Professor Parham AZIMI, PhD
Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Qazvin, Iran
E-mail: p.azimi@yahoo.com
E. SABERI, MSc.
Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Qazvin, Iran
E-mail: Sab_sml@yahoo.com

AN EFFICIENT HYBRID ALGORITHM FOR DYNAMIC FACILITY LAYOUT PROBLEM USING SIMULATION TECHNIQUE AND PSO

Abstract. *In this research, a new hybrid heuristic method is developed by combination of discrete Particle Swarm Optimization (PSO) algorithm and simulation technique to address Dynamic Facility Layout Problem (DFLP) which is the main contribution of the current study. DFLP is a NP-Hard problem, so there are several researches focused on heuristic algorithms to solve the problem. The proposed approach is the only research which uses simulation technique inside the PSO algorithm to enable it terminated faster. To show the efficiency of the proposed algorithm, several test problems have been examined and the results were compared to other algorithms. As the computational results show, the quality of solutions and the algorithm speed are suitable enough to be used in real world problems.*

Keywords: *Meta-heuristic algorithms, Particle Swarm Optimization (PSO), Discrete Event Simulation, Dynamic Facility Layout Problem (DFLP).*

JEL classification: C61, C15

1. Introduction

In today markets, one of the basic and important strategies in business environments for improving the productivity of a company is the cost reduction plan. So investigation of manufacturing cost problems in organization environments to achieve their predefined objectives is unavoidable. Facility Layout Problem (FLP) is one of the major problems in manufacturing companies which has direct effect on material handling costs. Specifying an arrangement of the facilities in working area, often referred as “Facility Layout Problem” [1]. Because of rapid changes and improvements in nowadays technology, and customer demands, it is desirable to have a plan for upcoming changes by developing a flexible layout. The term flexibility means the capability to easily modify, expand or reduce facility layout in the plant area [1]. For example, 75% of HP's products are less than 3 years old and this percentage increasing [2]. So if this company could not adapt with volatile conditions of global market competition, it has to be breakage. The main cost that related with facility layout is material handling costs. These costs are determined based on the amount of materials that flow between the facilities and the distance between locations of them [3].So considering to layout problem and represent an efficient layout for a planning horizon, help us to minimize material handling or flow cost. To realize the importance of this cost, Tompkins reported that in manufacturing organization between 15-70% of total expenses related to material handling cost , but with an effective layout this percentage reduce to 10-30% [1]. In classic layout problem or static facility layout problem (SFLP), usually some factors such as rapid demand change and short lifecycle did not have change. So significant part of SFLP cost i.e. material handling cost have no change. As a matter of fact SFLP is a plan for arranging the physical facilities within an area to achieve cost reduction objectives [3]. Layouts like SFLP that often material flow cost considered as a constant value, only studied in one period. But in competitive and volatile market conditions, material flow between departments may change during a planning horizon, the layouts comes to dynamic, and this problem is known as the dynamic facility layout problem (DFLP). As a result, the solution for the SFLP is a single layout, and the solution for the DFLP is a series of layouts [17].

2. Literature review

The first citation about facility layout problem was performed by Rosenblatt in 1978 [4]. Also he was the first researcher who developed a model for solving DFLP. He used dynamic programming algorithm to solve DFLP [17]. An algorithm for the single and multiple period dynamic layout problem is presented by Kouvelis in 1991. Although this method considers additional factors such as buffer space and changeover costs, it is computationally intractable in the multiple period case [7]. Balakrishnan (1993) proposed the fathoming procedure for the DFLP model developed by Rosenblatt (1986) [8]. Lacksonen (1994) proposed a two step algorithm which takes in to consideration the varying areas to solve DFLP. The first part of the problem uses QAP whereas, the second part uses mix integer programming to generate layouts [9]. Kaku and Mazzola (1997) made an attempt to consider tabu search heuristic to solve DFLP [10]. Balakrishnan and Cheng (2000) presented a genetic algorithm (GA) to solve the DFLP [11]. Baykasoglu and Gindy (2001) represented the SA based approach to test and solve the DLP. The proposed algorithm is tested using the data sets presented by Balakrishnan and Cheng (2000) and the results yield significant improvement compared to the existing one [12]. Erel et al. (2003) proposed a new heuristic to solve DFLP. They used weighted flow data from the various time periods to developed viable layouts, and they suggested a shortest path for the DFLP [13]. Corry and kozan (2004) discussed the dynamic reallocation of the departments to optimize the trade-off between material handling and rearrangement costs under similar conditions [14]. Enea et al. (2005) proposed a fuzzy model and a genetic search to solve facility layout problem [15]. Baykasoglu and Gindy developed an ant colony algorithm for solving the unconstrained and budget constrained dynamic layout problems [16]. McKendall et al. (2006) presented two SA heuristics. The first SA heuristic (SA I) is a direct adaptation of SA to the DFLP The second one (SAII) is the same as SA I, except that it has an added look-ahead/look-back strategy [17]. Drira et al. proposed a structure to analyze and survey facility layout problems [18]. Liang and Chao (2008) represented strategies of tabu search technique for facility layout optimization [19]. Sahin and Turkbey (2008) developed a new hybrid tabu-simulated annealing heuristic for the dynamic facility layout problem [20]. Dong et al. (2009) considered adding or removing department in DFLP using simulated annealing (SA) algorithm [21]. Sahin et al. proposed a simulated annealing heuristic for the dynamic layout problem with budget constraint [22]. McKendall and Hakobyan (2010) have presented a paper considering unequal-size departments, and developed SA algorithm for solving DFLP [23]. Azimi and Salehi (2011) used simulation-based heuristic for the DFLP [24].

3. Problem statement and mathematical model

The DFLP is the problem of finding positions of departments on the plant floor for multiple periods such that departments do not overlap, and the sum of the material handling and rearrangement costs is minimized. In other words, for each period in the planning horizon, the layout is determined such that the sum of the material handling cost for each layout and the cost of rearranging departments between each pair of consecutive layouts is minimized [23]. The quadratic assignment problem (QAP) for the DFLP that obtained from the Balakrishnan paper (1992) is presented as follows [29] :

Problem1:

s.t.

T:	The number of periods in the planning horizon,
N:	Both the number of departments and the locations,
i, k :	Notations of departments in the layout,
j , l :	Notations of locations in the layout,
A_{tij} :	Cost of shifting department i between locations j and l in period t,
f_{tik} :	Flow cost per unit distance from department i to k in period t,
d_{tjl} :	Distance from location j to l in period t.

X_{tij} =

Y_{tijl} =

Objective function (1) is used to minimize the sum of the rearrangement and material handling costs. Constraint set (2) restricts that each location is assigned to only one department in each period, and constraint set (3) ensures that exactly one department is assigned to each location in each period Constraint set (4) adds the rearrangement costs to the material flow cost if a facility is shifted between locations in consecutive periods. Finally, the restrictions on the decision variables are given in constraint set (5).

4. Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a stochastic based search algorithm widely used to find the optimum solution introduced by Kennedy and Eberhart in 1995 [26] . PSO is a kind of simulation that is inspired from studies of social behavior among movement of fishes or flocking of birds. Now it has been widely used to solve non-linear and multi-objective optimization problems. PSO is an effective optimization technique to search for global optimized solution but time of convergence is uncertain. Like other population based optimization methods the particle swarm optimization starts with randomly initialized population for individuals [27]. PSO algorithm uses a population (swarm) of individuals (solutions) called of particles. Each particle has information about its own position, velocity and fitness. With this information, each particle updates its personal best (**pbest**: best value of each individual so far) if an improved fitness value was found. On the other hand, the best particle in the whole population with its position and fitness value is used to update the global best (**gbest**: best particle in the whole population). Then the velocity of the particle is updated by using its previous velocity, the experiences of the personal best, and the global best in order to determine the position of each particle. As a result, PSO particles in each iteration of algorithm have 4 important vectors **(** : j-th dimension of position in particle i , : j-th dimension of velocity in particle i , : j-th dimension of best position (pbest) in particle i , : j-th dimension of global best position (gbest) in particle i **)** . The new velocities and positions of the particles for next iterations can be evaluated by using the equations 6 and 7 .

The new velocity of the particle i, is updated using Inertia, cognition and social terms. The current position of the particle i, is updated using previous position and current velocity of the particle. w is the inertia weight which is a parameter to control the impact of the previous velocities on the current velocity [25]. C_1 is a positive constant, called as coefficient of the self-recognition component, C_2 is also a positive constant called as coefficient of the social component [25]. r_1 and r_2 are uniform random numbers between [0,1], used to accordance of algorithm with natural universe. The parameter w regulates the trade-off between global (wide ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration while a small one tends to facilitates global local exploration. A suitable value for the inertia weight w usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution. Initially, the inertia weight is set as a constant [25]. Thus, an initial value is around 1.2 and gradually reducing towards 0 can be considered as a good value for w. If $C_1 > C_2$, particles tend to convergence in a local optimum, if $C_1 < C_2$, dispersal of particles and the time of finding optimum solution increased. Most particle swarm optimization algorithms are designed to search in continuous domains. By a little change in PSO parameters, we can use this algorithm in discrete domains and called it DPSO. In this algorithm, a particle's position is discrete but its velocity is continuous. In the discrete version, a particle moves in a state space restricted to zero and one on each dimension, where each represents the probability of bit taking the value 1. For the velocity value of each bit in a particle, Kennedy and Eberhart claimed that the higher value is more likely to choose 1, while the lower value favors the 0 choice [28]. Thus, the step for updating remains unchanged as shown in Eq. (6), except that and are integers in {0, 1} in binary case. The resulted changes in position are defined as follows:

$$[t+1]= \qquad (8)$$

In Eq. (8), denotes the sigmoid function or probability of j-th dimension of taking 1, is a random uniform random number between [0,1].

Eq. (9) helps for obtaining values of : = 1/ (1 + exp (−)) (9)

5. Description and encoding of DFLP

Fig.1 illustrates structure of facilities allocation, assuming T time periods, M locations and M facilities in each period. In each period every facility has a chance to allocate in each location. The value of permutation of M facilities with M locations, in one period is M , and in T periods is M^{T}.

Period 1																	Period T															
Facilities																	Facilities															
1				2				…				M					1				2				…				M			
Locations				Locations				Locations				Locations					Locations				Locations				Locations				Locations			
1	2	…	M	1	2	…	M	1	2	…	M	1	2	…	M		1	2	…	M	1	2	…	M	1	2	…	M	1	2	…	M

Figure 1. Allocation state of facilities to locations in DFLP

Six DFLP categories which proposed by Balakrishnan [11] have been listed in table 1. For instance, in smallest size of proposed DFLP (5T6M) by Balakrishnan [11], solution space has 1.93+E14 numbers of feasible solutions. It is clear that in larger size of DFLP, solution spaces will be very larger than smallest one. So the researchers have to find heuristic methods such as, ACO, SA, TS, GA, and PSO to solve the problem. Fig.2 is one of the example solution of 5T6M DFLP size. From the solutions, it is clear that DFLP solutions are discrete, so planning a suitable discrete PSO (DPSO) for this problem is unavoidable.

Table1: DFLP Categories proposed by [11]

T : periods	5	5	5	10	10	10
M : facilities	6	15	30	6	15	30
Problem size	5T6M	5T15M	5T30M	10T6M	10T15M	10T30M

Solution space	1.93E+14	3.82E+60	1.31E+162	3.74E+28	1.46E+121	

	T1	T2	T3	T4	T5
L1	2	3	6	6	6
L2	3	5	5	3	5
L3	4	4	1	4	4
L4	1	1	3	1	1
L5	6	2	2	2	3
L6	5	6	4	5	2

Figure 2. Example solution of 5T6M

6. Solving DFLP by PSO algorithm

When optimization problems have to be solved in discrete domain, first of all, their particles should be defined as well as possible. General structure of particles in DFLP is similar to Fig.2. Binary allocation model of Fig.2 is depicted in Fig3. Rows in Fig.3 are locations and columns are facilities, respectively. If a facility assigned to a location, the corresponding value in the row and in the column takes 1, and the all other values take 0. This method will prevent creating of any infeasible solution; because that facility must not assigned to any other locations and also that location must not assigned to any facilities. The two important procedures in DPSO encoding, are velocity and position update of particles.

	T1					
	M1	M2	M3	M4	M5	M6
L1	0	1	0	0	0	0
L2	0	0	1	0	0	0
L3	0	0	0	1	0	0
L4	1	0	0	0	0	0
L5	0	0	0	0	0	1
L6	0	0	0	0	1	0

⇨

T					
M1	M2	M3	M4	M5	M6
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	1	0	0
1	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0

Figure 3. Binary allocation model of DFLP particles

	M1	M2	M3	M4	M5	M6
L1	0.95	0.45	0.19	0.21	0.46	0.65
L2	0.2	0.58	0.35	0.39	0.34	0.48
L3	0.61	0.38	0.84	0.51	0.23	0.16
L4	0.31	0.67	0.47	0.37	0.72	0.56
L5	0.85	0.05	0.16	0.93	0.89	0.77
L6	0.13	0.57	0.14	0.81	0.53	0.21

Figure 4. Sigmoid values of first period of Figure 3

	M1	M2	M3	M4	M5	M6
L1	0.3265	0.1546	0.0653	0.0722	0.1581	0.2234
L2	0.0855	0.2479	0.1496	0.1667	0.1453	0.2051
L3	0.2234	0.1392	0.3077	0.1868	0.0842	0.0586
L4	0.1	0.2161	0.1516	0.1194	0.2323	0.1806
L5	0.2329	0.0137	0.0438	0.2584	0.2438	0.211
L6	0.054	0.2385	0.0586	0.3389	0.2218	0.0879

Figure 5. Relative sigmoid values of Figure 4

As mentioned, velocity of particles represents the probability of an assignment of a facility to a location. Suppose that the sigmoid function is used on values of the first period in Fig.3, and Fig.4 is resulting table. Each number in Fig.4 denotes the probability of allocating a facility to other locations. For example 0.95 in Fig.4 represents that facility M1 has 95% chance to be assigned the location (L1). Next stage is updating of particles positions related with these sigmoid values. In this paper, sigmoid relative values which are obtained from Eq. (10). In this equation q_{ij} is the relative probability of allocating facility i to location j and is the sum of sigmoid values of facility i over all locations. The biggest q_{ij} in each column takes 1 (because it has the maximum level of assignment probability) and the others taking 0. Fig.5 represents relative sigmoid values of fig.4.

Therefore comparative binary allocation model of fig.5 is obtained and depicted in Fig.6.

7. PSO parameters tuning

This paper used linear regression method in order to find the best PSO parameter values. First of all for every 48 DFLP test problems taken from [24], randomly generated values for PSO parameters are used and the problem were solved by PSO algorithm, and then results were recorded. In the next step, by using a linear regression method between PSO parameters values and the results of the first step, some fitness functions have been generated. At the third some step linear programming models were developed using the fitness functions as objective functions. In all models, the constraints include some restrictions to guarantee the necessary for convergence of the PSO algorithm according to [27].

Problem (2):

st:

The best parameter values that are obtained from the average solutions of these models are as follows:

C_1=1.5 **C_2=2.5** **W_{max}=0.9999** **W_{min}=0.4** **Max _it=100**

	M1	M2	M3	M4	M5	M6
L1	1	0	0	0	0	0
L2	0	0.2479	0.1496	0.1667	0.1453	0.2051
L3	0	0.1392	0.3077	0.1868	0.0842	0.0586
L4	0	0.2161	0.1516	0.1194	0.2323	0.1806
L5	0	0.0137	0.0438	0.2548	0.2438	0.211
L6	0	0.2385	0.0586	0.3389	0.2218	0.0879

⇨

	M1	M2	M3	M4	M5	M6
L1	1	0	0	0	0	0
L2	0	1	0	0	0	0
L3	0	0	1	0	0	0
L4	0	0	0	0	1	0
L5	0	0	0	1	0	0
L6	0	0	0	0	0	1

Figure 6. Binary allocation

8. Constructing initial PSO population using a simulation model

Fig. 7 is a simple 3T3M simulation model according to DFLP that created by ED 8.1 Software. In this model, the first column represents facilities which are in red, blue and green colors. The “Source” atoms in the second column produce products in the simulation model which are in fact the facilities of DFLP. The number of products in the simulation model is the number of replications that we need. In this study all simulation models have replicated 3000 times. Because, we have three periods, columns 4,5 and 6 are designed to simulate the periods. In each period, we have three locations. All products will be assigned randomly (the probability of assigning a facility to a location is 1/3) to the locations which are indicated by “Server” atoms in brown. For ensuring about constraints (2) and (3) in problem1, We have coded the simulation model in a way that all facilities are assigned to the locations simultaneously in each period and just one facility is assigned to each location, and vice versa . In this way, whenever a replication terminates, a random feasible solution of problem 1 is generated. One of the advantages of creating the simulation model is that it generates the solutions in a very short time, because it does not depend on the simulation clock. For example, 3000 replications of this model take almost under 10 second.

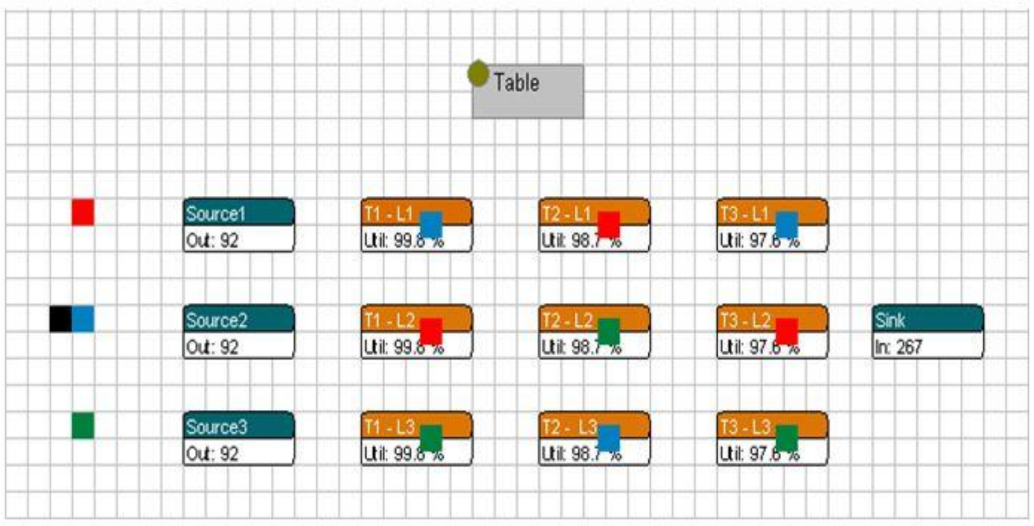

Figure 7: A simple DFLP with M=T=3

Finally 50 solutions which have the least objective function will be chosen as initial PSO population. Therefore, the simulation model will help PSO to start from potentially good solutions.

9. Computational Results

This section presents the computational results of the proposed DPSO algorithm applied to the 48 test problems according Table 1. We have used Enterprise Dynamics 8.1 as the simulation software and Matlab as the programming language for solving Problem 1, 2. All problems were run on a Pentium IV PC using core i5, 2.53GH, 6 GB RAM. In all experiments, we have used two algorithms. The first one which was denoted by DPSO, is a discrete PSO explained earlier and the second one which is denoted by HDPSO is a hybrid discrete PSO which uses simulation model results in the DPSO. All results were compared to the ones which were reported by McKendall et al. (2006). McKendall et al. (2006) used Simulated Annealing Algorithm (SAII) for DFLP and showed that their results were the best ones in comparison to other methods . Tables 2-7 summarize the results for each dataset. Column 1 is problem number. Columns 2, 3 and 4 are the best solutions of DPSO, HDPSO and SA II algorithms. Column 5 shows the improvement ratio of the best solutions comparison. Columns 6, 7 and 8-th are the average solutions of DPSO, HDPSO and SA II algorithms. Column 9 shows improvement ratio of the average solutions. Columns 10 and 11 and show the average running times in minutes. Last row in each table is average improvement ratio for best and average solutions.

Table 2: Computational results for 5T6M problems

	Best Solutions				Average Solutions				CPU time	
Problem No	DPSO	HDPSO	[17]	Dev (%)	DPSO	HDPSO	[17]	Dev (%)	HDPSO	[17]
1	105,066	106,518	106,419	-1.29	106,341	107,753	106,419	-0.07	0.13	0.17
2	108,054	104,924	104,834	0.09	110,841	104,974	104,834	0.13	0.13	0.17
3	105,103	104,364	104,320	0.04	106,571	104,408	104,320	0.08	0.13	0.17
4	106,399	106,443	106,399	0.00	106,482	109,900	106,399	0.08	0.13	0.17
5	106,874	105,628	105,628	0.00	108,554	105,668	105,628	0.04	0.13	0.17
6	104,262	104,053	103,985	0.07	106,759	104,053	103,985	0.07	0.13	0.17
7	107,834	105,237	106,439	-1.14	109,454	105,324	106,439	-1.06	0.13	0.17
8	106,095	103,876	103,771	0.10	108,877	104,021	103,771	0.24	0.13	0.17
Average				-0.27				-0.06		

Table 3: Computational results for 10T6M problems

	Best Solutions				Average Solutions				CPU time	
Problem No	DPSO	HDPSO	[17]	Dev (%)	DPSO	HDPSO	[17]	Dev (%)	HDPSO	[17]
9	215,204	213,805	214,313	-0.24	217,186	214,313	214,313	0.00	0.25	0.33
10	212,134	212,134	212,134	0.00	212,222	212,812	212,136	0.04	0.25	0.33
11	208,598	207,230	207,987	-0.37	209,784	207,926	208,070	-0.07	0.25	0.33
12	213,883	212,741	212,741	0.00	216,638	212,803	212,741	0.03	0.25	0.33
13	210,006	209,976	210,906	-0.44	211,728	210,926	210,974	-0.02	0.25	0.33
14	210,507	209,932	209,932	0.00	212,836	210,029	209,944	0.04	0.25	0.33
15	214,252	213,414	214,252	0.00	215,835	214,176	214,252	-0.04	0.25	0.33
16	213,657	212,588	212,588	0.00	215,245	212,588	212,588	0.00	0.25	0.33
Average				-0.13				-0.002		

Table 4: Computational results for 5T15M problems

	Best Solutions				Average Solutions				CPU time	
Problem No	DPSO	HDPSO	[17]	Dev (%)	DPSO	HDPSO	[17]	Dev (%)	HDPSO	[17]
17	481,094	480,604	480,496	0.02	481,377	481,565	481,621	-0.05	1.9	2.5
18	484,790	484,013	484,761	-0.15	486,235	485,100	485,388	-0.06	1.9	2.5

19	488,917	488,473	488,748	-0.06	490,044	489,811	490,053	-0.05	1.9	2.5
20	485,656	485,063	484,414	0.13	488,407	485,063	484,995	0.01	1.9	2.5
21	488,509	487,911	487,911	0.00	489,139	488,495	488,579	-0.02	1.9	2.5
22	487,414	487,214	487,147	0.01	487,628	487,717	487,814	-0.02	1.9	2.5
23	487,553	487,247	486,779	0.10	487,890	488,200	487,311	0.12	1.9	2.5
24	489,958	489,660	490,812	-0.24	490,921	491,194	491,620	-0.14	1.9	2.5
Average				-0.02				-0.03		

Table 5: Computational results for 10T15M problems

	Best Solutions				Average Solutions				CPU time	
Problem No	DPSO	HDPSO	[17]	Dev (%)	DPSO	HDPSO	[17]	Dev (%)	HDPSO	[17]
25	978,512	978,345	979,468	-0.11	978,552	978,474	981,528	-0.31	3.1	3.96
26	977,720	977,852	978,065	-0.04	978,612	978,676	978,630	0.00	3.1	3.96
27	982,517	982,396	982,396	0.00	982,645	982,581	984,235	-0.17	3.1	3.96
28	974,123	973,987	972,797	0.12	974,223	974,141	975,802	-0.17	3.1	3.96
29	977,694	977,909	978,067	-0.04	977,784	977,944	978,904	-0.11	3.1	3.96
30	968,004	967,847	967,617	0.02	971,017	970,818	970,553	0.03	3.1	3.96
31	979,259	978,083	979,114	0.01	981,641	979,940	981,196	0.05	3.1	3.96
32	983,022	982,973	983,672	-0.07	983,110	983,104	985,459	-0.24	3.1	3.96
Average				-0.01				-0.12		

Table 6: Computational results for 5T30M problems

	Best Solutions				Average Solutions				CPU time	
Problem No	DPSO	HDPSO	[17]	Dev (%)	DPSO	HDPSO	[17]	Dev (%)	HDPSO	[17]
33	576,772	576,700	576,741	-0.01	576,783	576,824	577,585	-0.14	3.3	4.4
34	568,006	567,929	568,095	-0.03	568,497	569,546	569,613	-0.01	3.3	4.4
35	574,638	574,596	574,036	0.10	575,307	576,195	576,074	-0.13	3.3	4.4
36	566,200	565,983	566,248	-0.05	566,374	566,190	567,297	-0.20	3.3	4.4
37	558,399	558,460	558,460	0.00	558,686	558,570	560,148	-0.28	3.3	4.4
38	566,911	566,743	566,597	0.03	567,007	566,815	567,215	-0.07	3.3	4.4
39	568,466	567,296	568,204	-0.16	568,637	567,394	569,469	-0.37	3.3	4.4
40	573,747	573,710	573,755	-0.01	573,793	574,368	575,270	-0.16	3.3	4.4
Average				-0.02				-0.17		

Table 7: Computational results for 10T30M problems

	Best Solutions				Average Solutions				CPU time	
Problem No	DPSO	HDPSO	[17]	Dev (%)	DPSO	HDPSO	[17]	Dev (%)	HDPSO	[17]
41	1,163,340	1,162,306	1,163,222	-0.08	1,163,450	1,162,348	1,164,101	-0.15	7.1	7.8
42	1,161,679	1,158,539	1,161,521	-0.26	1,160,520	1,160,580	1,164,108	-0.31	7.1	7.8
43	1,158,413	1,158,264	1,156,918	0.12	1,158,991	1,158,644	1,158,855	-0.02	7.1	7.8
44	1,145,968	1,146,103	1,145,918	0	1,148,884	1,147,040	1,148,331	-0.11	7.1	7.8
45	1,127,269	1,124,892	1,127,136	0	1,126,408	1,126,340	1,130,036	-0.33	7.1	7.8
46	1,145,296	1,147,028	1,145,146	0.01	1,147,725	1,147,517	1,145,777	0.15	7.1	7.8
47	1,140,866	1,139,585	1,140,744	-0.10	1,139,985	1,140,167	1,140,744	-0.07	7.1	7.8
48	1,163,080	1,161,437	1,161,437	0	1,162,313	1,162,345	1,166,384	-0.35	7.1	7.8
Average				-0.06				-0.15		

According to the computational results, HDPSO best solutions are better in 20 cases, 13 cases are equal and 15 cases are worse than SA II. Average solutions in 31 cases are better, 3 cases are equal and 14 cases are worse than SA II. Another criterion which is usual in heuristic algorithms comparisons is the average running times (CPU time). As the results show, HDPSO CPU times in all cases better than [17]. In summary, regarding the best solutions, HDPSO is -0.08% and for the average solutions is -0.09% better than [17].

10. Conclusions

DFLP is a combinatorial optimization problem which has some applications in real world problems. In this research, two discrete PSO-based heuristic algorithms were developed to solve the DFLP. The main idea of this research was improving the ability of PSO heuristic algorithm by utilizing the advantages of simulation modeling. In the first approach the DFLP was solved with DPSO and the results were recorded. In the second approach (HDPSO), a simulation model was designed and formulated and then 50 best solutions out of 3000 replications constructed the initial population of DPSO algorithm. Finally the first and the second approaches were compared to each other and also compared to some previous works. It was shown that HDPSO performs better than DPSO and also, the proposed HDPSO has very good computational efficiency when the problem size increases. Some recommendations for future works may be including the budget constraints into DFLP model or DFLP can be modeled and solved in a fuzzy environment.

REFERENCES

[1] **Tompkins, J. A., White, J. A., Bozer, Y. A., Frazelle, E. H., Tanchoco, J. M. &Trevino, J. (1996),** ***Facilities planning***; *New York: Wiley;*
[2] **Hammer, M. (1996),** ***Beyond Reengineering*;** *Harper Collins, New York;*
[3 Rezazadeh H, Ghazanfari M, Saidi Mehrabad M, Sadjadi Seyed Jafar. (2009), ***An Extended Particle Swarm Optimization Algorithm for Dynamic Facility Layout Problem***; *Journal of Zhejiang University Science A, 10(4):520-529;*
[4] **Rosenblat, M. J., and Lee, H. L. (1987),** ***A Robustness Approach to Facilities Design***; *International Journal of Production Research, 2 , 4, 479-486;*
[5] **Rosenblatt, M. J. (1986),** ***The Dynamics of Plant Layout*.** *Management Science; 32(1), 76–86;*
[6] **Benjaafar, S., Heragu, S. and Irani, S. (2002),** ***Next Generation Factory Layouts: Research Challenges and Recent Progress***; *Interfaces, 32(6), 58–76;*
[7] **Kouvelis, P., Kiran, A.S. (1991),** ***Single and Multiple Period Layout Models for Automated Manufacturing Systems****; European Journal of Operations Research 52(3):300–314;*
[8] **Balakrishnan, J. (1993),** ***Notes: The Dynamic of Plant Layout*.** *Management Science; 39, 5, 654-655;*
[9] **Lacksonen, T.A. (1994),** ***Static and Dynamic Layout Problems with Varying Areas***; *Journal of the Operational Research Society 1994;45:59–69;*
[10] **Kaku BK, Mazzola J.B. (1997),** ***A Tabu Search Heuristic for the Dynamic Plant Layout Problem****; INFORMS: Journal on Computing 1997;9:374–84;*
[11] **Balakrishnan, J., Cheng, CH. (2000),** ***Genetic Search and the Dynamic Layout Problem****; Computers and Operations Research, 27:587–93;*
[12] **Baykasoglu A, NNZ. (2001),** ***A Simulated Annealing Algorithm for Dynamic Layout Problem****; Computers and Operations Research, 28:1403–26;*
[13] **Erel E, Ghosh JB, Simon JT. (2003),** ***New Heuristic for the Dynamic Layout Problem****; Journal of the Operational Research Society,54:1202–75;*
[14] **Corry, P. and Kozen, E. (2004),** ***Ant Colony Optimization for Machine Layout Problems****; Computational optimization and applications, 28, 3, 287-310;*
[15] **Enea, M. et al. (2005),** ***The Facility Layout Problem Approach Using Fuzzy Model and a Genetic Search****; Journal of Intelligence Manufacturing I, 16, 303-316;*
[16] **Baykasoglu, A., Dereli, T. & Sabuncu, I. (2006),** ***An Ant Colony Algorithm for Solving Budget Constrained and Unconstrained Dynamic Facility Layout Problems****; Omega,2006;34(4), 385–396;*

[17] **McKendall, A. R. Jr., Shang, J. & Kuppusamy, S. (2006),** ***Simulated Annealing Heuristics for the Dynamic Facility Layout Problem****; Computers and operations Research,33(8), 2431–2444;*

[18] **Drira , A., Pierreval , H., Hajri-Gabouj, S. (2007),** ***Facility Layout Problems: A survey****; Annual Reviews in Control 31, 255–267;*

[19] **Şahin, R., Türkbey, O. (2008),** ***A New Hybrid Tabu-simulated Annealing Heuristic for the Dynamic Facility Layout Problem***; *International Journal of Production Research, 45, 1–19;*

[20] **Liang, L. Y., Chao, W. C. (2008),** ***The Strategies of Tabu Search Technique for Facility Layout Optimization****; Automation in Construction 17, 657–669;*

[21] **Dong, M., Wu, 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;*

[22] **Sahin, R., Ertogral, K., Turkbey, O. (2010),** ***A Simulated Annealing Heuristic for the Dynamic Layout Problem with Budget Constraint***; *Journal of Computer & Industrial Engineering, 59, 308-31;*

[23] **McKendall A.R. and Hakobyan, A. (2010),** ***Heuristics for the Dynamic Facility Layout Problem with Unequal-area Departments***; *European Journal of Operational Research, 201, 171–182;*

[24] **Azimi, P. and Salehi, M. A. (2011),** ***A Simulation-Based Heuristic for the Dynamic Facility Layout Problem****; International Bulletin of Business Administration, 11, 112-120;*

[25] **Umarani, R. et al. (2010),** ***PSO Evolution Overview and Applications***; *International Journal of Engineering Science and Technology Vol. 2(7), 2802-2806;*

[26] **Eberhart, R. Kennedy, J. (1995),** ***A New Optimizer Using Particles Swarm Theory****; Proceedings of the Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan),IEEE Service Center, Piscataway, NJ, 39-43;*

[27] **Eberhart RC., Shi, Y. (2001), Particle Swarm Optimization:** ***Developments, Applications and Resources****; Proceedings of the 2001 Congress on Evolutionary Computation, 1, 81-86. Piscataway: IEEE Press. [doi: 10.1109/CEC.2001.934374];*

[28] **Kennedy, J., Eberhart, R.C., (1997),** ***A Discrete Binary Version of the Particle Swarm Algorithm****; Proceedings of World Multi-Conf. on Systemics, Cybernetics and Informatics, 4104-4109;*

[29] **Balakrishnan, J., Jacobs, R.F., Venkataramanan, M.A. (1992),** ***Solutions for the Constrained Dynamic Facility Layout Problem****; European Journal of Operations Research, 57(2):280-286. [doi:10.1016/0377-2217(92)90049-F].*