A Tunned-parameter Hybrid Algorithm for Dynamic Facility Layout Problem with Budget Constraint using GA and SAA

Hani Pourvaziri [a,*], Parham Azimi [b]

[a] MSc, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[b] Assistant Professor, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

Received 12 December, 2012; Revised 15 September, 2013; Accepted 12 October, 2013

Abstract

A facility layout problem is concerned with determining the best position of departments, cells, or machines on the plant. An efficient layout contributes to the overall efficiency of operations. It's been proved that, when system characteristics change, it can cause a significant increase in material handling cost. Consequently, the efficiency of the current layout decreases or is lost and it does necessitate rearrangement. On the other hand, the rearrangement of the workstations may burden a lot of expenses on the system. The problem that considers balance between material handling cost and the rearrangement cost is known as the Dynamic Facility Layout Problem (DFLP). The objective of a DFLP is to find the best layout for the company facilities in each period of planning horizon considering the rearrangement costs. Due to the complex structure of the problem, there are few researches in the literature which tried to find near optimum solutions for DFLP with budget constraint. In this paper, a new heuristic approach has been developed by combining Genetic Algorithm (GA) and Parallel Simulated Annealing Algorithm (PSAA) which is the main contribution of the current study. The results of applying the proposed algorithm were tested over a wide range of test problems taken from the literature. The results show efficiency of the hybrid algorithm GA- to solve the Dynamic Facility Layout Problem with Budget Constraint (DFLPBC).
Keywords: Dynamic facility layout problem; Budget constraint; Genetic algorithm; Parallel Simulated annealing algorithm.

1. Introduction and Literature Review

Facility layout problem is concerned with determination of the most efficient arrangement of facilities within a factory. The facility can be defined as manufacturing cell, administrative office building or machine. An efficient layout causes an efficient material handling between facilities and thus decreases work in process (WIP) 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, inventory holding costs etc. the 20-50% of total operational cost and 15-70% of total cost of manufacturing systems depends on transportation cost. Thus, material handling cost is one of the most significant measures to determine the efficiency of a layout. The material handling cost is defined based on the flows of materials between departments and the distances between the locations of the departments. If the flows of materials don't change for a long time, the static layout problem could be used; but when demands change, the flows of materials between facilities change consequently. In this situation, the existing facility layouts are not suitable and efficient and need to be revised and rearranged.

Rearrangement of facilities to minimize the material handling costs and transportation cost is known as a dynamic facility layout problem (DFLP). The reasons for facility rearrangement in each organization are as follows:

1) Change in demands or products
2) Change in operation sequencing
3) Change in product strategies
4) Change in resources and facilities
5) Change in standard safety rules

Change in demand is the most important reason for rearranging facilities. Introducing new facilities or removing depreciation facilities can also cause a change in material handling pattern. Afentakis et al. (1990) stated that when these system's characteristics change, it can cause a significant increase in material handling requirements. These changes effect on the efficiency of present layout and necessitate the need to consider re-layout in the system. To solve this problem, the facility layout should be flexible and more comfortable to change. In this paper, the dynamic facility layout problem is studied and the material handling cost is used as efficiency index to evaluate the performance of our proposed algorithm with previous researches.

* Corresponding author E-mail: hani.pourvaziri@yahoo.com

Rosenblatt (1986) first introduced DFLP and developed an optimization approach based on dynamic programming to solve it. He showed that the number of layouts to be evaluated to guarantee optimality for a DFLP with N departments and T periods is$(N!)^T$. However, this approach is computationally intractable for real-life problems. Rosenblatt proposed two heuristics that were based on dynamic programming, each of which simply considers a set of limited good layouts for a single period. Because of the computational difficulties inherent in such a problem, several heuristics have since been developed. Urban (1998) developed a steepest-descent heuristic based on a pair-wise-exchange idea, which is similar to CRAFT. Lacksonen and Enscore (1993) introduced and compared five heuristics to solve the DFLP, which were based on dynamic programming, a branch and bound algorithm, a cutting plane algorithm, cut trees, and CRAFT.

It should be mentioned that in addition to the exact algorithms, many meta-heuristic algorithms have been reported in the literature. Balakrishnan and Cheng (2000) developed a genetic algorithm to solve the DFLP. Kaku and Mazzola (1997) used a tabu search (TS) heuristic. This TS heuristic is a two-stage search process that incorporates diversification and intensification strategies. Baykasoglu and Gindy (2001) proposed a simulated annealing (SA) heuristic for the DFLP, in which they used the upper and lower bound of the solution of a given problem instance to determine the SA parameters. Balakrishnan et al. (2003) presented a hybrid genetic algorithm for DFLP. Erel et al. (2005) proposed a new heuristic to solve the DFLP. They used weighted flow data from various time periods to develop robust layouts, and suggested the shortest path for solving the DFLP. McKendall and Shang (2006) developed three hybrid ant systems (HAS). McKendall et al. (2006) presented two (SA) heuristics. The first, (SA I), is a direct adaptation of SA for the DFLP. The second, (SA II), is the same as SA I, except that it incorporates an added look-ahead/look-back strategy. Rodriguez (2006) presented a hybrid meta-heuristic algorithm based on a genetic algorithm and tabu search. Krishnan et al. (2006) developed a new tool, the "Dynamic From–Between Chart", for an analysis of redesigned layouts. This tool models changes in the production rates using a continuous function. Balakrishnan and Cheng (2009) investigated the performance of algorithms under fixed and rolling horizons, differing shifting costs and flow variability, and forecast uncertainty. Gary et al. (2009) evaluate a multi-objective dynamic facility layout by ant system. Baykasoglu et al (2001) solved DFLP by simulated annealing for the first time. Sahin et al. (2010) presented simulated annealing algorithm (SAA) for the DFLP with budget constraint (DFLPBC). For an extensive review on the DFLP, one can refer to the work of Page, 1991; Balakrishnan and Cheng 1998; Tompkins al., 2003 Drira al., 2007. The studies described above share a common assumption that all departments are of equal size. However, some studies do not make this assumption. Two recent examples of such studies are McKendall and Hakobyan, 2010; Rodriguez et al., 2006; Gary et al., 2009.

It should be noticed that most previous researches did not consider the company budget for rearranging the departments. Because these rearrangements are costly activities, it is normal for a company to have a limited budget in this regard. According to the literature, there are just three studies on DFLP with the budget constraints; Balakrishnan et al. (1992), Baykasoglu et al. (2006), Sahin et al (2010). The last one which is the newest related research used a budget constraint for each period separately. They developed a simulated annealing algorithm for the problem and showed that their algorithm is more efficient than the two previous researches.

The assumptions for the DFLPBC are as follows:

- The flow between facilities is dynamic and deterministic
- Facilities and locations are of equal size
- The distances between facilities have been determined before.
- Budget allocated for each period is constant
- Leftover budgets from past periods are available for current period

This paper is organized in the following way. In section 2, the mathematical model of DFLPBC is presented. In section 3, a hybrid algorithm made by combining genetic algorithm and parallel simulated annealing algorithm (PSAA) is presented and the characteristics of the proposed GA-PSAA are specially described. In section 4, in order to obtain more accurate solutions, a parameter tuning according to Taguchi method is accomplished. In section 5, the computational results have been presented and compared for 48 numerical examples with small, medium and large sized in literatures. Finally, concluding remarks and suggestions for future researches are presented in section5.

2. Mathematical Model

The Dynamic Facility Layout Problem with Budget Constraint (DFLPBC) can be modelled as follows (Sahin et al., 2010):

The variables are:

$$X_{tij}\begin{cases}1 & \text{if facility } i \text{ is allocated to location } j \text{ in period } t\\ 0 & otherwise\end{cases}$$

B_t Available budget for period *t*.

LB_t Left-over budget from period *t* to period *t*+1.

The parameters and indexes are:

N The number of departments/locations

T The number of periods in the planning horizon

i, j, k, l	Index for departments/locations
t	Index for time periods
A_{tijl}	The cost of shifting department i from location j to l in period t
C_{tijkl}	The material transporting cost between the departments *i* in location *j* and the department *k* in location *l* in period *t*.
AB_t	Allocated budget for period *t*.

The mathematical model of DFLPBC is presented as below:

$$Minimize\ TC = \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl} * X_{t-1ij} * X_{til} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} C_{tijkl}\ X_{tij}\ X_{tkl} \quad (1)$$

St:

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

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

$$LB_t = B_t - \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl}\ y_{tijl} \ , \qquad t = 1,2,\dots,T \quad (4)$$

$$B_t = AB_t + LB_{t-1}, \qquad t = 1,2,\dots,T \quad (5)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{l=1}^{N} A_{tijl}\ y_{tijl} \le B_t, \qquad t = 1,2,\dots,T \quad (6)$$

$$X_{tij}, y_{tijl} \in \{0,1\} \quad , i, j, l = 1,2,\dots,N, \ t = 1,2,\dots,T \quad (7)$$

$$LB_t, B_t, AB_t \ge 0, \qquad t = 1,2,\dots,T \quad (8)$$

The objective function of this model is to minimize the total combined traveling and transporting material costs. Constraint set (2) ensures that each facility should be located in one position and constraint set (3) ensures that in each position only one facility should be allocated. Constraint set (4) is for equating the total available budget in a period to leftover budget from previous period and the allocated budget in the current period. Constraint set (5,6) represents the budget constraint for each period. Constraint set (7) restricts decision variables and constraint set (8) represents the non-negativity restriction on the decision variables.

3. Solution Methodology

Because of the hardness of optimally solving the problem with exact methods in reasonable CPU time, we should use heuristics or meta-heuristics algorithm to solve it. As in literatures the genetic algorithm had better performances in objective function and CPU time, we used genetic algorithm to solve this problem in this paper. Thus, we use a hybrid algorithm which is mixed of genetic algorithm and simulated annealing algorithm.

3.1. Genetic algorithm

GAs is stochastic evolutionary algorithms that are based on analogies with the principles of natural biological systems. GA is one of the best heuristic optimization methods which was originally developed by Holland (Mahmoudi et al., 2012). GA is especially suited 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 random to begin with but can be initialized with another heuristic algorithm. After initialization, a stochastic search process is employed to iteratively improve the average fitness of the population. Genetic algorithms employ standard operators to promote a population. Two of these main operators are mutation and crossover which are discussed in the following section.

3.2. Simulated annealing algorithm

Generally speaking, simulated annealing is a stochastic neighborhood search procedure to find the optimal solution in NP-hard problems. Ever since its introduction, the SA has shown a high performance in large combinatorial optimization problems (Naderi and Sadeghi, 2011). SA is an extension of a local search strategy designed to avoid getting trapped in a local optimum. This is accomplished by randomly generating neighbors and accepting a solution that worsens the value with 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 temperature T at iteration n. The method can avoid being stuck at a local optimum, and increase the probability of finding a global optimal solution. The acceptance probability *P* decreases over time as the temperature *T* decreases. Consequently, it performs a wide investigation of the solution space initially and then restricts the solution space gradually. Thus, algorithm converges to the best optimum solution.

The experimental results show that the performance of simulated annealing improves if the number of initial solutions from which the algorithm starts increases and space solution is searched in a parallel manner. This parallelism of simulated annealing is called Multiple

Independent Run (MIR) implementations. In MIR method, independent runs of sequential SA are executed in each processor and the best-found solution is selected as the result. Thus, the chance of algorithm to be converged to the global optimum increases. Because no communication of moves or solutions is required in this parallelization strategy, the time taken won't be increased drastically.

3.3. Hybrid GA-PSA algorithm

The normal GA cannot guaranty a complicated evolutionary process. This algorithm isn't 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 search engine like simulated annealing to find the global optimum solution. This hybrid algorithm which combines GA and SAA has both advantages of theses algorithms and help to improve solution performances.

In this hybrid algorithm, first, the GA generates initial population. Next, some of these solutions have been selected for PSAA 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 parallel local search process on the selected solutions starts. The best solutions gained by PSAA are combined with population of genetic algorithm which has been improved by crossover and mutation operators. The new population is evaluated and arranged and is used as the initial population of the next generation. This continues until stopping criteria are satisfied.

```
Initialize all control parameters
Generate initial random population
  While stopping criteria do not satisfy Do
        Evaluate the chromosomes
        Generate offspring with crossover
        Generate offspring with mutation
        Evaluate new population
  Parallel Simulated-annealing Phase
      Start PSAA by some of the best and the worst
       chromosomes
        For all the selected chromosomes Do
            While stopping criterion satisfied Do
                While system has reached thermal
                 equilibrium Do
                    Randomly generate neighbourhood
                    based on the current solution.
                    If better than the current state change
                    to the new state otherwise change to
                    the new state  according to the
                    transition  probability
                End while
              Decrease the temperature
          End while
        End for
Transfer the improved solutions to the genetic population
Combine all the generated population

  End while
```

Fig. 1. Pseudo code for genetic- Parallel Simulated-annealing algorithm

3.4. The chromosome structure of the algorithm

The first step to illustrate proposed hybrid algorithm is chromosome structure design. The most important feature of the chromosome is satisfying the constraints. Our designing chromosome is an $N * T$ matrix, in which N is the number of facilities and T is the periods of time. Each row of this matrix shows the layout of a period. In each row, the value of each gene represents the facility number and rank of it represents the position of facility in special period of time. For example, figure 2 illustrates the form of a chromosome in two periods of time. According to this chromosome, in period 1, the facility 2 is in position1, the facility4 is in position2, the facility 1 is in position 3 and finally the facility 3 is in position 4.

Period1	*2*	*4*	*1*	*3*
Period2	*3*	*2*	*4*	*1*

Fig. 2. An example of proposed chromosome structure

3.5. The crossover operator

Once a population has been created and the best elements selected, an operator is needed to evolve the Population. This operator is called crossover. Traditional crossover operators like one point crossover cannot be applied to DFLP because they often create invalid solutions. Therefore, a new crossover operator which is illustrated in Fig. 3, is used. The proposed crossover operator is totally different from that of the CVGA of Conway and Venkataramanan (1994) and the NLGA of Balakrishnan and Cheng (2000) and generates feasible solution string. Consequently, checking the feasibility of the solution is no more required. The crossover scheme 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 point 1, .., T).
3) The layout for the first period of offspring1 is generated in the following way: the segment from 1 to cross point1 of parent1 is directly copied to the offspring (segment1_off1_period1). The second segment of offspring1 in period 1 consists of elements (departments) of cross point1 up to n of parent1 in period1. But the order of placing these elements is according to order of them in parent 2. This structure is repeated for generating other periods of offspring1. This ensures feasibility in generated offspring and other offsprings which are produced like it.

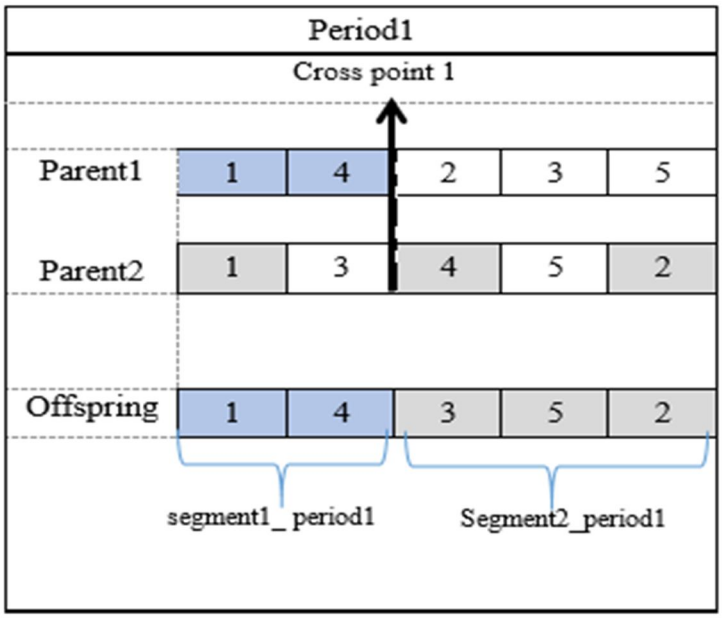

Fig. 3. An example of crossover operator

3.6. The mutation operator and neighborhood

The local transformations (or neighbourhood moves) in each iteration, that can be applied to the current solution(s), define a set of neighbouring solutions as: $N(s)$ = {new configuration obtained by applying a single move to the current solution}. Two swapping types are used. Swapping type 1 is used to define the neighbourhood $N(s)$ in local search (PSAA) and Swapping type 2 is used as mutation operator of genetic algorithm.

3.6.1. Swapping type 1

First, select a period (row) randomly. Then, select two different locations (columns) randomly from the selected period and swap the facilities in these locations (Fig 4).

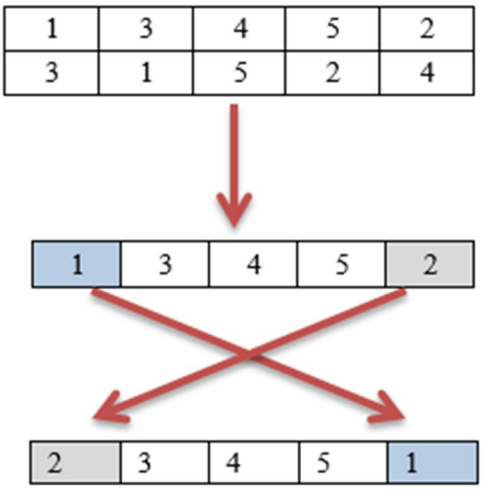

Fig. 4. An example of swapping type 1 (neighbourhood definition)

3.6.2. Swapping type 2:

Select two periods (rows) randomly from the current solution and swap the layout in these periods (Fig 5).

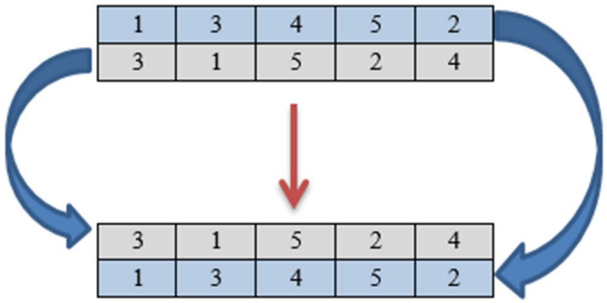

Fig. 5. An example of Swapping type 2 (mutation operator)

3.7. Cooling Schedule

The performance of this algorithm also depends on the cooling schedule, which is relevant to the temperature updating function. In the proportional decrement scheme, temperature decreases at the k and k +1 steps of the outer loop by:

$$T_{k+1} = \alpha T_k \quad (9)$$

Wherein α is cooling rate and is obtained by some experiments.

3.7. Stopping criteria

To limit the number of iterations of both GA and PSAA algorithms, some convergence experiments were performed and the best criterion was applied as follows:

Genetic algorithm will be stopped when one of the following situations is done:

1. 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 (according to experimental result this value is set to 40);
2. The total number of iterations reached to a predefined number (according to experimental result this value is set to 700).

For stopping PSAA in a temperature level first we define the set of *m* iterations as a round. The system reaches equilibrium at a certain temperature if the mean changes between two successive rounds of iterations remain constant within 0.95% confidence interval, we conclude that the system has reached thermal equilibrium and so we reduce the temperature; otherwise, we keep perturbing the solutions by creating neighbouring solutions, until reaching equilibrium. For the outer loop of PSAA the total number of iterations exceeds certain value (according to experimental result this value is set to 500) is set as stopping criteria.

4. Tuning the Parameters

The selection of parameters has a major influence on the efficiency of any meta-heuristic algorithms (Zoraghi et al., 2012). One of the most effective methods in selecting the appropriate parameters is Taguchi parameter tuning method. In the Taguchi method, the results are transferred into a measure called signal to noise (S/N) ratio. The frame of this ratio is different for each purpose. Eq(10) represents the (S/N) ratio formula for minimization objectives.

$$\frac{S}{N} = -10\log(\frac{1}{n}\sum_{i=1}^{n} y_i^2) \quad (10)$$

Where y is the response for the given factor level combination and n is the number of responses in the factor level combination. The assuming parameters of the proposed algorithm are: the crossover rate (P_c), the mutation rate (P_m), the initial temperature (T_0), cooling rate (α) and initial population (n_{pop}). These parameters are examined in 4 different levels. The best level of involved parameters is presented in Table 1.

Table 1
Factors and their best levels

Responses	P_c	P_m	T_0	α	$n_{pop,}$
Optimal level	0.8	0.15	1000	0.985	50

5. Computational Results

5.1. Analysis of results and comparisons

All computations were carried out on a Pentium V PC with 4 GByte RAM, 2.2 GHZ processor, Core i5. Minitab 16 software was used for tuning of parameters. To evaluate the performances of the proposed algorithm, 48 test problems with 6, 15 and 30 facilities for 5 and 10 periods of time were considered. The computational results for dynamic facility layout problem with budget constraint (DFLPBC) were compared to the results obtained by Sahin et al. (2010) and Baykasoglu et al. (2006) researches in literature. Tables 2, 3 and 4 show the comparing results of our proposed algorithm with Sahin et al. (2010) and Baykasoglu et al. (2006) in terms of objective function and CPU time for small, medium and large sized problems, respectively. The allocation of budget to periods of time is as the same as the one used by Sahin et al. (2010).

Column 7 shows the improvement values as the promotion percentages of the best solutions found by the proposed algorithm from the two best known solutions in the literature. The improvement is presented in Eq (11).

$$Improvement\% = \frac{min_{(Baykasoglu,Sahin)} - GA\_PSAA}{min_{(Baykasoglu,Sahin)}} * 100 \quad (11)$$

As the computational result shows, in Table 2 which is small-sized problems with N=6, the proposed algorithm has the 0.033% of the improvement factor. In Table 3 which is medium-sized problems with N=15, the proposed algorithm has the better solution with 0.5842% of improvement index and finally in Table 4 which is large-sized problems with N=30, the proposed algorithm has better performances according to the 1.4023 % improvement. To sum up, the proposed algorithm provides good solution quality in comparison to the algorithms developed in previous researches especially in large-sized problems.

In addition, for better understanding of the performance of hybrid GA-PSAA algorithm, we compared these algorithms by ANOVA test. To this aim, the cost function of algorithms is transformed to relative percentage deviation (RPD) as follows:

$$RPD_{ij} = \frac{Alg_{sol}(ij) - min_{sol}(j)}{min_{sol}(j)} \quad (12)$$

Where, $Alg_{sol}(ij)$ is objective function value for for test problem j by algorithm i and $min_{sol}(j)$ is the best value of the objective function between all algorithms for test problem j. The obtained results of ANOVA in Table 5 show that the algorithms are significantly different.

Table 2
Performance comparison: the improvement percentage and CPU time with N=6

T	*Problem no.*	*Budget type*	*Best Solution by Our GA-PSAA*	*Best Solution by Baykasoglu*	*Best Solution by Sahin et al*	*Improvement %*	*Average CPU Time (Min)*
5	P01	1	106,419	106,419	106,419	0.000	0.53
		2	106,419	106,419	106,419	0.000	0.425
		3	106,419	106,419	106,419	0.000	0.298
	P02	1	105,731	105,731	105,731	0.000	0.465
		2	105,731	106,802	105,731	0.000	0.232
		3	104,834	105,755	104,834	0.000	0.357
	P03	1	105,650	107,650	106,011	0.341	0.338
		2	106,113	108,113	107,609	1.390	0.334
		3	105,977	106,977	105,762	-0.203	0.468
	P04	1	107,124	108,260	106,583	-0.508	0.464
		2	107,424	109,474	107,984	0.519	0.456
		3	106,225	109,995	106,906	0.637	0.245
	P05	1	106,874	108,188	106,328	-0.514	0.213
		2	107,545	108,669	107,870	0.301	0.222
		3	107,813	106,834	106,328	-1.397	0.309
	P06	1	105,308	107,765	104,315	-0.952	0.211
		2	106,874	108,588	107,698	0.765	0.273
		3	107,132	109,568	104,262	-2.753	0.332
	P07	1	108,411	108,114	107,406	-0.936	0.169
		2	108,411	108,114	108,114	-0.275	0.141
		3	107,521	107,521	106,439	-1.017	0.181
	P08	1	108,306	107,248	107,248	-0.986	0.299
		2	108,603	108,603	107,248	-1.263	0.451
		3	107,724	108,603	107,248	-0.444	0.417
	P09	1	221,587	223,017	220,367	-0.554	0.531
		2	218,776	220,776	220,776	0.906	0.542
		3	217,580	221,010	217,251	-0.151	0.407
10	P10	1	214,430	217,412	217,106	1.233	0.469
		2	217,142	217,412	217,201	0.027	0.418
		3	210,318	217,412	212,134	0.856	0.293
	P11	1	215,278	219,024	214,960	-0.148	0.393
		2	217,934	219,024	215,622	-1.072	0.331
		3	212,733	219,024	215,393	1.235	0.273
	P12	1	213,321	217,350	216,828	1.617	0.188
		2	213,714	217,350	216,828	1.436	0.228
		3	210,214	217,350	216,828	3.050	0.468
	P13	1	216,026	217,142	211,960	-1.918	0.309
		2	216,505	217,142	213,304	-1.501	0.331
		3	215,630	219,835	211,620	-1.895	0.373
	P14	1	213,237	217,397	212,641	-0.280	0.625
		2	211,831	217,597	213,430	0.749	0.49
		3	216,515	217,397	213,424	-1.448	0.403
	P15	1	217,967	219,788	217,460	-0.233	0.28
		2	218,196	219,788	218,794	0.273	0.253
		3	215,088	219,788	214,823	-0.123	0.351
	P16	1	215,676	220,144	220,144	2.030	0.641
		2	219,558	220,055	220,144	0.226	0.487
		3	212,617	221,839	219,177	2.993	0.404
Average			161,176	163,331	161,357	0.033	0.360

Table 3
Performance comparison: the improvement percentage and CPU time with N=15

T	*Problem no.*	*Budget type*	*Best Solution by Our GA-PSAA*	*Best Solution by Baykasoglu*	*Best Solution by Sahin et al*	*Improvement %*	*Average CPU Time (Min)*
5	P17	1	475,893	512,046	481,675	1.2004	0.534
		2	479,752	517,302	481,682	0.4007	0.707
		3	489,084	512,481	480,453	-1.7964	0.628
	P18	1	489,289	500,284	484,799	-0.9262	0.539
		2	504,601	523,397	490,290	-2.9189	0.523
		3	489,963	525,379	486,726	-0.6651	0.658
	P19	1	503,344	508,011	489,583	-2.8108	0.581
		2	485,273	529,629	493,018	1.5709	0.622
		3	493,299	523,918	489,450	-0.7864	0.503
	P20	1	484,416	503,699	484,786	0.0763	0.500
		2	490,966	526,131	489,912	-0.2151	0.692
		3	479,566	519,095	484,954	1.1110	0.575
	P21	1	476,411	502,622	488,262	2.4272	0.537
		2	476,536	520,556	487,935	2.3362	0.590
		3	463,429	519,407	487,822	5.0004	0.565
	P22	1	473,903	499,891	486,493	2.5879	0.499
		2	479,353	520,721	488,199	1.8120	0.715
		3	476,553	517,139	487,360	2.2175	0.661
	P23	1	482,924	502,919	478,000	-1.0301	0.633
		2	466,539	521,767	487,007	4.2028	0.493
		3	466,539	515,327	486,801	4.1623	0.624
	P24	1	496,552	507,970	491,080	-1.1143	0.616
		2	490,567	530,135	494,369	0.7691	0.65
		3	483,419	522,490	491,237	1.5915	0.534
10	P25	1	982,740	1,039,960	981,531	-0.1232	0.988
		2	982,577	1,061,535	985,031	0.2491	0.903
		3	975,587	1,032,807	979,638	0.4135	0.869
	P26	1	976,290	1,022,447	979,655	0.3435	0.923
		2	984,131	1,041,725	981,478	-0.2703	1.173
		3	966,134	1,028,099	977,462	1.1589	1.159
	P27	1	976,625	1,068,402	984,103	0.7599	0.824
		2	978,507	1,068,886	993,049	1.4644	1.148
		3	978,538	1,036,409	983,112	0.4653	1.097
	P28	1	964,050	1,054,997	971,759	0.7933	1.253
		2	972,090	1,048,028	974,385	0.2355	0.914
		3	972,110	1,030,672	974,792	0.2751	0.987
	P29	1	984,539	1,051,395	978,456	-0.6217	0.920
		2	979,570	1,059,084	980,346	0.0792	1.173
		3	979,004	1,034,928	978,748	-0.0262	1.085
	P30	1	976,065	1,057,543	970,024	-0.6228	0.813
		2	970,152	1,061,751	972,765	0.2686	0.971
		3	970,040	1,036,017	970,435	0.0407	1.287
	P31	1	966,958	1,037,066	978,549	1.1845	1.142
		2	983,417	1,049,658	990,976	0.7628	0.981
		3	983,721	1,038,597	979,339	-0.4474	0.951
	P32	1	977,441	1,040,450	985,001	0.7675	1.196
		2	979,124	1,062,510	986,493	0.7470	1.057
		3	976,541	1,040,019	985,817	0.9409	1.134
Average			729,878	780,944	733,642	0.5842	0.815

Table 4
Performance comparison: the improvement percentage and CPU time with N=30

T	*Problem no.*	*Budget type*	*Best Solution by Our GA-PSAA*	*Best Solution by Baykasoglu*	*Best Solution by Sahin et al*	*Improvement %*	*Average CPU Time (Min.)*
5	P33	1	583,575	610,903	577,086	-1.1244	4.430
		2	584,933	606,465	579,704	-0.9020	4.280
		3	574,557	612,039	577,493	0.5084	3.240
	P34	1	560,525	576,350	571,846	1.9797	3.159
		2	571,685	608,461	572,396	0.1242	4.564
		3	601,895	607,954	570,537	-5.4962	5.564
	P35	1	566,321	586,831	579,113	2.2089	2.770
		2	569,503	614,621	579,406	1.7092	4.588
		3	556,226	613,072	574,225	3.1345	5.266
	P36	1	556,489	584,264	572,964	2.8754	6.006
		2	556,432	611,421	578,631	3.8365	3.674
		3	549,499	608,004	569,880	3.5764	4.265
	P37	1	555,513	570,492	559,934	0.7896	4.292
		2	553,472	607,299	559,078	1.0027	5.248
		3	556,096	599,439	559,506	0.6095	3.961
	P38	1	565,603	572,782	569,457	0.6768	2.558
		2	560,861	605,752	567,166	1.1117	3.061
		3	555,487	599,782	567,749	2.1598	5.722
	P39	1	572,862	571,703	569,470	-0.5956	3.678
		2	556,563	605,413	570,521	2.4465	3.962
		3	559,847	608,664	569,382	1.6746	4.661
	P40	1	556,186	596,744	579,411	4.0084	4.849
		2	568,469	618,022	586,310	3.0429	4.688
		3	560,413	614,882	577,719	2.9956	3.776
10	P41	1	1,159,518	1,218,971	1,171,634	1.0341	25.176
		2	1,158,313	1,228,403	1,172,520	1.2117	24.431
		3	1,126,384	1,223,520	1,171,500	3.8511	18.211
	P42	1	1,139,049	1,182,286	1,174,896	3.0511	17.634
		2	1,137,455	1,229,328	1,175,998	3.2775	25.788
		3	1,151,297	1,222,370	1,177,009	2.1845	31.726
	P43	1	1,157,704	1,188,620	1,169,208	0.9839	15.454
		2	1,150,810	1,229,156	1,179,660	2.4456	25.808
		3	1,136,389	1,224,383	1,164,129	2.3829	30.315
	P44	1	1,136,386	1,198,487	1,151,468	1.3098	34.210
		2	1,142,555	1,216,632	1,152,874	0.8951	20.778
		3	1,157,014	1,219,502	1,147,234	-0.8525	24.826
	P45	1	1,122,303	1,198,674	1,127,044	0.4207	24.413
		2	1,143,254	1,228,490	1,141,881	-0.1202	29.814
		3	1,125,378	1,237,846	1,129,703	0.3828	22.259
	P46	1	1,126,650	1,202,033	1,146,000	1.6885	14.189
		2	1,150,520	1,241,012	1,154,691	0.3612	16.970
		3	1,142,722	1,245,548	1,145,858	0.2737	32.450
	P47	1	1,219,744	1,210,573	1,210,573	-0.7576	20.403
		2	1,180,024	1,249,389	1,210,573	2.5235	22.326
		3	1,143,393	1,247,708	1,210,573	5.5494	26.669
	P48	1	1,202,789	1,209,088	1,189,154	-1.1466	27.392
		2	1,159,733	1,245,145	1,201,885	3.5072	26.686
		3	1,175,424	1,240,958	1,181,360	0.5025	27.259
Average			858,287	911,447	870,758	1.4023	14.321

Table 5
ANOVA results

Source	Degrees of freedom	Sum of squares	Mean square	F-Test	P-value
Algorithm	2	1959.11	979.557	236	0.0000
Error	429	1776.54	4.141		
Total	431	2996.87			

Therefore, in order to statistically rank the algorithms, it is necessary to use Tukey test. Table 6 shows the results of Tukey test based on the proposed GA-PSAA statistically outperforms other algorithms. Also, SA proposed by Sahin et al. (2010) is statistically better than ACO proposed by Baykasoglu et al. (2006).

Table 6
Tukey test of pairwise comparison of the algorithms

Algorithms	Lower	Upper	Significant difference
GA-PSAA & ACO	4.261	5.383	YES
GA-PSAA & SA	0.127	1.250	YES
SA & ACO	-4.695	-3.572	YES

5.2. Sensitivity analyses of the algorithms

In order to evaluate the effect of problem size (number of facilities and periods) on the quality of algorithms, this section provides some extensive experiment. By using the two-way ANOVA technique the 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 estimated and shown in Figures 6-7. As shown in Fig. 6, the increase in *N* factor leads to a tremendous decrement in the Baykasoglu's algorithm quality. But the SA by Şahin et al. (2010) and our GA-PSAA are highly robust against the increasing of the number of facilities. For N=6 both SA and GA-PSAA perform similarly. Nevertheless, the performance of GA-PSAA is statistically better than SA algorithm for N=15 and N=30.

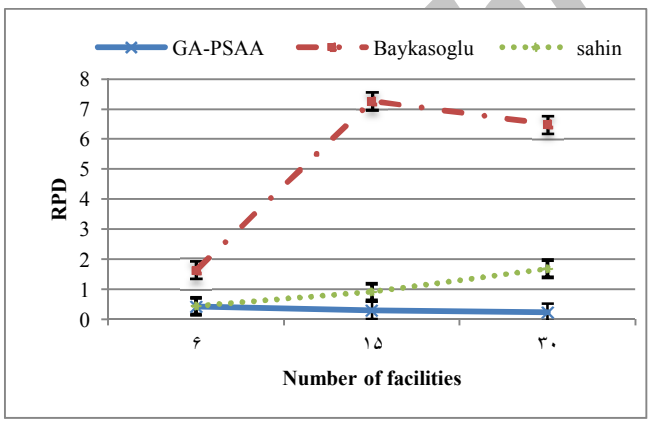

Fig. 6. Means plot and LSD intervals for interaction between the type of algorithm and number of facilities

Fig 7 makes a comparison in different number of periods. According to the Fig 7, 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 GA-PSAA can achieve better outcomes especially for T=10.

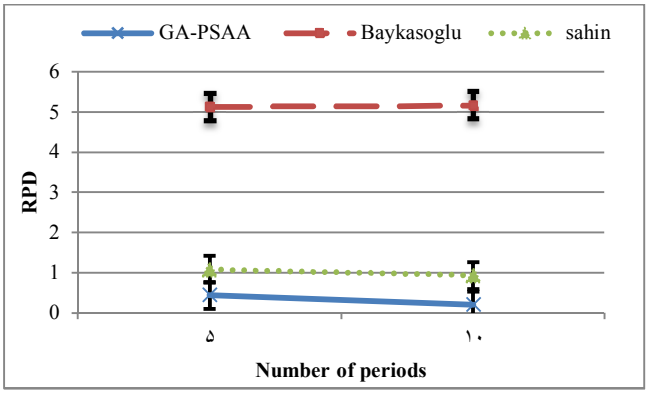

Fig. 7. Means plot and LSD intervals for interaction between the type of algorithm and number of periods

6. Conclusion and Suggestions for Future Works

In this study, an effective hybrid algorithm named GA-PSAA was developed to solve dynamic facility layout problem with budget constraint (DFLPBC). By using the advantages of genetic algorithm and parallel simulated annealing algorithm we could achieve better results compared with the previous studies. Performance of the proposed algorithm was tested for forty-eight instances taken from the literature. As the results show, the proposed algorithm can improve the objective function values, especially in large-sized problems. The following suggestions can be considered in future works:

- Material handling equipments and their maintenance time and cost can be considered.
- The money time value in different periods can be considered to model DFLPBC.
- Considering that the size of departments are not equal.
- A different chromosome structure can be used in a new hybrid algorithm.
- The budget constraint can include fuzzy inputs to model DFLPBC.
- Implementation of proposed method on other combinatorial optimization problems.

7. References

[1] Afentakis P, Millen R, Solomon MM. (1990). Dynamic layout strategies for flexible manufacturing systems. International Journal of Production Research;28:311–23.

[2] Balakrishnan, J., Cheng, C. H. (1998). Dynamic layout Algorithms: A state-of-the art survey. Omega.264, 507–521.

[3] Balakrishnan, J., Cheng, C. H. (2000). Genetic search And the dynamic layout problem: An improved algorithm. Computers and Operations Research 276, 587–593.

[4] Balakrishnan, J., Cheng, C.H., Daniel, G. Conway, A., Lau, C.M. (2003). A hybrid genetic algorithm for the dynamic plant layout problem. International Journal of Production Economics.86, 107–120.

[5] Balakrishnan, J., Cheng, C.H. (2009). The dynamic plant layout problem: Incorporating rolling horizons and forecast uncertainty. Omega, 37, 165 – 177.
[6] Balakrishnan, J., Jacobs, F. R., Venkataramanan, M. A. (1992). Solutions for the constrained dynamic facility layout problem. European Journal of Operational Research, 57(2), 280–286.
[7] Baykasoglu, A., Gindy, N. N. Z. (2001). A simulated annealing algorithm for dynamic facility layout problem. Computers and Operations Research. 2814, 1403–1426.
[8] Baykasoglu, A., Dereli, T., Sabuncu, I. (2006). An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems. Omega, 34(4), 385–396.
[9] Dong, M., Wua, C., Hou, F. (2009). Shortest path based simulated annealing algorithm for dynamic facility layout problem under dynamic business environment. Expert Systems with Applications.36, 11221–11232.
[10] Drira, A, Pierreval, H., Hajri-Gabouj, H. (2007). Facility layout problems: A survey. Annual Reviews in Control. 31, 255–267.
[11] Dunker, T., Radons, G., Westkamper, E. (2005). Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. European Journal of Operational Research. 1651, 55–69.
[12] Erel, J.B., Ghosh, J., Simon, J.T. (2005). New heuristic for the dynamic layout problem. Journal of the Operational Research Society.568, 1001.
[13] Gary, Y., Chen, K.J., Rogers, A. (2009). Multi-objective evaluation of dynamic facility layout using ant colony optimization. Proceedings of the 2009 Industrial Engineering Research Conference.
[14] Kaku, B., Mazzola, J. B. (1997). A tabu search heuristic for the plant layout problem. INFORMS Journal on Computing. 94, 374–384.
[15] Kouvelis, P., Kurawarwala, A. A., Gutierrez, G. J. (1992). Algorithms for robust single and multiple periods layout planning for manufacturing systems. European Journal of Operations Research. 632, 287–303.
[16] Krishnan, K.K., Cheraghi, S.H., Nayak, C. N. (2006). Dynamic from-between charts: A new tool for solving dynamic facility layout problems. International Journal ofIndustrial and Systems Engineering. 11/2, 182–200.
[17] Kulturel-Konak, S. (2007). Approaches to uncertainties in facility layout problems: Perspectives at the beginning of the 21st century. Journal of Intelligent Manufacturing. 182, 273–284.
[18] Lacksonen, T.A., Enscore, E.E. (1993). Quadratic assignment algorithms for the dynamic layout problem. International Journal of Production Research.313, 503–17.
[19] McKendall, A.R, Shang, J. (2006). Hybrid ant systems for the dynamic facility layout problem .Computers & Operations Research. 333, 790–803.
[20] McKendall, A.R., Shang, J., Kuppusamy, S. (2006). Simulated annealing heuristics for the dynamic facility layout problem, Computers & Operations Research. 33, 2431–2444.
[21] McKendall A.R., Hakobyan, A. (2010). Heuristics for the dynamic facility layout problem with unequal-area departments. European Journal of Operational Research. 201, 171–182.
[22] Mahmoudi, A., Shavandi,H., Nouhi, KH.,(2012). Analysing Price, Quality and Lead Time Decisions with the Hybrid Solution Method of Fuzzy Logic and Genetic Algorithm, Journal of Optimization in Industrial Engineering, 5(10), 1-9
[23] Naderi, B., Sadeghi, H. (2011). A Multi-objective Simulated Annealing Algorithm for Solving the Flexible no-wait Flowshop Scheduling Problem with Transportation Times. Journal of Optimization in Industrial Engineering. 5(11), 33-41
[24] Page, A.L. (1991). New product development survey:Performance and best practices. PDMA conference.
[25] Rodriguez, J.M., MacPhee, F.C., Bonham, D.J., Bhavsar, V.C. (2006). Solving the dynamic plant layout problem using a new hybrid meta-heuristic algorithm. International Journal of High Performance Computing and Networking. 45/6, 286–294.
[26] Rosenblatt M.J. (1986). The dynamics of plant layout. Management Science. 321, 76–86.
[27] Sahin, R., Ertogral, K., Turkbey, O. (2010). A simulated annealing heuristic for the dynamic facility layout problem with budget constraint. Computers & Industrial Engineering, 59, 308-313.
[28] Tompkins, J., White, J., Bozer, Y., Tanchoco, J. (2003). Facilities planning.third ed. John Wiley & Sons, New Jersey.
[29] Urban, T. L. (1998). Solution procedures for the dynamic facility layout problem. Annals of operations research.761, 323–342.
[30] Zoraghi, N., Najafi, AA., Niaki STA. (2012). An Integrated Model of Project Scheduling and Material Ordering: A Hybrid Simulated Annealing and Genetic Algorithm. Journal of Optimization in Industrial Engineering, 5(19), 19-27.