# An Ant Colony Optimization and Nelder-Mead Simplex Search Hybrid Algorithm for Unconstrained Optimization

N. Arun and V. Ravi
*Institute for Development and Research in Banking Technology, Castle Hills Road #1, Masab Tank, Hyderabad, INDIA*
*narun@mtech.idrbt.ac.in ; rav_padma@yahoo.com*

## Abstract

*In this paper we hybridize $ACO_R$, which is a meta-heuristic for optimization of continuous functions based on the Ant Colony Optimization framework with Nelder-Mead Simplex Search which is a direct search technique in a novel way and test its performance. In this study we use $ACO_R$ to do bulk of the search and use simplex search once the region of the global optimum is reached. This hybrid algorithm is able to converge faster to the global optimum for several test functions. The tuning of the parameters of the algorithm is also discussed.*

## 1. Introduction

In the recent past, meta-heuristics have been extensively employed to solve nonlinear programming problems (NLP). The reason for this popularity is that unlike the classical methods they can work in cases where (i) derivative information is not available (ii) the function being optimized has local optima (iii) the search space is non-convex. Whenever meta-heuristics are employed there is a trade-off between "exploration", which is the process of identifying promising regions in search space and "exploitation", which is the process of using the promising region to obtain solutions. Population based search techniques are good at exploration but once they identify a promising region of search space they take too long to converge to the optimum. Hence several hybrid meta-heuristics have been proposed which combine the strengths of meta-heuristics and local search techniques to achieve a good balance between exploration and exploitation. In the following, we first present a brief review of hybrid optimization algorithms not involving Ant Colony Optimization (ACO) and then a review of hybrid optimization algorithms involving ACO.

In the literature, we find several optimization algorithms which are hybrids of meta-heuristics and local search techniques. In [1] INESA is presented, which is a hybrid of Non-Equilibrium Simulated Annealing (NESA) and a simplex-like heuristic. This hybrid was applied to solve reliability optimization problems in complex systems. In [2] a hybrid of Genetic Algorithms (GA) and Nelder-Mead Simplex Search (NM) is presented. This hybrid was applied to solve benchmark test problems. In [3] a hybrid of Simulated Annealing (SA), Tabu Search (TS) and NM is presented. This hybrid has the property of indentifying the global minimum together with some local minima, which may be of interest. It was tested on benchmark problems. A hybrid of TS and NM is presented in [4]. It was tested on benchmark test problems and also on a problem involving the design of eddy current sensor for non-destructive control. In [5] a hybrid of Differential Evolution (DE) and a simplex like heuristic, which comprised only the reflection property of NM, is presented. Here, the NM part of the hybrid is invoked after DE identifies a promising region of search space. It was applied to solve a parameter estimation problem arising in biochemical engineering. A hybrid of Particle Swarm Optimization (PSO) and NM is presented in [6]. The idea consists of allowing NM to work on the best points in the population of PSO to quickly arrive at the global optimum.

In the ACO based hybrid algorithms we have Hybrid Continuous Interacting Ant Colony (HCIAC) [7], which is based on Continuous Interacting Ant Colony (CIAC) [8]. The CIAC algorithm introduces the notion of "heterarchy", which is a form of direct communication between the ants. Apart from using the pheromone trail, which acts as a form of indirect communication, the direct communication between ants in the form of "heterarchy" is also used in the algorithm. The HCIAC is a hybrid of CIAC and NM. Dynamic Hybrid Interacting Ant Colony (DHCIAC) is

presented in [9], which is aimed at optimization of dynamic functions instead of static ones.

Although CIAC and its variants have hybridized ACO and NM, the CIAC algorithm itself is a modified form of ACO metaphor, since it uses the notion of "heterarchy". $ACO_R$ [10], on the other hand is an elegant extension of the ACO metaphor to the realm of continuous optimization. It makes no major structural changes to ACO and bases its search function purely on the pheromone trail values. Hence we are motivated to try out the hybridization of $ACO_R$ with the fast direct search capability of NM.

## 2. Description of algorithms

The proposed algorithm is a hybrid of $ACO_R$ [10] and Nelder Mead Simplex Search [11]. We describe below, both the algorithms and then give the description of the proposed hybrid algorithm.

### 2.1. $ACO_R$ algorithm

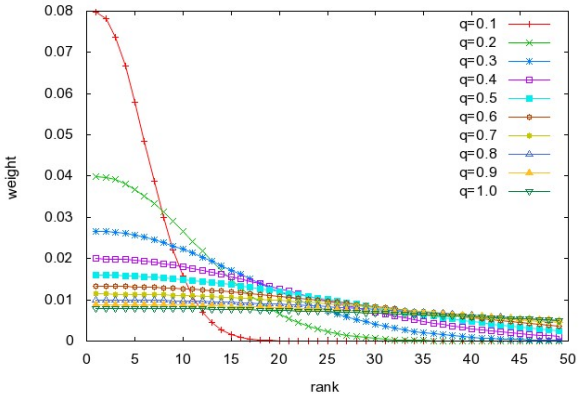
The $ACO_R$ algorithm for continuous optimization was propsed by Socha and Dorigo [10]. It is an extension of the original ACO algorithm [12] meant for combinatorial optimization. In ACO for combinatorial optimization the solution is constructed as a sequence of solution components. The number of solution components is finite. Each solution component has associated with it, a "pheromone" value which indicates the "goodness" of the solutions. Better solutions have higher pheromone values associated with them. The solution construction proceeds in a probabilistic fashion using the pheromone values. However, in continuous optimization, the problem is that the number of solution components is infinite and therefore a different approach was needed. The $ACO_R$ algorithm makes use of a solution archive (a set of solutions) to construct probability density functions, which model the fitness of the solutions in various regions of the search space. The solution construction is done by sampling these probability density functions. We use the notation in [10] and briefly describe the algorithm below. For a detailed explanation of the algorithm, the reader is referred to [10].

The algorithm makes use of a set of solutions called an archive $T$, which contains $k$ solutions $S_1, S_2, \ldots, S_k$. $f(S_1), f(S_2),\ldots, f(S_k)$ represent the objective function values of the solutions. Let the number of decision variables be $N$. $S_l^i$ represents the $i$th dimension of the $l$th solution. The solutions in the archive are sorted in the descending order of their fitness values. Ties are randomly broken. Weights are assigned to the solutions according to the following formula

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \quad (1)$$

$q$ is a parameter of the algorithm. It affects the "peakedness" of the distribution of weights. Smaller values of $q$ give rise to higher peakedness as opposed to higher values of $q$ (Fig 1). These weights are used for selecting existing solutions which will be used for the generation of new solutions. In this context, small values of $q$ cause the fittest solutions to be picked up for generating new solutions as opposed to higher values of $q$ which cause almost all solutions to be chosen with equal probability.

Figure 1: Effect Of Parameter 'q'




For each dimension a Gaussian kernel function is used as the probability density function. A Gaussian kernel function is a weighted sum of Gaussian functions. It is defined as

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i\sqrt{2\pi}} e^{\frac{-(x-\mu_l^i)^2}{2(\sigma_l^i)^2}} \quad (2)$$

The vector of means is given by

$$\mu^i = S_1^i, \cdots, S_k^i \quad (3)$$

The standard deviation is given by

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{\left|S_e^i - S_l^i\right|}{k-1} \quad (4)$$

$\xi$ is a parameter of the algorithm. The parameter $\xi$ influences the manner in which the solution archive

will be used in the search process. If the value of ξ is small, the new solutions will be close to the solutions in the archive and this increases the speed of convergence. On the other hand if the value of ξ is large, there will be greater diversification and convergence may be slow.

The $ACO_R$ algorithm constructs the solutions by sampling the Gaussian kernel function. The sampling is done in two steps. First, a solution is picked up from the archive. This is same as selecting the Gaussian function in the Gaussian kernel. The solution (Gaussian function) $l$ is chosen according to the probability

$$P_l = \frac{\omega_l}{\sum_i \omega_i} \tag{5}$$

Then the chosen Gaussian function is sampled. This process is repeated for each dimension. However, for each dimension a new solution is not chosen. The Gaussian functions corresponding to the already selected solution are used.

*Pseudo-Code for $ACO_R$ algorithm.*

1: Initialize the solution archive ***T*** to random solutions.
2: **repeat**
3: sort the solutions in the archive in descending order of fitness value. Break ties randomly. Compute weights according to (1).
4: **for** each ant ***m*** in **NUMANTS** do
5: select a solution $\boldsymbol{S_l}$ probabilistically according to (5).
6: **for** each dimension ***i*** do
7: generate a random number Z having standard normal distribution.
8: calculate $\sigma_l^i$ according to (4).
9: $S_m^i = S_l^i + Z\sigma_l^i$
10: **end for**
11: **end for**
12: **until** termination criterion is met

## 2.2. Nelder-Mead simplex search

Nelder-Mead simplex search was proposed by Nelder and Mead [11]. It is a heuristic technique (since it is based on rules of thumb and provides no guarantees) which is used for local search. It is based on a "simplex", which moves towards the global optimum based on the objective function values of the points making up the simplex.

The simplex is an *n*-dimensional figure. For solving a problem with *n* decision variables, we construct a simplex with (*n* + 1) dimensions. The simplex then moves towards the global optimum using the following four operations.

*Pseudo-Code for Simplex Search*

1: **REFLECTION:** $\boldsymbol{P_h}$, $\boldsymbol{P_s}$ and $\boldsymbol{P_l}$ denote the points with the highest, second highest and the lowest objective function values. Let their corresponding objective function values be $\boldsymbol{f_h}$, $\boldsymbol{f_s}$ and $\boldsymbol{f_l}$. Calculate the centroid $\boldsymbol{P_c}$ of the simplex by excluding $\boldsymbol{P_h}$. Reflect the highest point in the simplex about the centroid

$$P_r = P_c + \alpha(P_c - P_h) \tag{6}$$

α is the reflection coefficient (α > 0). We used α = 1 as suggested in [11]. Replace $\boldsymbol{P_h}$ by $\boldsymbol{P_r}$, if $f_s \geq f_r \geq f_l$ and repeat **step 1** again. If $\boldsymbol{f_r} < \boldsymbol{f_l}$ go to **step 2**, otherwise go to **step 3**.

2: **EXPANSION:** Calculate the point of expansion $\boldsymbol{P_e}$ by searching in the direction of $\boldsymbol{P_r}$.

$$P_e = P_c + \gamma(P_r - P_c) \tag{7}$$

γ is the expansion coefficient (γ > 1). We used γ = 2 as suggested in [11]. If $\boldsymbol{f_e} < \boldsymbol{f_r}$ , replace $\boldsymbol{P_h}$ by $\boldsymbol{P_e}$, otherwise replace $\boldsymbol{P_h}$ by $\boldsymbol{P_r}$ . Go to **step 1**.

3: **CONTRACTION:** If $f_h \geq f_r > f_s$ , $\boldsymbol{P_r}$ replaces $\boldsymbol{P_h}$. The point of contraction $\boldsymbol{P_{ct}}$ is calculated. Otherwise, $\boldsymbol{P_r}$ does not replace $\boldsymbol{P_h}$ and the point of contraction is calculated directly.

$$P_{ct} = P_c + \beta(P_h - P_c) \tag{8}$$

β is the contraction coefficient (0 < β < 1). We used β = 0.5 as suggested in [11]. If $f_{ct} \leq f_h$ , $\boldsymbol{P_{ct}}$ replaces $\boldsymbol{P_h}$ and go to **step 1**. Otherwise, go to **step 4**.

4: **SHRINKAGE:** If $\boldsymbol{f_{ct}} > \boldsymbol{f_h}$ then we shrink the simplex towards $\boldsymbol{P_l}$ (i.e., each point except $\boldsymbol{P_l}$ is moved towards $\boldsymbol{P_l}$).

$$P_i = P_l + \delta(P_i - P_l) \tag{9}$$

δ is the shrinkage coefficient (0 < δ < 1). We used δ = 0.5 as suggested in [11]. Go to **step 1**.

### 2.3. Ant colony optimization-simplex search (ACONM)

We now describe the hybrid of $ACO_R$ [10] and Nelder-Mead Simplex Search [11], which we shall call ACONM. The idea behind ACONM is very simple. It is to allow ACO to do the exploration, and once the algorithm identifies a promising region, to invoke the NM algorithm to quickly arrive at the global optimum (exploitation). The two important issues in the hybrid are:

1) The changeover from $ACO_R$ to NM.
2) Creation of the initial simplex for the NM algorithm.

**2.3.1. Changeover from $ACO_R$ to NM.** The point where the algorithm shifts from $ACO_R$ to NM is a crucial parameter of the algorithm. We used the standard deviation between solutions in the decision space to make this shift. The reason for choosing standard deviation between solutions in decision space is that, when the $ACO_R$ algorithm begins to converge on the global optimum, the solutions in the archive lie in the vicinity of the global optimum. They form a neighborhood which can be used by NM to quickly reach the global optimum. This is shown for Bohachevsky function [13] in Figs 2 and 3.

In each iteration, the standard deviation between solutions is calculated which we shall call η. ACO is allowed to run while η is greater than a user specified value $\eta_c$. Once η becomes less than $\eta_c$, we shift to NM.

If the value of $\eta_c$ is chosen to be a large value, then the changeover from ACO to NM happens early. It might result in a decrease in the number of function evaluation because NM uses fewer function evaluations when compared to meta-heuristics, but rate of successful minimization may also suffer because NM is not good at avoiding local minima. On the other hand if the changeover is delayed, the rate of successful minimization may be high because of $ACO_R$ performing exploitation, but the number of function evaluations could also be high.

Figure 2: Bohachevsky Function

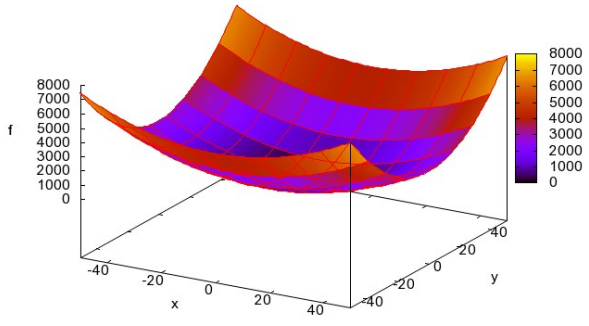
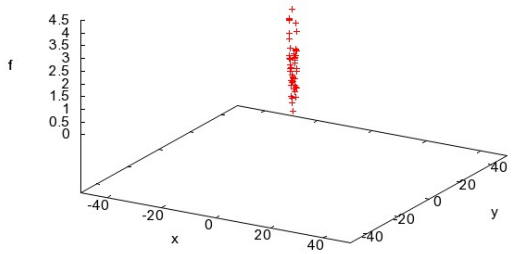
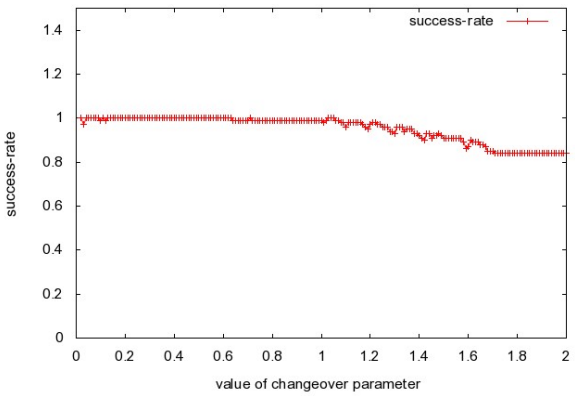


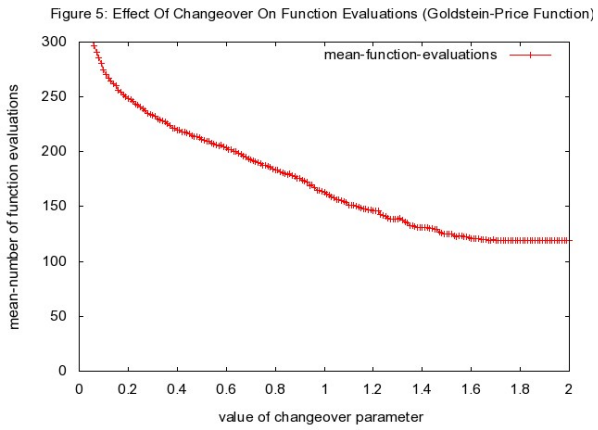

Figure 3: Population Of ACO_R Before Convergence For Bohachevsky Function



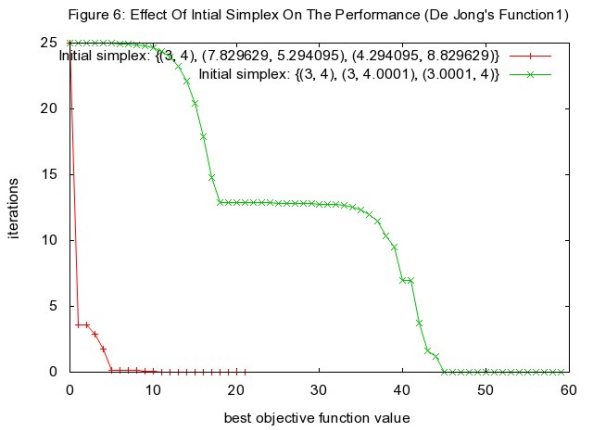

Its effect on the number of function evaluations and rate of successful minimization for Goldstein-Price function [14] is given in Figs 4 and 5.

Figure 4: Effect Of Changeover On Success Rate (for Goldstein-Price Function)






**2.3.1. Construction of the Initial Simplex.** NM is extremely sensitive to the initial simplex chosen. If the points of the initial simplex are such that their objective function values are more or less same, it doesn't give the algorithm much information about the landscape of the function being minimized. This will result in the algorithm making a large number of function evaluations to reach the optimum. Therefore, it is necessary to construct the initial simplex in such a way that the points are well spaced out in terms of their objective function values. Therefore before invoking the NM algorithm, we sort the solutions in the archive $T$ according to their fitness values, and divide the archive into $(n + 1)$ chunks ($n$ is the number of decision variables) and select the first solution in each chunk to create the initial simplex. This ensures that there is sufficient variation in the fitness values of the points making up the initial simplex. The effect of the initial simplex on the performance of NM for De Jong's function1 [15] is shown in Fig 6.




NM is invoked after the creation of the simplex and it runs until a specified termination criterion is met.

*Pseudo-Code for ACONM.*
1: Run $ACO_R$ while $\eta > \eta_c$
2: Sort the solutions in the archive according to their fitness values. Divide the archive $T$ into $n + 1$ chunks and create the initial simplex by picking the first solution from each chunk.
3. Run NM until termination criterion is met.

## 3. Experimental setup

The $ACO_R$ and ACONM algorithms were run on 18 test problems taken from the literature. In order to enable a fair comparison, the bounds fixed on the variables were same for both the algorithms.

In our experiments the standard deviation between solutions in the decision space is used as the termination criterion. The algorithms stops when it becomes less than or equal to a user specified value $\eta_t$ or the maximum number of iterations being exceeded. Arbitrarily small value of $\eta_t$ may be chosen based on the accuracy desired. We chose $\eta_t$ to be 10E-4 and the maximum number of iterations as 1000 times the number of dimensions.

We conducted 100 independent runs for each algorithm on each problem. The mean number of function evaluations and success rate over this set of 100 runs is reported. The algorithm is considered to have successfully converged to the global optimum, if the absolute difference between the known optimum and the best solution found by the algorithm is less than a user specified value which we call *maxerror*. Again the user can choose an arbitrarily small value based the requirements. We chose *maxerror* to be 10E-4 for both the algorithms.

The parameters used in $ACO_R$ are presented in Table 1. These parameters are the same as the ones used in [10]. We didn't resort to fine tuning of the parameters because, our aim is to compare the performance of $ACO_R$ with that of ACONM. The parameters used in ACONM are presented in Table 2. It contains the parameters of $ACO_R$ and that of NM together with the one new parameter which determines the point of changeover. Again the parameters of $ACO_R$ are same as those in the original paper and the parameters of NM are those suggested in [11]. Only the parameter $\eta_c$ was chosen by us. We chose its value as 1. The effect of $\eta_c$ on the performance of the algorithm was detailed in **2.3.1**. In case of a new problem the user should start with a reasonably small value like 1 and then tune it to get better performance.

The parameters shown in Tables 1 and 2 have not been varied across different problems.

| Table 1. Parameters of $ACO_R$ | |
|---|---|
| q | 0.1 |
| ξ | 0.85 |
| *numants* | 2 |
| K | 50 |

| Table 2. Parameters of ACONM | |
|---|---|
| q | 0.1 |
| ξ | 0.85 |
| *numants* | 2 |
| K | 50 |
| α | 1 |
| β | 2 |
| γ | 0.5 |
| δ | 0.5 |
| $\eta_c$ | 1 |

## 4. Results and discussion

In order to compare $ACO_R$ and ACONM both the algorithms were implemented in ANSI C and run on Pentium IV machine with Linux operating system. The results of the numerical experiments conducted are presented in the Table 3. The simple hybrid that we presented is based on the $ACO_R$ and the Nelder-Mead Simplex Search. It combines the strengths of the two in a simple way without introducing too many new parameters. The effect of the new parameter introduced shows that there is a trade-off between success rate and function evaluations when setting the point of changeover. If the changeover is done early, it results in fewer function evaluations but with the possibility of a lower success rate. On the other hand, if the changeover is done late both the success rate and number of function evaluations could be high. This simple hybrid has performed quite well on several of the test functions.

## 5. Conclusion

In this paper we have presented a simple meta-heuristic which is a hybrid of $ACO_R$ and Nelder-Mead Simplex Search. The hybrid algorithm makes use of the strengths of both the algorithms to achieve better performance. The $ACO_R$ algorithm is a powerful meta-heuristic for global optimization. It is good at avoiding the local optima and reaching the vicinity of the global optimum (exploration). However, once we reach the vicinity of the global optimum we could use a relatively simple heuristic namely the Nelder-Mead Simplex Search to quickly arrive at the global optimum (exploitation). This strategy resulted in fewer number

| Table 3. Results | | | | |
|---|---|---|---|---|
| | ACONM | | $ACO_R$ | |
| problem | srate | fevals | srate | fevals |
| Ackley [16] | 0.79 | 712.62 | 0.81 | 1251.97 |
| Bohachevsky [13] | 0.83 | 300.74 | 1.00 | 709.42 |
| Branin [14] | 1.00 | 539.23 | 1.00 | 901.72 |
| De Jong Function1 [15] | 1.00 | 184.21 | 1.00 | 565.3 |
| Easom [17] | 1.00 | 271.79 | 1.00 | 950.80 |
| Goldstein-Price [14] | 0.98 | 161.40 | 1.00 | 553.74 |
| Griewank10 [18] | 0.01 | 2102.00 | 0.28 | 2679.85 |
| Hartman 3 [14] | 1.00 | 179.90 | 1.00 | 698.18 |
| Hartman 6 [14] | 0.58 | 423.43 | 0.57 | 1183.54 |
| Rosenbrock [19] | 1.00 | 653.11 | 1.00 | 1314.38 |
| Schwefel [20] | 0.55 | 1112.76 | 0.59 | 1633.45 |
| Shekel5 [14] | 0.54 | 569.59 | 0.61 | 1161.34 |
| Shekel7 [14] | 0.67 | 502.11 | 0.59 | 1059.49 |
| Shekel10 [14] | 0.66 | 516.96 | 0.66 | 1067.33 |
| Shubert [21] | 0.84 | 1559.64 | 0.85 | 1991.94 |
| Rastrigin [22] | 0.56 | 807.16 | 0.63 | 1388.50 |
| Modified Himmelblau [23] | 0.80 | 416.02 | 0.77 | 844.33 |
| Zakharov [2] | 1.00 | 226.75 | 1.00 | 631.46 |

of function evaluations on several functions in a suite of 18 simple to difficult unconstrained benchmark problems. We showed the effect of the newly introduced parameter on the performance of the algorithm. The simulations show that the hybrid algorithm is able to converge to the optimum using fewer function evaluations for several problems. This is the significant result of the present study.

## 6. References

[1] V. Ravi, B. S. N. Murty and P. J. Reddy, "Nonequilibrium Simulated Annealing-Algorithm Applied to Reliability Optimization of Complex Systems," *IEEE Transactions on Reliability,* vol. 46, pp. 233-239, 1997.

[2] R. Chelouah and P. Siarry, "Genetic and Nelder-Mead algorithms hybridized for a more accurate global

optimization of continuous multiminima functions," *European Journal of Operational Research,* vol. 148, pp. 335-348, 2003.

[3] S. Salhi and N. M. Queen, "A Hybrid Algorithm for Identifying Global and Local Minima when Optimizing Functions with many Minima," *European Journal of Operational Research,* vol. 155, pp. 51-67, 2004.

[4] R. Chelouah and P. Siarry, "A hybrid method combining continuous tabu search and Nelder-mead simplex algorithms for the global optimization of multiminima functions," *European Journal of Operational Research,* vol. 161, pp. 636-654, 2005.

[5] T. R. Bhat, D. Venkataramani, V. Ravi and C. V. S. Murty, "An improved differential evolution method for efficient parameter estimation in biofilter modeling," *Biochemical Engineering Journal,* vol. 28, pp. 167-176, 2006.

[6] S.-K. S. Fan and E. Zahara, "A Hybrid Simplex Search and Particle Swarm Optimization for Unconstrained Optimization," *European Journal of Operational Research,* vol. 181, pp. 527-548, 2007.

[7] J. Dreo and P. Siarry, "Hybrid Continuous Interacting Ant Colony Aimed at Enhanced Global Optimization," *Algorithmic Operations Research,* vol. 2, pp. 52-64, 2007.

[8] J. Dreo and P. Siarry, "Continuous Interacting Ant Colony Algorithm Based on Dense Heterarchy," *Future Generation Computer Systems,* vol. 20, pp. 841-856, 2004.

[9] J. Dreo and P. Siarry, "An Ant Colony Algorithm Aimed at Dynamic Continuous Optimization," *Applied Mathematics and Computation,* vol. 181, pp. 457-467, 2004.

[10] K. Socha and M. Dorigo, "Ant Colony Optimization for continuous Domains," *European Journal of Operational Research,* vol. 185, pp. 1155-1173, 2008.

[11] J. A. Nelder and R. Mead, "A Simplex Method for Function Optimization," *The Computer Journal,* vol. 7, pp. 308-313, 1965.

[12] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge MA: MIT press, 2004.

[13] M. E. Bohachevsky, M. E. Johnson and M. L. Stein, "Generalized simulated annealing for function optimization," *Technometrics,* vol. 28, pp. 209-217, 1986.

[14] L. Dixon and G. Szego, *Towards Global Optimization.* New York: North Holland, vol. 2, 1978.

[15] De Jong, "An analysis of the behaviour of a class of genetic adaptive systems," PhD thesis, University of Michigan, 1975.

[16] R. Storn and K. Price, "Differential Evolution: A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.

[17] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* Berlin/Heidelberg/New York: Springer-Verlag, 1996.

[18] A. O. Griewank, "Generalized Descend for Global Optimization," *Journal of Optimization Theory and Applications,* vol. 34, pp. 11-39, 1981.

[19] H. P. Schwefel, *Evolution and Optimum Seeking.* New York: John Wiley and Sons, 1995.

[20] H. Muhlenbein, S. Schomisch and J. Born, "The Parallel Genetic Algorithm as Function Optimizer," in *Proceedings of the Fourth International Conference on Genetic Algorithms.* R. Belew and L. Booker Ed. Morgan Kaufman, 1991, pp. 271-278.

[21] A. V. Levy and A. Montalvo, "The Tunneling Algorithm for the Global Minimization of Functions," *Society for Industrial and Applied Mathematics,* vol. 6, pp. 15-29, 1985.

[22] A. Torn and A. Zilinskas, *Global Optimization.* Berlin: Springer-Verlag, 1989.

[23] D. M. Himmelblau, *Applied Nonlinear Programming.* New York: McGraw-Hill, 1972.