Journal of AI and Data Mining
Vol 4, No 1, 2016, 35-48 ***10.5829/idosi.JAIDM.2016.04.01.05***

Applications of two new algorithms of cuckoo optimization (CO) and forest optimization (FO) for solving single row facility layout problem (SRFLP)

M. Maadi[1], M. Javidnia[2*] and M. Ghasemi[2]

1. Department of Industrial Engineering, Damghan University, Damghan, Iran.
2. Department of Computer Engineering, Damghan University, Damghan, Iran.

Received 30 August 2015; Accepted 10 November 2015
**Corresponding: javidnia.mohammad@gmail.com (M. Javidnia).*

Abstract
Nowadays, due to the inherent complexity of the real optimization problems, it is a challenging issue to develop a solution algorithm to these problems. Single row facility layout problem (SRFLP) is an NP-hard problem of arranging a number of rectangular facilities with varying lengths on one side of a straight line with the aim of minimizing the weighted sum of the distances between all the facility pairs. In this work, the two new algorithms cuckoo optimization (CO) and forest optimization (FO) are applied and compared to solve SRFLP for the first time. The operators of these two algorithms are adapted according to the characteristics of SRFLP, and the results obtained are compared for two groups of benchmark instances of the literature. These groups consist of instances with the number of facilities less and more than 30. The results obtained from the two groups of instances show that the proposed cuckoo optimization algorithm (COA) has a better performance than the proposed forest optimization algorithm (FOA) in both aspects of finding the best solution and the computational time.

Keywords: *Facility Layout Problem (FLP), Single Row Facility Layout Problem (SRFLP), Cuckoo Optimization Algorithm (COA), Forest Optimization Algorithm (FOA).*

1. Introduction

In a facility layout problem (FLP), we wish to arrange a number of facilities in a given space to satisfy an objective function. Single row facility layout problem (SRFLP) is a specific case of FLP, which is the arrangement of n facilities on a line so as to minimize the transportation costs among facilities. SRFLP has attracted significant attention in the recent years [1]. Generally, SRFLP can be described as follows.

Assume that there are n rectangular facilities. They should be arranged on one side of a straight line in a given direction. The parameters involved in the problem are the length l_i(i=1, 2, …, n) of each facility i and ann × n matrix $C = [c_{ij}]$, where c_{ij} is usually the flow between the facilities i and j (i, j=1, 2, …, n with i<j). This matrix is a symmetric matrix. The distance between each pair of facilities is calculated as the distance between their centers. The objective of the problem is to arrange the facilities to minimize the weighted sum of the distances between all the facility pairs. Denoted byΠ_n, the set of all permutations π of N= {1, 2 … n}, SRFLP can be formulated as follows [2]:

$$\left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} d_{ij}^{\pi} \mid \pi \in \Pi_n \right\} \quad (1)$$

where, d_{ij}^{π} is the distance between the facilities i and j with respect to the permutationπ. In This model, all the permutations of N are searched to find a permutation with a minimum objective function value. One should note that due to the symmetrical property of the cost matrix C, there is no difference between c_{ij} andc_{ji}.

In 1974, Garey et al. showed that the minimum linear arrangement problem that is a special case of SRFLP is NP-hard, so by implication SRFLP is NP-hard too[3]. In this paper, with regard to this

model, two algorithms are proposed to solve SRFLP, and they are compared with each other.

In the literature, because of the theoretical attractiveness and numerous practical situations of SRFLP, many papers have been presented in this area. SRFLP has numerous practical applications in the real world. For example, arrangement of rooms in hospitals, departments in office buildings or supermarkets [4], arrangement of machines in flexible manufacturing systems [5], design of warehouse layouts, and assignment of files to disk cylinders in computers storage [6] are some of these applications mentioned in the literature.

In addition to these applications, there are some practical applications of a special case of SRFLP, in which the facilities have unit lengths. These applications include the design of error correcting codes [7], wire length minimization in VSLI design, graph drawing, and reordering of large sparse matrices [8, 9]. Regarding the numerous applications of SRFLP, several methods have been proposed in the literature to solve it. These methods can be categorized into exact, heuristic, and meta-heuristic algorithms. Exact methods have been applied to solve small instances of SRFLP to optimality, and their relaxations have been used to obtain good bounds for larger SRFLP instances. The first exact method used to solve SRFLP was a branch and bound algorithm presented by Simmons in 1969 [4].After that, different exact methods have been reported in the literature including combinatorial branch and bound, mathematical programming [10,4], dynamic programming [6], branch and cut [5,6,11], cutting planes approach [11,12], and semi-definite programming [7,13,14].

Since in this paper we study two meta-heuristic algorithms to solve SRFLP, we tend to review the heuristic and meta-heuristic algorithms suggested in the literature for SRFLP. For more information on the exact methods that have been presented in the literature to solve SRFLP, the researchers are referred to review the paper presented by Kothari and Ghosh [1].

Since the exact algorithms for SRFLP are computationally expensive, they have been applied to relatively small instances, with up to 42 facilities. The heuristic and meta-heuristic algorithms are faster than the exact methods but they do not guarantee an optimal solution [1]. In the recent years, different meta-heuristic algorithms have been applied to solve various engineering problems because of their complexity. For example, references [15-25] present different meta- heuristic algorithms for different engineering problems. In the literature, the heuristic and meta-heuristic algorithms used to solve SRFLP are divided into 2 groups of construction and improvement methods. Construction methods construct the sequence of facilities until a complete permutation is obtained. Neghabat in 1974 presented a constructive algorithm to obtain a complete solution by adding one machine at a time to the end of the current solution [26].A heuristic method based on the eigenvectors of a transformed flow matrix was introduced by Drezner in 1987 [27]. A linear mixed-integer formulation of the SRFLP and a penalty technique to solve it was presented by Heragu and Kusiak in 1991 [28]. Kumar et al. in 1995and Djellab and Gourgand in 2001 introduced a constructive greedy heuristic method to solve SRFLP [29, 30]. Also some heuristic procedures, used to extract a feasible solution to SRFLP from an optimal solution to the semi-definite programming (SDP) relaxation, were presented by Anjos et al. in 2005, Anjos et al. in 2008,Anjos and Yen in 2009, and Hungerlander and Rendle in 2011[7,11,13,31].

The improvement methods start with one or more permutations of facilities as the initial solutions, and improve them until the stopping criteria is reached or the solution cannot be improved. The seven meta-heuristic algorithms tabu search (TS), genetic algorithm (GA), particle swarm optimization (PSO), simulated annealing (SA), ant colony optimization (ACO), scatter search (SS), and imperialist competitive algorithm (ICA) have been reported in the literature to solve SRFLP. Also some hybrid algorithms based on SA, GA, ACO, PSO, etc. have been studied in different papers to solve SRFLP. Romero and Sanchez-Flores in 1990, Heragu and Alfa in 1992, and Gomes de Alvarenga et al. in 2000 applied SA to solve SRFLP [32,33,34]. Solimanpur et al. in 2005 presented an ant colony algorithm to solve this problem [35].Different papers including Datta et al. in 2011 and Ficko et al. in 2004usedGA to solve SRFLP [36,37]. Kumar et al. in 2008 and Kothari and Ghosh in 2014 presented a scatter search algorithm to solve SRFLP [38,39]. Gomes de Alvarenga et al. in 2000 and Samarghandi and Eshghi in 2010 applied TS to solve SRFLP [34,40]. In 2010, Samarghandi et al. presented PSO to solve this problem [41]. Akbari and Maadi and Lian et al. in 20ll proposed an ICA to solve SRFLP [42, 43]. Also there are some hybrid algorithms to solve SRFLP such as a hybrid algorithm based on SA and GA by Braglia in 1996 and a hybrid algorithm based on ACO and PSO by Teo and Ponnambalam in 2008 [44,45].

Cuckoo optimization algorithm (COA) is a new meta-heuristic algorithm introduced by Rajabioun in 2011 [46]. COA has proven its excellent capabilities such as faster convergence and better global minimum achievement rather than other meta-heuristic algorithms [47]. COA is applied to solve different non-linear problems, and has shown good capability in diverse optimization tasks. This algorithm has been tested so far on different types of practical instances in some scopes such as teleportation systems, machine error compensation, noise canceller design, chemical machine process, and machine process [48-55].This algorithm has been much better than the rest of the meta-heuristic algorithms, and has shown its efficiency [47]. This subject can be a motivation to apply COA in other scopes of optimization such as SRFLP.

Forest optimization algorithm (FOA) is a new evolutionary algorithm, which is inspired by few trees in the forest. This algorithm was introduced by Ghaemi and Feizi-Derakhshani in 2014 [56]. Since FOA is a newly introduced algorithm, there is no paper in the literature using it, and the present paper can be a start to apply FOA to solve different optimization problems. Also FOA can solve continuous problems, and this paper presents a forest-based algorithm that solves discrete SRFLP with changing FOA operators regarding the characteristics of SRFLP.

As only seven meta-heuristic algorithms have been reported in the literature to solve SRFLP, this paper introduces the two new COA and FOA to solve SRFLP for the first time. The performances of these two algorithms are also compared to each other using the SRFLP instances of the literature with different sizes.

Our paper is organized as follows. In section 2, the proposed COA to solve SRFLP is presented. Section 3 describes the proposed FOA for solving this problem. In section 4, computational results of the two proposed algorithms are compared, and it is followed by the conclusions in section 5.

2. Proposed cuckoo optimization-based algorithm

As stated earlier, Cuckoo optimization algorithm (COA) is an evolutionary algorithm inspired by the life of a bird named Cuckoo. This algorithm is based on the specific egg-laying and breeding of cuckoos. COA starts with an initial population of cuckoos that have some eggs to lay in some host birds' nest. Those eggs that are more similar to the host birds' eggs survive and can grow to become a mature cuckoo. Other eggs are detected by the host birds and are expelled out. In such cases, cuckoos migrate to places more suitable for generation survival and egg-laying. The number of grown eggs shows the nest suitability of the area. The goal of a cuckoo optimization problem is to find a situation in which a maximum number of eggs are saved. The environmental features and migration of groups of cuckoos hopefully lead them to converge and find the best environment for breeding and reproduction. This best environment is global or the best solution to the problem. In this process, after chicks become mature, they make some groups. Each group has its own typical habitat. The habitat with the best situation is the destination of the cuckoos of other groups. After moving, each group resides in the area near the current location. Regarding the number of each cuckoo eggs and the distance of cuckoo to the best residence, some egg laying radii are assigned to it. After that, cuckoos start to lay eggs in some random nests inside this radius. This process continues until the best position is obtained. In the best position, most of the cuckoo habitats are gathered around the same global solution [46]. Figure 1 shows a COA flowchart.

COA was introduced basically to solve continuous optimization problems. Until now, this algorithm has been modified to solve discrete optimization problems, and has had impressive results. In the next sections, the stages of the proposed COA to solve SRFLP are described. The implementation of COA in SRFLP is as follows.

2.1. Generating initial cuckoo habitat

As the goal of an optimization problem is to find an optimal solution in terms of the variables, a representation pattern that is usually a vector of the decision variables should be defined. For example, in genetic algorithm (GA), this vector is defined as chromosome and in particle swarm optimization (PSO), this array is named particle position. In COA, it is called habitat.

In an N_{var} dimensional optimization problem, a habitat is an array of $1 \times N_{var}$representing the current living position of cuckoos. This array is defined as follows:

$$\text{Habitat} = [x_1, x_2, \dots, x_{N_{var}}] \quad (2)$$

Like the other meta-heuristic algorithms, the profit of a habitat is obtained by evaluation of the profit function f_p at a habitat, as follows:

$$\text{Profit} = f_p(\text{habitat}) = f_p(x_1, x_2, \dots, x_{N_{var}}) \quad (3)$$

As it can be seen, COA is an algorithm that maximizes the benefit function.

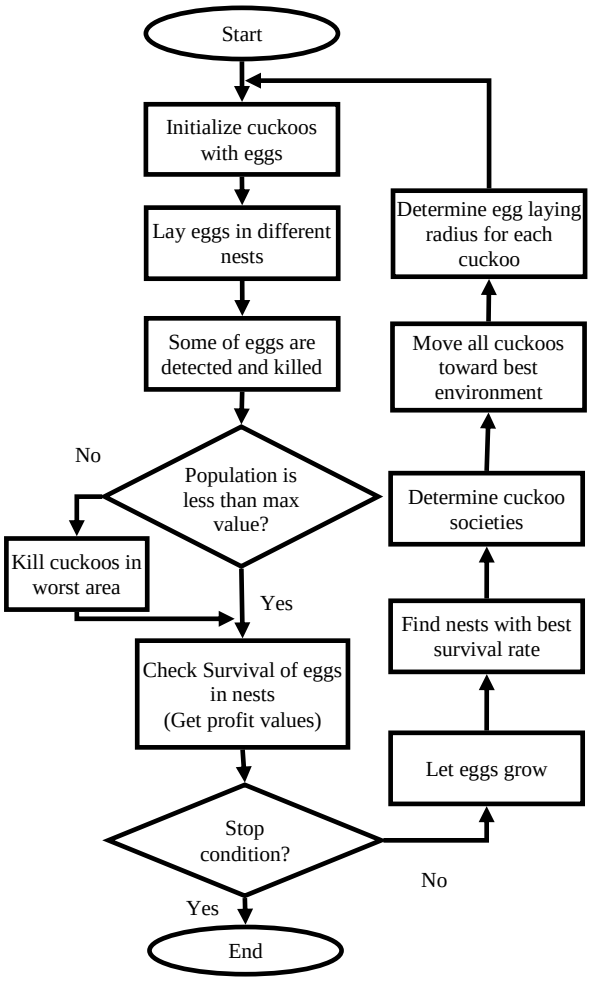

Figure 1. Flowchart of COA.

To use COA for solving the minimization problems, it is sufficient to multiply a negative mark in a cost function as follows:

$$\text{Profit} = f_p(\text{habitat}) = -f_p(x_1, x_2, \dots, x_{N_{var}}) \qquad (4)$$

To start COA, a matrix of size$N_{pop} \times N_{var}$ should be created, and for each habitat of initial population, some randomly-produced eggs are supposed to be assigned. The number of eggs that each cuckoo lays is between 5 and 20.These values are the upper and lower limits for egg-laying at different iterations. Another habit of real cuckoos is that they lay eggs within a maximum distance from their habitat. This maximum range is called the egg laying radius (ELR). In an optimization problem with an upper limit of var_{hi} and a lower limit of var_{low} for the variables, for each cuckoo, an ELR, which is proportional to the total number of eggs, number of current cuckoo's eggs, and also variable limits of var_{hi} and var_{low} is determined. The ELR is defined as follows:

$$\text{ELR} = \alpha \times \frac{\text{Number of current cuckoo's eggs}}{\text{total number of eggs}} \times (var_{hi} - var_{low}) \qquad (5)$$

where, α is an integer, supposed to handle the maximum value for ELR. To solve SRFLP with n facilities using COA, according to the problem definition, a habitat is represented as an n dimensional vector of integer numbers between 1 and n that shows a permutation allocating rectangular facilities to a straight line. For example, figure 2 shows a habitat of COA in SRFLP with 10 facilities.

2	3	6	8	9	10	1	4	7	5

Figure 2. Habitat representation.

Using (1) as the cost function that should be minimized for COA, the profit of each habitat can be calculated using (4). To generate the initial population, two procedures are used in this paper. The first one is based on theorem 1 in the paper presented by Samarghandi and Eshghi [40]. In this procedure, it is assumed that in the cost function coefficients, $c_{ij} = c$. Now, if we sort facilities in a non-descending order such that the shortest facility is denoted by 1 and the longest one by n, then figure 3 shows the optimum solution when n is an odd number, and figure 4 shows the optimum solution when n is an even number. With this procedure and another procedure that randomly generate the permutation allocating the rectangular facilities to a straight line, the initial population is created. After producing the initial population, the process of laying eggs will be started.

Figure 3. Optimal layout when n is odd.

Figure 4. Optimal layout when n is even.

2.2. Cuckoos' style for egg laying

In this stage, each cuckoo starts laying eggs randomly in some other host birds' nests within her ELR. A clear view of this concept is shown in figure 5.

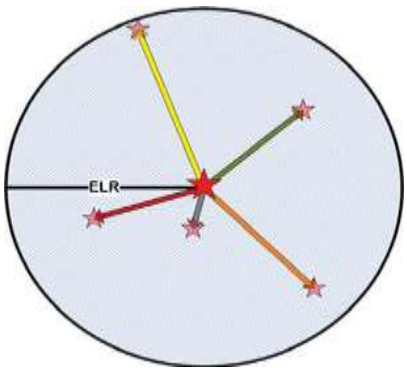

Figure 5. Random egg-laying in ELR. Central red star is the initial habitat of cuckoo with 5 eggs; pink stars are eggs' new nests.

After all cuckoos' eggs are laid in the host birds' nests, some of them that are less similar to the host birds' own eggs are detected by the host birds and thrown out of the nest. Thus after the laying process, p% of the eggs with less profit values will be destroyed. P is usually 10 in this step. The rest of the eggs grow in the host nests and are fed by the host birds. The important point about the cuckoo chicks is that only one egg per nest is allowed to grow because when a cuckoo egg hatches and the chick comes out, she throws out the host bird eggs, and if the host bird chicks are hatched sooner, the cuckoo chicks eat the most food that the host birds bring (because of their3 times bigger bodies, they push other chicks and eat more), and after some days, the host bird chicks starve to death and just the cuckoo chicks survive.

For implementation of this step of COA in SRFLP, in the proposed algorithm, because of the definition of habitat and structure of the problem, there is no ELR, and some changes in the structure of habitats are used for the laying eggs operator. For this aim, the swap operator is used. The steps of the proposed swap operator are as follow:

1- Consider a mature cuckoo habitat for laying eggs.
2- To lay eggs, exchange the position of two facilities of a habitat and make an egg.

Using this operator, the eggs are laid and the algorithm continues to the next section.

2.3. Immigration of cuckoos

After growing young cuckoos, they live in their own area and society for some time. At the time of spawning, migration to new and better habitats, where there is more chance for eggs to survive and more food available for young cuckoos, starts. After forming cuckoo groups in different areas (the search space of the problem), the society with the best profit is selected as the target point of a group for other cuckoos to migrate. When mature cuckoos live all over the environment, it is tough to recognize which cuckoo belongs to which group. To solve this problem, the grouping of cuckoos is done by the K-means clustering method (a k value between3 and 5 seems to be sufficient). After formation of groups, the average profit value is calculated to achieve the relative optimality of that group habitat. After that, the group with the highest amount of profit is selected as the target group, toward which other groups will migrate. During movement toward the target point, the cuckoos do not fly all the way to the destination habitat. They only fly a part of the way and also have deviation. This movement is shown in figure 6. As it is clear in this figure, each cuckoo flies only λ% of the total direction toward the destination habitat, and also has a deviation of φ radians. The two parameters λ and φ help cuckoos to search the environment more. λ is a random number between 0 and 1, and φ is a number between $-\pi/6$ and $\pi/6$. When all cuckoos migrate to the target habitat, and also the new habitats are specified, some eggs are given to the mature cuckoos. After that, considering the number of eggs for each cuckoo, an ELR is specified for it, and a new egg-laying process restarts.

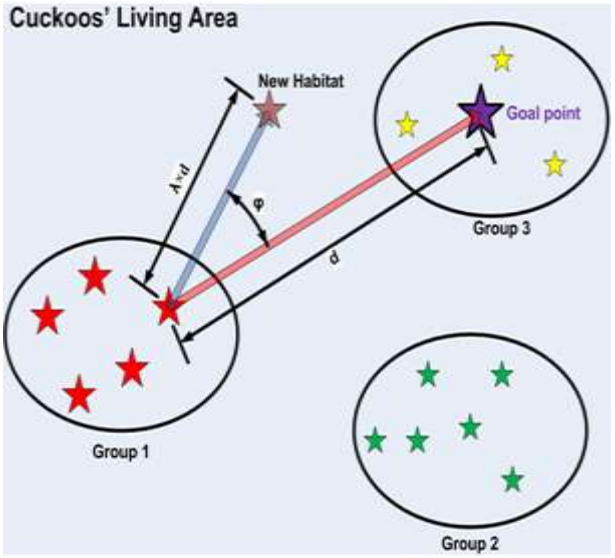

Figure 6. Immigration of a sample cuckoo toward a goal habitat.

To simulate this operator of COA for SRFLP, considering the characteristics of SRFLP, the steps proposed for the immigration operator are applied in the following procedure.

At first, the target habitat toward which the other cuckoos should migrate is considered (a habitat with the highest profit).Then a group of facilities in the target habitat is selected. Figure 7 shows an example of a target habitat with 10 facilities and selected group of facilities of the habitat. An array of a migrant cuckoo who wants to immigrate toward the target habitat is considered (Figure 8). The selected group of facilities that is in target habitat is copied exactly to the array of new position of the migrant cuckoo. For the other facilities in the new position of the migrant cuckoo that are not in the selected group, from the first cell of the migrant cuckoo habitat array, each facility that is not in the facilities of the selected group is placed in the cells of the array of the new position of the migrant cuckoos sequentially. Figure 9 shows the array of the new position of the migrant cuckoo.

8	5	2	1	6	4	3	9	10	7

Figure7. Target habitat and selected group of facilities.

4	7	6	5	10	9	3	8	2	1

Figure 8. A migrant cuckoo habitat.

7	5	10	1	6	4	3	9	8	2

Figure 9. New position of migrant cuckoo.

2.4. Eliminating cuckoos in worst habitats

Since there is always equilibrium in the birds' population in nature (because of food limitation, being killed by predators, and inability to find proper nest for eggs), a number$N_{max}$is determined as the maximum number of cuckoos that can live in the environment. In this step, only those N_{max} number of cuckoos that have better profit values survive, and the others demise. After some iteration, the algorithm moves to one best habitat with the characteristic of maximum similarity of eggs to the host birds and also with the maximum food resources. This habitat has the best profit value, and is the best solution to the problem. In COA, convergence of 95% of cuckoos to the same habitat puts an end to the algorithm. The stopping condition for this algorithm, like other evolutionary algorithms, can be the number of iterations. In the proposed algorithm, the number of iterations is considered as the stopping condition. The pseudo-code of the proposed COA used to solve SRFLP is described in algorithm 1.

Algorithm 1: Proposed COA

Input: number of facilities (*n)*, flow matrix (*F)*, facilities length matrix (*L)*, maximum number of iterations (maxiter), minimum number of eggs (minegg), maximum number of eggs (maxegg).

Output: An approximation of an optimal solution to the PCSP instance.

```
Initialize cuckoo habitats with N individuals using section 2.1.
Compute fitness for each cuckoo
Find best cuckoo as best cuckoo
for i=0 to maxiter
   for each cuckoo
      dedicate some eggs with random number between {minegg , maxegg}
   end

   for each cuckoo
      for each egg of selected cuckoo
         Perform laying operator using section 2.2.
   end

   Sort cuckoo habitats according to their fitness value and update best cuckoo
   Select the best N cuckoos and reduce the extended cuckoos.

   for each cuckoo
      Let new cuckoo populations immigrate toward best cuckoo using section 2.3.
   end
   Compute fitness for each cuckoo and update best cuckoo
end
Return the best cuckoo as the result
```

3. Proposed forest optimization-based algorithm

Forest optimization algorithm (FOA) is a new evolutionary algorithm that is inspired by few trees in the forest which can survive for several decades, while other trees could live for a limited period. As mentioned earlier, this algorithm was introduced by Ghaemi and Feizi-Derakhshani in 2014 [56].

FOA involves three main stages: local seeding of the trees, population limiting, and global seeding of the trees. Like other meta-heuristic algorithms, FOA starts with the initial population named trees. Each tree represents a solution of the optimization problem. A tree besides the cells of variables of the problem has a part that represents the age of the tree.

At first, the age of a tree is set '0'. After initialization of the trees, the operator of local seeding will generate new young trees (or seeds in fact) from the trees with age 0 and add new trees to the forest. Then all trees, except the new generated ones, get old, and their age increases by '1'. The next step is the population limiting. At this step, there is a control on the population of trees in the forest, and some trees will be omitted from the forest. The omitted trees form the candidate population for the global seeding stage.

In the third stage, which is the global seeding of the trees, a percentage of the candidate population is chosen to move far in the forest. The aim of global seeding is adding some new potential solutions to the forest in order to get rid of the local optimums. After that, the trees are ranked with regard to their fitness values. The tree with the best value of fitness function is chosen as the best tree, and its age is set '0' in order to avoid the aging of the best tree as the result of local seeding stage. In this way, it will be possible for the best tree to locally optimize its location by the local seeding operator.

These 3 stages will be continued until the stopping conditions are met. Figure 10 shows the flowchart of the forest optimization algorithm. As mentioned earlier, since FOA is a new introduced algorithm, there is no paper in the literature that uses this algorithm, and this paper can be regarded as a start to apply FOA to solve different optimization problems. Also as FOA is introduced to solve continuous non-linear programming, this paper introduces a discrete version of FOA to solve SRFLP. In the next sections, using operators of FOA, a forest optimization-based algorithm is introduced to solve SRFLP.

3.1. Initialize trees

As mentioned earlier, in FOA, the potential solution to each problem is considered as a tree. A tree is usually a vector of variables, and the variables have lower and upper limits. In addition to the variables, each tree has a part related to the age of the tree. The age of a tree for each newly generated tree is set '0' as a result of local seeding or global seeding. After the local seeding stage, the age of the trees, except for the new generated ones in the local seeding stage, increases by '1'. This increase in age is used later as a controlling population in the limiting stage. Thus in FOA, a tree is represented as an array of $1 \times (N_{var} + 1)$, where N_{var} is the diminution of the optimization problem, and one cell is considered as the age of a tree. Equation 6 represents a tree.

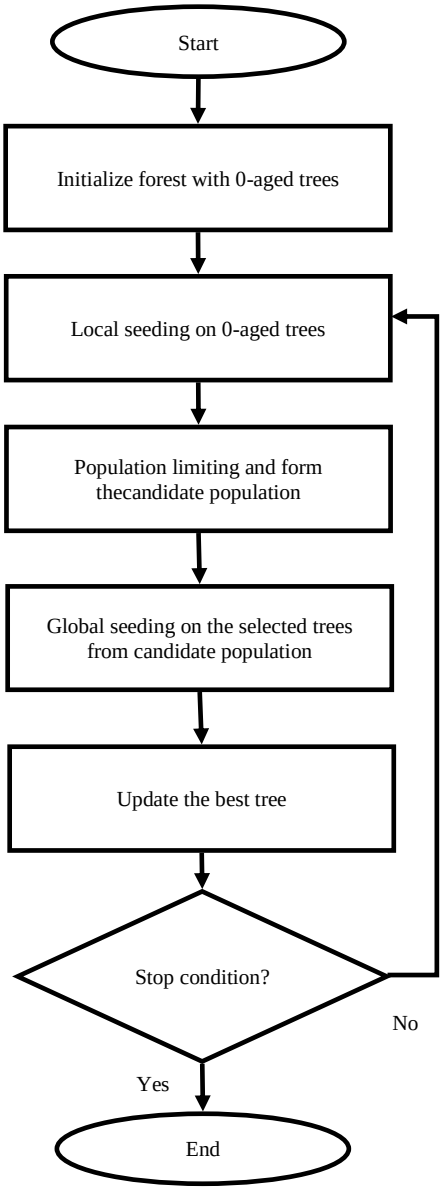

Figure 10. Flowchart of FOA.

$$\text{Tree} = [x_1, x_2, \ldots, x_{N_{var}}, \text{Age}] \quad (6)$$

In the definition of a tree, the age of a tree has a maximum range that is a predefined parameter named lifetime. This parameter should be determined at the beginning of the algorithm. When the age of a tree reaches the lifetime parameter, that tree is omitted from the population of the trees, and is added to the candidate population. The big number of lifetime parameter increases the age of the trees in each iteration, and the forest will be full of the old trees that do not take part in the local seeding stage. Otherwise, a very small value for this parameter causes the trees to get old very soon, and they will be omitted at the beginning of the competition. Thus this parameter should provide a good chance for a local search. The best lifetime value was determined to be 6 in FOA.

In SRFLP, a tree is represented as an n+1 dimensional vector, whose first n cells are integer numbers of 1 ton that represent a permutation, allocating rectangular facilities to a straight line, and the last cell is the age. Figure 11 shows a tree with 10 facilities.

4	7	6	5	10	9	3	8	2	1	0

Figure 11. A tree representation in SRFLP with 10 facilities and age of zero.

The fitness function for FOA is the same as that for COA, which is (1) and should be minimized. For generating the initial population, the same procedures used in COA are applied for FOA. All the initial trees in this step have the age '0'. With this population, the algorithm starts the next stage, which is the local seeding of the trees.

3.2. Local seeding of the trees

In the nature, during the seeding procedure of the trees, some seeds fall just near the trees, and after some time, they will grow. Now, the competition between the near-young trees starts, and those trees with better growing conditions such as enough sunlight and better location survive. The local seeding of the trees is simulated regarding this natural event. This operator is performed on the trees with the age '0', and adds some neighbors of each tree to the forest. The number of seeds that fall on the land near the trees and then turn into the trees as neighbors is considered as an FOA parameter named the local seeding changes or LSC. The value of this parameter used to solve different optimization problems depends on the dimension of the problem domain. In FOA, for problems with a dimension bigger than 5, 2/10 is recommended for LSC, and for dimensions less than 5, the LSC value will be 1. In local seeding

operator, at first, a variable of a tree with the age '0' is selected. After that, a random number named r is generated in the range of$(-\Delta x, \Delta x)$. Then the value for the selected variable is added to r. Δx is a small value, and it is smaller than the related variable's upper limit. Now, a new tree with age '0' is added to the forest.

For simulating the local seeding operator to solve SRFLP, the swap operator is used to generate new trees. In this operator, for each tree with age '0', two facilities are selected randomly, and their locations are exchanged. With this operator and using the LSC parameter, many new trees are added to the forest. In the next stage of the algorithm, a limitation on the number of trees should be considered as the operator of population limiting.

3.3. Population limiting

To avoid infinite expansion of the forest, after generating numerous trees in the stage of local seeding of the trees, the number of trees should be limited. In FOA, the two parameters lifetime and area limit restrict the population of the trees. At first, the trees with an age more than the lifetime parameter are removed, and will form the candidate population.

The second limitation is the area limit, in which after ranking the trees according to the fitness value, if the number of trees exceeds the limitation of the forest, extra trees are omitted from the forest and added to the candidate population. The value for the area limit parameter is considered to be the same as the number of the initial trees. After population limiting, the global seeding stage is performed on some percentage of the candidate population (described in the next section).

3.4. Global seeding of trees

There are different kinds of trees, and numerous animals and birds feed on the seeds and fruits of these trees in the nature. Thus in spite of local seeding of the trees, seeds are distributed in the entire forest, and consequently, the habitat of the trees expands. Also different natural processes such as wind and water flow help distributing the seeds in the entire forest widely. Using these natural processes, the global seeding stage is defined to simulate the distribution of the seeds of the trees in the forest.

The global seeding operator is performed on a determined percentage of the candidate population from the previous stage. This percentage is another parameter of the algorithm named transfer rate, and should be defined at the beginning of the algorithm.

The steps of global seeding of the trees are defined as follows. Regarding the transfer rate parameter, the trees from the candidate population are selected. Then some variables of each tree are selected randomly. After that, the value for each selected variable is changed with another randomly-generated value in the related variable range. As a result, a tree with age '0' is added to the forest. This operator performs a global search in the search space. During this stage, another parameter that is the number of variables, whose values will be changed, is defined as the global seeding changes or GSCs.

During the global seeding operator in the proposed algorithm to solve SRFLP, after determination of GSCs, the swap operator is carried out. The number of repetition of the swap operator is the number of GSCs. After producing new trees, these trees with age '0' are added to the forest.

3.5. Updating the best so far tree

In this stage, after sorting trees according to the value of their fitness function, the tree with the best fitness function value is selected as the best one. After that, the age of this tree turns to '0'. As mentioned earlier, in this way, it will be possible for the best tree to locally optimize its location by the local seeding operator.

3.6. Stop conditions

In FOA, the three stopping conditions predefined number of iterations, reaching the specified level of accuracy, and observance of no change in the fitness value of the best tree are defined. In the proposed forest optimization-based algorithm, the number of iterations is considered as the stopping condition. The pseudo-code of the proposed FOA to solve SRFLP is described in algorithm 2.

4. Comparison of cuckoo- and forest-optimization algorithms

Cuckoo optimization (CO) and forest optimization (FO) algorithms are coded in C#. The algorithms are run on an Intel (R) core(TM) i5-3210 CPU @2.05 Gigahertz and 4.00Gigabytes ram under the Windows 8.1 operating system.

The performance of the two algorithms is evaluated on several benchmark instances. These instances are divided into two groups of instances with the number of facilities less than 30, and instances with the number of facilities more than 30. All the instances are available at http://www.gerad.ca/files/sites/Anjos/flplib.html.

For instances with the number of facilities less than 30, the maximum number of iterations is set150, and for the other group of instances, it is considered as 300. For each instance, each algorithm is run 20 times. In the following section, the computational experience of the two algorithms for the two groups of instances is reported.

Algorithm2: Proposed FOA

Input: number of facilities (*n)*, flow matrix (*F)*, facilities length matrix (*L)*, number of initial trees (*N)*, maximum number of iterations (maxiter), local seeding change (LSC), global seeding change(GSC), transfer rate, life time, area limit.

Output: An approximation of an optimal solution to the SRFLP instance.

```
Initialize forest with N individuals using section 2.1.
The age of each tree is initially set zero.

For i=0 to maxiter
    for each trees with age 0
        for j=0 to LSC
            Perform local seeding on selected trees using section
            3.2.
        end
    end
    Increase the age of all trees by 1, except for the newly
    generated trees in this stage.
    Remove the trees with age bigger than lifetime parameter and
    add them to the candidate population.
    Sort trees according to their fitness value.
    Remove the extra trees that exceed the area limit parameter from
    the end of forest and add them to the candidate population.
    Choose transfer rate percent of the candidate population.
    For each selected tree
        Perform global seeding using section 3.4.
        Set the age 0
    end
end
Return the best tree as the result.
```

4.1. Instances with number of facilities less than 30

Initially, the algorithms are tested on some instances with $n \leq 30$ in the literature. The optimum solutions to this set of instances are known. Throughout the experiments, the following parameter values are used for FOA and COA, respectively.

For COA, the initial population of cuckoos is set 30. The lower limit for egg-laying is considered 3, and the upper limit for egg-laying is set 5. In FOA, the number of initial trees is considered to be 30, GSC= $1+\left\lfloor \frac{n}{10} \right\rfloor$, LSC=5, and the transfer rate is set (0.9*n).

The results obtained for the application of COA and FOA are shown in table 1. In this table, the first column shows the problem number. The number of facilities of instances is presented in the second column. The next column is the objective function value of optimum solution of instances from the literature. Columns 4, 5, and 6 are related to COA. Column 4 is the objective function value of the achieved COA solution. Column 5 shows the computational time of COA, and column 6 calculates the gap between objective function values of optimum solution and achieved COA solution of the instances.

As it can be seen in table 1, the proposed COA is able to achieve the best solutions reported in the literature. The next three columns 7, 8, and 9 are the results of applying FOA. In column 7, the value of objective function of the achieved FOA solution is shown. Column 8 shows the computational time of FOA, and column 9 calculates the gap between objective function values of optimum solution and achieved FOA solution of the instances. This gap is calculated as follows:

$$\text{Gap} = \frac{\text{achievedFOAsolution} - \text{optimumsolution}}{\text{optimumsolution}} \quad (7)$$

Regarding table 1, the average gap between objective function values of optimum solutions and achieved FOA solutions of all instances is 0.0017.

In the comparison of the two proposed algorithms, it is notable that the proposed COA has a better performance than FOA in both aspects of achieving the best solution and computational time. Table 2 shows the mean and standard deviation (SD) of objective function values of the instances. As mentioned earlier, each instance is run 20 times. Figure 12 depicts and compares the computational times of the two algorithms.

In figure 12, the horizontal axis shows the computational time, and the vertical axis depicts the problem number.

4.2. Instances with number of facilities more than 30

The same computational results for the second set of problems with the number of facilities more than 30 are shown in table 3.

According to this table, COA with the average gap of 0.0013 has a better performance than FOA with the average gap of 0.0017. Also the computational time of COA is less than FOA for all instances, and these results demonstrate a better performance of COA to solve SRFLP.

Table 4 shows the mean and standard deviations of the objective function values of instances of set 2. As mentioned earlier, each instance is run 20 times.

Figure 13 shows and compares the difference between the computational times of the two algorithms. In figure 13, the horizontal axis shows the computational time, and the vertical axis depicts two algorithms.

Table 1. Results for problem set1.

Problem Number	Number of Facilities	Optimal Value	FOA Result	FOA Time (Sec)	FOA Gap	COA Result	COA Time(Sec)	COA Gap
1	4	**78**	**78**	0	0	**78**	0	0
2	4	**638**	**638**	0	0	**638**	0	0
3	8	**801**	**801**	0	0	**801**	0	0
4	8	**2324.5**	**2324.5**	0	0	**2324.5**	0	0
5	9	**2469.5**	**2469.5**	0	0	**2469.5**	0	0
6	9	**4695.5**	**4695.5**	0	0	**4695.5**	0	0
7	10	**2781.5**	**2781.5**	0	0	**2781.5**	0	0
8	11	**6933.5**	**6933.5**	0	0	**6933.5**	0	0
9	15	**63.05**	**63.05**	0	0	**63.05**	0	0
10	20	**15549**	15592	1	0.002765451	**15549**	1	0
11	25	**4618**	4625	4	0.001515808	**4618**	3	0
12	25	**37116.5**	37129.5	12	0.000350249	**37116.5**	9	0
13	25	**24301**	24325	12	0.000987614	**24301**	8	0
14	25	**48291.5**	48344	14	0.001087148	**48291.5**	9	0
15	25	**15623**	15633	14	0.000640082	**15623**	10	0
16	30	**8247**	8313	25	0.00800291	**8247**	18	0
17	30	**21582.5**	21725.5	25	0.006625738	**21582.5**	17	0
18	30	**45449**	45627	24	0.003916478	**45449**	17	0
19	30	**56837.5**	57013.5	25	0.003096547	**56837.5**	18	0
20	30	**115268**	116067	26	0.006931672	**115268**	17	0

Table 2. Mean and SD results for instances of set one in 20 times of run.

Problem number	COA-Mean	COA-SD	FOA-Mean	FOA-SD
1	78	0	78	0
2	638	0	638	0
3	801	0	801	0
4	2327.8	3.012474066	2331.7	5.380520421
5	2474.1	5.412947441	2475.8	6.66708332
6	4702.1	5.319774431	4704.7	5.826662853
7	2786.7	4.236744033	2789.7	6.300793601
8	6941.5	4.716990566	6943.5	5.979130372
9	63.05	0	63.05	0
10	15588	22.25982929	15607.4	14.27585374
11	4622.8	6.57267069	4633	9.082951062
12	37154.6	39.49272085	37171.8	38.6144403
13	24330.4	26.83840532	24358	35.46124645
14	48336.6	57.97995343	48388.4	45.04220243
15	15661.2	35.10982768	15684.6	29.16847613
16	8268.8	20.31501907	8328.6	13.50185172
17	21599.6	18.84608713	21764.8	32.21335127
18	45513	39.91240409	45713.4	79.10625766
19	56928.5	70.48049376	57128.3	78.31714372
20	115467.2	177.3899659	116325.4	264.8882783

Table 3. Results for problem set2.

Problem Number	Number of Facilities	Best known value from literature	FOA Result	FOA Time (Sec)	FOA Gap	COA Result	COA Time(Sec)	COA Gap
1	40	107348.5	107886	39	0.005007056	107536.5	25	0.001751305
2	40	97693	97693	36	0	97693	24	0
3	40	78589.5	78689.5	37	0.001272435	78686.5	25	0.001234262
4	40	76669	77295	35	0.008164969	77191	23	0.006808488
5	40	103009	103127	35	0.001145531	103018	22	8.7371E-05
6	60	1477834	1500899	110	0.015607301	1477834	69	0
7	60	648337.5	649097.5	114	0.001172229	648792	69	0.000701024
8	60	841792	851120	109	0.011081122	841792	68	0
9	60	398468	398710	120	0.000607326	398480	69	3.01153E-05
10	60	318805	320678	114	0.005875065	318805	68	0
11	70	1518993.5	1530590	187	0.007634332	1530191	97	0.007371658
12	70	1441028	1443709	189	0.001860477	1441467	100	0.000304644
13	70	1518993.5	1527075.5	188	0.005320628	1520453.5	101	0.000961163
14	70	968796	969880	187	0.001118915	969464	98	0.000689516

15	70	4218017.5	4227125.5	190	0.002159308	4218797.5	101	0.000184921

Table 4. Mean and SD results for instances of set two in 20 times of run.

Problem number	COA-Mean	COA-SD	FOA-Mean	FOA-SD
N40_1	107791.8	237.6574741	107994.4	226.8453217
N40_2	97957.2	331.4131862	97997.2	414.8670872
N40_3	78997.1	414.9904216	79077.9	289.8569475
N40_4	77356.4	156.4106774	77598	310.6565628
N40_5	103483.6	427.486608	103765.4	586.7161153
N60_1	1478014	402.4922359	1501299	547.7225575
N60_2	648976.1	694.2137999	649159.4	652.3557695
N60_3	842162.4	828.2395789	851520	894.427191
N60_4	398775.8	264.9607518	399198	681.5937206
N60_5	318944.2	311.2606625	320707	884.421845
N70_1	1530418.2	260.7799843	1531022.8	276.3099712
N70_2	1441707	328.6335345	1444283.6	413.8892362
N70_3	1521250.7	711.4087785	1528030.2	962.3980856
N70_4	969872.4	704.0914714	970038.8	614.6276921
N70_5	4218891.9	544.4941926	4227575.3	640.7808317

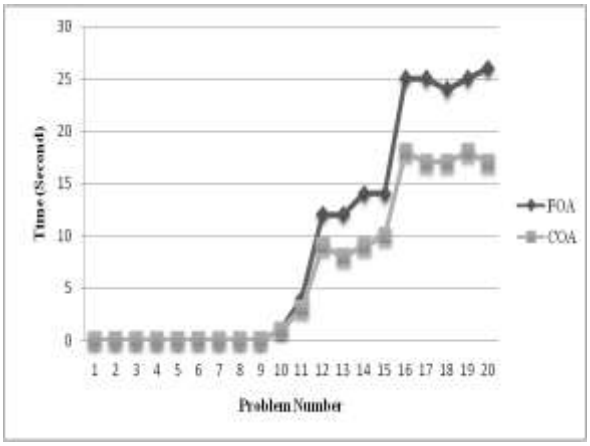

Figure 12. Comparison of computational times of two CO- and FO-based algorithms for instances with number of facilities less than 30.

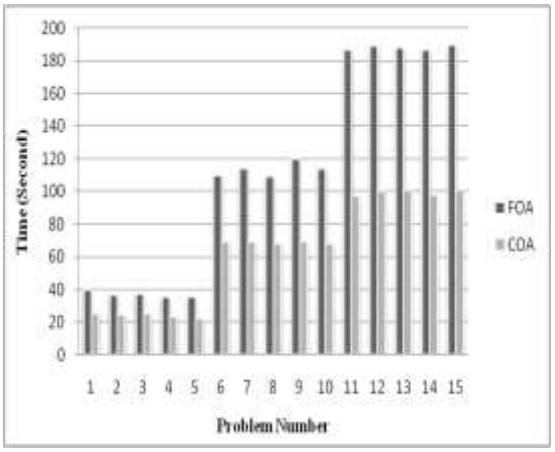

Figure 13. Comparison of computational times of two CO- and FO-based algorithms for instances with number of facilities more than 30.

5. Conclusion

This paper considers the single row facility layout problem (SRFLP). Some exact and heuristic methods have been developed in the recent years to solve this problem but, as this problem is NP-hard, finding optimum solutions for the large instances of this problem is not possible within a reasonable time. Nowadays different meta-heuristic algorithms are introduced in the literature to solve SRFLP but we still need to have a fast algorithm that can obtain a near-optimum solution for large instances of the problem. Cuckoo optimization (CO) and forest optimization (FO) algorithms are two meta-heuristic algorithms that have been introduced recently.

In the literature, these algorithms have not been applied to solve SRFLP yet. In this paper, the performance of COA and FOA to solve SRFLP is compared. As COA and FOA are defined to solve continuous optimization problems in the literature, to solve SRFLP that is a discrete problem, at first, a cuckoo-based algorithm and a forest-based algorithm are proposed. Then to test the performance of these two algorithms, two groups of instances with the number of facilities less and more than 30 are used. The computational results obtained for the two algorithms show that COA has a better performance than FOA in both aspects of achieving the best solution and computational time.

Reference

[1] Kothari, R. & Ghosh, D. (2012). The single row facility layout problem: state of the art, OPSEARCH, vol. 49, no. 4, pp. 442-462.

[2] Amaral, A. R. S. (2006). On the exact solution of a facility layout problem, European Journal of Operational Research, vol. 173, no. 2, pp.508–518.

[3] Garey, M. R., Johnson, D. S. &Stockmeyer, L. (1974). Some simplified NP-complete problems, Proceedings of the 6th Annual ACM Symposium on Theory of Computing, New York, United States, pp. 47–63.

[4] Simmons, D. M. (1969). One-dimensional space allocation: an ordering algorithm, Operations Research, vol. 17, no. 5, pp. 812–826.

[5] Heragu, S. S. & Kusiak, A. (1988). Machine layout problem in flexible manufacturing systems, Operations Research, vol. 36, no. 2, pp. 258–268.

[6] FPicard, J. C. & Queyranne, M. (1981). On the one-dimensional space allocation problem, Operations Research, vol. 29, no. 2, pp. 371–391.

[7] Anjos, M. F., Kennings, A. & Vannelli, A. (2005). A semidefinite optimization approach for the single row layout problem with unequal dimensions, Discrete Optimization, vol. 2, no. 2, pp. 113–122.

[8] Díaz, J., Petit, J. & Serna, M. (2002). A survey of graph layout problems, ACM Computing Surveys, vol. 34, no. 3, pp. 313–356.

[9] Petit, J. (2003). Experiments on the minimum linear arrangement problem, Journal of Experimental Algorithmics, vol. 8, Article 2.3.

[10] Heragu, S. S. & Kusiak, A. (1991). Efficient models for the facility layout problem, European Journal of Operational Research, vol. 53, pp. 1–13.

[11] Anjos, M. F. & Vannelli, A. (2008). Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes, INFORMS Journal on Computing, vol. 20, no. 4, pp. 611–617.

[12]Amaral, A. R. S. (2009). A new lower bound for the single row facility layout problem, Discrete Applied Mathematics. vol. 157, no. 1, pp. 183–190.

[13] Anjos, M. F. & Yen, G. (2009). Provably near-optimal solutions for very large single-row facility layout problems, Optimization Methods and Software. vol. 24, no. 4–5, pp. 805–817.

[14] Hungerländer, P. & Rendl, F. (2011). Semidefinite relaxations of ordering problems, Mathematical Programming, Series B, Vol. 140, no.1, pp. 77-97.

[15] Mousavi, Y. & Alfi, A. (2015). A memetic algorithm applied to trajectory control by tuning of fractional order proportional-integral-derivative controllers, Applied Soft Computing, vol. 36, pp.599–617.

[16] Arab, A, & Alfi A. (2015).An adaptive gradient descent-based local search in memetic algorithm for solving engineering optimization problems, Information Sciences, vol. 299, pp. 117–142.

[17] Alfi, A & Fateh, M. M. (2011). Intelligent identification and control using improved fuzzy particle swarm optimization, Expert Systems with Applications, vol. 38, pp.12312-12317.

[18] Alfi, A, & Modares, H. (2011). System identification and control using adaptive particle swarm optimization, Journal of Applied Mathematical Modeling, vol. 35, pp.1210-1221.

[19] Alfi, A. (2014). Swarm-based structure-specified controller design for bilateral transparent teleoperation systems via μ synthesis, IMA Journal of Mathematical control and Information. vol. 31, pp. 111–136.

[20] Alfi, A. & Fateh, M. M. (2011). Identification of nonlinear systems using modified particle swarm optimization: A hydraulic suspension system, Journal of Vehicle System Dynamics, vol. 46, no. 6, pp. 871-887.

[21] Alfi, A., Kalat, A. & Khooban, M. H. (2014). Adaptive fuzzy sliding mode control for synchronization of uncertain non-identical chaotic systems using bacterial foraging optimization, Journal of Intelligent and Fuzzy Systems, vol. 26, pp. 2567–2576.

[22] Alfi, A. & Khosravi, A. (2012). Constrained nonlinear optimal control via a hybrid BA-SD, International Journal on Engineering, vol. 25, no. 3, pp. 197-204.

[23] Alfi, A. (2011). PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems, ActaAutomatica, vol. 37, no. 5, pp. 541-549.

[24] Khooban, M. H, Alfi, A., Nazeri, D. & Abadi, M, (2013). Teaching-learning-based optimal interval type-2 fuzzy PID controller design: A nonholonomic wheeled mobile robots, Robotica, vol. 31, no.7, pp. 1059-1071.

[25] Shahamat, H. & Pouyan, A. A. (2015).Feature selection using genetic algorithm for classification of schizophrenia using fMRI data, Journal of AI and Data Mining. Vol. 3, no. 1, pp. 30-37.

[26] Neghabat, F. (1974). An efficient equipment layout algorithm, Operations Research, vol. 22, pp. 622–628.

[27] Drezner, Z. (1987). A heuristic procedure for the layout of a large number of facilities, Management Science, vol.7, no. 33, pp. 907–915.

[28] Heragu, S. S. & Kusiak, A. (1991). Efficient models for the facility layout problems, European Journal of Operational Research, vol. 53, pp. 1–13.

[29] Kumar, K. R., Hadjinicola, G. C. & Lin, T. L. (1995). A heuristic procedure for the single row facility layout problem, European Journal of Operational Research, vol.87, no.1, pp. 65-73.

[30] Djellab, H. & Gourgand, M. (2001). A new heuristic procedure for the single-row facility layout problem, International Journal of Computer Integrated Manufacturing, vol. 14, pp. 270–280.

[31] Hungerländer, P. & Rendl, F. (2011). A computational study for the single-row facility layout problem, Technical Report, Alpen-Adria-Universität Klagenfurt, Austria.

[32] Romero, D. & anchez-Flores, A. S. (1990). Methods for the one-dimensional space allocation

problem, Computers & Operations Research, vol. 17, pp. 465-473.
[33] Heragu, S. S. & Alfa, A. S. (1992). Experimental analysis of simulated annealing based algorithms for the facility layout problem, European Journal of Operational Research, vol. 57, pp. 190–202.

[34] Gomes de Alvarenga, A., Negreiros-Gomes, F. J. & Mestria, M. (2000). Meta-heuristic methods for a class of the facility layout problem, Journal of Intelligent Manufacturing, vol. 11, pp. 421–430.

[35] Solimanpur, M., Vrat, P. & Shankar, R. (2005).An ant algorithm for the single row layout problem in flexible manufacturing systems, Computers and Operations Research, vol. 32, pp. 583–598.

[36] Datta, D., Amaral, A. R. S. & Figueira, J. R. (2011). Single row facility layout problem using a permutation based genetic algorithm. European Journal of Operational Research., vol. 213, no.2, pp. 388–394.

[37] Ficko, M., Brezocnik, M. & Balic, J. (2004), Designing the layout of single- and multiple-rows flexible manufacturing system by genetic algorithms, Journal of Material Processing Technology, vol. 157-158, pp. 150–158.

[38] Kumar, S., Asokan, P., Kumanan, S. & Varma, B. (2008). Scatter search algorithm for single row layout problem in fms. Advances in Production Engineering & Management, vol. 3, no. 4, pp. 193–204.

[39] Kothari, R. & Ghosh, D. (2014). A scatter search algorithm for the single row facility layout problem, Journal of Heuristics, vol. 20, pp.125–142.

[40] Samarghandi, H. & Eshghi, K. (2010). An efficient tabu algorithm for the single row facility layout problem, European Journal of Operational Research, vol. 205, no. 1, pp. 98–105.

[41] Samarghandi, H., Taabayan, P. & Jahantigh, F. F. (2010). A particle swarm optimization for the single row facility layout problem, Computers and Operations Research, vol.58, no. 4, pp. 529–534.

[42] Akbari, M. & Maadi, M. (2011). Imperialist competitive algorithm for solving single row facility layout problem, 4th International Conference of Iranian Operations Research Society, Rasht, Guilan.

[43] Kunlei, L., Chaoyong, Z., Liang, G. & Xinyu, S. (2011). Single row layout problem using an imperialist competitive algorithm, Proceedings of the 41st International Conference on Computers & Industrial Engineering. Los Angeles, United States, pp. 578-586.

[44] Braglia, M. (1996). Optimization of a simulated annealing-based heuristic for single row machine layout problem by genetic algorithm, International Transactions in Operational Research, vol. 1, no. 3, pp. 37–49.

[45] Teo, Y. T. & Ponnambalam, S. G. (2008). A hybrid ACO/PSO heuristic to solve single row layout problem. In 4th IEEE conference on automation science and engineering. Washington, DC, United States.

[46] Rajabioun, R. (2011). Cuckoo optimization algorithm, Applied soft computing, vol. 11, no .8, pp. 5508-5518.

[47] Mahmoudi, S. & Lotfi, S. (2015). Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem. Applied soft computing, vol. 33, pp. 48-64.

[48] Shokri-Ghaleh, H. & Alfi, A. (2014). A comparison between optimization algorithms applied to synchronization of bilateral teleoperation systems against time delay and modeling uncertainties, Applied Soft Computing, vol. 24, pp. 447-456.

[49] Khajeh, M. & Golzary, A. R. (2014). Synthesis of zinc oxide nanoparticles-chitosan for extraction of methyl orange from water samples: Cuckoo optimization algorithm-artificial neural network, Spectrochimicaacta. Part A, Molecular and biomolecular spectroscopy, vol. 131, pp.189-194.

[50] Stryczek, R. (2014). A meta-heuristic for fast machining error compensation, Journal of Intelligent Manufacturing. pp. 1-12, doi: 10.1007/s10845-014-0945-0.

[51] Ahirwal, M. K. Kumar, A. & Singh. G. K. (2013). EEG/ERP adaptive noise canceller design with Controlled Search Space (CSS) approach in cuckoo and other optimization algorithms, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), vol. 10, pp. 1491-1504.

[52] Abolpour, B. & Mohebbi, A. (2013). Estimation of the compressive strength of 28-day-old concrete by use of an adaptive cuckoo–fuzzy logic model, Research on Chemical Intermediates, vol. 39, pp. 4001-4009.

[53] Teimouri, R. & Sohrabpoor, H. (2013). Application of adaptive neuro-fuzzy inference system and cuckoo optimization algorithm for analyzing electro chemical machining process, Frontiers of Mechanical Engineering, vol. 8, pp.429-442.

[54] Mellal, M. A. & Williams, E. J. (2014). Parameter optimization of advanced machining processes using cuckoo optimization algorithm and hoopoe heuristic, Journal of Intelligent Manufacturing, pp. 1-16.doi: 10.1007/s10845-014-0925-4.

[55] Shokri-Ghaleh, H. & Alfi, A. (2014). Optimal synchronization of teleoperation systems via cuckoo optimization algorithm. Nonlinear Dynamics, vol. 78, no. 4, pp. 2359-2376.

[56] Ghaemi M. & F .Derakhshani, M.(2014). Forest Optimization Algorithm, Expert Systems with Applications, vol. 41, no. 15, pp. 6676-6687.

پژوهش عملیاتی و داده‌کاوی

دانشگاه شاهرود

کاربرد دو الگوریتم جدید بهینه‌سازی فاخته و بهینه‌سازی جنگل برای حل مسأله‌ی چیدمان تسهیلات تک‌ردیفه

منصوره معادی[1]، محمد جاویدنیا[2,*] و مسعود قاسمی[2]

[1] گروه مهندسی صنایع، دانشگاه دامغان، دامغان، ایران.

[2] گروه مهندسی کامپیوتر، دانشگاه دامغان، دامغان، ایران.

ارسال ۲۰۱۵/۰۸/۳۰؛ پذیرش ۲۰۱۵/۱۱/۱۰

چکیده:

امروزه با توجه به پیچیدگی ذاتی مسائل بهینه‌سازی واقعی، همواره توسعه‌ی یک الگوریتم برای حل این مسائل به عنوان یک موضوع چالش برانگیز مطرح بوده است. مسأله‌ی چیدمان تسهیلات تک ردیفه، یک مسأله‌ی NP-hard برای چیدمان تعدادی تسهیلات مستطیلی شکل با طول متغیر بر روی یک سطر و با هدف حداقل سازی مجموع وزن‌دار فاصله‌ی بین هر جفت از تسهیلات است. در این مقاله برای اولین بار دو الگوریتم بهینه‌سازی فاخته و بهینه‌سازی جنگل برای حل مسأله‌ی چیدمان تسهیلات تک ردیفه به کار گرفته شده و با یکدیگر مقایسه شده‌اند. عملگرهای هر دو الگوریتم با توجه به ویژگی‌های مسأله‌ی فوق تغییر داده‌شده و با مسأله وفق داده شده‌اند و نتایج حاصل برای دو گروه از مثال‌های معیار موجود در ادبیات پژوهش مقایسه شده‌اند. در گروه اول تعداد تسهیلات کوچکتر مساوی ۳۰ و در گروه دوم تعداد تسهیلات بزرگتر از ۳۰ است. نتایج حاصل نشان‌دهنده‌ی عملکرد بهتر الگوریتم پیشنهادی مبتنی بر بهینه‌سازی فاخته نسبت به الگوریتم پیشنهادی مبتنی بر بهینه‌سازی جنگل، در هر دو گروه از مثال‌ها چه از نظر یافتن بهترین جواب و چه از نظر زمان محاسباتی الگوریتم است.

کلمات کلیدی: مسأله‌ی چیدمان تسهیلات، مسأله‌ی چیدمان تسهیلات تک ردیفه، الگوریتم بهینه‌سازی فاخته، الگوریتم بهینه‌سازی جنگل.