10.1109/IRANIANCEE.2010.5507009

Termite Colony Optimization: A Novel Approach for Optimizing Continuous Problems

Ramin Hedayatzadeh, Foad Akhavan Salmassi, Manijeh Keshtgari
Department of Information Technology
Shiraz University of Technology
Shiraz, Iran.
inbortend@gmail.com

Reza Akbari, Koorush Ziarati
Department of Computer Science and Engineering
Shiraz University
Shiraz, Iran
rakbari@cse.shirazu.ac.ir

Abstract- In this paper, a novel approach, called Termite colony optimization (or TCO), for optimizing numerical functions is presented. TCO is a population based optimization technique which is inspired from intelligent behaviors of termites. The proposed approach provides a decision making model which is used by termites to adjust their movement trajectories. Termites move randomly in the search space, but their trajectories are biased towards regions with more pheromones. TCO is compared with existing population-based algorithms on a set of well known numerical test functions. The experimental results show that the TCO is effective and robust; produce good results, and outperform other algorithms investigated in this consideration.

Keywords: Termite Colony Optimizatio, Numerical functions

I. INTRODUCTION

Optimization has been an active area of research for several decades. In optimization problems, the objective is to find the minimum or maximum of the function under consideration. There are many population based optimization techniques available for unconstrained numerical optimization. Genetic algorithms (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) are among the most popular optimization algorithms which employ a population of individuals to solve the problem on the hand.

GA is the most popular evolutionary algorithms, in which a population of individuals evolves according to a set of rules such as selection, crossover and mutation [1]. In such algorithm, exploitation is obtained through selection, where individuals move toward regions with higher fitness. Also, the exploration is provided by perturbing individuals using crossover and mutation operators.

ACO is an optimization algorithm inspired by foraging behaviors of ants. In order to find optimum solution, ant colonies use indirect communication which is based on the laying and detection of pheromone trails. This phenomenon is called stigmergy. ACO method explores the search space by allowing ant colonies to move randomly. Exploitation is achieved by biasing movements of the ants towards most profitable regions which contain more pheromones. ACO was first introduced by Dorigo [2] for routing in communication networks. After that, ACO based algorithms had been developed and tested successfully for a various engineering problems [3]-[5].

PSO is a swarm intelligence technique developed by Eberhart and Kennedy [6], inspired by the social behavior of bird flocking and fish schooling. PSO has been shown to successfully optimize a wide range of continuous functions. The search for an optimum in the PSO is an iterative process that is based on random decisions. In PSOs exploitation is obtained through selecting the particle with best fitness as *gbest* and moving toward that particle. Each particle memorizes its previous best position, represented as *pbest* to explore the space between its previous best position and global best position. Although, PSO provide powerful methods for optimization problems, it suffers from different problems such as premature convergence and stagnation. Different approaches such as inertia weight, and time-varying coefficients were proposed to alleviate these problems [7]-[10].

This work presents a novel optimization method based on intelligent behaviors of termites. TCO provides a colony of termites with a stochastic decision making process. The decision making process is used by the termites to select their movement patterns. A termite selects a movement pattern based on the local information which is obtained by sensing the nearby regions. A termite moves randomly, but its movement may influenced by observed pheromones in nearby regions. Random movement is biased by pheromone. A termite tends to move toward a region which contains more pheromones.

The rest of the paper is organized as follows. Section 2 introduces the intelligent behaviors of termite colonies and the ground principles of the proposed algorithm. Description of the TCO algorithm is presented in section 3. Section 4 reports test functions, experimental settings and the experimental analysis on the proposed approach in comparison with other algorithms. Finally, section 5 concludes this work.

Proceedings of ICEE 2010, May 11-13, 2010
978-1-4244-6760-0/10/$26.00 ©2010 IEEE

II. INTELLIGENT BEHAVIORS OF TERMITES

A colony of termites is a decentralized system which capable to perform complex tasks using relatively simple rules of individual termites' behavior. Such system contains simple individuals interacting locally with one another and with their environment. Despite a lack of centralized control or any organizational structure, local interactions among simple individuals cause a global pattern to emerge.

The ability of social insects to self-organize relies on four principles: positive feedback, negative feedback, randomness, and multiple interactions. A fifth principle, stigmergy, arises as a product of the previous four [11]. Such self organization is known generally as swarm intelligence.

A simple example of the hill building behavior of termites provides a strong analogy to the mechanisms of Termite and optimizing continuous problems in general. This example illustrates the four principles of self organization [11]. A similar analogy is often made with the food foraging behavior of ants.

A colony of termites has the ability to perform complex task by applying simple rules between its individuals. For example, consider a flat surface upon which termites and pebbles are distributed. The termites would like to build a hill from the pebbles. The termites try to collect all of the pebbles into one place. Termites act independently of all other termites, and move only on the basis of an observed local pheromone gradient. Pheromone is a chemical excreted by the insect which evaporates and disperses over time.

A termite moves randomly, but is biased towards the locally observed pheromone gradient. If no pheromone exists, a termite moves uniformly randomly in any direction. Each termite may carry only one pebble at a time. If a termite is not carrying a pebble and it encounters one, the termite will pick it up. If a termite is carrying a pebble and it encounters one, the termite will put the pebble down. The pebble will be infused with a certain amount of pheromone.

With these rules, a group of termites can collect dispersed pebbles into one place. Similar to other population based algorithms, the swarm intelligence principals such as positive feedback, negative feedback, randomness, multiple interaction, and stigmergy play important roles in social behaviors of termites.

III. TCO ALGORITHM

In this section we present the proposed TCO algorithm. The pseudocode of the TCO algorithm is represented in Figure 1. TCO employs stochastic process to find optimal solution. It employs a population of termites which move in a *D*-dimensional search space $S \subset R^D$ to find the optimal solution. Assume that we have a population of *N* termites. Each termite *i* in the population is associated with a position vector $\vec{x}_i = (x_{i1}, x_{i2}, ..., x_{iD})$, which represents a feasible solution for an optimal problem in the *D*-dimensional search space *S*. In TCO, each position vector $\vec{x}_i$ represents a hill with an associated quality which is represented as $fit(\vec{x}_i)$. The fitness value models amount of pheromones which are deposited on the hill. The TCO employs the following scenario to optimize a numerical function:

Initially, number of the termites, and maximum number of iterations, $\text{Iter}_{\max}$, are determined. After that, at the start time of the algorithm all of the termites are positioned randomly in the search space:

$$\vec{x}_i(0) = Init(i, S) \quad 1 \le i \le N \tag{1}$$

where $Init(i, S)$ is the initialization function which associates a random position to the termite *i* in the search space *S*. After initialization, the termites employ the following process to adjust their positions throughout iterations until the termination condition is met.

At each cycle of the algorithm, the fitness value of each termite is evaluated. The fitness value is used for computation of pheromone content at each location of termites. The pheromone content at the *j*-th location is computed based on the following equation:

$$\tau_i(t) = (1-\rho)\tau_i(t-1) + 1/(fit(x_i)+1) \tag{2}$$

where, ρ is the evaporation rate that is taken in range of [0..1], $\tau_i(t-1)$ and $\tau_i(t)$ respectively are the pheromone level at the current and previous locations of *i*-th termite.

After computing the pheromone levels at the locations of termites, each termite adjusts its trajectory based on local information and moves to new location. The termite movement is a function of pheromone level at the visited location and the distance between a termite location and the visited locations. Based on these, two different movement patterns introduced for termites. A local region around each termite is considered, and the number of visited position in the neighborhood of the termite is computed. If there is no visited position in the neighborhood of a termite, it moves randomly in its own nearby regions. Termites with one or more visited positions in their neighborhood may select a more profitable position and move toward that position.

A part of termites employ a random movement pattern in order to find more profitable regions. The random walk is performed by the termite in a region with radius τ. The search region is centered at current position of the termite. So the next position of a termite is updated using the following equation:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + Rw(\tau, \vec{x}_i(t-1)) \tag{3}$$

where $\vec{x}_i(t-1)$ represents the previous position of the termite which is replaced by the new the new position of that termite (i.e. $\vec{x}_i(t-1)$), and *Rw* is a random walk function that depends on the current position of the termite and the radius search τ . The initial value of radius τ is defined as a percentage of $|X_{\max} - X_{\min}|$, where $X_{\max}$ and $X_{\min}$ respectively represent the maximum and minimum values of the search space along a dimension. The value of τ is linearly decreased from $\tau_{\max}$ to $\tau_{\min}$ throughout iterations. The termites adjust their walking based on the τ . The large value of τ at the first iterations enables termites walking with large step size to explore wide regions in the search space. While the small values of τ in the last iterations encourage the scout termites to walk more precisely within small regions.

Initialization:
  Determine N, τ , and $\text{Iter}_{\max}$
  For *i* = 1 **to** N
    $\vec{x}_i(0) = Init(i, S) \quad 1 \le i \le N$
  Next *i*
Do
  Compute fitness of the termites
  For *i* = 1 **to** N
    $\tau_i(t) = (1-\rho)\tau_i(t-1) + 1/(fit(x_i)+1)$
  Next *i*
  For *i* = 1 **to** N
    Find the neighbor positions for termite *i*
      Select best neighbor position $\vec{b}_i$
      If termite *i* has neighbors **Then**
        If $\tau_i(t-1) < \tau_{b_i}(t-1)$ Then
          $\vec{x}_i(t) = \vec{x}_i(t-1) + \omega_b r_b(\vec{b}_i - \vec{x}_i(t-1))$
        End If
      Else
        $\vec{x}_i(t) = \vec{x}_i(t-1) + Rw(\tau, \vec{x}_i(t-1))$
      End If
  Next *i*
  Adjust radius τ and step size *s*
Until *iter*< $\text{Iter}_{\max}$

Figure 1. Pseudocode of TCO algorithm

The locally observed pheromones provide the ability for a termite to use a selection process in order to adjust its trajectories towards one of these pheromone gradients. In other words, the termite evaluates the provided pheromone information, and adjusts its trajectory towards a position with the highest level of pheromone. A termite *i* considers the local best position (denoted as $\vec{b}_i$) as its own promising position and move towards that if its current position has smaller level of pheromone compared to the best local position. The movement trajectory of the termite *i* is controlled using the following equation:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \omega_b r_b(\vec{b}_i - \vec{x}_i(t-1)) \quad if(\tau_i(t-1) < \tau_{b_i}(t-1)) \tag{4}$$

where $1 < w_b < 2 \,\&\, 0 < r_b < 1$ probabilistically controls the attraction of the termite towards local best position.

The TCO algorithm provides a colony of termites with two movement patterns. This heterogeneity result an effective population based algorithm for optimizing numerical functions. The random pattern is used by a part of termites that can not use social knowledge provided by the other termites. The TCO algorithm employs this pattern control diversity and stochasticity of the behaviors of the termite in the colony. Usually, increasing diversity of population is used as a way to mitigate stagnation problem. In TCO algorithm, diversity of the colony is effectively controlled by a part of termite with random walk. The proposed approach permanently encourages the population to exit from stagnation state by exploring the nearby regions using random termites. This approach provides excellent result in cases when the gradient does not point toward optimum solution or multiple local optima exist in direction of global optimum solution. The second movement pattern is used by the other part of termites. These termites utilize the social knowledge about more profitable regions and adjust their trajectories towards these regions. This pattern encourages the swarm to efficiently control the exploration and convergence towards global optimum. Also, it mitigates premature convergence and stagnation that may occur due to hard constriction on the movement trajectories of the individuals of a population.

IV. EXPERIMENTS

In this section, the experiments that have been done to evaluate the performance of the proposed TCO algorithm for a number of analytical benchmark functions are described. The performance of TCO is evaluated in comparison with three other population based algorithms.

A. Benchmarks

To test the performance of TCO, six well known benchmark functions are used here for comparison, both in terms of optimum solution after a predefined number of iterations and the rate of convergence to the optimum solution. TABLE I gives the test functions, mathematical expression, and their ranges. All test functions have optimum values equal to zero. The first two functions are unimodal, and the others are multimodal. A function is called unimodal, if it has only one optimum point. A multimodal function has two or more local optima.

To test the robustness of the algorithms, the most common initialization ranges used in the literature for these benchmarks considered in this paper. This commonly accepted approach, called asymmetric initialization, is used in this paper. In asymmetric initialization the individuals of a population are initiated only in the right side of search, while the symmetric initialization employs the entire search

space. Considering this approach, each termite has been initialized with a position which is randomly chosen in range $\left[\frac{X_{\max}}{2}, X_{\max}\right]$.

TABLE I. THE BENCHMARK FUNCTIONS: THEIR MATHEMATICAL REPRESENTATION AND RANGES.

Func.	Formula	Range
f_{Sph}	$f_{Sph}(x)=\sum_{i=1}^{n} x_i^2$	[-100, 100]
f_{Ros}	$f_{Ros}(x)=\sum_{i=1}^{n-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right)$	[-30, 30]
f_{Ras}	$f_{Ras}(x)=\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	[-5.12, 5.12]
f_{Shf6}	$f_{schf6}=0.5-\frac{\left(\sin\sqrt{\left(x_1^2+x_2^2\right)}\right)^2-0.5}{\left(1+0.001\left(x_1^2+x_2^2\right)\right)^2}$	[-100, 100]
f_{Gri}	$f_{Gri}(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]
f_{Ack}	$f_{Ack}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{1}^{n} x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{1}^{n}\cos(2\pi x_i)\right)+20+e$	[-30, 30]

B. Settings of the Algorithms

Simulations were performed to observe the performance of the proposed algorithm for finding optimum solutions. There is no algorithm in the literatures for numerical optimization based on intelligent behaviors of termites. The performance of the TCO is compared with the performance of the Genetic Algorithm (GA) [13], Ant Colony Optimization (ACO) [12], and Particle Swarm Optimization (PSO) [13] methods. These algorithms have a few parameters; part of them is common while another part is specific for each algorithm.

Common parameters are number of dimensions of the search space, maximum generation, population size, and total number of trials. For all test functions with exception of 2D function Schaffer's f6, three different dimension sizes, 10, 20 and 30 are tested. The corresponding maximum generations are 1000, 1500 and 2000 respectively. For Schaffer's f6 function, the maximum generation is set to 1000. For all the test functions, the population size is set to 100, and a total of 100 trials for each experimental setting are conducted.

The following settings were used for GA. The individuals of the populations are coded in binary and 48 bits are used for each function variable (the chromosome length is 48 bit). The scaling window technique (of length 5) is used for fitness scaling. The two-point crossover and genetic mutation are selected as genetic operators. The crossover probability is set at 0.6 and the mutation probability is set at $1/(chromlength \times sf)$. The selection of individuals corresponds to their fitness and an elitist strategy is used in which a single copy of best individual is preserved.

For the PSO algorithm, the swarm uses the acceleration coefficients $c_1 = c_2 = 1.49$, the inertia weight $\omega = 0.72$, and the maximum velocity $v_{\max}$ is limited to the domain of the test function.

For the TCO algorithm, the evaporation rate ρ was set at 0.8. The maximum and minimum values of $\tau_{\max}$ and $\tau_{\min}$ are respectively equal to $\frac{1}{2} * |X_{\max} - X_{\min}|$ and $\frac{1}{8} * |X_{\max} - X_{\min}|$. Finally, the same settings as describe in [12] was used for ACO algorithm.

C. Experimental Results

In the experiments the number of iterations to reach a predefined threshold was specified for each function. Different success criteria for different functions are presented in the literature. For Schaffer's f6, the success criterion is set to 0.0001, whereas for the other functions, the stopping criteria are set to 0.1. After the final iteration, if the minimum value was reached by the algorithm was not below the threshold, the trial was considered unsuccessful. The values of mean, standard deviations, success rate, and the average number of iteration before the algorithms reach the success criteria in terms of each test function which are obtained by the GA, ACO, PSO, and TCO algorithms are given in TABLE II. The X sign shows that the corresponding algorithm was failed in reaching success criteria in all trials. The average fitness of the algorithms which are smaller than E-15 are considered as 0. For ease of observation, the best results were found by the algorithms are shown in bold.

1) Performance Analysis

First, the performances of the algorithms are considered in terms of average optimum values for 100 trials. The results show that all of the algorithms provide good performance for Ackley function. However TCO produces better results than GA, ACO, and PSO strategies. It is clear from the result that for Rosenbrock function, the reduction of the quality of the average optimum was occurred for GA, ACO and PSO compared with TCO algorithm. As described previously the TCO algorithm employs different movement patterns which help the swarm to maintain global search and control convergence towards global optima. These capabilities help the algorithm to cope with the smooth slope of the Rosenbrock function. For the Schaffer's test function, GA, ACO, and PSO provides competitive results. However, the TCO algorithm provides considerable improvement on this function. For the other test functions, TCO algorithm outperforms GA, ACO and PSO strategies. In summary, the results show that the TCO produces consistent results for unimodal, and multimodal functions with dependent or independent variables.

TABLE II. AVERAGE FITNESS VALUES, STANDARD DEVIATION, AND SUCCESS RATE OF THE ALGORITHMS FOR THE BENCHMARK FUNCTIONS FOR 100 TRIALS

		GA				ACO				PSO				TCO			
Fun	Dim	Avg	Stdv	Iter	S. R.	Avg	Stdv	Iter	S. R	Avg	Stdv	Iter	S. R	Avg	Stdv	Iter	S. R
f_{Sph}	10	2.30E-02	1.51E-01	680.31	0.48	9.38E-01	1.36E+00	X	0.00	2.45E-01	3.18E-02	438.22	0.21	**3.28E-11**	**1.94E-11**	**270.15**	**1**
	20	8.23E-01	5.61E-01	X	0.00	2.17E+00	2.31E+00	X	0.00	8.09E-01	7.36E-02	X	0.00	**7.46E-10**	**2.95E-10**	**481.07**	**1**
	30	7.66E+00	3.27E+00	X	0.00	7.45E+00	4.82E+00	X	0.00	1.70E+00	9.47E-01	X	0.00	**5.66E-9**	**1.82E-9**	**596.30**	**1**
f_{Ros}	10	4.63E+01	3.38E+01	X	0.00	1.59E+01	8.53E+00	X	0.00	4.37E+00	2.38E+00	X	0.00	**8.97E-05**	**2.74E-05**	**683.05**	**1**
	20	1.03E+02	2.95E+01	X	0.00	4.62E+01	2.26E+01	X	0.00	7.74E+01	9.49E+01	X	0.00	**5.48E-04**	**3.12E-04**	**927.42**	**1**
	30	1.66E+02	5.95E+01	X	0.00	1.08E+02	4.95E+01	X	0.00	4.02E+02	6.33E+02	X	0.00	**8.21E-03**	**5.36E-03**	**1136.5**	**1**
f_{Ras}	10	1.39E+00	7.63E-01	X	0.00	1.35E-01	5.21E-02	**379.53**	0.64	2.65E+00	1.38E+00	X	0.00	**4.23E-7**	**3.45E-7**	890.52	**1**
	20	6.03E+00	1.45E+00	X	0.00	7.82E-01	4.93E-01	**640.12**	0.17	1.20E+00	3.32E+00	X	0.00	**7.30E-5**	**4.78E-5**	963.25	**1**
	30	1.04E+01	2.63E+00	X	0.00	2.32E+00	8.13E-01	X	0.00	3.24E+01	6.95E+00	X	0.00	**9.08E-04**	**6.06E-04**	**1290.4**	**1**
f_{Ack}	10	2.80E-02	1.72E-01	723.15	0.69	3.80E-06	7.12E-06	X	**1**	9.84E-13	9.62E-13	385.17	**1**	**3.55E-14**	**2.16E-14**	**664.20**	**1**
	20	5.49E-01	3.29E-01	1074.36	0.43	4.29E-05	6.55E-05	X	**1**	1.17E-08	1.58E-08	462.39	**1**	**6.92E-11**	**4.38E-11**	**724.81**	**1**
	30	8.20E-01	6.14E-01	1290.85	0.17	2.67E-03	3.81E-03	X	**1**	1.49E-06	1.86E-06	536.52	**1**	**9.84E-7**	**3.79E-7**	**894.35**	**1**
f_{Gri}	10	1.92E-01	8.37E-02	X	0.00	8.03E+00	7.25E+00	X	0.00	7.93E-02	3.34E-02	448.23	0.31	**5.27E-03**	**3.14E-03**	**385.22**	**1**
	20	2.76E-01	5.62E-02	X	0.00	3.91E+00	4.11E+00	X	0.00	3.05E-02	2.54E-02	479.05	0.53	**1.86E-05**	**5.08E-05**	**414.90**	**1**
	30	2.83E-01	7.41E-02	X	0.00	3.45E+00	3.92E+00	X	0.00	1.12E-02	1.42E-02	521.53	0.81	**5.45E-05**	**8.29E-06**	**465.03**	**1**
f_{Shf6}	2	6.85E-02	2.14E-02	X	0.00	1.56E-01	7.49E-02	X	0.00	3.84E-03	1.41E-03	239.27	0.68	**0.00E+00**	**0.00E+00**	**359.77**	**1**

2) Success Rate

The success rate which is related to the convergence of each method to the stopping criteria represents the stability of the algorithms. It is clear from the results that all the methods except GA have converged to the stopping criteria for Ackley function. The GA, ACO, and PSO algorithms have poor success rates on other test functions. The GA algorithm only achieved non zero success rate on Sphere function in 10 dimension and Ackley function. This algorithm fails in reaching success criteria for Rosenbrock, Rastrigrin and Griewank functions in 10, 20, and 30 dimensions as well as Schaffers' f6 function in 2 dimensions. Its success rates vary from 0 to 69% for other test functions in different dimensions. This problem occurs due inability of GA in providing proper balance between exploration and exploitation.

The PSO algorithm achieves a success rate 100% on Ackley function in 10, 20, and 30 dimensions. Also non zero success rates were found for Griewank, Schaffers' f6, and Sphere in 10 dimensions. All of the particles in PSO select the best particle as their interesting particle and adjust their trajectories toward that particle without considering other valuable information. So, they are gravitated rapidly toward the elite termite and the premature convergence will be occurred.

The ACO algorithm has good success rate in optimizing Ackley function. It has non zero success rate on Griewank function in 10, and 20 Dimensions. Its convergence rate drastically decreases in the case of Rosenbrock, Sphere, Rastrigrin, and Schaffer functions. This is caused due to very smooth slope of the Rosenbrock function nearby its global optimum position.

It is clear from the success rate results that the TCO method produces stable solutions for solving numerical optimization problems. The success rate of 100% on all the test functions in 10, 20, and 30 dimensions were reached by TCO algorithm.

3) Convergence Speed

The speed in converging to the global optimum is an important measure for evaluating the algorithm performance. The measure is evaluated by the average number of iteration before convergence. The columns "Iter" in TABLE II presents the convergence speeds of the algorithms. As can be seen, The ACO has the fastest convergence speed in optimizing Rastrigrin function in 10 dimensions. The TCO is the only algorithm which converged to the success criteria on Rosenbrock function. All the other algorithms never converged to the success criteria. The GA and PSO algorithms failed in optimizing Rastrigrin function, so their convergence speed is presented as X. The best convergence speed on Ackley was obtained by TCO. Similarly, TCO has the fastest convergence speed on the other test functions. In general, TCO has considerably the fast convergence speed.

4) Stability Analysis

The stability principle says that a population should not change its mode of behaviour every time the environment changes. In order to analyze the stability of the TCO algorithm, the trajectories of the global best termite in population are considered. We assume that stability occurs when the termites positioned at the search area defined by a threshold around the optimum position. The search region is defined as $S_{stab} = \{x : -\phi < x < \phi\}$, where ϕ is the threshold. We have used ϕ as 0.001. The trajectory of the termite outside of the search region is considered as undesirable movement and violates stability of the algorithm. The trajectories of the global best termite of in optimizing two dimensional test functions are given in Figure 2. The termite trajectories show that MLBCO algorithm guarantees stability on four test functions. After the algorithm stabled, the termite never escaped the search region S_{stab} . This implies that the population does not change its behaviour and stability principal is satisfied.

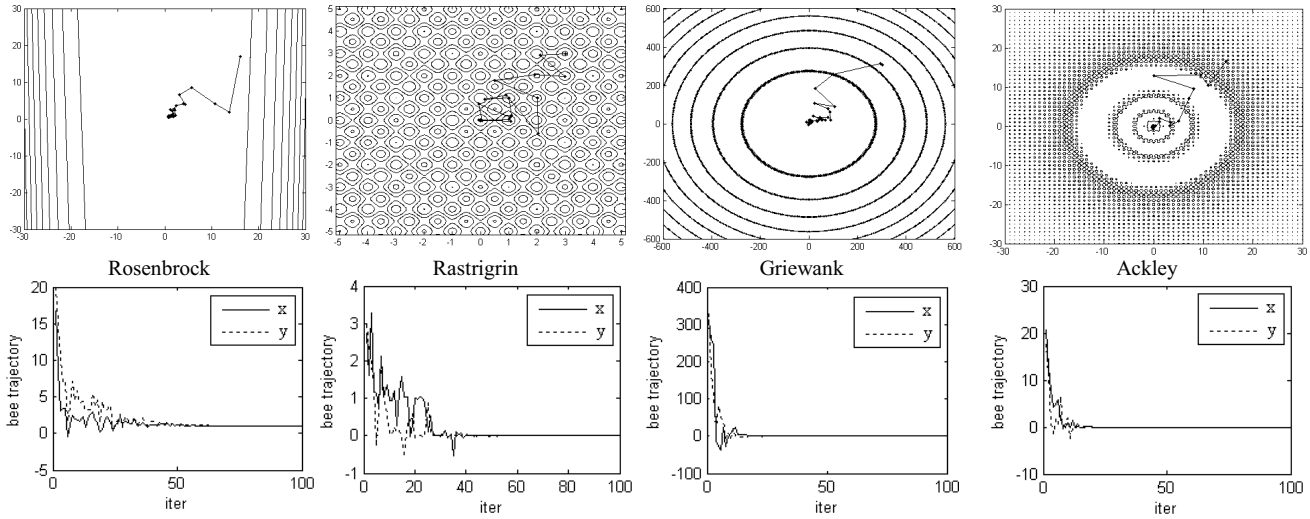

Figure 2. Convergence of the TCO algorithm on six test functions.

D. Disscussions

Stagnation and premature convergence are the major weaknesses of the optimization methods. The optimization methods try to cope with these problems by establishing proper balance between exploration and exploitation. In GA, exploitation is obtained through selection, where individuals move toward regions with higher fitness. Also, the exploration is provided using crossover and mutation operators. In ACO, the balance between exploration and exploitation is controlled using pheromone trails. In PSO, different approaches such as inertia weight, time-varying coefficients, etc. were proposed to alleviate these problems. Although, GA, ACO, and PSO alleviate these problems and provide powerful methods for continuous optimization, they still suffer from aforementioned deficiencies due to their intrinsic characteristics. TCO provides more flexibility to control the wandering and converging mechanism respectively in the first and the last iterations using the proposed decision making model. The average fitness, convergence speed, and success ratio are the main performance measures. The results show that TCO algorithm surpasses other algorithms for the average fitness. Considerable improvement on success rates, and convergence speeds have been obtained by TCO algorithm.

V. CONCLUSIONS

In this paper, we have described a novel optimization algorithm, called TCO, based on intelligent behavior of termites. Different types of movement patterns were introduced into the TCO algorithm aiming to maintain proper balance between global and local search by providing diversity into the swarm of termites. The TCO algorithm was compared with three other population based optimization algorithms on the six benchmark functions. The experiments results showed that the TCO is effective for numerical function optimization.

REFERENCES

[1] D. Srinivasan, T. H. Seow, “Evolutionary Computation”, CEC ’03, 8–12 Dec. 2003, Canberra, Australia, pp. 2292–2297, 2003.

[2] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[3] K. Sundareswaran, K. Jayant, and T. N. Shanavas, “Inverter Harmonic Elimination Through a Colony of Continuously Exploring Ants”, in *IEEE Transactions on Industrial Electronics*, Vol. 54, No. 5, pp. 2558-2565, 2007.

[4] T. Sum-im and W. Ongsakul, “Ant colony search algorithm for unit commitment,” in *Proc. IEEE International Conference on Industrial Technologies.*, Dec. 10–12, 2003, vol. 1, pp. 72–77.

[5] C. M. Colman, E. J. Rothwell, and J. E. Ross, “Investigations of simulated annealing, ant colony optimization and genetic algorithms for self structuring antennas,” *IEEE Transactions on Antennas Propagations*, vol. 52, no. 4, pp. 1007–1014, Apr. 2004.

[6] J. Kennedy, and R. Eberhart, “Particle swarm optimization,” in *Proceeding of IEEE Int. Conf. Neural Networks*, vol. 4, pp. 1942–1947, 1995.

[7] R.C. Eberhart, Y. Shi, “Tracking and optimizing dynamic systems with particle swarms”, *in Proceedings of IEEE Congress on Evolutionary Computation*, Seoul, Korea, pp. 94–97, 2001.

[8] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, “Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients”, in *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 240-255, 2004.

[9] P. N. Suganthan, “Particle swarm optimizer with neighborhood operator,” in *Proc. IEEE Int. Congress Evolutionary Computation*, vol. 3, pp. 1958–1962, 1999.

[10] P. N. Suganthan, “Particle swarm optimizer with neighborhood operator,” in *Proc. of IEEE Int. Congress Evolutionary Computation*, vol. 3, pp. 1958–1962, 1999.

[11] M. Roth, and S. Wicker, “Termite: Ad-Hoc Networking with Stigmergy”, *in Proceeding of IEEE International Conference on Global Economics*, pp. 2937-2941, 2003.

[12] K. Sundareswaran, K. Jayant, T. N. Shanavas, "Inverter Harmonic Elimination Through a Colony of Continuously Exploring Ants”,in *IEEE Trans. on Industrial Electronics*, vol. 54, no. 5, pp. 2558-2565, 2007.

[13] Srinivasan, D., Seow, T.H.: Evolutionary Computation, CEC ’03, 8–12, 4(2003), Canberra, Australia, pp. 2292–2297, Dec. 2003.