Iranian Journal of Management Studies (IJMS) *http://ijms.ut.ac.ir/*
Vol. 10, No. 2, Spring 2017 *Print ISSN: 2008-7055*
pp. 467-498 *Online ISSN: 2345-3745*
DOI: 10.22059/ijms.2017.216663.672285

Two Strategies Based on Meta-Heuristic Algorithms for Parallel Row Ordering Problem (PROP)

Mansoureh Maadi[1], Mohammad Javidnia[2*], Rasoul Jamshidi[1]

1. Department of Industrial Engineering, Damghan University, Damghan, Iran
2. School of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran

(Received: September 29, 2016; Revised: April 22, 2017; Accepted: May 2, 2017)

Abstract

Proper arrangement of facility layout is a key issue in management that influences efficiency and the profitability of the manufacturing systems. Parallel Row Ordering Problem (PROP) is a special case of facility layout problem and consists of looking for the best location of *n* facilities while similar facilities (facilities which has some characteristics in common) should be arranged in a row and dissimilar facilities should be arranged in a parallel row. As PROP is a new introduced NP-hard problem, only a mixed integer programming model is developed to formulate this problem. So to solve large scale instances of this problem, heuristic and meta-heuristic algorithms can be useful. In this paper, two strategies based on genetic algorithm (GA) and a novel population based simulated annealing algorithm (PSA) to solve medium and large instances of PROP are proposed. Also several test problems of PROP in two groups with different sizes that have been extracted from the literature are solved to evaluate the proposed algorithms in terms of objective function value and computational time. According to the results, in the first group of instances, both algorithms almost have equal performances, and in the second group PSA shows better performance by increasing the size of test problems.

Keywords

Facility layout problem, Parallel row ordering problem, Genetic algorithm, Population based simulated annealing algorithm.

* **Corresponding Author, Email:** javidnia.mohammad@gmail.com

Introduction

The facility layout problem (FLP) is concerned with finding the optimal and best facility arrangement in a given layout. Some examples of the FLP applications can be related to layouts in a library, hospital or service center, an equipment assignment, the construction of the new manufacturing units, or a workshop organization. According to the paper of Drira et al. (2007), facility layout problems are categorized into different problems depending on factors including the workshop characteristics, how the problem is addressed, and the approaches used to solve the problem. One of these categories that is based on layout configuration is named multi-row problems. In the literature, until now, different multi-row problems have been proposed and different algorithms have been used to solve them. These problems in two rows are Double Row Layout Problem (DRLP), Corridor Allocation Problem (CAP), and Parallel Row Ordering Problem (PROP) which are extensions of another category of facility layout problems named Single Row Facility Layout Problem (SRFLP) with different conditional assumptions. Afterwards, SRFLP, DRLP, CAP, and PROP are described, also a summary in Table 1, related methods for modelling and different proposed algorithms of the literature for solving these problems are declared. It is notable that these problems are NP-hard.

Single row facility layout problem (SRFLP) is a main problem that has attracted the notice of many researchers in recent years (Amaral, 2013b). SRFLP is concerned with arranging a number of rectangular facilities with different length on one side of a straight line. The aim of this problem is minimizing the weighted sum of the distance between all facility pairs. Numerous applications of SRFLP are mentioned in the literature including arrangement of rooms in hospital, departments in office building or in supermarkets (Heragu & Kusiak, 1988), arrangement of machines in flexible manufacturing systems (Picard & Queyranne, 1981), assignment of files to disk cylinders in computer's storage (Anjos et al., 2005). After introducing SRFLP by Simmons in 1969, researchers have proposed different

exact algorithms to solve this problem, but because these algorithms only were able to solve small instances with up to 42 facilities, in recent years, different heuristic and meta-heuristic algorithms have been proposed to solve medium and large instances of SRFLP.

The double row layout problem (DRLP) consists of arranging rectangular facilities of different widths on both side of a corridor in order to minimize the total cost of material handling. In this problem, no facilities are restricted to any row. In 2010, Chung and Tanchoco proposed DRLP and formulated it as a mixed integer programming (MIP) problem for which only small instances may be solved optimally. Also, an application of DRLP within a fabrication line producing liquid crystal display (LCD) was described in this paper. Until now, different meta-heuristic methods have been proposed to solve medium and large instances of DRLP in the literature.

Corridor allocation problem (CAP) that was introduced by Amaral (2012), explores an arrangement of facilities along two horizontal lines that are parallel to x-axis of Cartesian coordinate system named central corridor. The aim of CAP is the minimization of the total communication cost among facilities regarding two conditions: (1) No space is permitted between two adjacent facilities. (2) The leftmost point of the arrangement on both line of a corridor must have zero abscissa. In Amaral's (2012) study, a mixed integer programming model is proposed for problem instances of moderate size. After that, different meta-heuristic algorithms have been proposed to solve CAP in the literature. The applications of CAP include arrangement of rooms in office buildings, hospitals, shopping centers, and schools (Amaral, 2012).

Parallel row ordering problem (PROP) considers arrangement of N facilities along two rows and is proposed by Amaral (2013b). Assume $\{N_i\}_{i=1,2}$be a partition of N, such that $\cup_{i=1}^{2} N_i = N$and $N_1 \cap N_2 = \emptyset$. Let R={1,2} be a set of two rows. A one-to-one assignment of the set $\{N_i\}_{i=1,2}$ is done to set R so that the facilities pertaining to the subset N_i should be arranged along row r (for some r, 1≤r≤2). The objective of this problem is to order facilities in two rows with the aim of minimizing a cost function of the x-distances between facilities.

PROP has a number of practical applications, including arrangement of facilities along two parallel straight lines on a floor plan and in the construction of multi-floor buildings (Amaral, 2013b). Figure 1 shows arrangement of facilities along two parallel straight lines on a floor plan (Amaral, 2013b).

Table 1. Review of studies on facility layouts

Problem	Method	Researchers
SRFLP	Mixed integer programming	Love & Wong (1976), Amaral (2006a), Amaral (2008)
	Nonlinear programming	Heragu & Kusiak (1991)
	Dynamic programming	Picard & Queyranne (1981), Kouvelis & Chiang (1996)
	Cutting planes	Amaral (2009)
	Semidefinite programming (SDP)	Anjos et al. (2005), Anjos & Vannelli (2008), Anjos & Yen (2009), Hungerländer & Rendl (2013)
	Branch and Bound	Simmons (1969)
	Branch and Cut	Amaral & Letchford (2013)
	Path Relinking	Kothari & Ghosh (2012a)
	Greedy Search	Kumar et al. (1995), Djellab & Gourgand (2001)
	Local Search	Palubeckis (2015)
	Lin-Kernighan	Kothari & Ghosh (2013b)
	Simulated Annealing	Romero & Sanchez-Flores (1990), Heragu & Alfa (1992), de Alvarenga et al. (2000), Palubeckis (2017)
	Tabu Search	de Alvarenga et al. (2000), Samarghandi & Eshghi (2010), Kothari & Ghosh (2013c)
	Ant Colony Optimization	Solimanpur et al. (2005)
	Scatter Search	Satheesh Kumar et al. (2008), Kothari & Ghosh (2014c), Akbari & Maadi (2011), Kunlei et al. (2011)
	Particle Swarm Optimization	Samarghandi et al. (2010)
	Genetic Algorithm	Ficko et al. (2004), Datta et al. (2011), Kothari & Ghosh (2014b)
	Cuckoo Optimization Algorithm	Maadi et al. (2016)
	Forest Optimization algorithm	Maadi et al. (2016)
	Hybrid Algorithms	Braglia (1996), Teo & Ponnambalam (2008), Ozcelik (2012), Guan & Lin (2016)

Continue Table 1. Review of studies on facility layouts

Problem	Method	Researchers
DRLP	Mixed integer programming	Chung & Tanchoco (2010), Zhang & Murray (2012), Amaral (2013a), Anjos et al. (2016)
	Semidefinite programming (SDP)	Anjos et al. (2016)
	Local Search	Murray et al. (2013)
	Hybrid Algorithms	Zuo et al. (2014)
CAP	Mixed integer programming	Amaral (2012)
	Simulated Annealing	Ahonen et al. (2014)
	Tabu Search	Ahonen et al. (2014)
	Scatter Search	Ghosh & Kothari (2012)
	Hybrid Algorithms	Ghosh & Kothari (2012)
PROP	Mixed integer programming	Amaral (2013b)

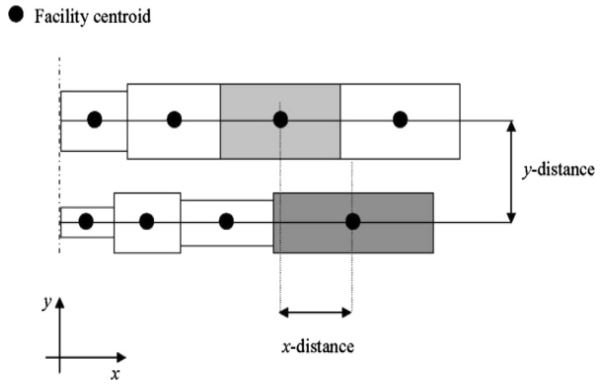

Fig. 1. Arrangement of facilities along two parallel straight lines on a floor plan (Amaral, 2013b)

Moreover, the difference between PROP, DRLP, and CAP is investigated. In DRLP and CAP, facilities are arranged along two parallel rows, in contrast to PROP, they are not restricted to any row. A difference between DRLP, CAP, and PROP is that PROP and CAP assume that the arrangement in both rows starts from a common point and no space is permitted between two adjacent facilities, while the DRLP does not make such assumptions. Also, in DRLP the distance between two rows is set to zero while it is not true in PROP and CAP.

As it can be seen in Table 1, only one paper about PROP has been presented in the literature, while this paper introduces the PROP and

provides a mixed integer programming (MIP) model that is able to solve small instances of PROP up to 23 facilities. In this paper, instances are commonly used as benchmark instances for the SRFLP to perform computational experiments. Using CPLEX12.4, the proposed MIP solves all instances to optimality.

Because PROP is an NP-hard problem, like other multi-row facility layout problems, to solve medium and large instances, a meta-heuristic algorithm is needed, so this paper can be a starting point to apply meta-heuristic algorithms to solve medium and large instances of PROP.

GA and SA as two basic and efficient meta-heuristic algorithms have shown great performance to solve different facility layout problems including SRFLP. In the literature, different GA and SA based algorithms are applied to solve SRFLP, so that these algorithms have shown better performance in comparison to other meta-heuristic algorithms of the literature. Hence, in this paper a genetic algorithm and a novel population based simulated annealing algorithm are proposed to solve medium and large instances of the PROP.

The rest of paper is organized as follows: In problem description section, the problem is described. Then, in meta-heuristic algorithms section proposed algorithms based on genetic algorithm and a novel population based simulated annealing algorithm are presented. In computational experiments section the proposed algorithms are evaluated on various instances with different sizes of $n \leq 23$ and $n > 23$ followed by the conclusions of the paper in *conclusion*.

Problem Description

In this section, we describe the mixed integer programming formulation of the PROP proposed by Amaral (2013b).

The parameters and indices are:

n	Number of facilities
t	Number of facilities with some characteristic in common
N_1	The set of first row facilities; $N_1 = \{1, \dots, t\}$
N_2	The set of second row facilities; $N_2 = \{t+1, \dots, n\}$
N	The set of all facilities; $N = N_1 \cup N_2$
i, j, k	Index for facilities
ℓ_i	The length of facility i; $i \in N$
c_{ij}	The average daily traffic (or flow) between facilities j and j; $i, j \in N$

We need the following vectors:

- A vector $a^1 = \left(a_{i,j}^1\right)_{i\,,j\,\in N_1; i<j}$ such that $a_{i,j}^1 = 1$ when facility i is to the left of facility j and $a_{i,j}^1 = 0$ otherwise.
- A vector $a^2 = \left(a_{i,j}^2\right)_{i\,,j\,\in N_2; i<j}$ such that $a_{i,j}^2 = 1$ when facility i is to the left of facility j and $a_{i,j}^2 = 0$ otherwise.
- A vector $d^1 = \left(d_{i,j}^1\right)_{i\,,j\,\in N_1; i<j}$ such that $d_{i,j}^1$ represents the x-distance between the centers of two facilities i and j placed at Row 1.
- A vector $d^2 = \left(d_{i,j}^2\right)_{i\,,j\,\in N_2; i<j}$ such that $d_{i,j}^2$ represents the x-distance between the centers of two facilities i and j placed at Row 2.
- A vector $d^{1,2} = \left(d_{i,j}^{1,2}\right)_{i\,\in N_1, j\,\in N_2; i<j}$ such that $d_{i,j}^{1,2}$ represents the x-distance between the centers of facility i placed at Row 1 and facility j placed at Row 2.

Define the following polytopes:

$$H_t^1 = \left\{ a^1 \in \mathbb{R}_+^{\mathrm{t}(\mathrm{t}-1)/2}:\ \begin{matrix} a_{i,j}^1 + a_{j,k}^1 - a_{i,k}^1 \le 1 & (i,j,k \in N_1; i<j<k) \\ -a_{i,j}^1 - a_{j,k}^1 + a_{i,k}^1 \le 0 & (i,j,k \in N_1; i<j<k) \end{matrix} \right\}$$

$$H_{n-t}^2 = \left\{ a^2 \in \mathbb{R}_+^{(n-\mathrm{t})(n-\mathrm{t}-1)/2}:\ \begin{matrix} a_{i,j}^2 + a_{j,k}^2 - a_{i,k}^2 \le 1 & (i,j,k \in N_2; i<j<k) \\ -a_{i,j}^2 - a_{j,k}^2 + a_{i,k}^2 \le 0 & (i,j,k \in N_2; i<j<k) \end{matrix} \right\}$$

$$D_t^1 = \left\{ d^1 \in \mathbb{R}_+^{\mathrm{t}(\mathrm{t}-1)/2}:\ d_{i,j}^1 \ge \frac{\left(\ell_{\mathrm{i}} + \ell_{\mathrm{j}}\right)}{2}\ \ (i,j \in N_1; i<j) \right\}$$

$$D_{n-t}^2 = \left\{ d^2 \in \mathbb{R}_+^{(n-\mathrm{t})(n-\mathrm{t}-1)/2}:\ d_{i,j}^2 \ge \frac{\left(\ell_{\mathrm{i}} + \ell_{\mathrm{j}}\right)}{2}\ \ (i,j \in N_2; i<j) \right\}$$

Then, a mixed integer programming formulation of the PROP is given by:

$$\text{Minimize} \left\{ \sum_{i,j\in N_1, i<j} c_{ij} d_{ij}^1 + \sum_{i,j\in N_2, i<j} c_{ij} d_{ij}^2 + \sum_{i\,\in N_1, j\,\in N_2; i<j} c_{ij} d_{ij}^{1,2} \right\} \tag{1}$$

$$\text{s.t:} \quad d_{i,j}^1 \ge \begin{pmatrix} \sum_{1\le k<i} \ell_{\mathrm{k}}\, a_{k,i}^1 + \sum_{i\le k<t} \ell_{\mathrm{k}} \left(1 - a_{i,k}^1\right) \\ -\sum_{1\le k<j} \ell_{\mathrm{k}}\, a_{k,j}^1 - \sum_{j\le k<t} \ell_{\mathrm{k}} \left(1 - a_{j,k}^1\right) + \dfrac{\left(\ell_{\mathrm{i}} - \ell_{\mathrm{j}}\right)}{2} \end{pmatrix} (i,j \in N_1; i<j) \tag{2}$$

$$d_{i,j}^1 \ge \begin{pmatrix} -\sum_{1\le k<i} \ell_{\mathrm{k}}\, a_{k,i}^1 - \sum_{i\le k<t} \ell_{\mathrm{k}} \left(1 - a_{i,k}^1\right) \\ +\sum_{1\le k<j} \ell_{\mathrm{k}}\, a_{k,j}^1 + \sum_{j\le k<t} \ell_{\mathrm{k}} \left(1 - a_{j,k}^1\right) + \dfrac{\left(\ell_{\mathrm{j}} - \ell_{\mathrm{i}}\right)}{2} \end{pmatrix} (i,j \in N_1; i<j) \tag{3}$$

$$d^1 \in D_t^1 \tag{4}$$

$$a^1 \in H_t^1 \tag{5}$$

$$a_{i,j}^1 \in \{0,1\} \quad (i,j \in N_1; i < j) \tag{6}$$

$$d_{i,j}^2 \geq \left(\begin{array}{c} \sum_{t \leq k < i} \ell_k a_{k,i}^2 + \sum_{i \leq k < n} \ell_k \left(1 - a_{i,k}^2\right) \\ - \sum_{t \leq k < j} \ell_k a_{k,j}^2 - \sum_{j \leq k < n} \ell_k \left(1 - a_{j,k}^2\right) + \frac{\left(\ell_i - \ell_j\right)}{2} \end{array} \right) (i,j \in N_2; i < j) \tag{7}$$

$$d_{i,j}^2 \geq \left(\begin{array}{c} - \sum_{t \leq k < i} \ell_k a_{k,i}^2 - \sum_{i \leq k < n} \ell_k \left(1 - a_{i,k}^2\right) \\ + \sum_{t \leq k < j} \ell_k a_{k,j}^2 + \sum_{j \leq k < n} \ell_k \left(1 - a_{j,k}^2\right) + \frac{\left(\ell_j - \ell_i\right)}{2} \end{array} \right) (i,j \in N_2; i < j) \tag{8}$$

$$d^2 \in D_{n-t}^2 \tag{9}$$

$$a^2 \in H_{n-t}^2 \tag{10}$$

$$a_{i,j}^2 \in \{0,1\} \quad (i,j \in N_2; i < j) \tag{11}$$

$$d_{i,j}^{1,2} \geq \left(\begin{array}{c} \sum_{1 \leq k < i} \ell_k a_{k,i}^1 + \sum_{i \leq k < t} \ell_k \left(1 - a_{i,k}^1\right) \\ - \sum_{t \leq k < j} \ell_k a_{k,j}^2 - \sum_{j \leq k < n} \ell_k \left(1 - a_{j,k}^2\right) + \frac{\left(\ell_i - \ell_j\right)}{2} \end{array} \right) (i \in N_1, j \in N_2; i < j) \tag{12}$$

$$d_{i,j}^{1,2} \geq \left(\begin{array}{c} - \sum_{1 \leq k < i} \ell_k a_{k,i}^1 - \sum_{i \leq k < t} \ell_k \left(1 - a_{i,k}^1\right) \\ + \sum_{t \leq k < j} \ell_k a_{k,j}^2 + \sum_{j \leq k < n} \ell_k \left(1 - a_{j,k}^2\right) + \frac{\left(\ell_j - \ell_i\right)}{2} \end{array} \right) (i \in N_1, j \in N_2; i < j) \tag{13}$$

Constraints (2) and (3) specify the x-distances between facilities of Row 1. In Row 2, Constraints (7) and (8) determine the x-distances between pairs of facilities and x-distances between two facilities from different rows are calculated in Constraints (12) and (13). In above formulations, Constraints (4) and (9) fortify that, minimum x-distance between two facilities in one row equals the sum of their half-lengths. Constraints (5) and (6) guarantee that for Row 1, a^1 is an incidence vector of a linear ordering, while Constraints (10) and (11) guarantee that for Row 2, a^2 is an incidence vector of a linear ordering.

The important point about this problem is that because the ordering of the PROP is only in one dimension of x-axis, the other components

of distance do not change between any two facilities. In fact, only x-distance between facilities is regarded in PROP.

Meta-Heuristic Algorithms

Regarding applications of different meta-heuristic algorithms in different areas of problem optimization such as inventory control (Orand et al., 2015), design of integrated logistics network (Yadegari et al., 2015), project scheduling (Zareei & Hassan-Pour, 2015), and parcel delivery services (Bahrami et al., 2016), in this section two proposed meta-heuristic algorithms including Genetic Algorithm (GA) and a Population-based Simulated Annealing (PSA) algorithm are applied to solve PROP. In solution representation section, the representation used for solutions of two algorithms is explained; then, to generate different and efficient initial population of two proposed algorithms, two procedures are introduced in generating initial population section. After that, two proposed algorithms are described in proposed genetic algorithm section and proposed population-based simulated annealing algorithm section.

Solution Representation

In this paper, a solution is represented by a vector of *n* elements that first *t* elements are related to facilities which should be arranged in the first row and the next *(n-t)* elements represent the facilities that should be placed in the second row. The representation of a solution of two algorithms with $n=10$ and $t= [\frac{n}{3}] = 3$ is depicted in Figure 2.

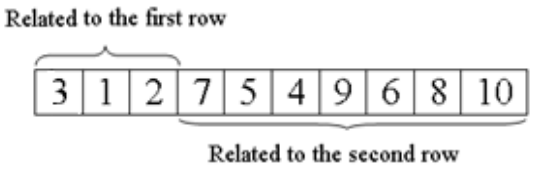

Fig. 2. A solution representation

It is notable that in PROP, at the beginning, the first *t* facilities (facilities 1, 2 … t) should be selected, hence, these *t* facilities cannot be exchanged with facilities in the parallel row and only their ordering

can be permuted in the first row. Thus, other *n-t* facilities are restricted to the parallel row and only their ordering can be changed in PROP to achieve a better solution. Regarding solution representation, the cost function of two proposed algorithms is Equation (1).

Generating Initial Population

To generate initial population of two algorithms, two procedures are applied in this paper. In the first procedure, using Theorem 1 of the paper of Samarghandi and Eshghi (2010) to solve SRFLP, one solution is created. In this theorem for sorting n facilities in a single row, it is assumed that the cost function coefficients of the problem are constant numbers (c_{ij}=c). now for a number of facilities such as facilities number 1 to number n, if we sort them in non-descending order in a way that the shortest facility is denoted by 1 and the longest one by *n*, the optimal solution when *n* is an odd number is shown in Figure 3 and for an even number of *n* the optimal solution is displayed in Figure 4. Using this theorem for two rows of the problem separately, a solution is created. The second procedure is randomly generating permutations in each row. In this procedure, at the beginning, the first *t* facilities are selected and laid in random ordering in the first *t* elements of a solution array and then other *n-t* facilities are randomly placed in the remaining elements of the solution. Using random permutation strategy to generate initial population guarantees a good diversification of the initial solutions. These two procedures assure that the search initiates in feasible space. Using two procedures, the initial population is created and the operators of algorithms can be started.

Fig. 3. Optimal solution for odd number of *n*

Fig. 4. Optimal solution for even number of *n*

Proposed Genetic Algorithm

Genetic algorithm (GA) is a stochastic search technique that is based on the concept of the survival of the fittest according to the Darwinian Evolution theory (Goldberg & Holland, 1988; Deb & Kalyanmoy, 2001). In GA, after defining solution representation which is named chromosome or individual and generating the initial population, this population is evolved over iterations using the main genetic operators such as selection, crossover and mutation. The central part of our GA implementation consists of selection, crossover and mutation operators on population that is generated using the method explained in the section above. Roulette wheel selection that is a fitness proportionate selection is used to select potentially useful solutions for recombination. A crossover operator with some crossover probability produces offspring individuals using the selected parent individuals. A mutation operator also explores the neighborhood of an offspring individual with another rate of mutation probability. Using these operators, the process of evolution of the population will be continued until some termination criteria are met. A predefined maximum number of iterations or the favorite amount of improvement in the cost function value are usually the termination criteria of the GA. For more information about GA, readers are referred to Goldberg et al. (1988). In the next sections the steps of proposed GA are described.

Crossover operator

Crossover is the operator of generating feasible offspring individuals using two parent individuals that are selected from the population. In proposed algorithm, because the first t facilities in the first row cannot be exchanged with other n-t facilities in the parallel row, one-point crossover operator is applied. In this operator regarding crossover rate, after selecting two parents using Roulette wheel selection method, the point of t is considered in two parents' arrays. The permutation of facilities in the first row of parent 1 and the permutation of facilities in the second row of parent 2 produce the array of the first offspring. The second offspring is created by the permutation of facilities in the second row of parent 1 and the permutation of facilities in the first

row of parent 2. Figure 5 shows the process of producing two offspring individuals in a problem with n=10 and $t= \lceil \frac{n}{3} \rceil$.

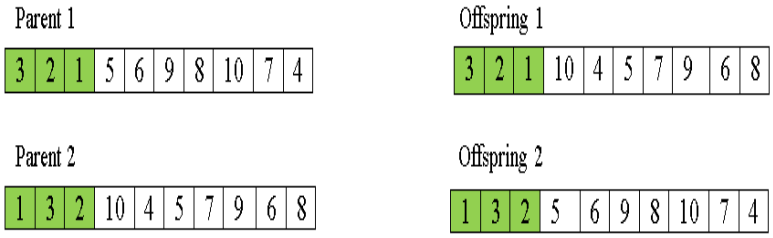

Fig. 5. Crossover operator

Mutation operator

Mutation operator in GA is used to hold the genetic diversity of population in all generations of the algorithm. In proposed GA, at first with regard to mutation rate, some offspring solutions are selected. In our proposed mutation operator for each solution, two new solutions are obtained as follows: Assuming one solution, initially two random facilities in the first row are swapped and a new solution is generated without any change in the ordering of the facilities in the parallel row, then another solution is created by swapping two random facilities in the parallel row without any change in ordering of facilities in the first row. New solutions that are feasible solutions are added to the population. In Figure 6, the performance of mutation operator is shown with an example.

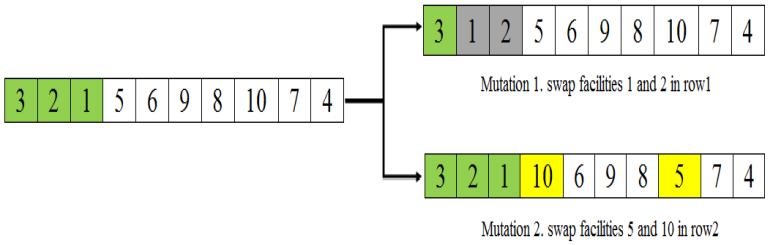

Fig. 6. Mutation operator

After an iteration of algorithm, the best solutions are selected as the population of the next iteration until the termination condition is met. The termination condition of proposed algorithm is considered as the

number of iterations as a parameter of the algorithm. The pseudo-code of proposed GA to solve PROP is described in Algorithm1.

Algorithm.1 Proposed GA

Input: number of facilities (*n)*, (*t)*, Flow matrix (*F)*, Facilities length matrix (*L)*, population size (*N)*, maximum number of iterations (maxiter), mutation rate (p_m) and crossover rate (p_c).

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

Compute $n\text{cross} = 2 \times \lfloor p_c \times N/2 \rfloor, \text{nmut} = 2 \times p_m \times \text{ncross}$
Create the initial population *init pop* of N individuals by using section 3.2.
Set the current population *pop* ← *init pop*.
Compute fitness (i) for each $i \in pop$
Find best fitness and its chromosome.

```
for i=0 to maxiter
    for j=0 to ncross
        P1, P2= select parents (pop)
        Offspring (j): permutation of facilities in the first row of P1 and permutation of
        facilities in the second row of P2.
        Offspring (j+1): permutation of facilities in the second row of P2 and permutation
        of facilities in the first row of P1.
        j=j+2
    End
    for k=0 to nmut
        Select a random individual from Offspring individuals as O1
        Mut[k] ← O1
        Mut[k+1]←O1
        Select two random numbers from {0, 1… t-1} as t1,t2
        Swap (t1, t2, Mut [k])
        Select two random numbers from {t, t+1… n-1} as t1,t2
        Swap (t1, t2, Mut [k+1])
        k=k+2
    End

    Replace the current population (pop) with the new population.
    Compute fitness (i) for each i ∈ pop and sort chromosomes according their fitness.
    Select the best N chromosomes from pop and reduce the extended population.
    Update the best fitness and its chromosome.
    i=i+1
End
Return the best chromosome.
```

Proposed Population-Based Simulated Annealing Algorithm

Simulated Annealing (SA) algorithm that is proposed by Kirkpatrick, Gellat, and Vecchi (1983), is an iterative and stochastic method inspired from the annealing process where the metal and other substances melt and then slowly cool to obtain a strong crystalline

structure. SA starts with an initial solution and subsequently may proceed to a neighboring solution by a random move, until the stopping conditions are met. In the literature, SA has attracted the consideration of many researchers because of not getting stuck in a local minimal by accepting worse solutions. For more information about SA, readers are referred to the paper of van Laarhoven and Aarts (1987), and Kirkpatrick, Gellat, and Vecchi (1983).

The proposed PSA algorithm starts with an initial population of solutions in contrast to regular SA algorithm that starts with one solution. Because PROP is a permutation based problem, with increase in the number of facilities, the solution space will be increased more and more. So searching for all solutions of the medium and large instances of PROP is not possible in reasonable time (We know in a PROP instance with n facilities, $t!\times(n-t)$ different solutions can be created). Because of existing different proposed algorithms in the literature to solve facility layout problems, the initial solutions of proposed algorithms are produced randomly using uniform distribution in order to better search for the solution space. This process in different papers is introduced as the best method to produce initial solutions in different types of facility layout problems such as Hosseini-Nasab and Emami (2013), Kunlei et al. (2011), and Samarghandi et al. (2010). Also, it helps to get rid of local optimums. In addition, another method that has been introduced to produce efficient initial solutions in SRFLP is applied to solve PROP.

Like proposed GA, proposed PSA algorithm starts with generating initial population according to the section Generating Initial Population. After that, for each solution of the population, a number of solutions are produced using neighborhood structure. In this process for each solution, M neighboring solutions are generated that M is a parameter of proposed PSA algorithm and should be determined. Regarding SA mechanism, new produced solutions are accepted or rejected. In neighborhood structure to perform new neighboring solutions, the swap operator is used as follows: Consider a solution with t facilities in the first row and n-t facilities in the parallel row. To generate one new solution, at first one row is selected randomly and

two random facilities of the row are swapped. This process will be continued until *M* new solutions are generated. After producing neighboring solutions, SA mechanism is started. The pseudo-code of the proposed PSA algorithm is described in Algorithm 2.

SA algorithm begins with initial high temperature of T_0 and gradually reduces until final temperature of T_f, through cooling process. T_0 and T_f are parameters of the SA. The initial temperature should be high enough to allow a move to any neighboring solution and not a very high temperature that makes the search process inefficient. The final temperature is usually set to zero. In this paper, we use a linear cooling schedule, which updates temperature at each time using the expression of $T_i=\alpha\times T_{i-1}$, which α is another parameter of SA that is cooling ratio. In proposed algorithm, the initial temperature (T_0)is considered the cost function value of the worst solution among initial population and T_f is set to zero. Such as proposed GA, the stopping condition of proposed PSA algorithm is considered as the number of iterations as a parameter of the algorithm.

Algorithm 2 Proposed PSA algorithm

Input: number of facilities (*n*),(*t*), Flow matrix*(F)*, Facilities length matrix(*L)*, population size (*N)*,maximum number of iterations (maxiter), number of neighborhoods (*M)* and Cooling rate*(α).*

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

```
Create the initial population (init pop) of N individuals by using section 3.2.
Set the current population pop ← init pop.
Compute fitness (i) for eachi ∈ pop.
Find the worst fitness value and copy that to temp//initial temperate
for i=0 to maxiter
    for j=0 to N
        S ←pop (j)
        f ← fitness(S)
        for k=0 to M do
            S'← S
            Generate a random number from{0, 1} as rand
            if(rand= =0)
                    Select two random numbers from {0, 1… t-1} as t1,t2
                    Swap (t1, t2, S')
            else
                    Select two random numbers from {t, t+1… n-1} as t1,t2
                    Swap (t1, t2, S')
```

```
            End
            f' ←Compute fitness(S')
            if (f'<f)
                S ← S'
            else
                Generate a random number from[0, 1] as r
                if (exp(-(f'-f/temp) > r)
                    S ← S'
                End
            End
            k=k+1
        End
        j=j+1
    End
    Update the best solution.
    temp= α ×temp
    i=i+1;
End
Return the best solution.
```

Computational Experiments

The performances of proposed genetic and simulated annealing algorithms are evaluated on several instances in different sizes available in the literature. These two proposed algorithms are implemented in C# and are run on an Intel (R) core (TM) i5-3210 CPU @ 2.5 Gigahertz and 4.00 Gigabytes ram under the Windows 8.1 Operating system.

All PROP instances look like SRFLP instances with an additional parameter *t* (Amaral, 2013b). In fact, a PROP instance is composed of the following data: A positive integer of *n*, vector of positive integers $(l_i)_{i=1,2,\dots,n}$ that l_i is the length of the facility *i*, symmetric cost matrix of non-negative integers $(c_{ij})_{i=1,2,\dots,n,j=1,2,\dots,n}$, and an integer number in [1, *n*-1] as *t*.

The instances used in this paper are divided into two general groups. The first group that is related to the instances with size $n \leq 23$ has optimal solution. These instances are introduced and solved by Amaral (2013b)[1]. Another group of instances with size $23 < n \leq 70$ does not have optimal solution and is tested in this paper for the first time.

1. The instances are available at http://www.gerad.ca/files/sites/Anjos/flplib.html

All instances used in this group to test the PROP were originally used in the literature to test SRFLP and are available at the mentioned site. For each instance, both algorithms are run 20 times. Now, the results of experiments on different instances in different sizes are described separately.

Instances with Size n≤23

First, the proposed algorithms are tested on instances in the literature with size n≤23. As said before, these instances were solved using a MIP model in the paper of Amaral (2013b) and have optimal solutions group, Instance s11 is from Simmons (1969), Am15 is from Amaral (2006), and others are from Amaral (2013b). Throughout the experiments, the parameter values that are used for GA and PSA algorithm in this group of instances are as follows respectively: For GA, the initial population is set to 50. The crossover and mutation rates are considered 0.9 and 0.2. The maximum number of iterations is determined as 100. In PSA, the initial population is set to 5. The number of neighborhood solutions is set to 20. Maximum number of iterations is determined as 100 and α=0.85. For instances of this group, two proposed algorithms were able to obtain the optimal solution of all the instances that are introduced by Amaral (2013b). In addition, the proposed algorithms consume less time to obtain optimal solution in comparison to the MIP model of Amaral (2013b). In Figure 7, Figure 8, Figure 9 and Figure 10 computational time of two proposed algorithms for different parameters of t are compared.

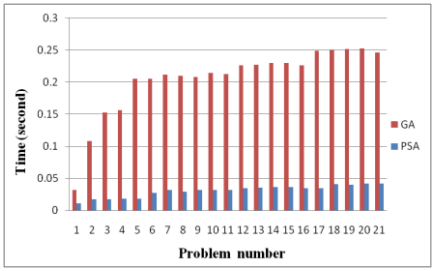

Fig. 7. Comparison of computational time of proposed GA and PSA algorithm on instances with size n≤23 and t= $\lfloor \frac{n}{2} \rfloor$

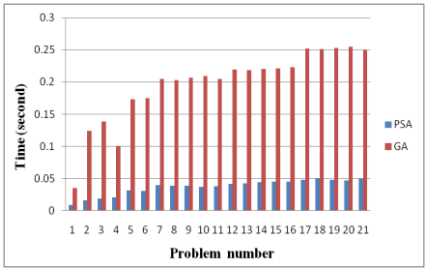

Fig. 8. Comparison of computational time of proposed GA and PSA algorithm on instances with size n≤23 and t= $\lfloor \frac{n}{3} \rfloor$

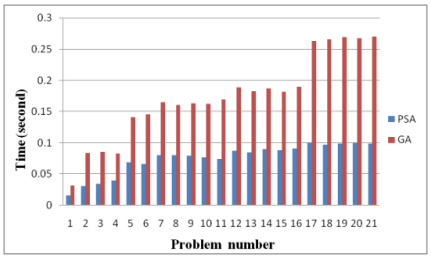

Fig. 9. Comparison of computational time of proposed GA and PSA algorithm on instances with size n≤23 and t= $[\frac{n}{4}]$

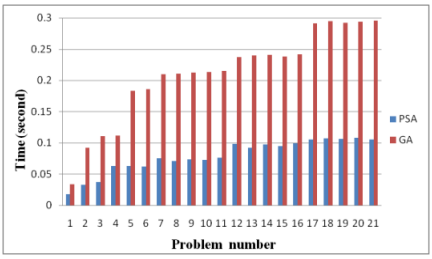

Fig. 10. Comparison of computational time of proposed GA and PSA algorithm on instances with size n≤23 and t= $[\frac{n}{5}]$

Obtaining the optimal solution for all instances of this group in a reasonable and short computational time is a sign of the ability of two proposed algorithms to solve different PROP instances, so it seems that these algorithms can be useful to solve real large instances.

Instances with Size 23<*n*≤70

Next, the performance of genetic and population-based simulated annealing algorithms are evaluated on the second group of instances with *n*=30, 40, 56, 60 and 70 and parameter *t*= $[\frac{n}{2}]$,$[\frac{n}{3}]$, $[\frac{n}{4}]$ and $[\frac{n}{5}]$.As said before, the optimal solution of these instances are not in the literature and these instances are solved in this paper for the first time. In this group, instances with *n*=30 are from Anjos and Vannelli (2008), instances with *n=40* are from Hungerländer and Rendl (2013), instances with *n*=56 are from Anjos and Yen (2009), and instances with *n*=60 and *n*=70 are from Anjos et al. (2005). All instances of this group are solved with different values of parameter *t* consisting of $[\frac{n}{2}]$, $[\frac{n}{3}]$, $[\frac{n}{4}]$ and $[\frac{n}{5}]$. For this group of instances, since the number of facilities is large, selection of appropriate values for proposed algorithm parameters is very important. Therefore, before solving this group of instances, using Taguchi method, parameters of proposed algorithms should be tuned. So in continue, at first, parameter tuning is applied to determine the parameters of two proposed algorithms, then, results of applying proposed algorithms to solve this group of instances are described.

Parameter Tuning

The most experimental design to tune parameters of algorithms is Taguchi method. In this method, a large number of factors can be studied using a small number of required experiments (Phadke, 1989).

In Taguchi method, the factors of algorithm are divided into two main categories of controllable and noise factors. The aim of this method is minimizing the effect of noise factors and determining the optimal level of controllable factors. To measure the stability of a process, the signal-to-ratio is used in this method (Hsu, 2012). Here the term signal denotes the desirable value (mean response variable) and noise signifies the undesirable value (standard deviation). The formulation of S/N estimates how samples deviate from the center of population and maximizing the *S/N* ratio is the aim (Tang et al., 2012). The formula of calculation of *S/N* is related to the objective function of the problem. In the Taguchi approach, the objective functions are classified into three groups of the smaller-the-better type, the nominal-the-best type and the larger-the-better type. As the objective function of PROP is categorized in the smaller- he better type, related *S/N* ratio is calculated as follows:

$$\text{S/N ratio} = -10\log_{10}[\frac{1}{N}\sum_{i=1}^{N} Y_i^2] \quad (14)$$

In Equation (2), Y_i^2 denotes the response value (the value of objective function) of the *i*th instance. In what follows, at first, the levels of the factors are introduced, then, after considering one instance, Taguchi method will be executed to determine the best level of each factor. The proposed algorithms include four factors. Table 2 shows the considered levels for these four factors for PSA algorithm and Table 3 Shows this information for GA. The number of degrees of freedom should be determined at first in order to select an appropriate orthogonal array, so the total degree of freedom is set to eight; therefore, the appropriate array must have at least eight trials. We apply the array L_9 for the experiments.

Table 2. The controlled factors of PSA and their levels

factors	Description	Level 1	Level 2	Level 3
Init_ sol	Number of Initial Solutions	10	20	30
Num_neigh	Number of neighborhood of each solutions	20	30	40
Alpha	The Cooling ratio	0.85	0.9	0.95
Maxiter	maximum number of iterations	100	200	300

Table 3. The controlled factors of GA and their levels

factors	Description	Level 1	Level 2	Level 3
Init_ pop	Number of Initial populations	50	100	150
Cross_rate	The Crossover rate	0.7	0.8	0.9
Mut_rate	The Mutation rate	0.05	0.1	0.15
Maxiter	maximum number of iterations	150	300	450

We use instances with 70 facilities for the experiments. The objective function should be transformed into its average relative percentage deviation (RPD) to calculate the performance of algorithm. PRD is defined as follow:

$$\mathrm{RPD_i} = \frac{1}{\mathrm{M}}\left[\frac{\sum_{\mathrm{j=1}}^{\mathrm{J}} \frac{\mathrm{cost_{ij}} - \mathrm{LB_i}}{\mathrm{LB_i}}}{\mathrm{J}}\right] \quad (15)$$

where J is number of replication, M is number of instance in each group (is equal to 5), $cost_{ij}$ is the total cost obtained for instance i in replication j and $\mathrm{LB_i}$ is the minimum total cost obtained for instance i. Because of the optimal values of instance is not known, $\mathrm{LB_i}$ is equal to the best total cost obtained by the proposed algorithms for instance i. The results are transformed into *S/N* ratio according to the formula presented below:

$$S/N\ ratio = \eta_k = -10log_{10}[\frac{1}{N}\sum_{i=1}^{N} {RPD_i}^2] \quad (16)$$

where η_k is the $S/N\ ratio$ for trial k and is number of objective functions (is equal to 4 for t= $[\frac{n}{2}]$,$[\frac{n}{3}]$, $[\frac{n}{4}]$ and $[\frac{n}{5}]$). Table 4 shows the orthogonal array L_9 and the RPD for each trial and S/N ratio for each experimental trial for PSA and Table 5 shows this information for GA.

In these tables RPD_1 for $t = [\frac{n}{2}]$, RPD_2 for $t = [\frac{n}{3}]$, RPD_3 for $t = [\frac{n}{4}]$, RPD_4 for $t = [\frac{n}{5}]$. For each level of control factors, the S/N ratio is averaged and its value is plotted against each control factor of PSA algorithm in Figure 11. Also, Figure 12 shows this information for GA. Since, the aim is to maximize the S/N ratio, the level with highest S/N ratio is selected as the best level.

Table 4. L_9 Orthogonal array and data for the screen experiment (PSA Algorithm)

Init_ sol	Num_ neigh	Alpha	Maxiter	RPD_1	RPD_2	RPD_3	RPD_4	SNRA1
10	20	0.85	100	0.0123	0.0116	0.0167	0.0197	36.2330
10	30	0.90	200	0.0039	0.0022	0.0012	0.0026	51.5341
10	40	0.95	300	0.0042	0.0017	0.0004	0.0005	52.7949
20	20	0.90	300	0.0015	0.0008	0.0006	0.0012	59.3494
20	30	0.95	100	0.1386	0.1270	0.1341	0.1418	17.3612
20	40	0.85	200	0.0017	0.0003	0.0006	0.0006	60.3019
30	20	0.95	200	0.0041	0.0044	0.0049	0.0084	44.8842
30	30	0.85	300	0.0008	0.0004	0.0002	0.0001	66.6012
10	20	0.85	100	0.0123	0.0116	0.0167	0.0197	36.2330

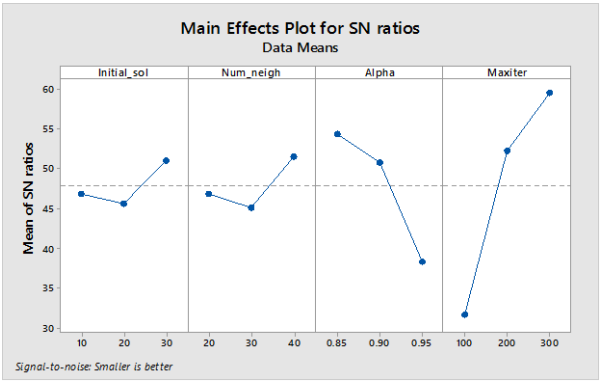

Fig. 11. The mean S/N ratio plot for PSA algorithm

Table 5. L_9 orthogonal array and data for the screen experiment (GA)

Init_ pop	Cross_ rate	Mut_ rate	Maxiter	RPD_1	RPD_2	RPD_3	RPD_4	SNRA1
50	0.7	0.05	150	0.07504	0.07105	0.07847	0.01630	23.6902
50	0.8	0.10	300	0.03015	0.01958	0.02042	0.02510	32.3300
50	0.9	0.15	450	0.02017	0.01416	0.01555	0.01351	35.8861
100	0.7	0.10	450	0.02238	0.01524	0.00829	0.01493	35.9144
100	0.8	0.15	150	0.02859	0.02024	0.02494	0.02534	32.0570
100	0.9	0.05	300	0.02534	0.01968	0.02274	0.02290	32.8587
150	0.7	0.15	300	0.02601	0.01627	0.01328	0.01175	35.0314
150	0.8	0.05	450	0.01992	0.01376	0.01624	0.01566	35.6236
150	0.9	0.10	150	0.02749	0.01795	0.02866	0.02623	31.8922

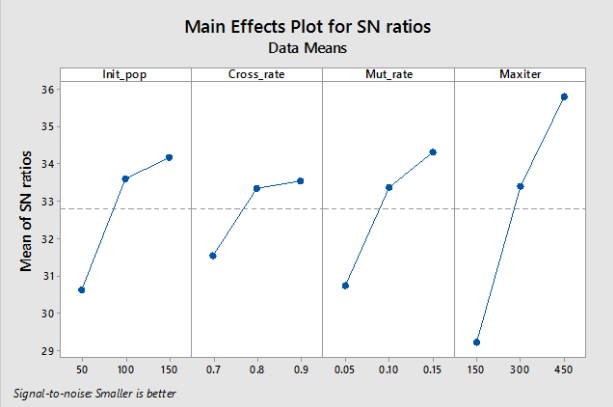

Fig. 12. The mean S/N ratio plot for GA algorithm

Results of applying two proposed algorithms on instances with size 23<*n*≤70

After setting the parameters, for evaluation of the proposed algorithms to solve this group of instances, the relative percentage deviation (RPD) is computed as follows:

$$\mathrm{RPD_i} = \frac{\sum_{j=1}^{J} \frac{\mathrm{cost_{ij}} - \mathrm{LB_i}}{\mathrm{LB_i}}}{J} \times 100 \qquad (17)$$

where J is number of replication (is equal to 20), $cost_{ij}$ is the total cost obtained for instance i in replication j and LB_i is equal to the best total cost obtained by the proposed algorithms for instance i.

Tables 6, 7, 8 and 9 show the evaluation of two proposed algorithms on instances of this group with different values of *t* in both terms of cost function value and computational time as well as RPD. In these tables, values in bold-face indicate the best found cost function value for an instance. Regarding these tables, it can be observed that for instances with size *n*=30, the best obtained cost function values of two algorithms in all tables are equal. Whereas the computational time of PSA algorithm is less than GA in all instances. As for other instances of this group, it can be seen that in PSA algorithm almost the value of cost function of all instances is equal or less than GA, and proposed PSA algorithm has better performance

rather than GA in both aspects of cost function value and computational time.

Table 6. GA and PSA algorithm results on instances with size 23<n≤70 and t= $[\frac{n}{2}]$

Problem name	Num. of facilities	GA result	GA time(s)	GA RPD	PSA result	PSA time(s)	PSA RPD
N30_1	30	**4174**	5.515	2.5874	**4174**	2.832	0.047
N30_2	30	**11154.5**	5.613	2.1142	**11154.5**	2.991	0.058
N30_3	30	**23127**	5.099	1.7821	**23127**	2.922	0.070
N30_4	30	**32651.5**	5.112	2.3318	**32651.5**	2.944	0.089
N30_5	30	**60353**	5.216	1.9521	**60353**	3.001	0.042
N40_1	40	55541.5	11.012	1.8121	**55526.5**	6.151	0.391
N40_2	40	**50399**	10.561	1.7159	**50399**	6.018	0.291
N40_3	40	**42118.5**	10.212	2.0015	**42118.5**	6.219	0.382
N40_4	40	**40998**	10.641	1.9725	**40998**	6.181	0.512
N40_5	40	52623	11.001	2.0259	**52562**	6.237	0.403
sko56-1	56	32454	25.901	2.0222	**32292**	15.001	0.177
sko56-2	56	259531	25.818	2.3361	**259500**	15.317	0.261
sko56-3	56	86215	26.061	1.8882	**85881**	15.161	0.212
sko56-4	56	159092	26.005	2.2393	**158939**	14.992	0.196
sko56-5	56	301663.5	25.012	2.1913	**299429.5**	14.981	0.201
A-60-01	60	772746	39.146	0.7607	**772202**	26.069	0.081
A-60-02	60	430679	38.871	0.7125	**430384**	26.153	0.094
A-60-03	60	331190.5	38.755	0.8329	**331140.5**	25.905	0.075
A-60-04	60	201451	39.156	0.8074	**201052**	26.879	0.090
A-60-05	60	165174	39.312	0.8116	**165099**	26.019	0.087
A-70-01	70	784096	60.164	0.7607	**779563**	40.897	0.091
A-70-02	70	738576	60.154	0.7001	**738304**	41.092	0.099
A-70-03	70	767811.5	60.516	0.8321	**764463.5**	40.919	0.089
A-70-04	70	493976	61.289	0.7156	**491217**	41.354	0.098
A-70-05	70	2188127.5	61.541	0.8028	**2187780.5**	41.253	0.086

Table 7. GA and PSA algorithm results on instances with size 23<n≤70 and t= $[\frac{n}{3}]$

Problem name	Num. of facilities	GA result	GA time(s)	GA RPD	PSA result	PSA time(s)	PSA RPD
N30_1	30	**5310**	6.931	1.7099	**5310**	3.916	0.0338
N30_2	30	**14894.5**	6.722	1.9321	**14894.5**	3.616	0.0412
N30_3	30	**27306**	6.515	2.0125	**27306**	3.203	0.0501
N30_4	30	**44498.5**	6.991	1.6991	**44498.5**	3.109	0.0421
N30_5	30	**68998**	6.812	1.8972	**68998**	3.141	0.0363
N40_1	40	**83103.5**	12.808	0.8229	**83103.5**	7.431	0.0501
N40_2	40	63303	12.901	0.8561	**63212**	7.615	0.0404
N40_3	40	**46444.5**	12.813	0.8912	**46444.5**	7.687	0.0498

Continue Table 7. GA and PSA algorithm results on instances with size 23<n≤70 and t= $\lfloor \frac{n}{3} \rfloor$

Problem name	Num. of facilities	GA result	GA time(s)	GA RPD	PSA result	PSA time(s)	PSA RPD
N40_4	40	**45142**	12.991	0.9057	**45142**	7.591	0.0523
N40_5	40	58561	12.901	0.8661	**58492**	7.573	0.0411
sko56-1	56	43037	29.991	1.0965	**42946**	17.701	0.1308
sko56-2	56	309523	30.156	1.1125	**309277**	17.584	0.1936
sko56-3	56	107863	30.117	1.2351	**107469**	17.612	0.2126
sko56-4	56	200531	29.788	1.4083	**200421**	17.629	0.1121
sko56-5	56	374636.5	30.012	0.9934	**374202.5**	17.681	0.1712
A-60-01	60	996339	44.912	0.3954	**996191**	31.056	0.0101
A-60-02	60	**494862**	45.011	0.3148	**494862**	31.346	0.0122
A-60-03	60	411991.5	45.199	0.3569	**411379.5**	30.936	0.0115
A-60-04	60	267530	44.961	0.5421	**267511**	31.004	0.0128
A-60-05	60	187904	44.981	0.4901	**187572**	30.911	0.0131
A-70-01	70	965581	74.105	0.3954	**963209**	50.812	0.0116
A-70-02	70	962106	75.097	0.4165	**961762**	51.019	0.0120
A-70-03	70	971729.5	74.962	0.5108	**970083.5**	50.666	0.0168
A-70-04	70	625524	75.034	0.3368	**625518**	51.199	0.0121
A-70-05	70	2793270.5	74.649	0.4289	**2789182.5**	50.727	0.0109

Table 8. GA and PSA algorithm results on instances with size 23<n≤70 and t= $\lfloor \frac{n}{4} \rfloor$

Problem name	Num. of facilities	GA result	GA time(s)	GA RPD	PSA result	PSA time(s)	PSA RPD
N30_1	30	**6791**	7.012	2.1940	**6791**	3.981	0.2356
N30_2	30	**18928.5**	7.121	2.2878	**18928.5**	3.997	0.2639
N30_3	30	**34523**	7.091	2.0061	**34523**	3.992	0.3152
N30_4	30	**52710.5**	7.053	2.2215	**52710.5**	3.989	0.2069
N30_5	30	**89548**	7.190	2.9925	**89548**	3.988	0.2912
N40_1	40	95555.5	14.081	0.9552	**98512.5**	8.001	0.0414
N40_2	40	73756	14.215	0.8910	**73752**	7.981	0.0347
N40_3	40	65285.5	14.011	1.0159	**65280.5**	7.925	0.0369
N40_4	40	**63314**	14.128	0.7643	**63314**	8.012	0.0509
N40_5	40	74010	14.021	0.9268	**74006**	7.991	0.0382
sko56-1	56	**49495**	32.012	0.7471	**49475**	20.012	0.0917
sko56-2	56	362975	31.911	0.9125	**362393**	19.981	0.0931
sko56-3	56	128851	32.128	0.7169	**128843**	19.391	0.0896
sko56-4	56	239085	32.241	0.8215	**239048**	20.019	0.0945
sko56-5	56	468219.5	31.961	0.7658	**467565.5**	19.882	0.0792
A-60-01	60	1203452	55.311	0.4702	**1203171**	38.057	0.0215
A-60-02	60	557733	55.194	0.6251	**557293**	37.976	0.0222
A-60-03	60	518039.5	55.097	0.5136	**517645.5**	38.023	0.0301
A-60-04	60	**315892**	54.99	0.4413	**315892**	38.152	0.0299
A-60-05	60	246545	55.1	0.5109	**245884**	38.025	0.0267
A-70-01	70	1077997	80.821	0.4701	**1076667**	55.323	0.0164
A-70-02	70	1107759	81.137	0.3898	**1107427**	54.957	0.0139
A-70-03	70	1164060.5	81.144	0.4976	**1163514.5**	54.481	0.0198
A-70-04	70	751420	81.342	0.4612	**751412**	55.022	0.0153
A-70-05	70	3294405.5	80.912	0.5123	**3294390.5**	54.151	0.0168

Table 9. GA and PSA algorithm results on instances with size 23<n≤70 and t= $\lfloor \frac{n}{5} \rfloor$

Problem name	Num. of facilities	GA result	GA time(s)	GA RPD	PSA result	PSA time(s)	PSA RPD
N30_1	30	**7289**	7.927	1.3362	**7289**	4.251	0.1207
N30_2	30	**19785.5**	7.836	1.5912	**19785.5**	4.398	0.1591
N30_3	30	**39524**	7.991	1.6985	**39524**	4.414	0.1398
N30_4	30	**59587.5**	7.983	1.4113	**59587.5**	4.298	0.1429
N30_5	30	**104449**	8.001	1.7821	**104449**	4.329	0.1942
N40_1	40	108232.5	15.036	1.6984	**108168.5**	8.001	0.0428
N40_2	40	**78263**	14.997	1.3262	**78263**	8.070	0.0378
N40_3	40	71439.5	15.055	1.1892	**71428.5**	8.015	0.0529
N40_4	40	**69254**	15.025	1.3657	**69254**	8.015	0.0491
N40_5	40	81719	14.991	1.3156	**81644**	8.018	0.0366
sko56-1	56	55459	42.004	0.5471	**55248**	30.001	0.1151
sko56-2	56	400744	41.991	0.5138	**398605**	29.956	0.1249
sko56-3	56	141395	41.971	0.5945	**141393**	29.981	0.1068
sko56-4	56	264140	42.051	0.6159	**264078**	30.012	0.1315
sko56-5	56	539411.5	42.023	0.5681	**539340.5**	29.973	0.1437
A-60-01	60	1302994	61.529	0.4322	**1302507**	45.198	0.0131
A-60-02	60	680369	61.641	0.3831	**679772**	44.986	0.0193
A-60-03	60	583081.5	62.287	0.3612	**582558.5**	44.699	0.0201
A-60-04	60	345366	61.676	0.4142	**342637**	45.154	0.0142
A-60-05	60	275042	62.239	0.5171	**273604**	44.789	0.0169
A-70-01	70	1274578	96.695	0.4322	**1272547**	70.982	0.0261
A-70-02	70	1232527	97.132	0.4697	**1230359**	70.955	0.0202
A-70-03	70	**1355131.5**	96.167	0.3319	**1355131.5**	71.524	0.0191
A-70-04	70	827554	96.214	0.4189	**827460**	70.867	0.0297
A-70-05	70	3696487.5	97.162	0.3046	**3696447.5**	71.571	0.0253

Conclusion

The facility layout problems encompass a large category of optimization problems that have various names according to their different characteristics, such as the workshop characteristics, the methods to formulate and tackling different versions of basic layout problems and different methods to solve facility layout problems. Based on layout configuration, multi-row problems are a category of facility layout problems that has attracted the notice of researchers recently. In the literature, three problems of double row layout problem, corridor problem and parallel row ordering problem are introduced as multi-row problems that arrange the facilities in two

rows considering different assumptions. These problems are NP-hard. As exact methods in facility layout problems are only able to solve small and medium instances, in the literature, meta-heuristic algorithms have been applied to solve medium and large instances of different categories of facility layout problems that have shown acceptable performances. In fact, for each category of facility layout problems, after introducing different models to formulate and exact algorithms to solve, meta-heuristic algorithms have been used to solve medium and large facility layout problems.

As PROP was introduced in 2013, only there is one paper in the literature that proposed a MIP formulation for PROP. In the mentioned paper, only instances with size $n \leq 23$ were introduced and solved to optimality in a long computational time. So, in this paper both genetic and a novel population based simulated annealing algorithms are implemented for the PROP, for the first time to solve medium and large instances of the PROP. In the literature, GA and SA have shown good performance comparing to other meta-heuristic methods to solve different facility layout problems.

In this paper, describing steps of two proposed algorithms, initially the algorithms are evaluated on several instances from the basic paper of PROP (Amaral, 2013b) with size $n \leq 23$. Results show that two proposed algorithms are able to achieve optimal solutions, although the computational time of proposed PSA is less than GA in all instances. Moreover, the algorithms are further evaluated on instances with sizes n= 30, 40, 56, 60 and 70. These problems do not have optimal solutions and are introduced and solved in this paper for the first time. Results show for instances with size $n=30$, the best obtained cost function values of two proposed algorithms are similar but PSA algorithm consumes less computational time rather than GA. For larger instances with sizes n=40, 56, 60 and 70, the proposed PSA algorithm presents a better performance obtaining smaller cost function values and consuming less computational time. Generally, it can be said that the proposed PSA has shown better performance rather than the proposed GA to solve medium and large instances of the PROP.

Regarding different multi-row problems and different methods to solve them that were described, it seems that because in this paper meta- heuristic algorithms are applied to solve medium and large instances of the PROP for the first time, applying different meta-heuristic algorithms to solve this problem can be considered for future works. Also adding some constraints and assumptions to the multi-row problems according to real assumptions of facility layout problems in industries and generating new problems can be considered as an another suggestion to continue this paper.

References

Ahonen, H., Gomes de Alvarenga, A., & Amaral, A. (2014). Simulated annealing and tabu search approaches for the corridor allocation problem. *European Journal of Operational Research*, *232*(1), 221-233.

Akbari, M., & Maadi, M. (2011). Imperialist competitive algorithm for solving single row facility layout problem. Paper presented at the 4th *International Conference of Iranian Operations Research Society*, (In Persian).

Amaral, A. R. (2006). On the exact solution of a facility layout problem. *European Journal of Operational Research*, *173*(2), 508-518.

Amaral, A. R. (2008). An exact approach to the one-dimensional facility layout problem. *Operations Research*, *56*(4), 1026-1033.

Amaral, A. R. (2009). A new lower bound for the single row facility layout problem. *Discrete Applied Mathematics*, *157*(1), 183-190.

Amaral, A. R. (2012). The corridor allocation problem. *Computers & Operations Research*, *39*(12), 3325-3330.

Amaral, A. R. (2013a). Optimal solutions for the double row layout problem. *Optimization Letters*, *7*(2), 407–413.

Amaral, A. R. (2013b). A parallel ordering problem in facilities layout. *Computers & Operations Research*, *40*(12), 2930-2939.

Amaral, A. R., & Letchford, A. N. (2013). A polyhedral approach to the single row facility layout problem. *Mathematical programming*, *141*(1-2), 453-477.

Anjos, M. F., Fischer, A., & Hungerländer, P. (2016). Solution approaches for the double-row equidistant facility layout problem. In *Operations Research Proceedings* (pp. 17-23). Springer.

Anjos, M. F., Kennings, A., & Vannelli, A. (2005). A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discrete Optimization*, *2*(2), 113-122.

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*, *20*(4), 611-617.

Anjos, M. F., & Yen, G. (2009). Provably near-optimal solutions for very large single-row facility layout problems. *Optimization Methods & Software*, *24*(4-5), 805-817

Bahrami, F., Safari, H., Tavakkoli-Moghaddam, R., & Modarres Yazdi, M. (2017). On modelling door-to-door parcel delivery services in Iran. *Iranian Journal of Management Studies*, *9*(4), 883-906.

Braglia, M. (1996). Optimisation of a simulated-annealing-based heuristic for single row machine layout problem by genetic algorithm. *International Transactions in Operational Research, 3*(1), 37-49.

Chung, J., & Tanchoco, J. (2010). The double row layout problem. *International Journal of Production Research, 48*(3), 709-727.

Datta, D., Amaral, A. R., & Figueira, J. R. (2011). Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research, 213*(2), 388-394.

Deb, K., & Kalyanmoy, D. (2001). *Multi-objective optimization using evolutionary algorithms*. New York, NY: John Wiley & Sons, Inc.

Djellab, H., & Gourgand, M. (2001). A new heuristic procedure for the single-row facility layout problem. *International Journal of Computer Integrated Manufacturing, 14*(3), 270-280.

Drira, A., Pierreval, H., & Hajri-Gabouj, S. (2007). Facility layout problems: A survey. *Annual Reviews in Control, 31*(2), 255-267.

Ficko, M., Brezocnik, M., & Balic, J. (2004). Designing the layout of single- and multiple-rows flexible manufacturing system by genetic algorithms. *Journal of Materials Processing Technology, 157*, 150-158.

Ghosh, D., & Kothari, R. (2012). *Population heuristics for the corridor allocation problem*. Indian Institute of Management Ahmedabad (IIMA), Research and Publication Department.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*(2), 95-99.

Gomes de Alvarenga, A., Negreiros-Gomes, F. J., & Mestria, M. R. (2000). Metaheuristic methods for a class of the facility layout problem. *Journal of Intelligent Manufacturing, 11*(4), 421-430.

Guan, J., & Lin, G. (2016). Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem. *European Journal of Operational Research, 248*(3), 899-909.

Heragu, S. S., & Alfa, A. S. (1992). Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operational Research, 57*(2), 190-202.

Heragu, S. S., & Kusiak, A. (1988). Machine layout problem in flexible manufacturing systems. *Operations Research, 36*(2), 258-268.

Heragu, S. S., & Kusiak, A. (1991). Efficient models for the facility layout problem. *European Journal of Operational Research*, 53(1), 1-13.

Hosseini-Nasab, H., & Emami, L. (2013). A hybrid particle swarm

optimisation for dynamic facility layout problem. *International Journal of Production Research*, *51*(14), 4325-4335.

Hsu, C.-M. (2012). Improving the lighting performance of a 3535 packaged hi-power LED using genetic programming, quality loss functions and particle swarm optimization. *Applied Soft Computing*, *12*(9), 2933-2947.

Hungerländer, P., & Rendl, F. (2013). Semidefinite relaxations of ordering problems. *Mathematical Programming*, *140*(1), 77-97.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680.

Kothari, R., & Ghosh, D. (2012a). *Path relinking for single row facility layout*. Indian Institute of Management, Ahmedabad.

Kothari, R., & Ghosh, D. (2013b). Insertion based Lin–Kernighan heuristic for single row facility layout. *Computers & Operations Research*, *40*(1), 129-136.

Kothari, R., & Ghosh, D. (2013bc). Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods. *European Journal of Operational Research*, *224*(1), 93-100.

Kothari, R., & Ghosh, D. (2014a). An efficient genetic algorithm for single row facility layout. *Optimization Letters*, *8*(2), 679-690.

Kothari, R., & Ghosh, D. (2014b). A scatter search algorithm for the single row facility layout problem. *Journal of Heuristics*, *20*(2), 125-142.

Kouvelis, P., & Chiang, W.-C. (1996). Optimal and heuristic procedures for row layout problems in automated manufacturing systems. *Journal of the Operational Research Society*, *47*(6), 803-816.

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*, *87*(1), 65-73.

Kunlei, L., Chaoyong, Z., Liang, G., & Xinyu, S. (2011). Single row facility layout problem using an imperialist competitive algorithm. Proceedings from *the 41st International Conference on Computers & Industrial Engineering.*

Love, R., & Wong, J. (1976). On solving a one-dimensional space allocation problem with integer programming. *INFOR: Information Systems and Operational Research*, *14*(2), 139-143.

Maadi, M., Javidnia, M., & Ghasemi, M. (2016). Applications of two new algorithms of cuckoo optimization (CO) and forest optimization (FO) for solving single row facility layout problem (SRFLP). *Journal of Artificial Intelligence and Data Mining*, *4*(1), 35-48.

Murray, C. C., Smith, A. E., & Zhang, Z. (2013). An efficient local search heuristic for the double row layout problem with asymmetric material flow. *International Journal of Production Research*, *51*(20), 6129-6139.

Orand, S. M., Mirzazadeh, A., Ahmadzadeh, F., & Talebloo, F. (2015). Optimization of the inflationary inventory control model under stochastic conditions with Simpson Approximation: Particle swarm optimization approach. *Iranian Journal of Management Studies*, *8*(2), 203.

Ozcelik, F. (2012). A hybrid genetic algorithm for the single row layout problem. *International Journal of Production Research*, *50*(20), 5872-5886.

Palubeckis, G. (2015). Fast local search for single row facility layout. *European Journal of Operational research*, *246*(3), 800-814.

Palubeckis, G. (2017). Single row facility layout using multi-start simulated annealing. *Computers & Industrial Engineering*, *103*, 1-16.

Phadke, M. S. (1989). *Quality Engineering Using Robust Design*. NJ, US: Prentice Hall.

Picard, J.-C., & Queyranne, M. (1981). On the one-dimensional space allocation problem. *Operations Research*, *29*(2), 371-391.

Romero, D., & Sánchez-Flores, A. (1990). Methods for the one-dimensional space allocation problem. *Computers & Operations Research*, *17*(5), 465-473.

Samarghandi, H., & Eshghi, K. (2010). An efficient tabu algorithm for the single row facility layout problem. *European Journal of Operational Research*, *205*(1), 98-105.

Samarghandi, H., Taabayan, P., & Jahantigh, F. F. (2010). A particle swarm optimization for the single row facility layout problem. *Computers & Industrial Engineering*, *58*(4), 529-534.

Satheesh Kumar, R., Asokan, P., Kumanan, S., & Varma, B. (2008). Scatter search algorithm for single row layout problem in FMS. *Advances in Production Engineering & Management*, *3*, 193-204.

Simmons, D. M. (1969). One-dimensional space allocation: An ordering algorithm. *Operations Research*, *17*(5), 812-826.

Solimanpur, M., Vrat, P., & Shankar, R. (2005). An ant algorithm for the single row layout problem in flexible manufacturing systems. *Computers & Operations Research*, *32*(3), 583-598.

Tang, C.-Y., Wu, Y.-L., & Peng, C.-C. (2012). Fundamental matrix estimation by multiobjective genetic algorithm with Taguchi's method. *Applied Soft Computing*, *12*(1), 553-558.

Teo, Y. T., & Ponnambalam, S. (2008). A hybrid ACO/PSO heuristic to solve single row layout problem. Proceedings from CASE 2008: *4th IEEE International Conference on the Automation Science and Engineering*, Washington, DC.

Van Laarhoven, P. J., & Aarts, E. H. (1987). *Simulated annealing: Theory and applications.* NY: Springer, 7-15.

Yadegari, E., Najmi, H., Ghomi-Avili, M., & Zandieh, M. (2015). A flexible integrated forward/reverse logistics model with random path-based memetic algorithm. *Iranian Journal of Management Studies*, *8*(2), 287.

Zareei, M., & Hassan-Pour, H. A. (2015). A multi-objective resource-constrained optimization of time-cost trade-off problems in scheduling project. *Iranian Journal of Management Studies*, *8*(4), 653-685.

Zhang, Z., & Murray, C. C. (2012). A corrected formulation for the double row layout problem. *International Journal of Production Research*, *50*(15), 4220-4223.

Zuo, X., Murray, C. C., & Smith, A. E. (2014). Solving an extended double row layout problem using multiobjective tabu search and linear programming. *IEEE Transactions on Automation Science and Engineering*, *11*(4), 1122-1132.