Applied Soft Computing 24 (2014) 457–469

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A hybrid multi-population genetic algorithm for the dynamic facility layout problem

Hani Pourvaziri [a], B. Naderi [b,*]

[a] Department of Industrial and Mechanical Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran
[b] Department of Industrial Engineering, Faculty of Engineering, University of Kharazmi, Karaj, Iran

ARTICLE INFO

Article history:
Received 28 February 2013
Received in revised form 10 June 2014
Accepted 27 June 2014
Available online 22 July 2014

Keywords:
Dynamic facility layout problem
Multi-population genetic algorithm
Heuristics

ABSTRACT

Due to inherent complexity of the dynamic facility layout problem, it has always been a challenging issue to develop a solution algorithm for this problem. For more than one decade, many researchers have proposed different algorithms for this problem. After reviewing the shortcomings of these algorithms, we realize that the performance can be further improved by a more intelligent search. This paper develops an effective novel hybrid multi-population genetic algorithm. Using a proposed heuristic procedure, we separate solution space into different parts and each subpopulation represents a separate part. This assures the diversity of the algorithm. Moreover, to intensify the search more and more, a powerful local search mechanism based on simulated annealing is developed. Unlike the available genetic operators previously proposed for this problem, we design the operators so as to search only the feasible space; thus, we save computational time by avoiding infeasible space. To evaluate the algorithm, we comprehensively discuss the parameter tuning of the algorithms by Taguchi method. The perfectly tuned algorithm is then compared with 11 available algorithms in the literature using well-known set of benchmark instances. Different analyses conducted on the results, show that the proposed algorithm enjoys the superiority and outperformance over the other algorithms.

1. Introduction

The facility layout problem consists of the determination of the most efficient facilities arrangement within a factory. The facility can be defined as a manufacturing cell, an administrative office building or a machine. An efficient layout causes an efficient material handling (MH) between facilities and consequently decreases the work-in-process and inventory holding costs. An efficient layout also contributes to the overall efficiency of operations and can save many overall costs such as production costs and inventory holding costs.

For manufacturing plants, material handling cost is the most frequently used indicator to determine the efficiency of a layout; since it represents 20–50% of the total operating cost and 15–70% of the total cost of manufacturing a product [1]. Throughput time, quality and inventory levels are also indirectly related to the material flow. The material handling cost is calculated based on flows of materials between facilities and distances between the locations of the facilities. If the flows of materials do not change for a long time, the static facility layout problem can be used to specify the best layout. Meller and Gau [2] provided a comprehensive review of recent studies about the facility layout models and algorithms.

In real world industry, companies often work in dynamic and changeable environments. Thus, we should analyze the current layout and restructure the layout according to the changes occurred. Francis et al. [3] discuss some reasons to modify the layout. Among all, one can refer to the following: changes in the product/process design, adding/deleting a product, significant increase/decrease in the demand of a product, the replacement of equipment, the adoption of new safety standards and unexplainable.

Afentakis et al. [4] stated that when system characteristics change, the MH costs will likely significantly increase. An important way to reduce this increment is to rearrange the workstations (facilities). But the rearranging of workstations may impose some cost such as labor cost, equipment cost, and the cost of production loss. Consequently, there is a tradeoff between increased MH cost and the rearrangement cost for the shift from the present layout to a new layout. The problem of compromising a balance between MH cost and the rearrangement cost is known as the dynamic facility layout problem (DFLP).

In the DFLP, the future is divided into a number of periods of time where a period may be defined, for instance, as a week, a month, or a

* Corresponding author. Tel.: +98 9128305906.
E-mail address: bahman_naderi62@yahoo.com (B. Naderi).

http://dx.doi.org/10.1016/j.asoc.2014.06.051
1568-4946/

year. For each period, data related to the flow of materials between each pair of facilities are forecasted. It is assumed that the flow does not change throughout a period. The objective of DFLP is to obtain layouts for each period in the planning horizon in order to minimize the sum of the material handling and rearrangement costs. The assumptions for the standard DFLP are explained as follows [5]:

1. The distances between locations are known and constant.
2. Planning horizon is divided to T period.
3. In each time period, each facility should be put on only one location and one facility can be placed to only one location.
4. In each time period, the material flow between each pair of facilities is known and does not change over the period.
5. The objective is to obtain the layout plan (i.e., the layout for all periods) which minimizes the sum of the material handling and rearrangement costs.

In this paper, we present a novel hybrid multi-population genetic algorithm (HMPGA) to solve DFLP. We also introduce a heuristic algorithm to generate the individuals of the initial populations. We describe a representation of layouts that can be used efficiently in genetic encoding. In our multi-population strategy each population evolves independently from each other. After a pre-determined number of generations, all the populations are combined and make a new population (main population). The main population continues to evolve until stopping criteria is satisfied. In this way, we can guarantee that the various parts of the solution space are most likely searched. After a comprehensive parameter tuning, the algorithm is evaluated using advanced statistical tools. To evaluate the performance of the proposed algorithm, we compare it with 11 available algorithms in the literature on well-known set of benchmark instances presented in Balakrishnan and Cheng [6].

The structure of this paper is as follows: In "Literature review" section, a review of related works to DFLP is presented. In "Problem formulation" section, the mathematical formulation of the problem is described. In "The proposed hybrid multi-population genetic algorithm" section, we introduce the proposed hybrid multi-population genetic algorithm. The experimental evaluation of the proposed algorithm is given in "Parameter tuning" section. A summary and discussion on future works concludes this paper.

2. Literature review

The DFLP was first introduced in detail by Rosenblatt [7]. The author proposed an optimization approach based on dynamic programming. He considered only a limited set of good layouts for each period. But, this approach is computationally unaffordable for real size problems; because DFLP is NP-hard and only small problems can be solved optimally in a reasonable computational time. In other words, the number of possible solutions for a DFLP instance with N departments and T periods is $(N!)^T$. Thus, most of the effective algorithms for the DFLP are heuristics and meta-heuristics.

Urban [8] proposed a heuristic algorithm based on the computerized relative allocation of facilities technique (CRAFT). Lacksonen and Enscore [9] introduced five heuristics to solve the DFLP. The heuristics considered in their study was based on dynamic programming, branch and bound, cutting plane algorithm, cut tree algorithm, and CRAFT.

There have been several meta-heuristics suggested for DFLP as Well. Conway and Venkataraman [10] solved the DFLP by a simple genetic algorithm. Balakrishnan and Cheng [6] presented a nested loops genetic algorithm (NLGA) to solve the DFLP. In the inner loop, genetic operators were utilized to improve some of the individuals in the population. The outer loop consists of periodically replacing a number of poor individuals in the population whereupon it ensures that the inner loops work with different populations. This expands the search space and should lead to high quality solutions. Kaku and Mazzola [11] presented a tabu search (TS) for the DFLP. This TS is a two-stage search process that incorporates the diversification and intensification strategies. Balakrishnan et al. [12] presented two heuristics that improved Urban's steepest-descent pairwise exchange heuristic. The first heuristic uses Urban's heuristic to generate solutions for the DFLP. Then, the solutions generated for each forecast window is improved using a backward-pass pairwise exchange heuristic, and finally the best solution is selected. The second heuristic combines Urban's heuristic with dynamic programming. Balakrishnan et al. [13] presented a hybrid genetic algorithm for the DFLP. They proposed a heuristic crossover operator based on dynamic programming (DP). First, each multi period layout in the parent pool is decomposed into some single period layouts. Then, after removing the duplicated layouts, best combination among all the layouts are obtained by using DP.

Erel et al. [14] also purposed a three-stage heuristics to solve the DFLP using dynamic programming. In the first phase, a set of good layouts is obtained by weighted flow data from the T time periods. In the second stage, the set of solutions obtained in the first stage and dynamic programming is used to obtain solutions to the DFLP. In the third stage, a random descent pairwise interchange strategy is used to improve the solutions obtained in the second stage. McKendall et al. [15] developed two simulated annealing algorithms (SA). The first one is a direct adaptation of SA, and the second algorithm is the same as the first one but with a look-ahead/look-back strategy.

Rodriguez et al. [16] presented a hybrid metaheuristic based on the genetic algorithm and tabu search for the DFLP in which tabu search was used as a local optimizer to find all global optima. The parallelization of the GA and TS were based on an asynchronous master-slave model which leads to a substantial economy in computation time. Baykasoglu et al. [17] added the budget constraint for total rearrangement costs over the whole planning horizon and proposed an ant colony algorithm (ACO) as solution method. In the first part, the ACO was used to solve the unconstrained dynamic facility layout problems. In the second part, the budget constraints are added to the problems. The rearrangement cost values that were obtained from the solutions of the unconstrained problems were used to determine the budget constraints. The budget constraints for each period may be found by adding 10% more to the rearrangement costs of the unconstrained problem.

Three hybrid ant systems were developed to by McKendall and Shang [5] to solve DFLP. The first, called HAS I, was a direct implementation of the HAS for the DFLP in which after performing a predefined pairwise exchanges using the pheromone trail matrix, a random descent pairwise exchange heuristic would be used to improve the layout plans. The second heuristic, called HAS II, was closely like HAS I, except that instead of the random descent pairwise exchange heuristic, a local search heuristic based on SA was accomplished. The third heuristic, called HAS III, was exactly like HASI, except that the random descent pairwise exchange heuristic was improved by a look-ahead/look-back strategy.

More recently, Balakrishnan and Cheng [18] introduced a new approach for DFLP called rolling horizon problem. In the standard DPLP the multi-period plan is created at the beginning of period one. But, in the rolling horizon problem, against DFLP, after the first period, the data for period one is dropped and the multi-period plan is recalculated according to the remaining periods of planning horizon.

Sahin and Turkbey [19] introduced a new hybrid tabu-simulated annealing algorithm for DFLP. Their algorithm was basically an SA approach enhanced with tabu list to avoid cycling and reduces

computation time. They also solved the same problems using a pure SA algorithm and a pure tabu search algorithm for comparison purposes.

McKendall and Liu [20] presented three TSs for DFLP. The first algorithm was a direct implementation of TS. The second one was improved by adding some diversification and intensification strategies. In other words, it uses dynamic tabu tenure length and a diversification and intensification strategy similar to the one described in Chiang and Kouvelis [21].The third heuristic, called probabilistic TS, was a modification of the first TS in which instead of accepting the best allowable move, a move from the top M admissible moves is randomly selected. Baykasoglu and Gindy [22] showed the application of simulated annealing to the DFLP. For an extensive review of the problem, assumptions and solution techniques for DFLP, one can refer to Balakrishnan and Cheng [23].

The studies mentioned above share a common assumption that all the departments are of equal size. There are some other studies that do not consider this assumption. Two recent examples of such studies are Dunker et al. [24] and McKendall and Hakobyan [25]. Table 1 shows more recent solution techniques available in the literature for the DFLP.

3. Problem formulation

The mathematical formulation of the discrete representation of the DFLP is presented below. This model is in form of quadratic binary integer programming. This formulation of the model is presented by McKendall et al. [15]. The parameters and indices are:

N	The number of departments/locations
T	The number of time periods.
i,j,k,l	Index for departments/locations
t	Index for time periods
$A_{t,i,j,l}$	The cost of shifting department i from location j to l in period t
$C_{t,i,j,k,l}$	The cost of material flow between department i placed in location j and k placed in location l in period t.

The decision variables are:

$X_{t,i,j}$	Binary variable taking value 1 if department i is assigned to locations j in period t, and 0 otherwise.
$Y_{t,i,j,l}$	Binary variable taking value 1 if department i is shifted from locations j to location l in period t, and 0 otherwise.

The model is as follows.

$$MinZ = \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{t,i,j,l} \times Y_{t,i,j,l} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} C_{t,i,j,k,l} \times X_{t,i,j} + X_{t,k,l} \quad (1)$$

Subject to.

$$\sum_{j=1}^{N} X_{t,i,j} = 1 \qquad i = 1, 2, \ldots, N; \quad t = 1, 2, \ldots, T \quad (2)$$

$$\sum_{i=1}^{N} X_{t,i,j} = 1 \qquad j = 1, 2, \ldots, N; \quad t = 1, 2, \ldots, T \quad (3)$$

$$Y_{t,i,j,l} = X_{t-1,i,j} \times X_{t,i,l} \qquad i, j, l = 1, 2, \ldots, N; \quad t = 2, \ldots, T \quad (4)$$

$$X_{t,i,j} \in \{0, 1\} \qquad i, j = 1, 2, \ldots, N; \quad t = 1, 2, \ldots, T \quad (5)$$

$$Y_{t,i,j,l} \in \{0, 1\} \qquad i, j, l = 1, 2, \ldots, N; \quad t = 2, \ldots, T \quad (6)$$

The objective function (1) minimizes the total cost of layout rearrangements and material handling cost. Constraint set (2) ensures that each location department is assigned to only one location department during each period, and constraint set (3) assures that exactly one department is assigned to each location in each period. Constraint set (4) specifies if a department is shifted between locations in two consecutive periods. Lastly, constraint sets (5) and (6) determine the decision variables.

To formulate a problem with N departments/locations and T time periods, this model requires $N^3T + N^2T$ binary variables and $N^3T + 2NT$ constraints.

4. The proposed hybrid multi-population genetic algorithm

Population-based algorithms, particularly genetic algorithms, perform well on a wide class of problems. A typical genetic algorithm consists of one single population; however, better results can be achieved by introducing multiple populations in parallel. The noteworthy features of hybrid multi-population genetic algorithm against other studied methods were proven in Toledo et al. [26]. They used a hybrid multi-population genetic algorithm to solve sets of hard and large test instances of multi-level capacitated lot sizing problem with backlogging and achieved high quality solutions compared with other methods from the literature. Despite the successful results reported for multi-population hierarchically structured procedures [27], there is no cases have been applied to solve dynamic facility layout problem, specially with a hybrid feature. In addition, since the multi-population genetic algorithm let the sub-population evolves separately, it might be able to explore more extreme solutions [28]. Therefore, we have been thinking of developing a novel multi-population genetic algorithm to solve DFLP. The individuals of each population are generated purposefully by a heuristic. Then, each population is treated and evolved independently. We assume that the size, the crossover and the mutation probability of each subpopulation may be different.

After a predetermined number of generations (isolation time) all the populations share their information, i.e. the quality individuals/chromosomes. In other words, some of chromosomes with the lowest total cost are combined together and make a new population, named main population. The contribution of each population in forming the main population is determined according to a parameter called migration rate. Thus, the main population is a combination of best individuals of initial populations. The main population continues to evolve by the genetic operators.

To enhance the process of evolving the main population, a local search algorithm is employed. The multi-population mechanism ensures the diversification capability of the algorithm while the local search is allocated to improve the intensification capability. The local search is used to explore the neighbors of the suitable individuals. The local search algorithm could be either an iterative local search or a local search based on meta-heuristics. Among different possible options, we utilize the simulated annealing algorithm, because it shows high performance in the literature of the similar problems [29]. At each iteration, the best chromosome is selected as the initial solution of the local search, and it is improved by ameliorator search strategy until a stopping criteria is met. Then, the improved chromosome is transformed to the main population which is upgraded by genetic operators. Fig. 1 shows the outline of the proposed hybrid multi-population genetic algorithm.

4.1. Initial individuals and populations

The initial solutions have a great impact on the performance of meta-heuristics. That is, if the GA starts from good initial population, it would likely lead to better final solutions. If all the initial solutions are generated by a rule, the algorithm likely loses its diversity as a result of generating solutions close to a particular solution space. On the other hand, if all initial solutions are generated randomly, the algorithm loses its computational time to reach

Table 1
Meta-heuristic and hybrid heuristic for the DFLP.

Authors	Method	Descriptions
Conway and Venkataramanan [10]	Genetic algorithm	A direct implementation of genetic algorithm
Balakrishnan and Cheng [6]	Genetic algorithm	A new cross over, mutation and replacement strategy for genetic algorithm
Kaku and Mazzola [11]	Tabu search	Introducing a TS heuristic for DPLP
Baykasoglu and Gindy [22]	Simulated annealing	A new and improved models and algorithms to solve DPLP
McKendall et al. [15]	Heuristic simulated annealing	Heuristic SA with alook-ahead/look-back strategy added
Balakrishnan et al. [12]	Hybrid dynamic programming	An improved dynamic pair-wise exchange based on dynamic programming
Erel et al. [14]	Hybrid dynamic programming	A new method in which designing for each period is done by a heuristic
Balakrishnan et al. [13]	Hybrid genetic algorithm	A heuristic cross over using dynamic programming
Rodriguez et al. [16]	Genetic algorithm-Tabu search	A new hybridization GA and TS to improve convergence speed
Baykasoglu et al. [17]	Hybrid ant colony	Solving DFLP with and without the budget constraints by ACO
McKendall and Shang [5]	Hybrid ant system	A hybrid SA-ant colony to solve the DFLP
Sahin and Turkbey [19]	Simulated annealing	A heuristic SA for DFLP the budget constraints
McKendall and Liu [20]	Tabu search heuristics	TS heuristics with diversification, intensification strategies and a probabilistic TS

acceptable solutions. Care must be taken to have both diversifying and intensifying solutions in the population.

In the literature, the initial population is widely generated randomly. After initial experiments, we come to this conclusion that using three parallel populations is the best selection for this type of problem. The initial individuals of two populations are obtained by a heuristic and the individuals of the third population are generated randomly.

The proposed heuristic can be described as follows. First, we consider DFLP with relaxed conditions in which the binary constraint of variables is ignored.

In other word the relaxed model is exactly as the original model except that the constraint $X_{t,i,j} \in \{0, 1\}$ is changed to $0 \leq X_{t,i,j} \leq 1$. Through this relaxation a pure integer nonlinear programming model is changed to a nonlinear programming (NLP) model. The relaxed model is solved using the well-known NLP solver, CONOPT [30] via the GAMS modeling language. Thus we are able to solve the problem in reasonable time. The solutions obtained by this nonlinear model are used as a probability distribution of assigning facilities to locations for generating initial populations. That is, $X_{t,i,j}$ which was a binary variable in the original model, now is transformed to a relaxed variable which can take any real value between 0 and 1. The definition of $X_{t,i,j}$ can be interpreted as the probability of assigning department i to location j during period t. As mentioned before, this concept plays a key role in generating populations, where it is the probability distribution for assigning each facility to each location during a period. In fact, the optimal values of the relaxed DFLP are empirical distributions used by the HMPGA to generate initial populations. The individuals of the first subpopulation are generated according to these probabilities. For better comprehending, suppose that after solving the relaxed model, we have $X_{1,1,1} = 0.9$ and $X_{1,1,2} = 0.1$. This means the probability of assigning facility 1 to location 1 in period 1 equals 0.9. Likewise, probability of assigning facility 1 to location 2 in period 1 is equal to 0.1. Therefore, in 90 percent of the first subpopulation's chromosomes, facility 1 in period 1 is assigned to location 1 and in 10 percent of the first subpopulation's chromosomes, facility 1 in period 1 is assigned to location 2.

The individuals of the first subpopulation generated by the heuristic above mentioned are expected to have better fitness and more likely be attracted by the most promising regions of search space. Therefore, subpopulation 1 searches in the best region of solution space. The subpopulation 2 is generated reversely. That is, the probability of assigning department i to location j at period t equals $1 - X_{t,i,j}$. Thus, a non-promising region is represented by individuals of subpopulation 2. The individuals of subpopulation 3 are fully randomly generated in order to give the chance of visiting throughout the solution space.

The procedure: the proposed multi-population genetic algorithm

Solve the relaxed problem.
Initialize the algorithm (subpopulations and parameters)
While *the first stopping criterion is not met* **do**
  Evolve each population by genetic operators
Endwhile
Generate the main population by combining the subpopulations
While *the second stopping criterion is not met* **do**
  Evolve the main population by genetic operators
  Apply the local search mechanism on the best individual
Endwhile

Fig. 1. The outline of the proposed hybrid multi-population genetic algorithm.

4.2. Representation scheme

Each solution in the population is represented by T substrings each of which corresponds to a layout for a certain period. The representation is based on the permutation of N department numbers. By scanning from right to left, the first department number is located into location 1; the second department number is located in location 2, and so on. Consider a problem with 5 departments and 3 periods. One possible solution is shown by Fig. 2.

4.3. Crossover and mutation operators

Once all the individuals of three subpopulations have been generated, two genetic operators, called crossover and mutation, evolve these individuals. The crossover operator combines two solutions to generate new individuals. It should work so as to avoid generating infeasible solutions; otherwise, it loses the computational time due to search the infeasible space. The previous crossover operators applied to DFLP often generate infeasible solutions. Refs. [10,12] use the crossover operators that might generate infeasible solutions.

Unlike the previous crossover, we propose a novel crossover operator that only generates feasible solutions. Consequently, checking the feasibility of the solutions is no longer required. The proposed crossover operator has the following steps.

(1) Two strings from the parent pool are randomly selected.
(2) A cross point is randomly selected for each period (cross points 1,. . ., T).
(3) For the first period, the department numbers before the cross point 1 of Parent 1 are directly copied in the offspring. The remaining department numbers are put into empty positions according to their relative positions in Parent 2. The procedure is repeated for subsequent periods.

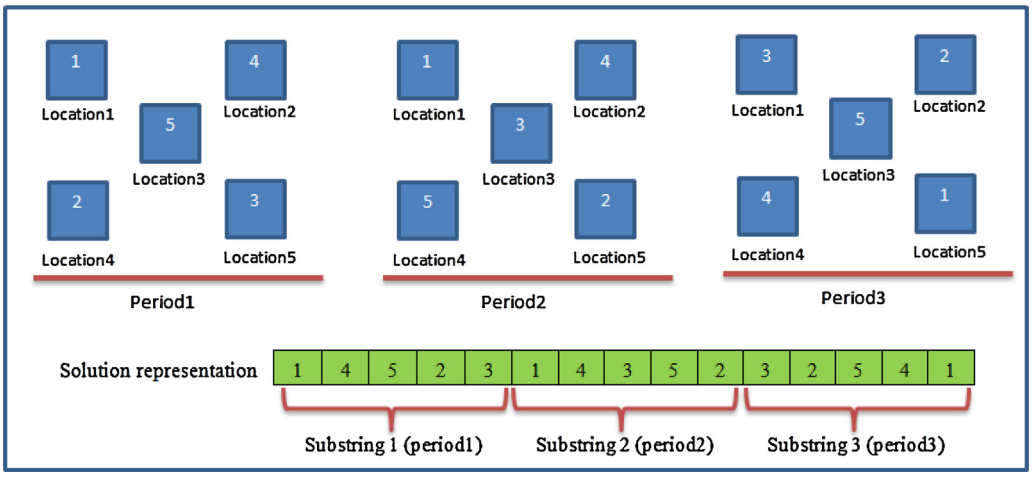

Fig. 2. The representation of a solution with 5 departments and 3 periods.

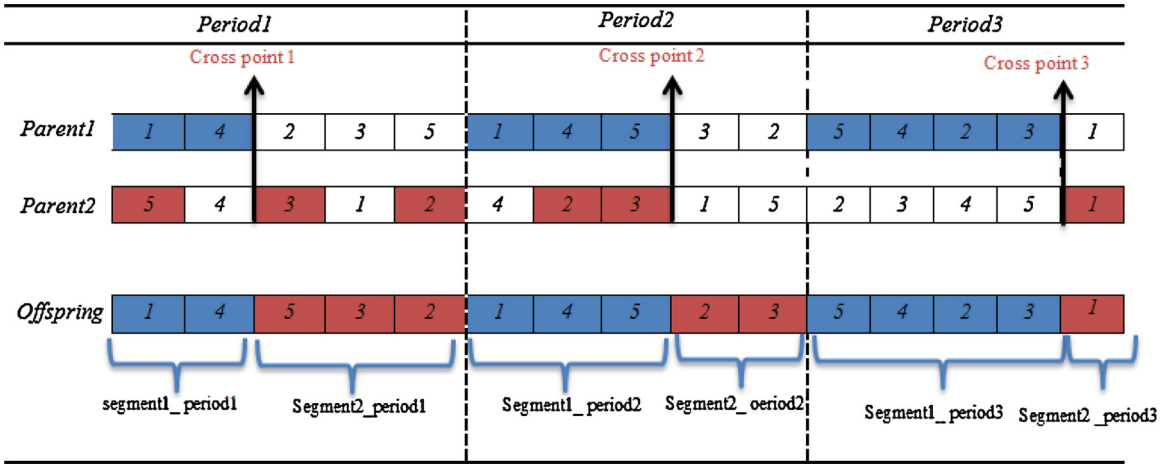

Fig. 3. The example for crossover operator.

Fig. 3 shows the procedure of the crossover for a problem with 5 departments and 3 periods.

The second genetic operator is mutation. The purpose is to make a tremendous change in offspring to avoid getting trapped into local optima. We apply the following procedure as the mutation operator. First, two period is randomly selected. Then, the layouts in these two periods are swapped. Fig. 4 shows the procedure for a problem with 5 departments and 3 periods.

4.4. Local search mechanism

Once the high-performance regions of the search space are identified by the GA, it may be useful to use a local search to optimize the best members of the population [31].The purpose, behind use a local search next to GA is to use of the power of the GA to work on the solution in a global sense while allowing the best solution to locally be optimized. SA has been proven effective for local search [29]. The concept of joining SA and GA has been proposed by Huntley and Brown [32] and Grefenstette [31]. Toledo et al. [27] applied SA for a deeper search in the neighborhood of the best chromosome found by the multi-population GA and showed the superior performance of the proposed hybridized multi-population GA against a non-hybridized one. Consequently, our hybrid algorithm incorporates the best features of genetic algorithm (searching larger regions of solution spaces) and simulated annealing (refining exhaustive solution of local region) to evolve the main population. In this way we apply SA to intensify the search for solutions in the neighborhood of the best individuals found by the main population.

SA is a general probabilistic local search algorithm proposed by Kirkpatrick et al. [33]. The main idea of SA search process to prevent getting trapped in a poor local optimum is to accept not only better solutions but also the worse neighbor solutions with a certain probability. This probability depends on the difference between the solution value of $f(x)$ and neighborhood value $f(x_n)$ and the so-called

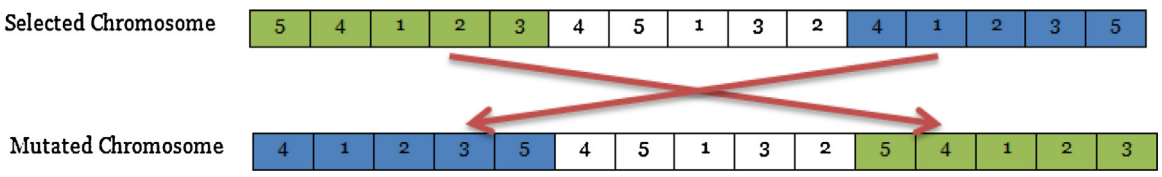

Fig. 4. The example for mutation operator.

temperature T at iteration n ($prob = \min\{1, e^{-(f(x_n)-f(x))/T_n}\}$). The way that T decreases is called annealing schedule. The general applied local search is much like SAI proposed by McKendall et al. [15]. Next sub-sections describe a more detail of local search algorithm.

4.4.1. Neighborhood structure

One of the first considerations in simulated annealing will be how you move from one state to another. This means that it is necessary to define a neighborhood. We defined the neighborhood structure as swapping the locations of two facilities. First, a substring (period) is selected randomly. Then, two different locations (bit) are randomly selected and the values inside them (facilities) are swapped. For example, as shown in Fig. 5, the location of facilities four and two in period two is exchanged.

4.4.2. Initial temperature

One of the main input parameter is initial temperature T_0. A suitable initial temperature T_0 is high enough that results in an average increase of acceptance probability *prob*. Since the *prob* is dependent on both temperature and difference degree of state (δ), it is rational to determine the initial temperature proportional to the difference degree of state values in SA. To this aim, first, a large set of solutions is randomly generated and then the standard division (*St.Dev*) of objective function for those random solutions is calculated. Then, the initial temperature is computed by the following formula:

$$T_0 = \frac{-St.Dev}{\ln(0.5)} \tag{7}$$

Therefore, the initial temperature results the probability of accepting a worse neighboring solution be at least 0.50.

4.4.3. Cooling rate

The cooling schedule is used to reduce the current temperature in each iteration n is determined by the following equation:

$$T_n = T_0 \alpha^{r-1} \tag{8}$$

where T_0 is the initial temperature, α is called the cooling ratio and is determined according to parameter tuning. $r-1$ is the number of temperature reductions. Before reducing the temperature, a number of moves need to be performed in order to ensure that the system is at a steady state. Therefore, the temperature should be reduced after a certain number of attempted exchanges.

4.4.4. Stopping criteria

To limit the number of iterations of both inner and outer loop of SA algorithms, some convergence experiments were performed and the best criterion was applied as follows. For stopping SA in a temperature level, first, a set of iterations are defined as a round (each round contain 100 iterations). If the change between the best known solutions of two successive rounds remains constant, we conclude that the system has reached thermal equilibrium and therefore we reduce the temperature; otherwise, we keep perturbing the solutions by creating neighboring solutions, until reaching equilibrium.

For the outer loop of SA, if the current temperature receive to a predefined temperature, denoted by T_{min}, the algorithm is stopped. T_{min} is set to value 0.01. Consequently, a certain number of iterations are carried out at each temperature and then the temperature is decreased. This is repeated until the system freezes into a steady state and the stopping criterion of outer loop is satisfied.

5. Parameter tuning

Parameters tuning highly affects the performance of metaheuristics algorithms [34]. To conduct an experiment for such an analysis, there are several statistical designs [35]. The most frequently used design is a full factorial one. Yet, it is not efficient when the number of factors increases, because it needs many trials. As a result, it becomes costly and difficult to perform. Among the alternative statistical design to tune the algorithm, Taguchi method is more effective because it can study a large number of factors with a small number of required experiments [36]. This is done though a set of orthogonal arrays.

In Taguchi method, the factors are separated into two main classes: controllable and noise factors. This method minimizes the effect of noise and determines the optimal level of controllable factors. The signal-to-noise ratio, which is closely related to Taguchi quality loss, is used to measure the stability of a process [37]. Here, the term "signal" denotes the desirable value (mean response variable) and "noise" denotes the undesirable value (standard deviation). Formulation of the S/N is an estimate of how samples deviate from the center of population [38]. The aim is to maximize the signal-to-noise ratio. The formula is used to calculate the S/N ratio is related to the objective function. In the Taguchi approach, the objective function is categorized into three groups of the smaller-the better, the larger-the better, and the nominal value-is expected. Since our objective function is minimization of total cost, its corresponding S/N ratio is:

$$S/N \text{ ratio} = -10\log_{10}\left[\frac{1}{M}\sum_{i=1}^{M} Y_i^2\right] \tag{9}$$

where Y_i^2 denotes the response value of the ith instance. In what follows, the levels of the parameters are first introduced. Then, after generating random instance, the Taguchi method will be carried out to determine the best level of each factor. Finally the analysis of variance (ANOVA) is carried out to show the relative magnitude of factor's effects on the algorithm's performance.

5.1. The Taguchi experimental design

The proposed algorithm includes 8 factors. Table 2 shows the considered levels for these 8 factors. In order to select an appropriate orthogonal array, first we should determine the number of degrees of freedom. The total degree of freedom is 22; therefore, the appropriate array must have at least 22 trials. We use the array L_{32} for the experiment and thus we have 32 different HMPGA.

In order to run designed experiments, we generate a set of instances as follows: we have $N = 10, 20, 30$ departments and $T = 5$, 10 periods. Therefore, each trial (experiment) consists of 6 different instances based on problem size. For each instance five replications are considered. The material handling and rearrangement costs are randomly generated using the procedure described in [10]. The stopping criterion used when testing all instances is set to a CPU time limit fixed to N^2T second. This stopping criterion allows for more time as the number of departments or periods increases. To calculate the performance of algorithm in each instance, the objective function is transformed into its relative percentage deviation (*RPD*) defined as follow:

$$RPD_i = \underset{j}{\text{Mean}} \frac{total\ cost_{i,j} - LB_i}{LB_i} \tag{10}$$

where $total\ cost_{i,j}$ is the total cost obtained for instance i in replication j and LB_i is the minimum total cost obtained for instance i. The results are transformed into S/N ratio according to the formula presented below:

$$S/N \text{ ratio}: \quad \eta_k = -10\log_{10}\left[\frac{1}{M}\sum_{i=1}^{M} RPD_i^2\right] \tag{11}$$

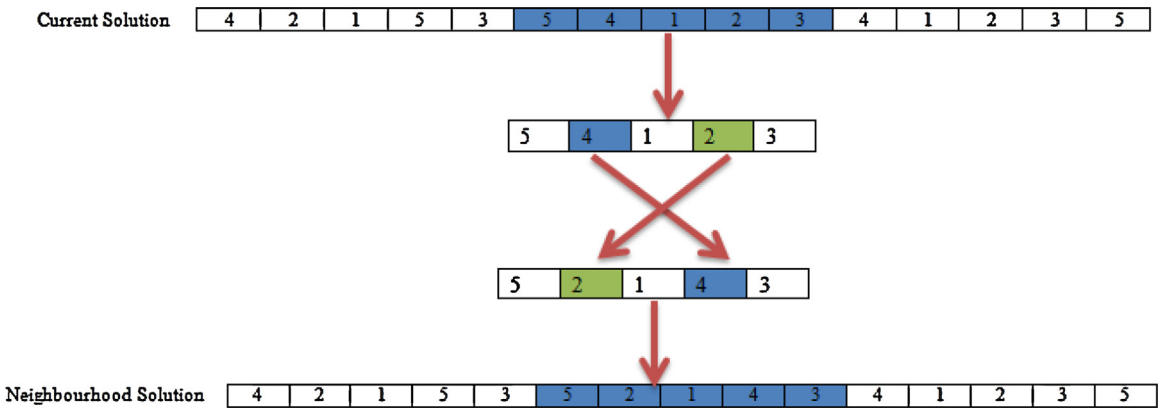

Fig. 5. The example of neighborhood structure.

Table 2
The controlled factors and their levels.

Factors		Levels			
Symbol	Description	Level 1	Level 2	Level 3	Level 4
A	Migration rate of populations	(0.1,0.3,0.6)	(0.3,0.3,0.4)	(0.3,0.1,0.6)	(0.1,0.6,0.3)
B	Percent of cross-mute in pop 1	(0.8,.05)	(0.8,0.09)	(0.9,0.05)	(0.9,.09)
C	Percent of cross-mute in pop 2	(0.8,0.05)	(0.8,0.09)	(0.9,0.05)	(0.9,.09)
D	Percent of cross-mute in pop 3	(0.7,0.1)	(0.8,0.1)	(0.7,0.15)	(0.8,0.15)
E	Initial size of pop 1	50	100	200	300
F	Initial size of pop 2	50	100	200	300
G	Initial size of pop 3	100	200	300	400
H	Cooling rate	0.980	0.995		

Table 3
L32 ($2^1 \times 4^7$)orthogonal array and data for the screen experiment.

Exp no.	Control factors								Witness						*S*/*N* ratio
	A	B	C	D	E	F	G	H	RPD_1	RPD_2	RPD_3	RPD_4	RPD_5	RPD_6	
1	1	1	1	1	1	1	1	1	1.220	2.660	2.220	3.352	4.666	5.312	−10.953
2	1	2	2	2	2	2	2	1	0.980	1.214	1.124	1.548	1.540	1.829	−2.941
3	1	3	3	3	3	3	3	1	1.254	1.140	1.547	1.880	2.011	3.781	−6.555
4	1	4	4	4	4	4	4	1	0.050	1.050	1.814	2.550	3.880	6.901	−10.886
5	2	1	1	2	2	3	3	1	0.880	0.096	2.118	1.115	1.954	3.839	−6.210
6	2	2	2	1	1	4	4	1	0.050	0.087	1.250	1.054	1.771	2.590	−3.197
7	2	3	3	4	4	1	1	1	1.210	1.147	1.871	1.998	2.143	2.590	−5.558
8	2	4	4	3	3	2	2	1	0.874	0.873	1.412	1.572	2.541	4.431	−7.281
9	3	1	2	3	4	1	2	1	1.210	1.540	2.210	1.540	2.552	4.550	−8.051
10	3	2	1	4	3	2	1	1	0.457	1.214	1.697	1.137	2.792	3.989	−6.926
11	3	3	4	1	2	3	4	1	1.350	1.412	2.350	1.574	3.452	3.898	−8.121
12	3	4	3	2	1	4	3	1	0.890	1.211	1.719	1.443	1.872	2.629	−4.701
13	4	1	2	4	3	3	4	1	0.457	0.958	2.545	0.985	3.239	6.390	−9.993
14	4	2	1	3	4	4	3	1	1.540	2.654	2.269	2.214	3.229	5.699	−10.168
15	4	3	4	2	1	1	2	1	1.021	1.241	2.112	3.123	3.210	3.951	−8.524
16	4	4	3	1	2	2	1	1	0.520	1.590	1.256	1.954	2.322	3.687	−6.561
17	1	1	4	1	4	2	3	2	1.360	2.264	1.965	2.689	2.995	6.785	−10.856
18	1	2	3	2	3	1	4	2	0.360	0.951	2.621	1.025	2.365	4.436	−7.562
19	1	3	2	3	2	4	1	2	0.985	1.006	3.145	1.162	2.022	3.745	−7.179
20	1	4	1	4	1	3	2	2	1.320	0.956	2.365	1.521	3.120	4.797	−8.584
21	2	1	4	2	3	4	1	2	0.541	1.006	1.214	2.360	1.625	3.236	−5.535
22	2	2	3	1	4	3	2	2	0.230	0.980	1.250	0.955	1.637	2.396	−2.977
23	2	3	2	4	1	2	3	2	0.845	1.812	2.572	1.913	1.204	3.630	−6.827
24	2	4	1	3	2	1	4	2	0.580	1.220	3.025	1.250	2.102	4.055	−7.456
25	3	1	3	3	1	2	4	2	0.587	0.956	1.562	1.256	2.211	6.025	−8.890
26	3	2	4	4	2	1	3	2	1.210	1.369	2.689	1.220	3.125	3.967	−7.966
27	3	3	1	1	3	4	2	2	1.362	1.214	2.541	1.129	1.259	4.654	−7.572
28	3	4	2	2	4	3	1	2	0.689	0.987	2.039	1.698	1.199	3.057	−5.068
29	4	1	3	4	2	4	2	2	1.123	1.036	2.980	1.652	2.210	3.212	−6.864
30	4	2	4	3	1	3	1	2	1.658	1.063	2.958	1.362	2.112	3.149	−6.822
31	4	3	1	2	4	2	4	2	0.969	1.562	3.012	0.817	3.186	3.358	−7.602
32	4	4	2	1	3	1	3	2	1.320	1.652	2.413	1.295	1.257	5.115	−8.208

Table 4
The ANOVA of the *S*/*N* ratio.

Factors	Description	Degrees of freedom	Sum of squares	Mean square	*F*-ratio	*P*-value	Percent contribution
A	Migration rate of populations	3	33.785	11.262	7.8	0.007	24.440
B	Percent of cross-mute in pop 1	3	22.101	7.367	5.1	0.025	15.987
C	Percent of cross-mute in pop 2	3	28.985	9.662	6.69	0.011	20.967
D	Percent of cross-mute in pop 3	3	18.516	6.172	4.28	0.039	13.394
E	Initial size of pop 1	3	4.357	1.452	1.01	0.434	3.151
F	Initial size of pop 2	3	7.066	2.355	1.63	0.25	5.111
G	Initial size of pop 3	3	10.423	3.474	2.41	0.135	7.540
H	Cooling rate	1	0.013	0.013	0.01	0.926	0.009
Error		9	12.99	1.443			9.397
Sum		31	138.235				1

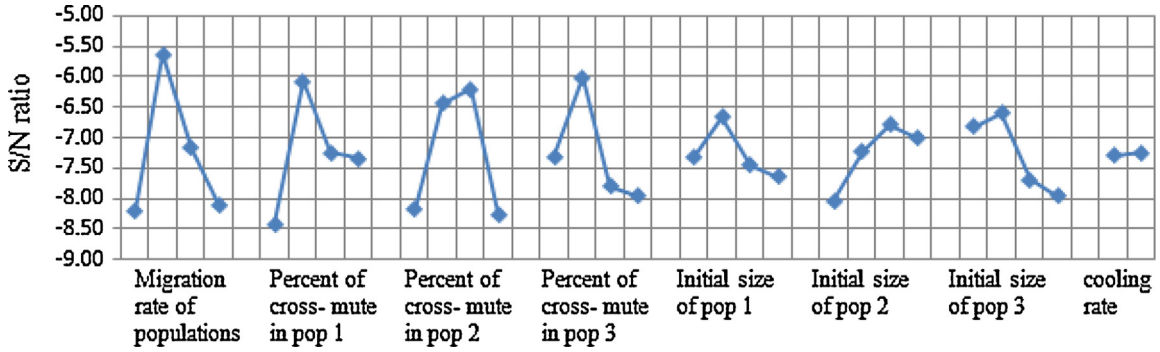

Fig. 6. The mean *S*/*N* ratio plot for each level of the factors.

where η_k is the *S*/*N* ratio for trial *k* and *M* is number of instance (is equal to 6). Table 3 shows the orthogonal array L_{32} and the *RPD* of each instance in each trial and *S*/*N* ratio for each experimental trial. For each level of control factors, the *S*/*N* ratios are averaged and its value is plotted against each control factor in Fig. 6. Since, the aim is to maximize the *S*/*N* ratio, the level with highest *S*/*N* ratio is selected as best level (Table 5).

5.2. Analysis of variance

The experiment was analyzed by means of a multifactor analysis of variance (ANOVA) technique. The purpose of the ANOVA is to determine the relative importance of each single factor in terms of their main effects on the response variable. Since the ANOVA is parametric, three main hypotheses should be checked: normality, homoscedasticity and independence of residuals. The resulting residuals from the experimental data were analyzed and all three hypotheses were valid. Table 4 shows the results of ANOVA. According to the obtained results, the most important parameter is migration rate with 24.4% of total variation. After this parameter, percent of cross-mute in populations are the second important parameters. Cooling rate is the least important parameter with only 0.009% of total variation.

Table 5
The best levels of the factors.

Factors	Description	Best levels
A	Migration rate of populations	(0.3,0.3,0.4)
B	Percent of cross-mute in pop 1	(0.8,.09)
C	Percent of cross-mute in pop 2	(0.9,0.05)
D	Percent of cross-mute in pop 3	(0.8,0.1)
E	Initial size of pop 1	100
F	Initial size of pop 2	200
G	Initial size of pop 3	200
H	Cooling rate	0.995

6. Experimental evaluation

This section experimentally evaluates the performance of the proposed HMPGA. Then, we compare the performance of the algorithm with 11 available algorithms for the similar problem. It should be mentioned that the results of each algorithm are directly extracted from the original papers. The algorithms are

- Simulated annealing algorithms (SAI and SAII) by McKendall et al. [15],
- Hybrid ant systems (HAS) by McKendall and Shang [5],
- Nested loop genetic algorithm (NLGA) by Balakrishnan and Cheng [6],
- Genetic algorithm (CONGA) by Conway and Venkataramanan [10],
- Ant colony optimization (ACO) by Baykasoglu et al. [17],
- Dynamic programming approach (DP) of Erel et al. [14],
- Simulated annealing (SA_EG) by Baykasoglu and Gindy [22],
- Hybrid genetic algorithm (HGA) of Balakrishnan et al. [13],
- Hybrid tabu-simulated annealing (TABUSA) by Sahin and Turkbey [19]
- Heuristic tabu search (HTABU) by McKendall and Liu [20].

In this regard, we use the instances of a well-known benchmark of [6]. In this benchmark, the rearrangement cost in not related to the rearrangement distance and is constant across all periods. Considering three layout sizes of 6, 15, and 30 departments and two different planning horizons of 5 and 10 periods, there are $2 \times 3 = 6$ different size. There exist eight problems for each size. Therefore, there are 48 test problems on which problems 1–8 are related to layouts with 6 facilities in 5 period of time, problems 9–16 are related to layouts with 15 facilities in 5 period of time and so on. The proposed HMPGA algorithm was tested on a Pentium-III at 1400 MHz.

Table 6
Average relative percentage deviation.

T	N	Algorithms											NO.OF
		SAI	SAII	TABUSA	HAS	HTABU	SA_EG	HGA	ACO	NLGA	CONGA	DP	HMPGA	
5	6	0.00	0.00	0.013	0.00	0.00	0.00	0.02	0.02	0.08	0.68	0.01	0.00	8.5×10^4
	15	0.68	0.66	0.63	0.65	0.62	0.13	1.22	5.15	6.39	5.97	1.14	0.09	9.37×10^5
	30	0.74	0.69	0.57	0.81	0.40	1.62	1.12	6.45	8.05	13.05	1.07	0.05	1.92×10^7
10	6	0.00	0.012	0.00	0.00	0.00	0.03	0.09	0.66	0.3547	2.1025	0.02	0.0000	1.4×10^5
	15	0.40	0.29	0.22	0.38	0.27	0.30	0.76	4.04	6.7219	9.2126	0.79	0.0774	6.2×10^6
	30	0.59	0.393	0.39	0.66	0.22	1.48	0.94	7.09	7.0735	19.6919	0.81	0.0415	8.95×10^7
Average		0.403	0.340	0.305	0.417	0.253	0.595	0.690	3.902	4.780	8.453	0.641	0.044	19.3×10^6

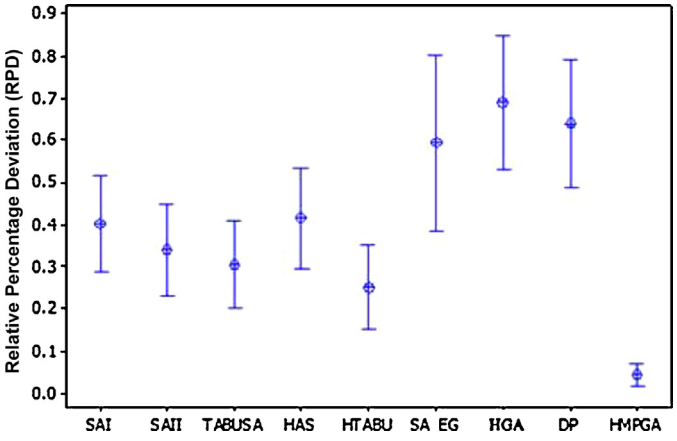

Fig. 7. The average *RPD* and a 0.95% confidence interval.

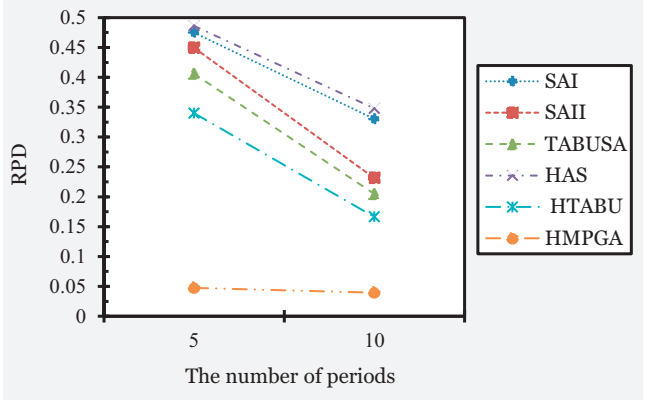

Fig. 8. The average *RPD* of the tested algorithms versus the number of periods.

The performance of algorithms in each problem is measured by using the *RPD* presented in Eq. (10).

$$RPD_{i,j} = \frac{total\ \cos t_{i,j} - LB_i}{LB_i} \quad (10)$$

where $total\ cost_{i,j}$ is the total cost obtained for test problem i by algorithm j and LB_i is the minimum total cost obtained for test problem i. Table 6 shows the average of obtained *RPD*s in terms of problem size. The last column of Table 6 shows the number of objective function (NO.OF) evaluated by HMPGA to find the optimal solution.

As it can be seen in Table 6, the best performing algorithm is the proposed HMPGA with average *RPD* of 0.044. The second best algorithm is HTABU with average *RPD* of 0.253. The 3th and 4th ranks are obtained by TABUSA and SA II with average *RPD* of 0.305 and 0.240, respectively. The worst performing algorithms are CONGA, NLGA and ACO with average *RPD* of more than 3. Since CONGA, NLGA and ACO perform significantly poor, there is no reason to study more on their efficiency.

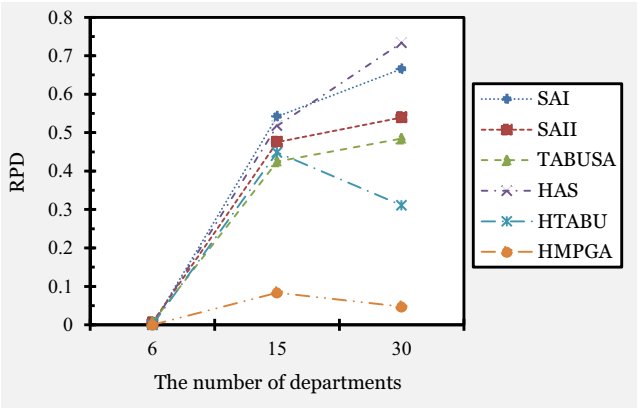

Fig. 9. The average *RPD* of the tested algorithms versus the number of departments.

To study deeper about the performance of the rest algorithms, a 0.95% confidence interval in terms of the *RPD* performance measure in all scale problems is shown in Fig. 7. The results show that HMPGA obtains not only the lowest average *RPD* but also the lowest deviation which means it provides a robust performance.

Table 7
ANOVA results.

Source	Degrees of freedom	Sum of squares	Mean square	*F*-test	*P*-value
Algorithm	5	4.492	0.898	7.06	0
Error	282	35.871	0.127		
Total	287	40.362			

Table 8
Tukey test of pairwise comparison of the algorithms.

Algorithms	Lower	Upper	Significant difference at 99.36%
HMPGA & SAI	0.1515	0.5664	YES
HMPGA & SAII	0.0893	0.5042	YES
HMPGA & TABUSA	0.0542	0.4691	YES
HMPGA & HAS	0.166	0.5809	YES
HMPGA & HTABU	0.002	0.4169	YES
SAI & SAII	−0.2696	0.1453	NO
SAI & TABUSA	−0.3048	0.1102	NO
SAI & HAS	−0.193	0.2219	NO
SAI & HTABU	−0.357	0.0579	NO
SAII & TABUSA	−0.2426	0.1723	NO
SAII & HAS	−0.1308	0.2841	NO
SAII & HTABU	−0.2948	0.1201	NO
TABUSA & HAS	−0.0957	0.3192	NO
TABUSA & HTABU	−0.2597	0.1552	NO
HAS & HTABU	−0.3715	0.0435	NO

Table 9
Number of achievements to best solution.

T	*N*	Algorithms					
		SAI	SAII	TABUSA	HAS	HTABU	HMPGA
5	6	8	8	7	8	8	8
	15	1	0	1	1	1	3
	30	0	0	0	0	2	6
10	6	8	7	8	8	8	8
	15	0	1	2	1	2	3
	30	0	1	0	0	3	4
Sum		17	17	18	18	24	32

In addition, the results show that the difference between the performance of SA_EG, DP, HGA and the rest algorithms is relatively large. So, we do not consider them in the future analysis.

To statistically analyze the results of the rest algorithms, we conduct the analysis of variance test where the algorithm type is the single factor. The results of ANOVA presented in Table 7, shows that the algorithms are significantly different. This motivates using Tukey Test to statistically compare algorithms. Table 8 shows the results. The proposed HMPGA statistically outperforms all the other algorithms. While SA I, SA II, TABUSA, HAS and HTABU are not statistically different.

We also compare the algorithms by the number of times that each one obtains the best solution in the 48 problems. Table 9 shows the results. Interestingly, the proposed HMPGA achieve the best solutions for 32 problems out of 48 ones. While the second ranked algorithm of HTABU obtains 24 best solutions. The algorithms of SA I and SA II yield the lowest number with 17 instances.

To further analyze the results, we study the possible effects of the number of periods and departments on the performance of algorithms. Figs. 8 and 9 show the average *RPD* obtained by the algorithms versus the number of periods and departments. The HMPGA maintains a robust performance in various ranges of periods. As is clear in Fig. 9, in larger instances, the gap of performance of HMPGA becomes more.

To statistically show the effect of problem size (number of facilities and periods) on six algorithms, a two-way ANOVA is applied.

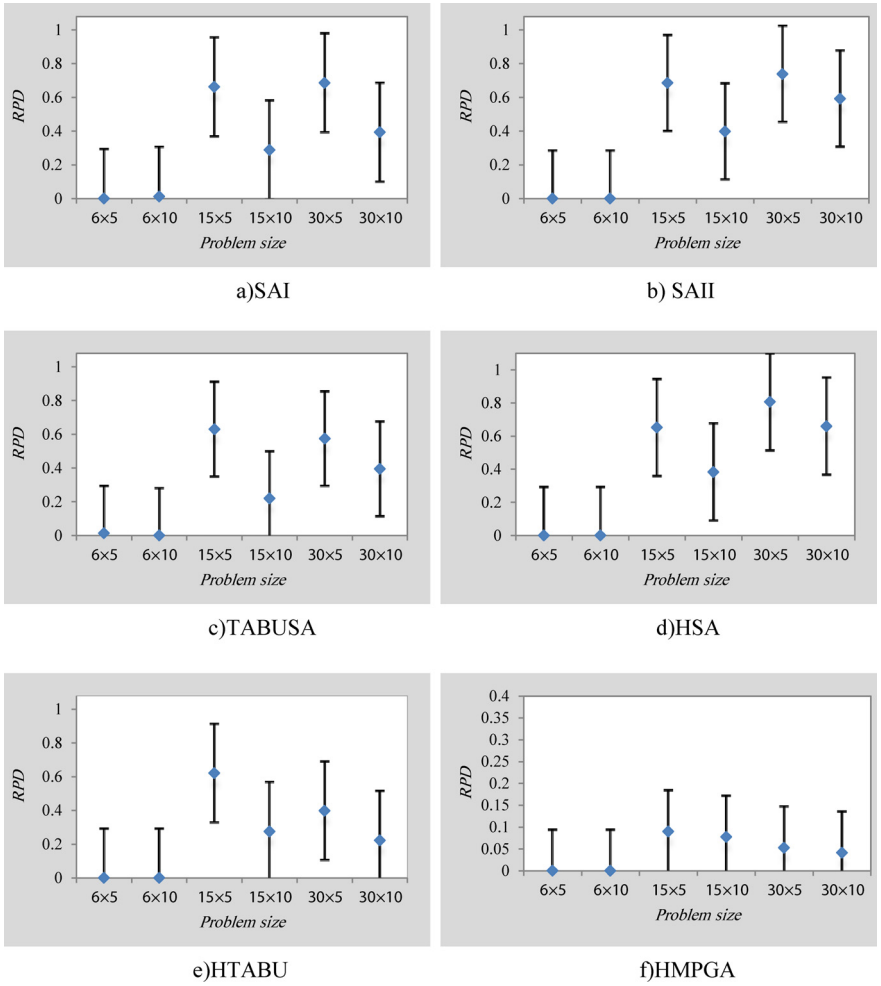

Fig. 10. Means plot of the algorithms vs. the problems size: (a) simulated annealing I, (b) simulated annealing II, (c) hybrid tabu-simulated annealing, (d) hybrid ant systems, (e) heuristic tabu search, and (f) hybrid multi-population genetic algorithm.

Means plot and Fisher's least significant difference (LSD) intervals (at the 95% confidence level) for the interaction among the factors of type of algorithm and problem size are shown in Fig. 10.

As can be seen, all the algorithms perform at the same level for the problem sized 6×6 and 6×10. But, as the problem size is increasing, more fluctuation in *RPD* is observed. These fluctuations are so high that signifies that the algorithms' performance change in different sizes. Anyhow, HMPGA, because of a sophisticated parameter tuning, leads to the highest robustness among the other algorithms such that it can statistically perform similar in different problem sized. In order to give a general perspective to the reader about the algorithms' computational time, they are reported in Appendix B.

7. Conclusion and future work

This study introduced a novel solution approach for DFLP in which a hybrid multi-population genetic algorithm with an effective structure which is used to generate initial populations. It was found that the proposed approach gives promising solution in reasonable CPU time and performs well. It was also discovered that the quality of the solution is largely related to the initial setting of parameters such as cross over and mutation rate, population size and migration rate. For this reason, we accomplish a comprehensive exploration by Taguchi method to find best value of these control parameters. Then we implement this tuned algorithm on the data set used by [6] on which an extensive computational work is performed. The data contains problems for 6, 15, 30 departments for 5 and 10 periods. The results obtained by HMPGA were compared with eleven other algorithms from the literature. The results show proposed method generally act more effectively than presented algorithm in literature. Although the performance of algorithms are being worse as the size of the problems increase; but the HMPGA can eliminate this deterioration better than other algorithms and perform more robustly. In other word, by increasing the size of the problem and search space, efficiency of the algorithm will be more and more revealed.

The following suggestions are given for future studies:

1. Material handling equipment (AGVs) and its maintenance cost can be considered.
2. Considering the rearrangement cost as a separate objective function. Thus a multiple objective cases of dynamic facility layout problem (MODFLP) can be modeled and solved.
3. A new solution representation which is flexible for neighbor generation mechanisms or for the mutation operators and can be devised to improve the effectiveness of this heuristics and other ones in terms of solution quality and computational time requirements.
4. Considering time-value of money wherein re-arrangement costs are not fixed and change during the planning horizon.
5. The implementation of the proposed method on other combinatorial optimization problems.
6. Designing an algorithm which need fewer parameters or has the ability of self-adaptation.

Appendix A.

See Tables 10–15.

Table 10
Computational results for problems with $N = 6$ and $T = 5$.

Problem no.	SAI	SAII	TABUSA	HAS	HTABU	SA_EG	GA	ACO	NLGA	CONGA	DP	HMPGA
P01	106,419	106,419	106,419	106,419	106,419	106,419	106,419	106,419	106,419	108,976	106,419	106,419
P02	104,834	104,834	104,834	104,834	104,834	104,834	104,834	104,834	104,834	105,170	104,834	104,834
P03	104,320	104,320	104,320	104,320	104,320	104,320	104,320	104,320	104,320	104,520	104,320	104,320
P04	106,399	106,399	106,399	106,399	106,399	106,399	106,515	106,509	106,515	106,719	106,509	106,399
P05	105,628	105,628	105,737	105,628	105,628	105,628	105,628	105,628	105,628	105,628	105,628	105,628
P06	103,985	103,985	103,985	103,985	103,985	103,985	104,053	104,053	104,053	105,605	103,985	103,985
P07	106,439	106,439	106,439	106,439	106,439	106,439	106,439	106,439	106,978	106,439	106,447	106,439
P08	103,771	103,771	103,771	103,771	103,771	103,771	103,771	103,771	103,771	104,485	103,771	103,771
Average	105,224	105,224	105,238	105,224	105,224	105,224	105,247	105,247	105,315	105,943	105,239	105,224

Table 11
Computational results for problems with $N = 6$ and $T = 10$.

Problem no.	SAI	SAII	TABUSA	HAS	HTABU	SA_EG	GA	ACO	NLGA	CONGA	DP	HMPGA
P09	214,313	214,313	214,313	214,313	214,313	214,313	214,313	217,251	214,397	218,407	214,313	214,313
P10	212,134	212,134	212,134	212,134	212,134	212,134	212,134	216,055	212,138	215,623	212,134	212,134
P11	207,987	207,987	207,987	207,987	207,987	207,987	207,987	208,185	208,453	211,028	207,987	207,987
P12	212,530	212,741	212,530	212,530	212,530	212,747	212,741	212,951	212,953	217,493	212,741	212,530
P13	210,906	210,906	210,906	210,906	210,906	211,072	210,944	211,076	211,575	215,363	211,022	210,906
P14	209,932	209,932	209,932	209,932	209,932	209,932	210,000	210,277	210,801	215,564	209,932	209,932
P15	214,252	214,252	214,252	214,252	214,252	214,438	215,452	215,504	215,685	220,529	214,252	214,252
P16	212,588	212,588	212,588	212,588	212,588	212,588	212,588	214,621	214,657	216,291	212,588	212,588
Average	211,830	211,857	211,830	211,830	211,830	211,901	212,020	213,240	212,582	216,287	211,871	211,830

Table 12
Computational results for problems with $N = 15$ and $T = 5$.

Problem no	SAI	SAII	TABUSA	HAS	HTABU	SA_EG	GA	ACO	NLGA	CONGA	DP	HMPGA
P17	480,453	480,496	480,453	480,453	480,453	481,378	484,090	501,447	511,854	504,759	482,123	480,547
P18	484,761	484,761	484,761	484,761	484,761	478,816	485,352	506,236	507,694	514,718	485,702	480,141
P19	489,058	488,748	489,058	488,748	488,748	487,886	489,898	512,886	518,461	516,063	491,310	487,878
P20	484,405	484,414	484,446	484,446	484,446	481,628	484,625	504,956	514,242	508,532	486,851	481,905
P21	487,882	487,911	487,822	487,722	487,753	484,177	489,885	509,636	512,834	515,599	491,178	482,122
P22	487,162	487,147	486,493	486,685	486,493	482,321	488,640	508,215	513,763	509,384	489,847	482,647
P23	487,232	486,779	486,268	486,853	486,268	485,384	489,378	508,848	512,722	512,508	489,155	483,235
P24	491,034	490,812	490,551	491,016	490,551	489,072	500,779	512,320	521,116	514,839	493,577	490,533
Average	486,498	486,384	486,232	486,336	486,184	483,833	489,081	508,068	514,086	512,050	488,718	483,626

Table 13
Computational results for problems with $N = 15$ and $T = 10$.

Problem no.	SAI	SAII	TABUSA	HAS	HTABU	SA_EG	GA	ACO	NLGA	CONGA	DP	HMPGA
P25	980,087	979,468	978,848	980,351	980,906	982,298	987,887	1,017,741	1,047,596	1,055,536	983,070	979,087
P26	979,369	978,065	977,338	978,271	978,004	973,179	980,638	1,016,567	1,037,580	1,061,940	983,826	977,469
P27	983,912	982,396	981,172	978,027	982,944	985,364	985,886	1,021,075	1,056,185	1,073,603	988,635	978,521
P28	974,416	972,797	971,720	974,694	971,720	974,994	976,025	1,007,713	1,026,789	1,060,034	976,456	972,487
P29	977,188	978,067	976,781	979,196	977,100	975,498	982,778	1,010,822	1,033,591	1,064,692	982,893	974,108
P30	970,633	967,617	968,362	971,548	967,617	968,323	973,912	1,007,210	1,028,606	1,066,370	974,436	967,854
P31	979,198	979,114	978,660	980,752	978,576	977,410	982,872	1,013,315	1,043,823	1,066,617	982,790	973,217
P32	984,927	983,672	982,888	985,707	983,341	985,041	987,789	1,019,092	1,048,853	1,068,216	988,584	981,971
Average	978,716	977,650	976,971	978,568	977,526	977,763	982,223	1,014,192	1,040,378	1,064,626	982,586	975,589

Table 14
Computational results for problems with $N = 30$ and $T = 5$.

Problem no.	SAI	SAII	TABUSA	HAS	HTABU	SA_EG	GA	ACO	NLGA	CONGA	DP	HMPGA
P33	576,039	576,741	574,624	576,886	574,577	583,081	578,689	604,408	611,794	632,737	579,741	575,247
P34	568,316	568,095	568,256	570,349	567,481	573,965	572,232	604,370	611,873	647,585	570,906	565,632
P35	573,739	574,036	572,865	576,053	571,462	577,787	578,527	603,867	611,664	642,295	577,402	573,214
P36	567,911	566,248	566,231	566,777	564,868	572,139	572,057	596,901	611,766	634,626	569,596	564,529
P37	559,277	558,460	557,356	558,353	555,628	563,503	559,777	591,988	604,564	639,693	561,078	554,143
P38	566,077	566,597	566,599	566,792	565,100	570,905	566,792	599,862	606,010	637,620	567,154	562,251
P39	567,131	568,204	567,628	567,131	566,993	571,499	567,873	600,670	607,134	640,482	568,196	561,257
P40	576,014	573,755	573,487	575,280	573,023	581,614	575,720	610,474	620,183	635,776	575,273	567,325
Average	569,313	569,017	568,381	569,703	567,392	574,312	571,458	601,568	610,624	638,852	571,168	565,450

Table 15
Computational results for problems with $N = 30$ and $T = 10$.

Problem no	SAI	SAII	TABUSA	HAS	HTABU	SA_EG	GA	ACO	NLGA	CONGA	DP	HMPGA
P41	1,164,359	1,163,222	1,161,751	1,166,164	1,159,589	1,174,815	1,169,474	1,223,124	1,228,411	1,362,513	1,171,178	1,160,354
P42	1,162,665	1,161,521	1,160,656	1,168,878	1,157,942	1,173,015	1,168,878	1,231,151	1,231,987	1,379,640	1,169,138	1,158,872
P43	1,157,693	1,156,918	1,155,406	1,166,366	1,154,799	1,166,295	1,166,366	1,230,520	1,231,829	1,365,024	1,165,525	1,151,251
P44	1,149,048	1,145,918	1,144,821	1,148,202	1,143,110	1,154,196	1,154,192	1,200,613	1,227,413	1,367,130	1,152,684	1,141,254
P45	1,126,432	1,127,136	1,125,968	1,128,855	1,123,446	1,140,116	1,133,561	1,210,892	1,215,256	1,356,860	1,128,136	1,122,352
P46	1,145,445	1,145,146	1,143,480	1,141,344	1,141,144	1,158,227	1,145,000	1,239,255	1,221,356	1,372,513	1,143,824	1,132,541
P47	1,148,083	1,140,744	1,145,830	1,140,773	1,145,951	1,157,505	1,145,927	1,248,309	1,212,273	1,382,799	1,142,494	1,141,413
P48	1,166,672	1,161,437	1,164,322	1,166,157	1,160,484	1,177,565	1,168,657	1,231,408	1,245,423	1,383,610	1,167,163	1,161,958
Average	1,152,550	1,150,255	1,150,279	1,153,342	1,148,308	1,162,717	1,156,507	1,226,909	1,226,744	1,371,261	1,155,018	1,146,249

Appendix B. Computational time and systems of the algorithms

See Table A1.

Table A1
Computational time and systems of the algorithms (in second).

T	N	Algorithms					
		SAI	SAII	TABUSA	HAS	HTABU	HMPGA
		Pentium IV 2.4 GHz PCspan		Pentium D 3.4 GHz PC	Pentium IV 2.4 GHz PC	AMD Athlon 2600+ 1.92 GHz PC	Pentium-III 1.4 GHz
5	6	10.2	10.2	1.8	30	1	3
	15	123	123	244	360	10	58
	30	264	264	5471	1500	180	574
10	6	19.8	19.8	7.2	120	1	9.8
	15	237	237	1504	900	35	204
	30	468	468	26829	2700	720	2198

References

[1] J.A. Tompkins, J.A. White, Y.A. Bozer, E.H. Frazelle, J.M.A. Tanchoco, J. Trevino, Facilities Planning, Wiley, New York, 1996.
[2] R.D. Meller, K.Y. Gau, The facility layout problem: recent and emerging trends and perspectives, J. Manuf. Syst. 15 (5) (1996) 351–366.
[3] R.L. Francis, L.F. McGinnis, J.A. White, Facility Layout and Location: An Analytical Approach, Prentice Hall, New Jersey, 1992.
[4] P. Afentakis, R. Millen, M.M. Solomon, Dynamic layout strategies for flexible manufacturing systems, Int. J. Prod. Res. 28 (1990) 311–323.
[5] A.R. McKendall, J. Shang, Hybrid ant systems for the dynamic facility layout problem, Comput. Oper. Res. 33 (3) (2006) 790–803.
[6] J. Balakrishnan, C.H. Cheng, Genetic search and the dynamic layout problem, Comput. Oper. Res. 27 (6) (2000) 587–593.
[7] M.J. Rosenblatt, The dynamics of plant layout, Manag. Sci. 32 (1) (1986) 76–86.
[8] T.L. Urban, A heuristic for the dynamic facility layout problem, IIE Trans. 25 (4) (1993) 57–63.
[9] T.A. Lacksonen, E.E. Enscore, Quadratic assignment algorithms for the dynamic layout problem, Int. J. Prod. Res. 31 (3) (1993) 503–517.
[10] D.G. Conway, M.A. Venkataramanan, Genetic search and the dynamic facility layout problem, Comput. Oper. Res. 21 (8) (1994) 955–960.
[11] B.K. Kaku, J.B. Mazzola, A tabu-search heuristic for the dynamic plant layout problem, INFORMS J. Comput. 9 (4) (1997) 374–384.
[12] J. Balakrishnan, C.H. Cheng, D.G. Conway, An improved pair-wise exchange heuristic for the dynamic plant layout problem, Int. J. Prod. Res. 38 (13) (2000) 3067–3077.
[13] J. Balakrishnan, C.H. Cheng, D.G. Conway, C.M. Lau, A hybrid genetic algorithm for the dynamic plant layout problem, Int. J. Prod. Econ. 86 (2003) 107–120.
[14] E. Erel, J.B. Ghosh, J.T. Simon, New heuristic for the dynamic layout problem, J. Oper. Res. Soc. 54 (2003) 1275–1282.
[15] A.R. McKendall, J. Shang, S. Kuppusamy, Simulated annealing heuristics for the dynamic facility layout problem, Comput. Oper. Res. 33 (8) (2006) 2431–2444.
[16] J.M. Rodriguez, F.C. MacPhee, D.J. Bonham, V.C. Bhavsar, Solving the dynamic plant layout problem using a new hybrid meta-heuristic algorithm, Int. J. High Perform. Comput. Netw. 4 (5/6) (2006) 286–294.
[17] A. Baykasoglu, T. Dereli, I. Sabuncu, An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems, Omega 34 (2006) 385–396.
[18] J. Balakrishnan, C.H. Cheng, The dynamic plant layout problem: Incorporating rolling horizons and forecast uncertainty, Omega 37 (2009) 165–177.
[19] R. Sahin, O. Turkbey, A new hybrid tabu-simulated annealing heuristic for the dynamic facility layout problem, Int. J. Prod. Res. 47 (24) (2009) 6855–6873.
[20] A.R. McKendall, W.H. Liu, New Tabu search heuristics for the dynamic facility layout problem, Int. J. Prod. Res. 50 (3) (2011) 867–878.
[21] W. Chiang, P. Kouvelis, An improved tabu search heuristic for solving facility layout design problems, Int. J. Prod. Res. 34 (1996) 2565–2585.
[22] A. Baykasoglu, N.N.Z. Gindy, A simulated annealing algorithm for dynamic layout problem, Comput. Oper. Res. 28 (2001) 1403–1426.
[23] J. Balakrishnan, C.H. Cheng, Dynamic layout algorithms: a state-of-the-art survey, Omega 26 (4) (1998) 507–521.
[24] T. Dunker, G. Radons, E. Westkamper, Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem, Eur. J. Oper. Res. 165 (1) (2005) 55–69.
[25] A.R. McKendall, A. Hakobyan, Heuristics for the dynamic facility layout problem with unequal-area departments, Eur. J. Oper. Res. 201 (2010) 171–182.
[26] C.F.M. Toledo, R.R.R. Oliveira, P.M. França, A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging, Comput. Oper. Res. 40 (4) (2013) 910–919.
[27] C.F.M. Toledo, M.S. Arantes, R.R.R. Oliveira, B. Almada-Lobo, Glass container production scheduling through hybrid multi-population based evolutionary algorithm, Appl. Soft Comput. 13 (2013) 1352–1364.
[28] P.C. Chang, S.H. Chen, The development of a sub-population genetic algorithm II (SPGA II) for multi-objective combinatorial problems, Appl. Soft Comput. 9 (2009) 173–181.
[29] D.T. Pham, D. Karaboga, Intelligent Optimisation Techniques. Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, SpringerVerlag, London, 2000, pp. 302.
[30] A.S. Drud, CONOPT – a large-scale GRG code, ORSA J. Comput. 6 (2) (1992) 207–216.
[31] J.J. Grefenstette, Incorporating problem specific knowledge into genetic algorithms, in: L. Davis (Ed.), Genetic Algorithms and Simulated Annealing, 1987, pp. 42–60.
[32] C.L. Huntley, D.E. Brown, Parallel genetic algorithms with local search, Comput. Oper. Res. 18 (3) (1991) 275–289.
[33] S. Kirkpatrick, C.D.J.R. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[34] B. Naderi, M. Khalili, R. Tavakkoli-Moghaddam, A hybrid artificial immune algorithm for a realistic variant of job shops to minimize the total completion time, Comput. Ind. Eng. 56 (4) (2009) 1494–1501.
[35] D.C. Montgomery, Design and Analysis of Experiments, 5th ed., Wiley, New York, 2000.
[36] S.M. Phadke, Quality Engineering Using Robust Design, Prentice Hall, Englewood Cliffs, NJ, 1989.
[37] C.M. Hsu, Improving the lighting performance of a 3535 packaged hi-power LED using genetic programming, quality loss functions and particle swarm optimization, Appl. Soft Comput. 12 (2012) 2933–2947.
[38] C.Y. Tanga, Y.L. Wub, C.C. Peng, Fundamental matrix estimation by multiobjective genetic algorithm with Taguchi's method, Appl. Soft Comput. 12 (2012) 553–558.