an intelligent way to choose their initial solution.

	Search space	Required accuracy	Max. number of FEs
Tripod	$[-100,100]^2$	0.0001	10,000
Compression spring	3 variables objective 2.625421 (granularity 0.001 for x_3)	10^{-10}	20,000
Gear train	4 variables	10^{-13}	20,000
Rosenbrock F6	$[-100,100]^{10}$	0.01	100,000

Table 1: The mini-benchmark.

The results reported in this section are averages and standard deviations over 100 simulations. In order to conduct a fair comparison, the same number of function evaluations was used for both EUS and the proposed algorithm. For the proposed method, half the number of function evaluations was used for SDS and the other half was used for EUS.

All the tests are run on an Apple MacBook computer with Intel Core Due 2 processor running at 2.0 GHz with 2GB of RAM. Mac OS X 10.5.6 is the operating system used. All programs are implemented using MATLAB version 7.6.0.324 (R2008a) environment.
The two methods were compared based on the following criteria:

1) *Average gap*, Avg. GAP, where the optimality gap for a given solution **x** and the best solution **x*** is defined as:

$$GAP = |f(\mathbf{x}) - f(\mathbf{x}^*)| \tag{2}$$

2) *Success rate (SR)* which is the percentage of trials where a method converges with the required accuracy.

3) *Execution time* (measured in seconds).

Table 2 shows the results of comparing SDS with EUS on the mini-benchmark problems. The statistically significant best solutions have been shown in **bold** (using the non-parametric statistical test called Wilcoxon's rank sum test for independent samples [5] with $\alpha = 0.05$). In terms of GAP and SR, the results show that SDS outperforms EUS on all the problems. The execution time of both methods is generally the same. Hence, the results indicate that using SDS improves the performance of EUS by choosing a good starting point for EUS. Table 3 compares the performance of SDS with the standard PSO 2007 (SPSO-2007) as reported by [4]. The results show that SDS outperforms SPSO on Tripod and Gear train while SPSO outperforms SDS on Compression spring and Rosenbrock F6.

	EUS			SDS		
f	*Avg. GAP*	*SR*	*Exec. Time(s)*	*Avg. GAP*	*SR*	*Exec. Time(s)*
Tripod	1.12e+00 (7.15e-01)	20%	6.00e-01(2.71e-01)	**5.40e-01 (6.10e-01)**	52%	5.84e-01(1.44e-01)
Compression spring	2.00e+04 (1.41e+05)	0%	1.66e+00(8.30e-02)	**3.35e-02 (1.93e-02)**	5%	1.66e+00(1.67e-01)
Gear train	9.45e-07 (8.07e-07)	0%	1.47e+00(1.19e-01)	**3.72e-10 (6.85e-10)**	13%	1.45e+00(2.48e-01)
Rosenbrock F6	3.83e+02 (3.32e+02)	4%	3.92e+01(1.23e+01)	**7.58e+01 (1.75e+02)**	24%	4.12e+01(1.22e+01)

Table 2. Comparison between EUS and SDS over 100 runs.

f	SPSO-2007	SDS
Tripod	50%	52%
Compression spring	35%	5%
Gear train	6%	13%
Rosenbrock F6	82%	24%

Table 3. Success rates over 100 runs.

5 Conclusions and Future Work

In this paper, a new variant of SDS that could be used to solve continuous nonlinear function optimization problems was proposed. The proposed method was compared with two methods on four benchmark problems with encouraging results. This study showed that SDS could also be used as an intelligent way to choose a good starting point for a local search method.

Future work will investigate the performance of the proposed method on more problems (including real-world problems). Alternative intensification and diversification strategiesq will also be investigated.

Acknowledgment

The intensification process used during the diffusion phase of the proposed approach was suggested by Prof. Fred Glover in a different study with the first author.

References

[1] Bishop, J.M., (1989). Stochastic Searching Networks. Proc. 1st IEE Conf. on Artificial Neural Networks, pp 329–331, London.

[2] Bishop, J.M. & Torr, P., (1992). The Stochastic Search Network. In R. Linggard, D.J. Myers, C. Nightingale (eds.), Neural Networks for Images, Speech and Natural Language, pp370–387, New York, Chapman & Hall.

[3] CIRG (2011). Stochastic Diffusion Search. http://www.reading.ac.uk/cirg/sds/cirg-stochasticdiffusionsearch.aspx (access date: 4-Mar-2011).

[4] Clerc, M. (2010). Beyond Standard Particle Swarm Optimization. International Journal of Swarm Intelligence Research, vol. 1(4), pp. 46-61.

[5] F. Wilcoxon (1945). Individual comparisons by ranking methods. Biometrics, vol. 1, pp. 80-83.

[6] Gardeux, V., Chelouah, R., Siarry, P. & Glover, F., (2009). Unidimensional search for solving continuous high-dimensional optimization problems. In the proceedings of the 2009 9th International Conference on Intelligent Systems Design and Application, Pisa, Italy, pp. 1096-1101.

[7] Nasuto, S.J., Bishop, J.M. & Lauria, L., (1998). Time Complexity of Stochastic Diffusion Search. Neural Computation '98, Vienna, Austria.

[8] Nasuto, S.J. & Bishop, J.M., (1999). Convergence Analysis of Stochastic Diffusion Search. Journal of Parallel Algorithms and Applications 14:2, pp 89–107.

[9] Whitaker, R.M., Hurley, S., (2002). An agent based approach to site selection for wireless networks. Proc ACM Symposium on Applied Computing (Madrid). 574–577.