7

Cockroach Swarm Optimization – Modifications and Application

Joanna Kwiecien
AGH University of Science and Technology
Department of Automatics and Robotics, Krakow, Poland

CONTENTS

7.1 Introduction

During the last decade, a number of computational intelligence algorithms have been developed. The cockroach swarm optimization algorithm discussed here, is one of many interesting metaheuristic search algorithms inspired by nature. The CSO algorithm was described by Chen and Tang in 2010 [1]. It mimics the food foraging behavior of swarms of cockroaches and involves the procedures concerning the individuals' movement within the group. A coordination system of a group of cockroaches, being a decentralised system, is responsible for the organisation of tasks required for resolving a specific task.

Cockroaches, like many other animals, appear to employ some mechanisms by which a swarm may assemble. They make decisions based on their interaction with peers. Although cockroaches are known to have a variety of habits, most available relate to swarming, chasing, dispersing, and ruthless behavior. The CSO algorithm belongs to a group of population methods, because it applies a population of current solutions which can generate new solutions after proper selection and modification. Therefore, in its structure one can find three kinds of simple behaviors applied to search space: chase-swarming that reflects that cockroaches can communicate with each other, dispersion that reflects scattering or escaping from light (it may improve the survival rate), and ruthlessness, when the strong cockroach eats the weak one randomly at intervals of time. Cockroaches change their positions with time. Hence, the CSO algorithm uses a population of solutions for every iteration, and the cockroach updates its solution based on either the best solution obtained (within its visual scope or the best solution obtained so far) or a random movement.

When applying the CSO algorithm, it is necessary to make decisions regarding the determination of its structure to be suitable for the solution of a given problem, as well as carefully select the quantitative and qualitative parameter values, being typical for the algorithm under consideration, followed by an algorithm performance evaluation. In order to cope with the application of the CSO algorithm to solve the traveling salesman problem, some adaptations of the algorithm are presented. It should be mentioned that many researchers described the traveling salesman problem in respect to its practical use, and the TSP is one of the most famous combinatorial problems. A lot of real problems can be formulated as the TSP problem [2]. As indicated in [3], path planning and wayfinding tasks as TSP-like tasks are quite common in human navigation.

The rest of this chapter is organized as follows: in Section 7.2 a brief introduction to the CSO algorithm is presented. The pseudo-code for the basic version of CSO algorithm is also discussed. Section 7.3 provides some modifications of the CSO algorithm including inertia weight, stochastic constriction factor, hunger component, and neighborhood scheme. In Section 7.4, the application of the CSO algorithm for the traveling salesman problem is described. Therefore, we present how to adapt the CSO algorithm to solve this problem, concerning the movement performance. Various modifications of movements, e.g. approaching the best local or global individual with the use of crossover operators in chase-swarming, or relocating based on 2-opt moves are used. Finally, in Section 7.5 some concluding remarks are provided.

7.2 Original CSO algorithm in brief

7.2.1 Pseudo-code of CSO algorithm

The main steps of the CSO algorithm are outlined in the pseudo-code form (Algorithm 9).

Algorithm 9 Pseudo-code of the original CSO.

determine the $D-th$ dimensional objective function $OF(.)$
initialize the CSO algorithm parameter values such as k – the number of cockroaches in the swarm, *visual* – the visual scope, *step* - the step of cockroaches, *Stop* – termination condition
generate a population of k individuals, the $i-th$ individual represents a vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$
for each $i-th$ cockroach from swarm **do**
    evaluate quality of the cockroach X_i using $OF(.)$ function
end for
find the best solution P_g in initial population
while termination condition not met **do**
    for $i = 1$ to k **do**
        for $j = 1$ to k **do**
            if the objective function of the $j-th$ cockroach is better than the objective function of the $i-th$ cockroach, within its visual scope **then**
                move cockroach i towards j
            end if
            if the position of cockroach i is local optimum **then**
                move cockroach i towards P_g
            end if
        end for j
    end for i
    if new solution is better than P_g **then**
        update P_g (cockroach i is a current global solution)
    end if
    for $i = 1$ to k **do**
        move cockroach randomly using formula
        $X_i = X_i + rand\,(1, D)$
        if the new position is better than P_g **then**
            update P_g
        end if
    end for
    select cockroach h randomly
    replace cockroach h by the best global cockroach
end while
return the best one as a result

7.2.2 Description of the original CSO algorithm

The CSO algorithm starts with a fixed number of cockroaches and in each cycle of the algorithm, the cockroaches' positions are updated to search for a better solution. In the CSO algorithm, individuals have a randomly generated position at the beginning. The search space, in which the cockroaches move,

has dimension D. Therefore, the position of the $i-th$ individual in the CSO algorithm can be represented by D-dimensional vector.

At the start, we should have defined an objective function $OF(.)$ (step 1) and the CSO algorithm parameters (step 2) such as number of cockroaches (k), visual range ($visual$), $step$ - a fixed value, and the stopping criterion ($Stop$). The visibility parameter denotes the visual distance of cockroaches. In the third step, we have to initialize the swarm with random solutions. The $i-th$ individual denotes a D-dimensional vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$ for $i = 1, 2$, ..., k. The location of each cockroach is a potential solution, and the objective function value corresponding to each solution is calculated (step 5). After such evaluation, the swarm's global best position is kept and considered as global optimum P_g (step 7). In step 8, the main loop of the CSO approach is started until the stopping criterion is fulfilled. The stopping criterion can include, for example, the maximum number of iterations, the number of iterations without improvement, the computational time, obtaining an acceptable error of a solution.

In the chase-swarming procedure (steps 9 to 18), the locally best individuals form small swarms and follow the best one. In its basic version, the individuals perform movement towards that one that enjoys the best target function value, relating to the range of their visibility. Therefore, the new position of strongest cockroaches P_i moving forward to the global optimum P_g can be computed as (step 12):

$$X_i = X_i + step \cdot rand \cdot \ (P_g - X_i) \tag{7.1}$$

The locally best individuals create small swarms and follow the best one in the group. Note that within this procedure, the new direction of each individual X_i moving to its local optimum P_i in the range of its visibility (step 15) is obtained from:

$$X_i = X_i + step \cdot rand \cdot \ (P_i - X_i) \tag{7.2}$$

There is a possibility that a cockroach moving in a small group will be strongest if it finds a better solution, because individuals follow in other ways than their local optimum. If it is better than P_g with respect to the objective function $OF(.)$, the global solution will be updated (step 20). A lonely cockroach within its own scope of visibility, is its local optimum and it moves forward to the best global solution. In addition, a dispersion procedure is incorporated into the running process (steps 22 to 28). The dispersion of individuals is performed as random movements (step 24) using the formula: $X_i = X_i + rand\,(1, D)$, where $rand\,(1, D)$ is a D-dimensional random vector (D is the space dimension) within a certain range. It is a simulation of the appearance of a light or a brisk movement. In step 26, the position of the best cockroach is updated. From time to time, the ruthless behavior can occur when the current best individual replaces a randomly chosen individual (step 30). This situation corresponds to the depletion of nutritional resources that may lead to cannibalism among cockroaches.

When the stopping criterion has been met, the algorithm will stop and the best position of cockroaches will be returned as the result obtained after all cycles of the CSO.

7.3 Modifications of the CSO algorithm

Since its introduction in 2010 [1], the CSO algorithm has had some improvements aimed at guaranteeing stability and improving diversity. Here, some modifications of the CSO algorithm are briefly described.

7.3.1 Inertia weight

Modified CSO [4] extends CSO and introduces an inertia weight w in chase-swarming behavior. Other procedures remain without changes. Therefore, in the chase-swarming procedure, the cockroach position which moves forward to the global optimum P_g is changed as:

$$X_i = w \cdot X_i + step \cdot rand \cdot \ (P_g - X_i) \tag{7.3}$$

and each individual X_i moves to its local optimum P_i in the range of its visibility as follows:

$$X_i = w \cdot X_i + step \cdot rand \cdot \ (P_i - X_i) \tag{7.4}$$

Here, w is used for affecting the convergence in CSO. While a large inertia weight w supports a global search, a small value favours a local search. According to [4], it is a constant, and the value of this factor usually is from the range [0.5, 1].

7.3.2 Stochastic constriction coefficient

To maintain the stability of a swarm, a stochastic constriction factor (SCF) to control cockroach movements during chase-swarming procedure was proposed in [5]. SCF supports generation of different values as a constriction factor in cycle (each iteration). In order to insure the speed and convergence of the CSO algorithm, the chase-swarming procedure with the introduction of SCF can be expressed as follows:

$$X_i = \begin{cases} \xi(X_i + step \cdot rand \cdot \ (P_i - X_i)), & \text{if } X_i \neq P_i \\ \xi(X_i + step \cdot rand \cdot \ (P_g - X_i)), & \text{if } X_i = P_i \end{cases} \tag{7.5}$$

7.3.3 Hunger component

An improved cockroach swarm optimization (ICSO) including hunger behavior was presented in [6]. It prevents local optimum and enhances diversity of

population. In the algorithm, hunger behavior occurs after the chase-swarming procedure.

A variable t_h is called a hunger threshold. It is defined, when the cockroach is hungry and this threshold is reached. It is a random number [0,1]. The formula for hunger behavior is given by:

$$x_i = x_i + (x_i - ct) + x_f \tag{7.6}$$

where x_i is cockroach position, (x_i-ct) denotes cockroach migration from its current position, c denotes the controlling speed of the migration at time t, and x_f is food location.

7.3.4 Global and local neighborhoods

Because after some iterations all cockroaches are near around P_g, two kinds of neighborhood structures were presented in [7]. One, namely the local structure, where each cockroach current position at the $G-th$ population ($X_{i,G+1}$) is computed by the best position ($P_{i,G}$) found so far in a small neighborhood. In turn, the second structure can be the entire population at the $G-th$ current generation. Therefore, the new position of the $i-th$ cockroach ($F_{i,G}$) in chase-swarming behavior is formulated as follows:

$$F_{i,G} = \begin{cases} X_{i,G} + (P_{i,G} - X_{i,G}) + (X_{r1} - X_{r2}), & \text{if } rand(0,1) < 0.5 \\ X_{i,G} + (P_{g,G} - X_{i,G}) + (X_{r3} - X_{r4}), & \text{otherwise} \end{cases} \tag{7.7}$$

where the indices r_1, r_2, r_3, r_4 belong to [1, the number of cockroaches], $rand(0,1)$ is a uniformly distributed random number.

It should be mentioned that in G+1 generation, $X_{i,G+1}$ is chosen between $X_{i,G}$ and $F_{i,G}$ according to the following selection rule:

$$X_{i,G+1} = \begin{cases} F_{i,G}, & \text{if } OF(F_{i,G}) < OF(X_{i,G}) \text{ for minimization problems} \\ X_{i,G}, & \text{otherwise} \end{cases} \tag{7.8}$$

7.4 Application of CSO algorithm for traveling salesman problem

7.4.1 Problem description

The goal of the traveling salesman problem is to find the path of shorter length or minimum cost between all the requested points (cities), visiting each point exactly once and returning to the starting point. The salesman starts at a point, visiting all the points one by one and returns to the initial point. For a

given set of cities from 1 to N and a cost matrix $C = [c_{i,j}]$, where each value $c_{i,j}$ represents the cost of traveling between cities i and j, each possible tour can be represented as a permutation $\pi = (\pi(1), ..., \pi(N))$, where $\pi(i) \in N$ represents the city visited in step i, $i = 1, ..., N$. Therefore, the goal is to find the minimal cost of the closed tour which minimizes the following objective function [8]:

$$f(\pi) = \sum_{i=1}^{N-1} c_{\pi(i),\pi(i+1)} + c_{\pi(N),\pi(1)} \tag{7.9}$$

If distances $c_{i,j}$ and $c_{j,i}$ between two cities i and j are equal then we have a symmetric TSP, and it can be defined as an undirect graph. If the problem is an asymmetric TSP, it can be defined as a direct graph. If all cities are described by their coordinates (x,y) and distances between cities are Euclidean, we have a Euclidean TSP. As mentioned in Section 7.1, TSP has a great importance in many research and practical problems. It belongs to the class of $NP - complete$ problems and many approximate algorithms are used to solve its instances.

7.4.2 How can the CSO algorithm be used for this problem?

Below, a brief description of the CSO algorithm for solving TSP is provided according to [9]. A correct representation of the individual, distance or the method of the individuals' movement, with the determination of the quality of adjustment, constitute the main elements that require the process of algorithm adaptation to the problem being solved. For the adaptation, owing to the easy implementation, a representation of solutions with the permutation of the set of n cities is used, and the initial positions of individuals is generated randomly.

A cockroach, in the CSO algorithm, is a combination of the visited cities and a decreased fitness value is indicative of better individuals. The basic aspect of the cockroach swarm optimization algorithm adaptation consists in the definition of the scope of visibility and of the movement. When solving a traveling salesman problem, the position of a cockroach represents a solution to the TSP problem, coded as the permutation. It should be emphasized that the solution notation in the form of permutation requires a proper definition of movement. Therefore, the objective function for each individual, and the position of individuals are denoted by $f(\pi)$ and π, respectively. It should be noted that the solution quality is determined by the path length. Hence, upon generation of initial solutions, their quality is estimated. In this stage, a cockroach should evaluate its position itself first, before swarming with others. The smaller the value of $f(\pi)$ is, the better the quality solution is. The purpose of the subsequent steps is to improve solutions and find the path with shorter length. Therefore, k individuals are required to find a closed tour and encounter others in the searching process.

Next, the strategy of updating the current solution is generated within the chase-swarming procedure, where the new position of the cockroach is explored and evaluated. Movement in chase-swarming procedure can be determined by crossover operators known from genetic algorithms. It should be mentioned that traditional crossover methods may not provide a good tour; therefore we have to use such a crossover operator that ensures path modification without cycles or inconsistent components. In [9] partially-mapped crossover (PMX), order crossover (OX), and sequential constructive crossover (SCX) were investigated in the considered procedure. If the path of cockroach j is shorter than the tour of cockroach i, within its *visual*, then cockroach i moves towards j according to crossover operators. On the other case, if the path of cockroach i is local optimum (within its *visual*) then cockroach i goes to P_g (with crossover operator). When those cockroaches finish this procedure, they will compare the quality of the routes visited in the current position and the better one is selected as a candidate solution. It should be empasized that if two cockroaches have no common positions in their permutations, they cannot move with respect to each other.

In the basic version of CSO presented in the literature, the random step is involved in the dispersing procedure. It is helpful to escape from the local optimum. In the TSP, random movement can be represented by the swap operator that generates a new cockroach position by exchanging two randomly selected cities.

As was shown in [9], in order to increase the efficiency of the algorithm, an alternative approach could be used. Such modified dispersion based on 2-opt moves allows for the adaptation of the CSO algorithm to the TSP. The modification consisted in the reduction of the set of pairs of candidate edges, because only selected edges could be exchanged. The selection of edges for exchange was performed randomly, with the assumption that each edge in the graph was exchanged with at least one randomly selected edge. If a new solution obtained as a result of an edge without creating cycles or inconsistent components exchange was better than the previous one, it was saved as a current solution. This procedure can be outlined in the pseudo-code form (Algorithm 10).

Algorithm 10 Pseudo-code of the dispersion procedure.

```
1:  for each individual do
2:      for each position t_a in solution do
3:          choose the random set of edges to remove (lk)
4:          for each edge b = 1 : lk do
5:              remove edge a = (t_a, t_{a+1}) and b
6:              add new pair of edges according to 2-opt moves
7:              calculate the path length
8:              if the new solution is better then
9:                  update current solution
10:             end if
```

11: **end for**
12: **end for**
13: **end for**

The whole searching procedure of the best path ceases if the maximum number of iterations is met.

7.4.3 Description of experiments

The main purpose of the research on the CSO algorithm modification in the traveling salesman problem was to determine the evaluation of the usability of this method. For experiments, the selected benchmarks taken from the library TSPLIB available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ were used. The dimensions of the selected problems range from 51 to 225 cities.

In all cases the population size was considered to be the size of the test problem, *visual* was around 40 % of population size. In turn, the maximum number of iterations was set to 1000. It is important to remark that, in all experiments for each instance, fixed parameters were used during all iterations. During preliminary researches we were interested in testing three crossover operators and finding the best one for a sample instance containing 51 towns ($eil51$). An analysis of the influence of those operators on the quality of the results obtained was made (see [9]). For the random initial cockroach population and dispersion based on the 2-opt moves, it was concluded that the best result was obtained for the sequential crossover (SCX) in the chase-swarming procedure [9]. Therefore, all the results presented in this chapter are based on the SCX operator. For each case the CSO algorithm ran 20 times with the same control parameters. For each instance, global minimum values were computed for each run and compared with the reference values.

7.4.4 Results obtained

Owing to the application of the SCX operator and modification of the dispersion procedure, based on 2-opt movements it is possible to minimise the obtained errors of results. In the majority of the presented test cases, the average values of deviation from the reference values did not exceed 4%, which indicated potential possibilities of the CSO algorithm application in solving the traveling salesman problem.

Table 7.1 summarizes the best values of the path length on selected instances for 20 trials. The first column indicates the test instance, the second column (Ref) shows the reference valus, the third column ($Best$) presents the best solutions found by CSO, the fourth column (Av) gives the average solution of 20 independent runs. The last two columns Dev_best and Dev_av stand for the relative deviation of the found solution from

the reference value (Ref), calculated as $Dev\_best = \frac{Best-Ref}{Ref} \cdot 100\%$ and $Dev\_av = \frac{Av-Ref}{Ref} \cdot 100\%$.

TABLE 7.1
Results of experiments obtained by CSO approach with SCX and directed dispersion.

Instance	Ref	Best	Av	Dev_best [%]	Dev_av [%]
eil51	426	428.83	438.16	0.66	2.85
berlin52	7542	7544.37	7834.12	0.03	3.87
st70	675	677.11	695.53	0.31	3.04
eil76	538	552.39	571.05	2.68	6.14
pr76	108159	109049.00	110962.90	0.82	2.59
kroA100	21282	21381.30	21774.12	0.47	2.31
rat195	2323	2436.80	2499.78	4.90	7.61
ts225	126643	128881.00	129964.15	1.77	2.62

Analyzing the performance of this approach in comparison with the reference solutions (relative deviations of the found solutions from Table 7.1), one can see that for five instances ($eil51$, $berlin52$, $st70$, $pr76$, $kroA100$) CSO with SCX and directed dispersion finds the best solutions near optimal, with values of Dev_best less than 1 %. It should be mentioned that the best found solutions are the same as in [9]. In all of the test instances, the presented approach gives Dev_av between 2.31 % and 7.61 %. Therefore, CSO can produce quite good results, but it requires good modifications of key components and the settings of parameters.

7.5 Conclusions

In this chapter the CSO algorithm, and some of its modifications were presented. As shown in Section 7.4, the CSO algorithm can be used to solve the traveling salesman problem. As described the basic problem of the CSO adaptation to discrete optimization problems consists in the proper definition of the solution and the consequential modification of the individuals' movement performance manner. One may choose to adapt some operators of genetic algorithms. Results of the SCX operator in the chase-swarming procedure, together with the modified dispersion procedure (using the 2-opt movements) are described. Certain adaptations may be designed in such a way as to trigger a specific strategy of movements. When analyzing the results obtained, one can observe that by introducing presented modifications into the framework of the CSO algorithm, this method can produce satisfactory solutions. In [10], the use of other modifications for solving the TSP problem is demonstrated.

References

1. Z.Chen, H. Tang. "Cockroach swarm optimization" in *Proc. of 2nd International Conference on Computer Engineering and Technology (ICCET)*, 2010, pp. 652-655.
2. R. Matai, S.P. Singh, M.L. Mittal. "Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches" in *Traveling Salesman Problem, Theory and Applications*, pp.1-24, InTech 2010.
3. J.M. Wiener, T. Tenbrink. "Traveling salesman problem: The human case" in *KI: Themenheft KI und Kognition*, vol.22, pp. 18-22, 2008.
4. Z. Chen. "A modified cockroach swarm optimization" in *Energy Procedia*, vol. 11, pp. 4-9, 2011.
5. I.C. Obagbuwa, A.O. Adewumi, A.A. Adebiyi. "Stochastic constriction cockroach swarm optimization for multidimensional space function problems" in *Mathematical Problems in Engineering*, Article ID 430949, 2014.
6. I.C. Obagbuwa, A.O. Adewumi. "An improved cockroach swarm optimization" in *The Scientific World Journal*, Article ID 375358, 2014.
7. L. Cheng, Y. Song, M. Shi, Y. Zhai, Y. Bian."A study on improved cockroach optimization algorithm" in *Proc. of the 7th International Conference on Computer Engineering and Networks*, 2017.
8. D. Davendra, I. Zelinka,R. Senkerik, M. Bialic-Davendra. "Chaos Driven Evolutionary Algorithm for Salesman Problem" in *Traveling Salesman Problem, Theory and Applications*, pp. 55-70, InTech 2010.
9. J. Kwiecien. "Use of different movement mechanisms in cockroach swarm optimization algorithm for traveling salesman problem" in *Artificial Intelligence and Soft Computing*, vol. 9693, pp. 484-493, 2016.
10. L. Cheng, Z. Wang, S. Yanhong, A. Guo. "Cockroach swarm optimization algorithm for TSP" in *Adv. Eng. Forum*, vol. 1, pp. 226-229, 2011.