RAIRO-Oper. Res. 55 (2021) S2433–S2453
https://doi.org/10.1051/ro/2020127

RAIRO Operations Research
www.rairo-ro.org

A NEW SOFT COMPUTING ALGORITHM BASED ON CLOUD THEORY FOR DYNAMIC FACILITY LAYOUT PROBLEM

Seyed Shamsodin Hosseini[1], Parham Azimi[1,*], Mani Sharifi[1] and Mostafa Zandieh[2]

Abstract. This paper deals with dynamic facility layout problem (DFLP) in a plant which is concerned with determining the best position of machines in the plant during a multi-period planning horizon. The material handling costs and machines rearrangement costs (MRC) are used to determine the best layout. In addition to the positions of machines, the details of transportation such as type of transporters and sequence of transportation operations have a direct effect on material handling costs (MHC). Therefore, it is more realistic to consider the transportation details during DFLP optimization. This paper proposes a new mathematical model to simultaneously determine the best position of machines in each period and to plan the transportation operations. Minimizing sum of MHC and MRC is considered as the objective function. A new hybrid meta-heuristic approach has been developed by combining modified genetic algorithm and cloud-based simulated annealing algorithm to solve the model. Finally, the proposed methodology is compared with two meta-heuristics on a set of test problems.

Mathematics Subject Classification. 68U20, 90-08, 91B70.

Received October 26, 2018. Accepted November 12, 2020.

1. Introduction

Facility layout problem (FLP) is determination of the most efficient arrangement of facilities in plant, office buildings or service facilities. The facility can be defined as machine, manufacturing cell or administrative office. An efficient layout causes a well-organized material handling between facilities, contributes to the overall efficiency of operations, and can save operational costs such as inventory holding costs, production costs, etc. According to Tompkins *et al.* [36], in a manufacturing system, the material handling causes between 20% and 50% of the total operating costs, and it is generally agreed that effective facility planning can reduce these costs by at least 10%–30%. Therefore, the FLP is a strategic decision having vital importance for manufacturing facilities. The material handling cost (MHC) is one of the most significant measures to determine the efficiency of a layout. It is defined as the product of the flow of materials between machines and the distance between the locations of the machines. If the flow of materials doesn't change during the planning horizon time, the static

Keywords. Dynamic facility layout problem, transporters, modified genetic algorithm, cloud-based simulated annealing algorithm.

[1] Department of Industrial Engineering, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran.

[2] Department of Industrial Management, Management and Accounting Faculty, Shahid Beheshti University, GC, Tehran, Iran.
*Corresponding author: p.azimi.qiau@gmail.com

facility layout problem (SFLP) can be considered and, it has been intensively studied in the literature. For a recently review of the SFLP, see Drira *et al.* [12].

Nowadays, the flow of the materials may change very fast during the planning horizon. Changes in the material flows may be the results of many factors, such as the introduction of new products, the change in operation sequencing, the change in the design of an available product, the stopping of products from production line, the shortening of products life cycle, the change in product strategies, and the change in the amount of production, etc. These changes affect the efficiency of current layout during the planning horizon and necessitate considering re-layout in the system. The problem that considers these changes in the flow of the materials between the machines during the planning horizon is known as the dynamic facility layout problem (DFLP). The DFLP contains a multi-period planning horizon. A layout design for the DFLP is a series of layouts and each layout is related with a period. Therefore, the objective function in a DFLP is defined as the minimization of MHC in all periods plus the rearrangement costs of machines. The rearrangement costs are imposed whenever the position of a machine in consecutive time periods is changed. Hence, the DFLP is responsible to establish the trade-off between the MHC and machine rearrangement costs (MRC).

In addition to the MHC and MRC, the time needed to transport the materials between machines is an important factor in lots of manufacturing system. It directly impacts on the production time. Therefore, to have a more realistic DFLP model, transportation time must be considered. To calculate the transportation time, the details of transportation must be considered. These details consist of type of the transporters and sequence of transportation operation. The proposed mathematical model in this paper take all these details into account. The sum of MHC and MRC is considered as the objective function and transportation time is inserted the model as a constraint. Since, the model is an extended version of general DFLP, it belongs to the NP Hard class of problems. We propose a new hybrid meta-heuristic algorithm that can solve the model efficiently.

The rest of the paper is structured as follows. Section 2 presents the related works. Section 3 provides a definition of the problem along with the notations and assumptions and mathematical model. Section 4 explains the optimization algorithm and its implementation on the problem. In Section 5 the efficiency of the proposed algorithm is evaluate against CPLEX and two meta heuristic algorithms. Finally, Section 6 concludes the paper and provides some suggestions for future studies.

2. Literature review

Rosenblatt [31] presented the first study on the DFLP. It is shown that the number of layouts to be evaluated to guarantee optimality for a DFLP with N machines and T periods is $(N!)^T$. Therefore, checking all possible solutions is computationally intractable for real-life problems. Based on dynamic programming, Rosenblatt [31] developed two heuristics each of which considers a set of limited good layouts for a single period. Since, the problem is inherently computationally intractable, several heuristics solution approach have been developed so far. A steepest-descent heuristic based on a pair-wise-exchange idea was developed by Urban [38]. Lacksonen and Enscore [21] presented and compared five heuristics to solve the DFLP. These heuristics used the principles of dynamic programming, a branch and bound algorithm, a cutting plane algorithm, cutting tree algorithm, and computerized relative allocation of facilities technique (CRAFT).

Many heuristic algorithms for the DFLP have been reported in the literature. Kaku and Mazzola [18] used a tabu search (TS) heuristic. This TS heuristic is a two-stage search process that incorporates diversification and intensification strategies. Balakrishnan and Cheng [3] developed a genetic algorithm (GA) to solve the DFLP. Baykasoglu and Gindy [7] proposed a simulated annealing (SA) heuristic for the DFLP. Balakrishnan *et al.* [6] presented a hybrid GA for DFLP. Erel *et al.* [13] proposed a new heuristic to solve the DFLP suggesting the shortest path for solving the DFLP. McKendall and Shang [25] developed three hybrid ant systems. McKendall *et al.* [26] presented two SA heuristics. While the first, (SAI), is a formally SA for the DFLP, the second, (SAII), is the SAI added look-ahead/look-back strategy. Rodriguez *et al.* [30] presented a hybrid heuristic algorithm which incorporates GA and tabu search. Krishnan *et al.* [20] developed a new tool, the Dynamic From–Between Chart, for an analysis of rearrangement departments. This tool models the changes in the production rates using

a continuous function. Balakrishnan and Cheng [4] investigated the performance of algorithms under fixed and rolling horizons, different shifting costs and flow variability, and forecast uncertainty. Şahin and Türkbey [32] presented a new hybrid heuristic algorithm based on the simulated annealing added with tabu list. Ulutaş and Islier [37] proposed a clonal selection algorithm to solve the DFLP. Rezazadeh *et al.* [29] presented the discrete particle swarm optimization algorithm for the DFLP. McKendall and Liu [24] presented three TS heuristics for the DFLP. The first heuristic is a simple TS heuristic. The second heuristic adds diversification and intensification strategies to the first heuristic, and the third heuristic is a probabilistic TS heuristic. Kaveh *et al.* [19] studied the dynamic facilities layout problem with ambiguity in information flow. They considered the material flow as fuzzy numbers with different membership functions and modeled the problem in fuzzy programming. Two intelligent algorithms of SA and hybrid GA were proposed to solve the model. The numerical results showed that SA algorithm solve these type problems better than hybrid genetic algorithm (HGA). Yao *et al.* [41] describes a connection layout model based on the total transfer efficiency of all the passengers. They emphasized on the connection characters between various traffic modes and subway stations. They established connection facility layout model of subway stations with the aim of improving the transfer efficiency. Meaningful results were obtained from the connection layout model of subway stations. Hosseini *et al.* [16] proposed a robust approach based on integrating three meta-heuristics: imperialist competitive algorithms, variable neighborhood search, and simulated annealing, to efficiently solve the DFLP in manufacturing systems. Qudeiri *et al.* [28] studied layout design optimization of dynamic environment flexible manufacturing systems. They presented a two-stage approach to investigate the best locations of the machine in flexible manufacturing system. In the first stage, the throughput of randomly selected locations of the machine is computed by proposing a production simulation system. In the second stage, the generated locations are fed into artificial neural network to find a relation between a machine's location and the throughput. From the experimental results showed that their hybrid algorithm can exhibits very good performance in not only solution quality but also in terms of the robustness and computational time. For an extensive review on the DFLP, the readers are referred to the work of Moslemipour *et al.* [27].

Hosseini and Seifbarghy [15] presented a new multi-objective model for DFLP. They considered the selection of the best material handling equipment between each pair of machines. In their model, machine rearrangement costs and MHC were considered as different objectives. A new metaheuristic algorithm, called multi-objective water flow like algorithm, to optimize multi-objective dynamic facility layout problem. Computational results indicate the high efficiency of the proposed algorithm with respect to the other algorithm in solving the proposed model. Horta *et al.* [14] proposed a mathematical programming approach to optimize layout of a fresh products' warehouse. Products are transported from the picked up point at the receiving dock to the store point. Their problem considered the different routes during the picking and delivering the products. Therefore, their model incorporated both layout and routing problem. Hu [17] proposed a high-fidelity simulation method for the evaluation of passenger flow organization and facility layout at metro stations. Passenger flow movements were modeled by simulation. The results showed that this method is capable to evaluate passenger flow organization through visual and quantitative analyses for improving facility layout design. Liu *et al.* [22] proposed an improved wang-landau (WL) sampling algorithm to solve the DFLP. They combined the WL with some heuristic strategies. Four groups of cases were used to evaluate the proposed algorithm. The computational results showed that the proposed algorithm is effective in solving the DFLP. A novel modified backtracking search algorithm (MBSA) was proposed by Vitayasak *et al.* [39] to solve the stochastic DFLP. They investigated three MBSA against the classical backtracking search algorithm (BSA)and a GA by using 11 test problem available in the literature. The computational results showed that the best MBSA was able to find better solutions with respect to the GA. On the other hand, the computational time required by the three MBSAs was 70% less than the GA. For an extensive review on the DFLP, the readers are referred to the work of Moslemipour *et al.* [27].

None of the studies mentioned above considers the transportation time. In this study, a new mathematical model to simultaneously determine the layout of machines and planning the transporters. The model is an extended version of DFLP and so is NP Hard. Consequently, developing efficient solution approach is expected.

This paper propose a new hybrid meta heuristic approach by combining modified genetic algorithm (MGA) and cloud based simulated annealing algorithm (CBSA) to solve the model.

3. Problem description

As mentioned above, the problem under study is determining the position of machine and planning the transportation operation in a manufacturing system during a multi-period planning horizon. Consider that the manufacturing system produces several products. There are some predefined locations in the shop floor each of which must be assigned by a machine. The jobs arrive to the system and pass among the machines to be processed. Jobs are transported between machines that exist in their process route. Therefore, each job has a number of transportation operations. For example consider the process route of job 1 as 1→3→5→6→7→9. The first transportation operations of the job 1 is shown by $O_{1,1}$. The concept of transportation operations 1 is to take job 1 from machine 1 and move it to machine 3. Based in this explanation, the job number 1 contains 5 transportation operations each of which must be performed by the same or different transporters. Transportation is done by transporters such as AGV, manual trucks and lift tracks. To transfer jobs between each pair of machines, one of the available transporters is selected. In response to a request, the transporter perform an empty travel from the machine of its last delivery towards the machine of the current request; then, the transporter perform a full travel to the machine destination of the transfer request. Consequently, it is very important how to assign the transportation operations to the transporters. On the other hand the layout of machines has a direct impact on the transportation cost and time. An efficient layout will lead to a decrement in transportation costs. Therefore, decisions related to transportation operations must be made considering machine layout. To this aim this section present an integrated mathematical model. The developed model is able to simultaneously make suitable decisions to locate the machines and assign transportation operations to transport the jobs between machines. Next subsections presents the indices, parameters, decision variables and assumptions used in the model.

3.1. Notations

Indexes

i, j	Index of jobs.
k, l	Index of operations.
m, n	Index of machines.
p, q	Index of locations.
t	Index of time periods.

Parameters

$F_{m,n,t}$	The material flow between machine m and n in period t.
$D_{p,q}$	The distance between location p and q.
$a_{i,k,m}$	1 If $O_{i,k}$ is done by machine m 0 otherwise.
$re_{m,p,q,t}$	The cost of shifting machine m from location p to q in period t.
Max_allow_t	Maximum allowable time to perform transportation operation in period t.

Independent decision variables

$X_{i,k,j,l,tr,t}$	1 If $O_{i,k}$ is done after $O_{j,l}$ by transporter tr in period t 0 otherwise.
$Y_{m,p,t}$	1 if the machine m is assigned to the location p in period t 0 otherwise.
$Z_{i,k,tr,t}$	1 if transportation operation k of job i is performed by transporter tr in period t.

Dependent decision variables

$t_{m,n,t}$	The time interval between the machine m and n in period t.
$C_{i,k,t}$	Completion time of transportation operation k of job i in period t.
$P_{i,k,tr,t}$	The time needed to perform transportation operation k of job i by transporter tr in period t.

3.2. Assumptions

- The material flow between facilities is forecasted for each period and will be fixed during that period.
- The time needed to perform transportation operation is limited.
- A machine can process only one part at a time.
- A transporter can perform only one transportation operation at a time
- The sizes of the facilities and locations are equal.
- The distances between locations are known in advance.
- Facilities are not allowed to be re-located at the end of each period.

3.3. Mathematical model

According to the problem description and the assumption, a new mathematical model is developed. The model belongs to the integer nonlinear models in which the objective function is minimizing sum of MHC and MRC. The problem can be formulated as follows:

$$\text{OF} = \text{MIN} \sum_{\forall t} \sum_{\forall m} \sum_{\forall n} \sum_{\forall p} \sum_{\forall q} D_{p,q} F_{m,n,t} Y_{m,p,t} Y_{n,q,t} + \sum_{\forall t>1} \sum_{\forall m} \sum_{\forall n} \sum_{\forall p} \sum_{\forall q} re_{m,p,q,t} Y_{m,p,t} Y_{m,q,t-1} \tag{3.1}$$

subject to:

$$\sum_{\forall m} Y_{m,p,t} = 1 \qquad \forall p,t \tag{3.2}$$

$$\sum_{\forall p} Y_{m,p,t} = 1 \qquad \forall m,t \tag{3.3}$$

$$\sum_{\forall tr} Z_{i,k,tr,t} = 1 \qquad \forall i,k,t \tag{3.4}$$

$$C_{ik,t} - P_{i,k,tr,t} \geq C_{j,l,t} - M \cdot (1 - X_{i,k,j,l,tr,t}) \qquad \forall i,j,k,l,t,tr \tag{3.5}$$

$$C_{j,l,t} - P_{j,l,tr,t} \geq C_{i,k,t} - M \cdot X_{i,k,j,l,tr,t} \qquad \forall i,j,k,l,t,tr \tag{3.6}$$

$$M\,(1 - X_{i,k,j,l,tr,t}) \geq 2 - Z_{i,k,tr,t} - Z_{j,l,tr,t} \qquad \forall i,j,k,l,t,tr \tag{3.7}$$

$$P_{i,k,tr,t} \geq (t_{m,n,t} a_{i,k,m} a_{i,k+1,n}) \qquad \forall i,k,m,n,t,tr \tag{3.8}$$

$$t_{m,n,t} \geq D_{p,q} Y_{m,p,t} Y_{n,q,t} \qquad \forall m,n,p,q,t \tag{3.9}$$

$$C_{i,k,t} \geq Z_{i,k,tr,t} P_{i,k,tr,t} \qquad \forall i,k,t,tr \tag{3.10}$$

$$C\_max_t \geq C_{i,k,t} \qquad \forall i,k,t \tag{3.11}$$

$$C\_max_t \leq Max\_allow_t \qquad \forall t \tag{3.12}$$

$$X_{i,k,j,l,tr,t} \quad \in \{0,1\} \qquad \forall i,k,j,l,t,tr \tag{3.13}$$

$$Y_{m,p,t} \in \{0,1\} \qquad \forall m,p,t \tag{3.14}$$

$$Z_{i,k,tr,t} \in \{0,1\} \qquad \forall i,k,t,tr. \tag{3.15}$$

The objective function (3.1) is used to minimize the sum of the material handling and rearrangement costs. Constraint set (3.2) requires every machine to be assigned to a location in each period and (3.3) ensures that exactly one machine is assigned to each location within each period. Constraint set (3.4) guarantee that each a transportation operation of each job is performed by a transporter. Constraint set (3.5) and Constraint set (3.6) implies that a transporter is not allowed to start a transportation operation until it has completed the

previous transportation operation. Constraint set (3.7) implies control the relation between variables. Constraint set (3.8) calculates the time needed to perform a transportation operation. Constraint set (3.9) calculates the time distance between two machines. Constraint set (3.10) control the completion time of each transportation operation. Constraint set (3.11) and (3.12) control maximum allowable time to perform transportation operation in each period. Constraint set (3.13)–(3.15) control the value of decision variables.

4. Solution methodology

Because of the hardness of optimally solving the model with exact methods in a reasonable time, we should use meta-heuristic algorithms to solve it. While the genetic algorithms have better performances in terms of objective function and CPU time, the simulated annealing algorithm has some advantages such as ease of implementation and the ability to provide reasonably good solution for many combinatorial problems. Therefore, we use a hybrid algorithm combining the advantages of GA and SA to solve the model. In the proposed algorithm the GA has been enhanced by a heuristic mutation and the SA has been enhanced by a heuristic temperature decrement.

4.1. Genetic algorithm with modified mutation

4.1.1. Genetic algorithm

Genetic algorithm (GA) is a stochastic evolutionary algorithm that is based on analogies with the principles of natural biological systems. GA is especially suitable for solving complex optimization problems. In general, GA consists of simultaneously evaluating multiple regions of the solution space during each iteration. It studies a collection of solutions called population and so performs a multidirectional search. The population is usually generated randomly; but also it can be generated in a heuristic manner. After generating the initial population (initial solutions), a stochastic search process is employed to iteratively enhance the quality of the solutions. Genetic algorithms employ standard operators to transition a population. Two of these main operators are mutation and crossover, discussed in the subsequent sections.

4.1.2. Modified genetic algorithm

In GA, cross over operator results in convergence of the algorithm. In order to prevent quick convergence, the mutation operator is applied. Therefore, it is more intelligent to apply the mutation only when the chromosomes are going to be very similar. In the proposed modified GA (MGA) the mutation operator is applied only when the similarity rate in the population exceeds a predefined value. Consequently, mutation operator is applied only in some generations and this leads to avoid computational efforts.

4.2. The simulated annealing algorithm based on cloud theory

4.2.1. Simulated annealing algorithm

Generally speaking, simulated annealing algorithm (SA) is a stochastic local search procedure to find the near optimal solution in combinatorial optimisation problems [34]. The SA, though by itself is a local search algorithm, avoids getting trapped in a local minimum. The main idea of SA search process to prevent getting trapped at a poor local optimum is to accept not only better solutions but the worse neighbour solutions with a certain probability. The probability of accepting a worse move is a function of both the temperature of the system and the change in the cost function. It can be appreciated that as the temperature of the system decreases the probability of accepting a worse move is decreased. The method can avoid being stuck at a local optimum, and increases the probability of finding a global optimal solution. Consequently, it initially performs a wide investigation of the solution space and then restricts the solution space gradually. Thus, the algorithm converges to the best optimal solution.

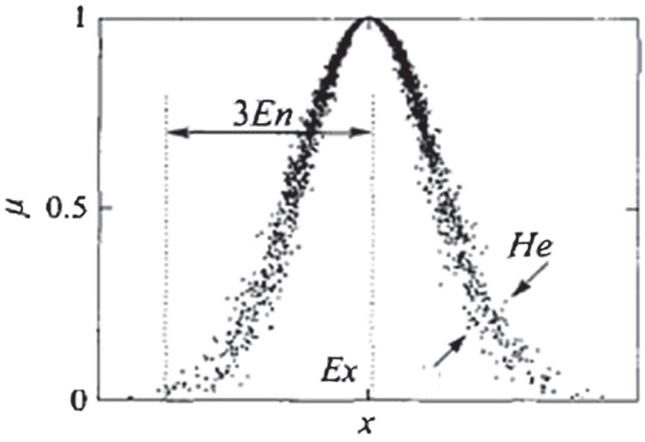

FIGURE 1. Three digital characteristics of a normal cloud [23].

4.2.2. Basic concepts of cloud theory

Cloud theory is a new theory handling uncertainty based on the uncertain transition between quality and quantity. The cloud model was presented by Deyi *et al.* [10] basing on the traditional fuzzy sets theory and probability statistics. It provides a model that facilitates fuzziness and randomness of concepts and transforms the concepts to a uniform presentation in a numerical domain [11]. The basic concepts of cloud theory can be sketched as follows:

Suppose that L is the language value of domain v, and mapping: $M_D(x) : v \to [0,1], \forall x \in v, x \to M_D(x)$, if the distribution of M_D is normal, it is named a normal cloud model. This map produces a group of random numbers, with stable tendency that is distinguished by expectation Ex, entropy En and Hyper entropy, He. These three parameters reflect the quantitative features of the concept M_D, Deyi and Yi [9]. Ex determines the centre of the cloud and En determines the range of the cloud. Figure 1, displays that about 99.74% of the total cloud drops distribute between $Ex - 3En$ and $Ex + 3En$. Hyper entropy determines the degree of cloud drops dispersive, the larger the He is the more dispersive the cloud drops locate. If the three digital characteristics (Ex, En, He) and certain v_0 are given, a normal distribution of Y condition cloud generator (NYCCG) can create a drop of cloud $(drop(x_i, u_0))$, as follows Deyi and Yi [9]:

$$
\begin{aligned}
&\text{INPUT} : \{Ex, En, He\}, n, v_0 \\
&\text{OUTPUT} : \{(x_1, v_0), \ldots, (x_n, v_0)\} \\
&\text{FOR } i - 1 \text{ to } n \\
&En' = randn(En, He) \\
&x_i = Ex \pm En' \sqrt{-2\ln(v_0)} \\
&drop(x_i, v_0)
\end{aligned}
$$

where $randn\ (En, He)$ produces a random number with normal distribution whose expectation is En and standard deviation is He.

4.2.3. Cloud-based simulated annealing algorithm

The cloud-based simulated annealing algorithm (CBSA) follows the Metropolis rule. Because the normal cloud model has the characteristics of randomness and stable tendency [11], cloud theory and Y condition normal cloud are used to generate a continuous annealing temperature. Elasticity is the main feature that allows us to dynamically scale down annealing temperature to avoid problems like being trapped in the local minimum.

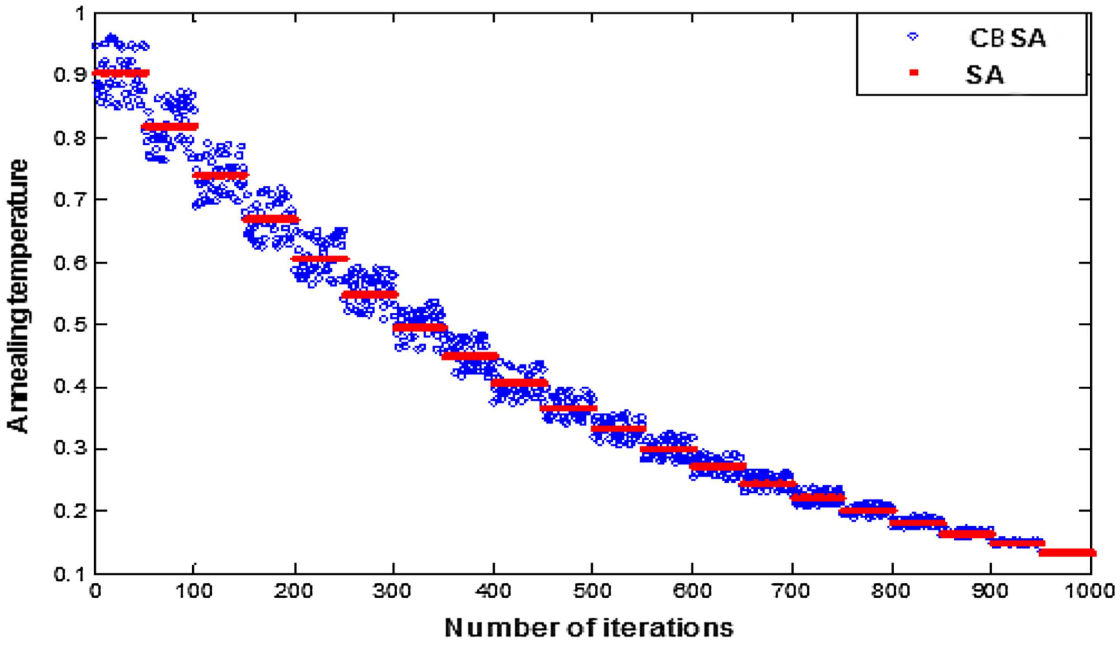

FIGURE 2. Temperature annealing process in CBSA and SA.

Moreover, the stable tendency of annealing temperature can protect the excellent individual effectively and fix the global minimum positions quickly as well [23].

Figure 2 shows the annealing temperature with each state in CBSA and SA. It clearly shows that the gradual decrease in annealing temperature is ideally achieved in CBSA. Also, the changeable bound of annealing temperature of CBSA is far wider than that of SAA. Hence, the annealing process of CBSA is more consistent with the physical annealing process [9]. Therefore, the CBSA overcomes the disadvantage of SA with respect to annealing phase.

4.3. Hybrid MGA_CBSA algorithm

The traditional GA cannot guarantee a complicated evolutionary process. This algorithm is not also capable to do a complete local search on solution space. Therefore, we can combine the power of genetic algorithm in global search with a local searches engine like parallel cloud based simulated annealing algorithm to find the global optimum solution. This hybrid algorithm that combines MGA and CBSA has both advantages of these algorithms and help to improve solution performances.

In this hybrid algorithm, first the MGA generates initial population. Next, some of these solutions have been selected for CBSA as initial solutions. To have variety in the proposed algorithm, we consider that some bad, normal and good solutions could be selected with the same chances. Then, the local search process on the selected solutions starts. The best solution gained by CBSA combine with population of MGA that is improved by crossover and mutation operators. The new population is evaluated and arranged and used as the initial population of the next generation. This continues until stopping criteria are satisfied (Fig. 3).

4.3.1. The chromosome structure of the algorithm

The first step to illustrate proposed hybrid algorithm is chromosome structure. The most important feature of the chromosome is satisfying the constraints. Our designed chromosome is a two sub-matrix each of which corresponds to a special area of decision making. The first sub-matrix is a N^*T matrix, in which N is the

Procedure of MGA-CBSA

Generate the initial population
Until termination condition is meet **Do**
Evaluate the chromosomes
Apply the crossover operator
Calculate the similarity coefficient of chromosomes
Apply the mutation operator based on similarity coefficient
Start CBSA by some of the best and the worst chromosomes
  For all the selected chromosomes **Do**
    Set repetition counter $k = 0$
    Determine the initial temperature T_k
    Until outer loop termination condition is meet **Do**
      Until inner *loop termination condition is meet* **Do**
        Generate a neighbour solution ω_2 from current solution ω_1
        Calculate $\Delta_{\omega 1, \omega 2} = f(\omega_2) - f(\omega_1)$
          If $\Delta_{\omega 1, \omega 2} \leq 0$, then $\omega_1 = \omega_2$
          Else
          Generate the temperature T_k' based on cloud theory
          Set $\omega_1 = \omega_2$ with probability $exp\left(\frac{-\Delta}{T_k'}\right)$
          End
        End
    $k = k + 1$
    $T_k = T_{k-1}\alpha$
    End
  End
Transfer the improved solutions to the genetic population and combine all the generated population
End

FIGURE 3. The Pseudo-code of proposed MGA_CBSA.

number of facilities, and T is the number of periods. Each row of this matrix shows the layout of a period. In each row, the value of each gene represents the facility number and the rank of it represents the position of facility in a special period of time.

Sub-matrix 2 shows the transportation operations. It consists of T matrix each of which is related to transportation operation in a period. Each matrix is $3*N$. The first row shows the product number. Second row is related to the handling operation and the last row present the transporter number. For example, Figure 4 illustrates the form of a chromosome in a system with five periods of time and two products. Product 1 has three operations and product 2 has two operations. According to the first column, the transportation operation number one of product 1 is performed by transporter 2, the transportation operation number one of product 2 is performed by transporter 2 and so on. Also, according to this chromosome, in period 1, the facility 6 is in position 1, the facility 3 is in position 2, the facility 7 is in position 3, and the facility 8 is in position 4 and so on.

4.3.2. The crossover operator

Once a population is created and the best elements selected, an operator is needed to evolve the population. This operator is called crossover. The crossover operator applied for the layout sub-matrix is illustrated in Figure 5. The proposed crossover operator will generate feasible string. Consequently, checking the feasibility of the solution is no more required. The crossover scheme has the following steps:

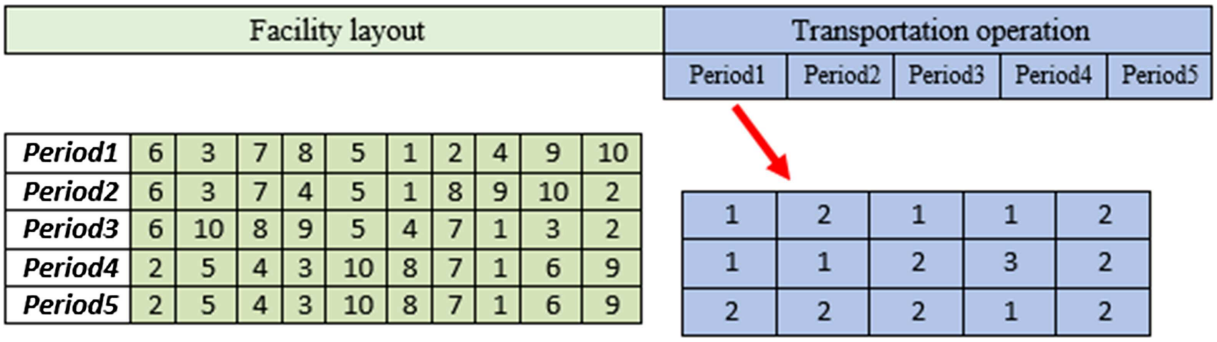

FIGURE 4. An example of the proposed chromosome structure.

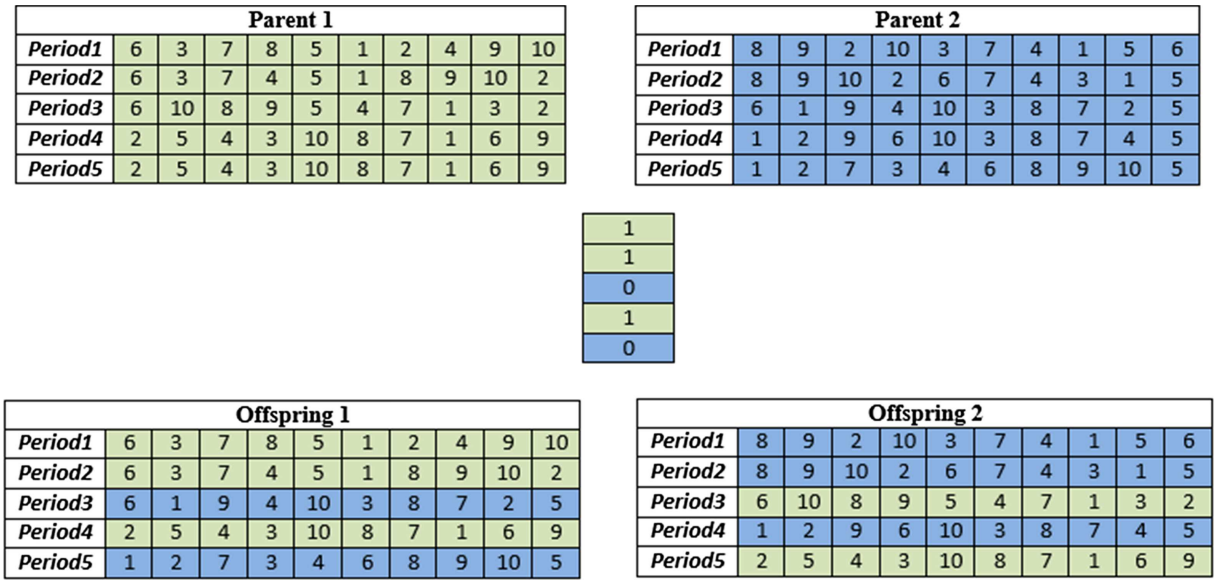

FIGURE 5. An example of crossover operator (layout structure).

(1) Two strings are randomly selected.
(2) A binary vector is randomly generated.
(3) The layout for the offspring in each period is determined according to reference vector, in a way that if it equals 1, the layout is just like parent 1 and if it equals 0, the layout is just like parent 2.

For sub-matrix related to transportation operations, a single point cross over is used. This type of crossover operator has the following steps (Fig. 6).

(1) Two parents are selected.
(2) A cross point is randomly selected.
(3) The columns before the cross point of Parent 1 and the columns after the cross point of Parent 2 are copied in the offspring 1. The *vice versa* is applied to generate offspring 2.

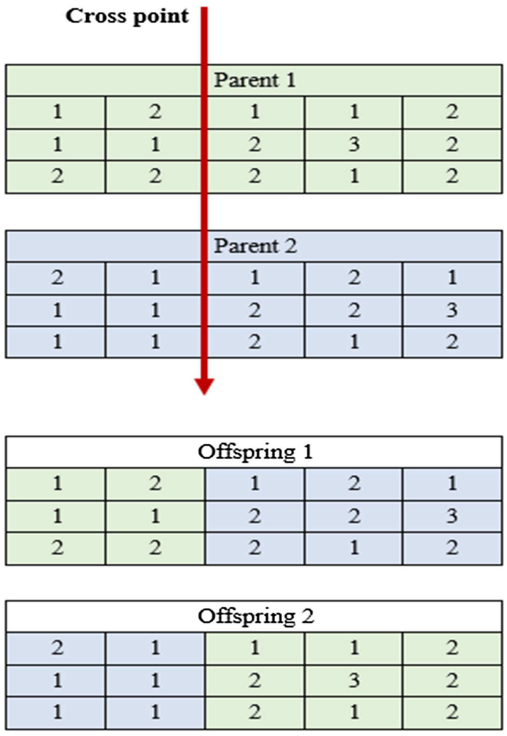

FIGURE 6. An example of crossover operator (transportation structure).

4.3.3. The mutation operator

We propose an adaptive mutation based on similarity coefficient method (SCM). The basic idea of the SCM was originally developed for grouping machines in production environment in Witte [40]. When GA is continuously converging, the similarity among the individuals of GA population becomes higher. Therefore, the fitness values of the individuals are significantly similar to each other and, the variety of the population is reduced which definitely deteriorate the performance of GA. Introducing different chromosomes into the current population can inhibit GA from fast converging. In this work, we modified the basic idea of SCM for the DFLP environment. We can calculate the similarity coefficient as follows:

$$\mathrm{SC}_{ab} = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} \partial\left(X_{ita}, X_{itb}\right)}{N \times T} \tag{4.1}$$

where X_{ita} and X_{itb} are locations of facility i in period t in chromosomes a and b, N is the number of facilities and, T is the number of periods. For comparing the similarity between two chromosomes, we consider the similarity of each gene that can be expressed as follows:

$$\partial\left(X_{ita}, X_{itb}\right) = \begin{cases} 1 & \text{if } X_{ita} = X_{itb} \\ 0 & \text{otherwise.} \end{cases}$$

The average similarity coefficient of the population is calculated as follows:

$$\overline{\mathrm{SC}} = \frac{\sum_{a=1}^{N-1} \sum_{b=a+1}^{N} \mathrm{SC}_{ab}}{\binom{N}{2}} \tag{4.2}$$

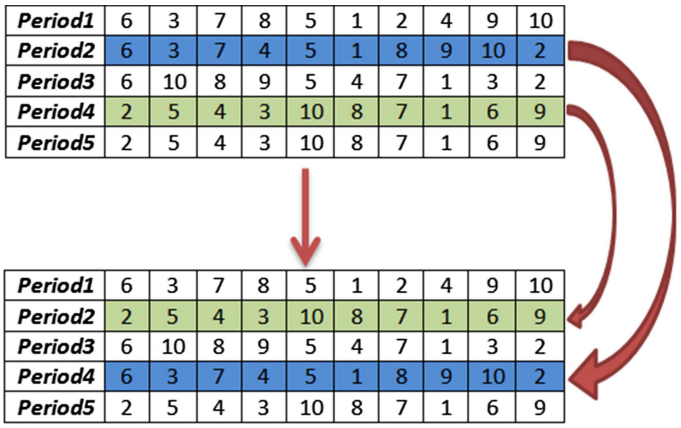

FIGURE 7. An example of mutation operator (layout structure).

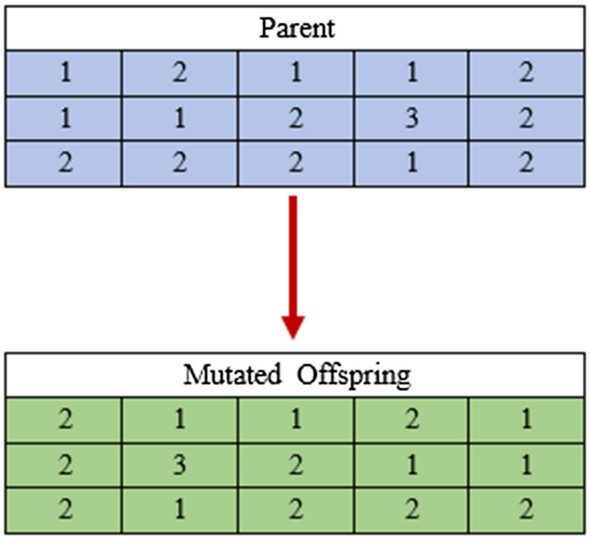

FIGURE 8. An example of mutation operator (transportation structure).

where w is number of chromosomes in the population. Considering a pre-defined threshold similarity coefficient (γ), the specified mutation operator will be automatically involved in the GA loop by the following condition:

$$\begin{cases} \text{apply Mutation oprator to GA loop} & \text{if } \overline{SC} > \gamma \\ \text{do not use Mutation oprator to GA loop} & \text{otherwise} \end{cases}$$

γ is determined according to some experiments.

In order to apply mutation operator for layout sub-matrix first, two periods (rows) from the current solution are randomly selected and then swap the layout in these periods (Fig. 7). In order to apply mutation operator for transportation sub-matrix first, a periods from the current solution is randomly selected and then all the transportation operations are randomly arranged (Fig. 8).

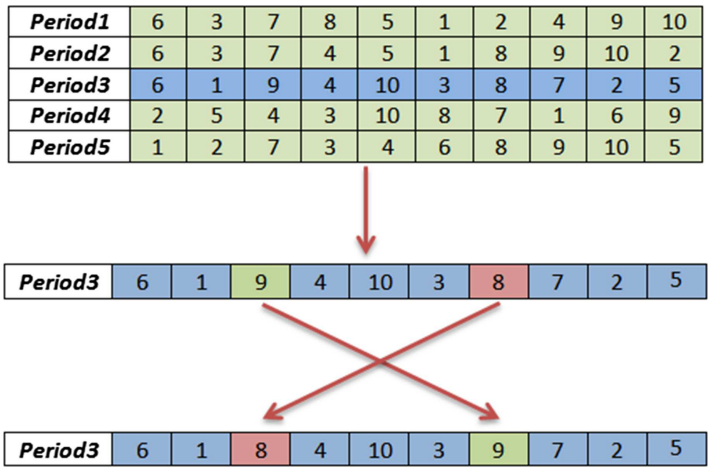

FIGURE 9. An example of neighbourhood definition (layout structure).

4.3.4. The neighbourhood structure

The neighbourhood moves is applied to the current solution to move it to a new positon in solution space. 2-change neighbourhood is used to define the neighbourhood. First, we select a period (row) randomly. Then, two different locations (columns) from the selected period are chosen and the facilities in these locations are swapped (Fig. 9).

4.3.5. Initial temperature

The initial temperature (T_0) is one of the main input parameters of the SA-based algorithms. A suitable initial temperature is that leads to a high acceptance probability. The acceptance probability (*prob*) is calculated by the following formula:

$$prob = \exp\left(\frac{-\Delta}{T'_k}\right) \tag{4.3}$$

where T'_k is the temperature in iteration k and Δ is the normalized value of difference between current and neighbourhood objective function as follows:

$$\Delta = \left|\frac{F(X_{\text{new}}) - F(X_{\text{old}})}{F(X_{\text{old}})}\right|. \tag{4.4}$$

According to equations (4.3) and (4.4) the acceptances probability is related to the both temperature and the difference between current and neighbourhood objective function. Consequently, to determine the initial temperature, it is more rational to consider the difference range between objective function of solutions. To take this issue into account, we first randomly generate a large set of solutions and compute the standard division ($St.Dev$) of objective function for those solutions. Then, equation (4.5) is applied to compute the initial temperature.

$$T_0 = \left(\frac{-St.Dev}{\ln(0.5)}\right). \tag{4.5}$$

4.3.6. Cooling schedule

The annealing temperature in general SA is a certain value in every stage; but the temperature update function in CBSA is dynamic. In CBSA the annealing temperature is specified continuously based on the cloud theory and Y condition normal cloud. Algorithm 1 is used to generate the continuous temperature so that

the annealing temperature changes randomly around the given reference, like cloud (Fig. 2). This leads to a high increment in diversity. Therefore, by preserving the diversity of searching process, the probability of being trapped in the local optimums is reduced.

Algorithm 1. Generating the temperature annealing in the stage k.

/ Y condition cloud generator /
$En = T_k, He = T_k, v_0 = 1 - T_k$
$En' = En + He - rand(0,1)/3$
/ $rand(a,b)$ generates random numbers which are uniformly distributed between a and b /
$T'_k = En'\sqrt{-2ln(v_0)}$

In Algorithm 1, T_k is the reference temperature at k_{th} step of the outer loop is determined by the following equation:

$$T_k = T_{k-1}\alpha \tag{4.6}$$

where α is the cooling rate obtained by conducting some experiments. T'_k is the temperature that is used to calculate acceptance function. Therefore, using a Y condition cloud generator a group of new randomly values that are distributed around the base temperature.

4.3.7. Stopping criteria

In order to limit the number of iterations of both MGA and CBSA algorithms, some convergence experiments were performed and the best criterion is applied as follows:

Genetic algorithm will be stopped if the fitness function of the elite chromosome (chromosome with the best fitness value) does not change more than 0.5% after a pre-determined number of successive generations. This value is determined as 40 by the experimental design.

For stopping CBSA in a temperature level first, we define the set of m iterations as a round. If the mean change between two successive rounds of iterations remains constant within 0.95% confidence interval, the thermal equilibrium of the system is proven, and so the temperature is reduced; otherwise, the algorithm keep perturbing the solutions by creating neighbouring solutions. For the outer loop of CBSA, a certain number of iterations are set as stopping criteria, and this value is determined to 100 as a result of experimental design.

5. Computational results

In this section the performances of proposed algorithm is evaluated. In the proposed algorithm two heuristic of decreasing temperature based on cloud theory and similarity coefficient method were applied. The proposed algorithm is named MGA_CBSA and is compared with GA_CBSA and MGA_SA. In GA_CBSA the decreasing temperature is based on cloud theory but similarity coefficient method has not been applied. In MGA_SA, similarity coefficient method is used but the temperature decreasing is not based on cloud theory. To do this, a set of test problems are defined, where they are randomly generated based on a uniform distribution shown Table 1. Furthermore Table 2 shows the machines to complete the processing routes of each job. Note that throughout this paper all calculations were executed on a computer with Core i7-CPU 3.20 GHz, RAM 8.00 GB. Moreover algorithms were coded using MATLAB software (Version 7.10.0.499, R2010a).

5.1. Small-sized test problems

In order to evaluate the capability of the metaheuristic algorithms to find the global optimum solution, first the results of the metaheuristic algorithms is compared against the IBM ILOG CPLEX 12 solver for small-sized problem instances. To this aim, 12 small-sized test problems are used. Each instance was allowed a maximum of 7200 s of CPU time (2 h) using the CPLEX solver. The computational results are presented in Table 3. The % GAP as the deviation percentages of the best solutions found by the meta-heuristic algorithm from the CPLEX

Table 1. The framework of generating random instances.

Number of product	[6, 15, 30]
Number of periods	[5,10]
Distance between locations (meter)	Uniform [10, 50]
Product demand	Uniform [10, 30]
Process times (second)	Uniform [10, 50]

Table 2. Alternative machines for products operations.

product_route{1}	[1 2 3 4];
product_route{2}	[1 2 4];
product_route{3}	[1 3 8 9 10];
product_route{4}	[1 8 9 10];
product_route{5}	[1 3 5 8 10];
product_route{6}	[1 2 3 4 5 7 8 10];
product_route{7}	[1 3 4 5 6 8 10];
product_route{8}	[1 5 6 8 8 10];
product_route{9}	[1 5 6 8 10];

is shown in Table 3. The GAP is presented in equation (5.1). Average CPU time for each problem is also given in second.

$$\text{GAP}\% = \frac{\text{Sol}_{\text{Alg}} - \text{Sol}_{\text{CPLEX}}}{\text{Sol}_{\text{CPLEX}}} * 100. \tag{5.1}$$

The differences between the global optimum of the CPLEX and the proposed metaheuristic algorithms are sufficiently small. For problems 1–9, on average, the metaheuristic algorithm MGA_CBSA, GA_CBSA and MGA_SA have been able to find the solutions with a gap of 3.1%, 4.1% and 7.1% respectively. These results indicate high capabilities of the metaheuristic algorithms in achieving near optimum solutions. To statistically evaluate the algorithm's performance, Tukey least significant difference (HSD) intervals are applied. The results, presented in Figure 10, indicate that there is no significant difference between metaheuristic algorithms and CPLEX solver. Thus, it can be concluded that the metaheuristic algorithms are capable to find the high quality solutions. As the problem size increases, the CPLEX optimality gap will increase so that for problems 10–12 the CPLEX cannot find the global optimum solution in allowable time. Therefore, further comparisons are performed only between metaheuristic algorithms.

5.2. Large-sized test problems

In order to compare the meta-hubristic algorithms in large sized problems, 30 test problems with 6, 15 and 30 facilities and 5 and 10 periods of time is considered. Each department-period combination, contains five different problems. The computational results are shown in Figures 11 and 12. As shown in Figure 11, the MGA_CBSA is able to find better solution with respect to two other algorithms. Moreover, the high performance of MGA_CBSA is clearer in larger sized problems. This proves the applicability of proposed MGA_CBSA in large sized of real-world problems. Also, according to Figure 12, the GA_CBSA has the lowest capability in terms of CPU Time. The main reason of this weakness is that in GA_CBSA the mutation operator is applied in all of generation and this leads to increase computational time.

In order to statistically analyse the results shown in Figures 11 and 12 an analysis of variance test based on algorithms' relative percentage division (RPD) is performed. RPD is a normalized measure of objective

TABLE 3. Comparison between the solutions of CPLEX and the proposed algorithms (for small-sized problems).

	N	T	CPLEX sol. (Opt %)		MGA_CBSA			GA_CBSA			MGA_SA		
			OF	Time	OF	Time	GAP	OF	Time	GAP	OF	Time	GAP
1	4	2	83487 (0)	185	83487	64	0.000	83487	69	0.000	83487	67	0.000
2			82378 (0)	188	82378	76	0.000	82378	71	0.000	82378	70	0.000
3			80734 (0)	185	80734	85	0.000	82541	89	0.022	85143	88	0.055
4	4	3	82129 (0)	359	88549	94	0.078	89248	98	0.087	92417	99	0.125
5			89257 (0)	354	89257	87	0.000	89257	103	0.000	94127	98	0.055
6			89674 (0)	361	92514	98	0.032	93625	108	0.044	94217	101	0.051
7	5	2	110124 (0)	1245	119527	134	0.085	124095	169	0.127	127434	141	0.157
8			109871 (0)	1237	112954	127	0.028	112841	175	0.027	118207	134	0.076
9			109351 (0)	1281	115927	137	0.060	117254	172	0.072	122418	141	0.119
10	5	3	125581 (35)	7200	120514	197	–0.040	122354	235	–0.026	123271	198	–0.018
11			139524 (37)	7200	123417	186	–0.115	125176	241	–0.103	128241	185	–0.081
12			128519 (41)	7200	122519	197	–0.047	124327	229	–0.033	128041	201	–0.004

Notes. T = number of periods. N = number of machines. OF = objective function. Opt% = CPLEX optimality gap.

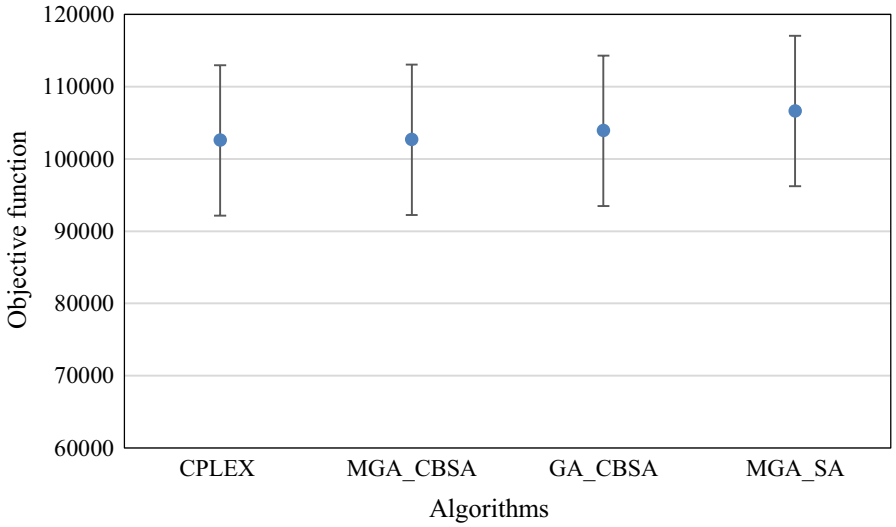

FIGURE 10. Mean plot and Tukey's Honest Significant Difference (HSD) at the 95% confidence level.

functions for in each instance. Equation (5.2) is used to calculate the RPD.

$$\mathrm{RPD}_{ij} = \frac{\mathrm{Alg}_{\mathrm{sol}}(ij) - \min_{\mathrm{sol}}(j)}{\min_{\mathrm{sol}}(j)} i = 1, \ldots, 3, j = 1, \ldots, 30 \tag{5.2}$$

where RPD_{ij} is the RPD of algorithm i for problem j, $\mathrm{Alg}_{\mathrm{sol}}(ij)$ is objective function value of algorithm i for problem j problem and $\min_{\mathrm{sol}}(j)$ is the best value of the objective function between all algorithms for problem j. The results of ANOVA presented in Table 4, show that the algorithms are significantly different. This makes necessary using Tukey Test to statistically compare algorithms. Figure 13 shows the simultaneous Tukey honest significant different (HSD). As shown in Figure 13, the proposed MGA_CBSA statistically outperforms all the other algorithms in objective function and CPU Time. While GA_CBSA is statistically better than MGA_SA in terms of objective function and MGA_SA is statistically better than GA_CBSA in terms of CPU Time.

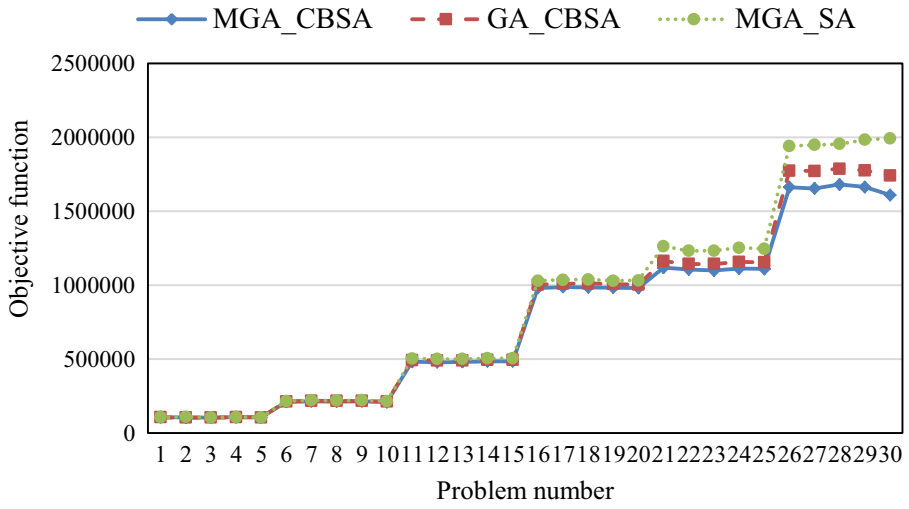

FIGURE 11. Comparison of the algorithms according to objective function metric.

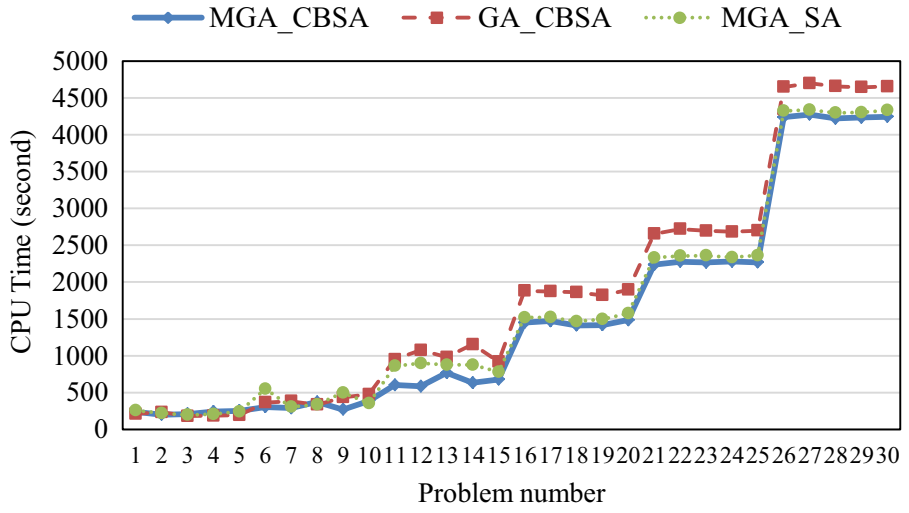

FIGURE 12. Comparison of the algorithms according to CPU Time metric.

5.3. Sensitivity analysis of the algorithms

In this section, we apply an extensive experiment to evaluate the effect of problem size (number of machines and periods) on the quality of algorithms. To statistically analyse the effect of problem size on three algorithms, the Tukey's test (at the 95% confidence level) is applied. Means plot and Tukey's honest significant difference (HSD) intervals are shown in Figure 14. It clearly shows the interaction among the factors of type of algorithm and problem size. According to the results, it can be noted that increasing number of machines factor leads to a tremendous decrement in the MGA_SA quality. Our MGA_CBSA are highly robust against the increasing of number of machines. It perform similar for $N = 6$, $N = 15$ and $N = 30$. The performance of GA_CBSA is statistically better than MGA_SA.

TABLE 4. ANOVA results.

	Source	Sum of squares	Degrees of freedom	Mean square	*F*-Test	*P*-value
Objective function	Algorithm	813.35	2	406.673	26.21	1.24E-09
	Error	1350.09	87	15.518		
	Total	2163.43	89			
CPU Time	Algorithm	6643.5	2	3321.74	9.3	0.0002
	Error	31071.5	87	357.14		
	Total	37715	89			

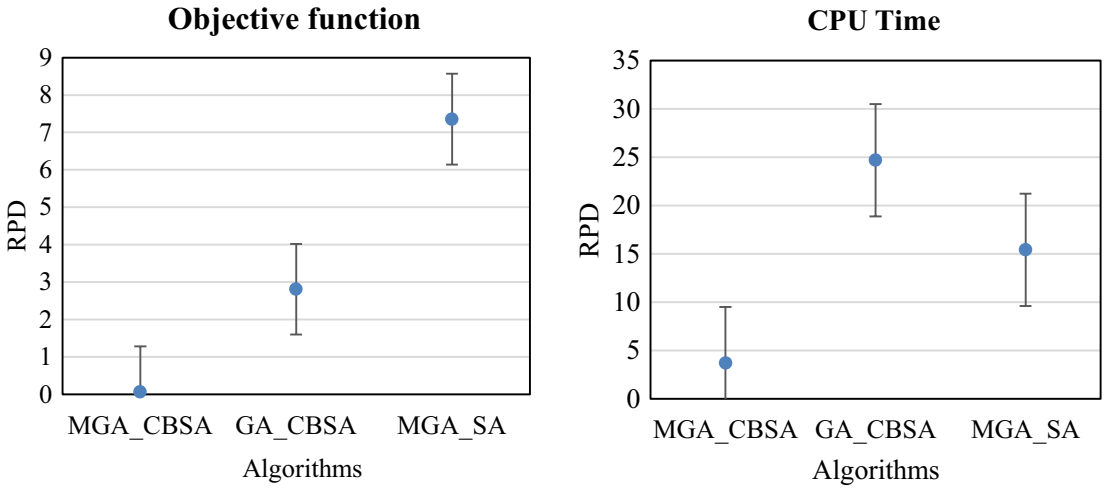

FIGURE 13. Means plot and Tukey intervals (at the 95% confidence level).

Figure 15 makes a comparison of different number of periods. According to the Figure 15 it can be proved that the number of periods is the least effective factor in algorithms' performance, so that all the algorithms can statistically perform similarly in different sizes of T and be robust. However, the MGA_CBSA can achieve better outcomes especially for $T = 10$.

6. CONCLUSIONS

Cloud theory has a very promising potential for solving optimization problems. The successful integration of the cloud theory and the other heuristics extend the research area of cloud theory and also introduce a new way for the solving optimization problems. From this point, an effective hybrid algorithm named modified genetic algorithm-cloud based simulated annealing algorithm (MGA_CBSA) was developed to efficiently solve DFLP in this paper. It makes the first attempt to show how the cloud theory can be applied to DFLP. The performance of the proposed MGA_CBSA was examined with the data set. The results show that the proposed hybrid heuristic obtains promising results for a large width of problems and has also a reasonable computational time when compared to the other algorithms. Also increasing the size of the problem affects slightly the MGA_CBSA. Although this paper considers complex and realistic characteristics of DFLP, a more complicated formulation can be derived by taking into account the pickup/delivery points, aisles, maintenance of material handling equipment, and several floors in the future. The analysis of systems with limited buffer capacity is considerably

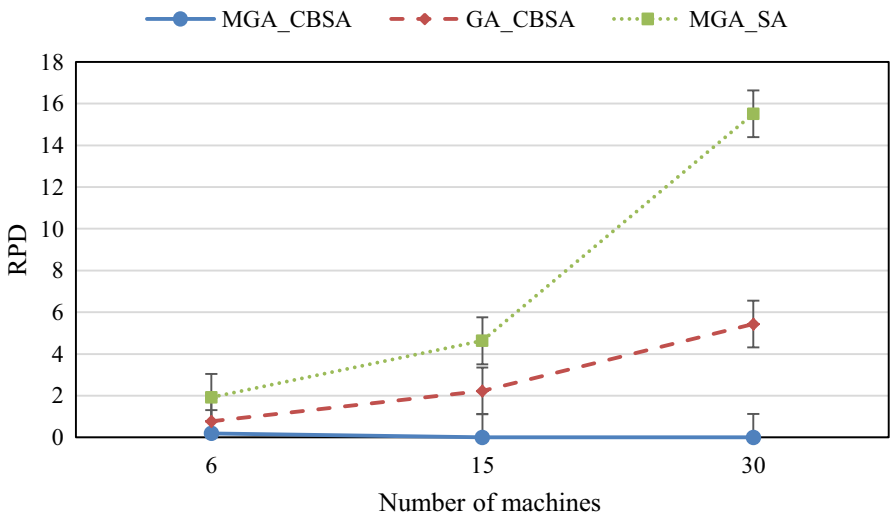

FIGURE 14. Means plot and HSD intervals for interaction between the type of algorithm and number of facilities.

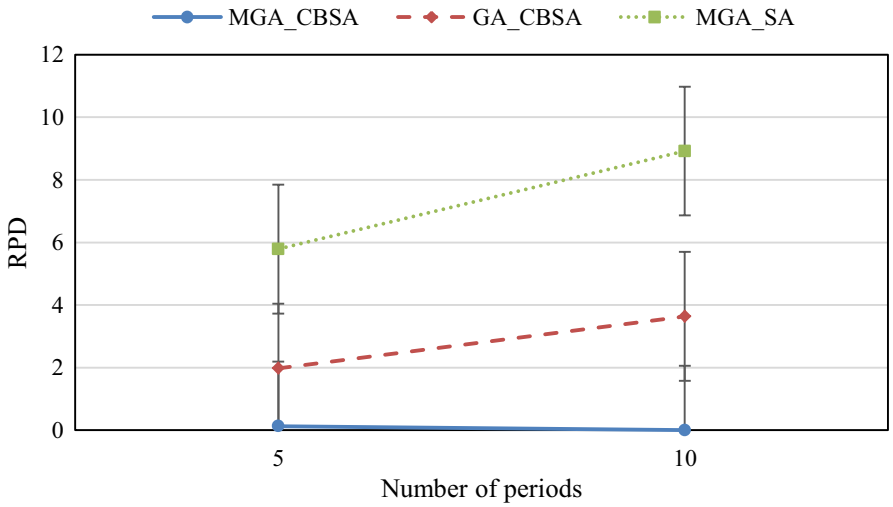

FIGURE 15. Means plot and HSD intervals for interaction between the type of algorithm and number of facilities.

more complex and is worthy of future research. We also suggest to compare the performance of proposed MGA_CBSA for other combinatorial optimization problems.

REFERENCES

[1] P. Afentakis, R. Millen and M.M. Solomon, Dynamic layout strategies for flexible manufacturing systems. *Int. J. Prod. Res.* **28** (1990) 311–323.

[2] J. Balakrishnan and C.H. Cheng, Dynamic layout algorithms: a State-of-the-art Survey. *Omega* **26** (1998) 507–521.

[3] J. Balakrishnan and C.H. Cheng, Genetic search and the dynamic layout problem. *Comput. Oper. Res.* **27** (2000) 587–593.

[4] J. Balakrishnan and C.H. Cheng, The dynamic plant layout problem: incorporating rolling horizons and forecast uncertainty. *Omega* **37** (2009) 165–177.

[5] J. Balakrishnan, F.R. Jacobs and M.A. Venkataramanan, Solutions for the constrained dynamic facility layout problem. *Eur. J. Oper. Res.* **57** (1992) 280–286.

[6] J. Balakrishnan, C.H. Cheng, D.G. Conway and C.M. Lau, A hybrid genetic algorithm for the dynamic plant layout problem. *Int. J. Prod. Econ.* **86** (2003) 107–120.

[7] A. Baykasoğlu and N.N.Z. Gindy, A simulated annealing algorithm for dynamic layout problem. *Comput. Oper. Res.* **28** (2001) 1403–1426.

[8] A. Baykasoğlu, T. Dereli and I. Sabuncu, An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems. *Omega* **34** (2006) 385–396.

[9] L. Deyi and D. Yi, Artificial Intelligence with Uncertainty. Chapman & Hall / CRC, London (2005).

[10] L. Deyi, M. Haijun and S. Xuemei, Membership clouds and membership cloud generators. *J. Comput. Res. Dev.* **32** (1995) 15–20.

[11] K. Di, L. Deyi and L. Deren, Cloud theory and its applications in spatial data mining knowledge discovery. *J. Image Graph* **4** (1999) 930–935.

[12] A. Drira, H. Pierreval and H. Hajri-Gabouj, Facility layout problems: a survey. *Ann. Rev. Control* **31** (2007) 255–267.

[13] E. Erel, J.B. Ghosh and J.T. Simon, New heuristic for the dynamic layout problem. *J. Oper. Res. Soc.* **54** (2003) 1275–1282.

[14] M. Horta, F. Coelho and S. Relvas, Layout design modelling for a real world just-in-time warehouse. *Comput. Ind. Eng.* **101** (2016) 1–9.

[15] S.S. Hosseini and M. Seifbarghy, A novel meta-heuristic algorithm for multi-objective dynamic facility layout problem, *RAIRO:OR* **50** (2016) 869–890.

[16] S. Hosseini, A. Al Khaled and S. Vadlamani, Hybrid imperialist competitive algorithm, variable neighborhood search, and simulated annealing for dynamic facility layout problem. *Neural Comput. App.* **25** (2014) 1871–1885.

[17] M. Hu, A high-fidelity three-dimensional simulation method for evaluating passenger flow organization and facility layout at metro stations. *Simulation* **93** (2017) 841–851.

[18] B.K. Kaku and J.B. Mazzola, A tabu-search heuristic for the dynamic plant layout problem. *Informs J. Comput.* **9** (1997) 374–384.

[19] M. Kaveh, V. Majazi Dalfard and S. Amiri, A new intelligent algorithm for dynamic facility layout problem in state of fuzzy constraints. *Neural Comput. App.* **24** (2014) 1179–1190.

[20] K.K. Krishnan, S.H. Cheraghi and C.N. Nayak, Dynamic From-Between Chart: a new tool for solving dynamic facility layout problems. *Int. J. Ind. Syst. Eng.* **1** (2006) 182–200.

[21] T.A. Lacksonen and E.E. Enscore, Quadratic assignment algorithms for the dynamic layout problem. *Int. J. Prd. Res.* **31** (1993) 503–517.

[22] J. Liu, W. Dawen, H. Kun and X. Yu, Combining Wang–Landau sampling algorithm and heuristics for solving the unequal-area dynamic facility layout problem. *Eur. J. Oper. Res.* **262** (2017) 1052–1063.

[23] P. Lv, L. Yuan and J. Zhang, Cloud theory-based simulated annealing algorithm and application. *Eng. Appl. Artif. Intell.* **22** (2009) 742–749.

[24] A.R. McKendall Jr and W.H. Liu, New Tabu search heuristics for the dynamic facility layout problem. *Int. J. Prod. Res.* **50** (2012) 867–878.

[25] A.R. McKendall Jr and J. Shang, Hybrid ant systems for the dynamic facility layout problem. *Comput. Oper. Res.* **33** (2006) 790–803.

[26] A.R. McKendall Jr, J. Shang and S. Kuppusamy, Simulated annealing heuristics for the dynamic facility layout problem. *Comput. Oper. Res.* **33** (2006) 2431–2444.

[27] G. Moslemipour, T.S. Lee and D. Rilling, A review of intelligent approaches for designing dynamic and robust layouts in flexible manufacturing systems. *Int. J. Adv. Manuf. Technol.* **60** (2012) 11–27.

[28] J.A. Qudeiri, U. Umer, F.A. Khadra, H.M.A. Hussein, A. Al-Ahmari, S. Darwish and M.H. Abidi, Layout design optimization of dynamic environment flexible manufacturing systems. *Adv. Mech. Eng.* **7** (2015) 1–11.

[29] H. Rezazadeh, M. Ghazanfari, M. Saidi-Mehrabad and S.J. Sadjadi, An extended discrete particle swarm optimization algorithm for the dynamic facility layout problem. *J. Zhejiang Univ. Sci. A* **10** (2009) 520–529.

[30] J.M. Rodriguez, F.C. MacPhee, D.J. Bonham and V.C. Bhavsar, Solving the dynamic plant layout problem using a new hybrid meta-heuristic algorithm. *Int. J. High Perform Comput. Networking* **4** (2006) 286–294.

[31] M.J. Rosenblatt, The dynamics of plant layout. *Manage. Sci.* **32** (1986) 76–86.

[32] R. Şahin and O. Türkbey, A new hybrid tabu-simulated annealing heuristic for the dynamic facility layout problem. *Int. J. Prod. Res.* **47** (2009) 6855–6873.

[33] R. Şahin, K. Ertoğral and O. Türkbey, A simulated annealing heuristic for the dynamic facility layout problem with budget constraint. *Comput. Ind. Eng.* **59** (2010) 308–313.

[34] R. Sharpe and B.S. Marksjö, Facility layout optimization using the metropolis algorithm. *Environ. Plann B* **12** (1985) 443–453.

[35] S.P. Singh and R.R.K. Sharma, A review of different approaches to the facility layout problems. *Int. J. Adv. Manuf. Technol.* **30** (2006) 425–433.

[36] J.A. Tompkins, J.A. White, Y.A. Bozer and J.M.A. Tanchoco, Facilities Planning. John Wiley & Sons, New York (2003).

[37] B.H. Ulutas and A.A. Islier, A clonal selection algorithm for dynamic facility layout problems. *J. Manuf. Syst.* **28** (2009) 123–131.

[38] T.L. Urban, A heuristic for the dynamic facility layout problem. *IIE Trans.* **25** (1993) 57–63.

[39] S. Vitayasak, P. Pongcharoen and C. Hicks, A tool for solving stochastic dynamic facility layout problems with stochastic demand using either a Genetic Algorithm or modified Backtracking Search Algorithm. *Int. J. Prod. Econ.* **190** (2017) 146–157.

[40] J. Witte, The use of similarity coefficients in production flow analysis. *Int. J. Prod. Res.* **18** (1980) 503–514.

[41] L. Yao, L. Sun, W. Wang and X. Xia, Connection facility layout model of subway stations. *Adv. Mech. Eng.* **7** (2015) 457508.