Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

An extended discrete particle swarm optimization algorithm for the dynamic facility layout problem

Hassan REZAZADEH†, Mehdi GHAZANFARI, Mohammad SAIDI-MEHRABAD, Seyed JAFAR SADJADI

(*Department of Industrial Engineering, Iran University of Science and Technology, Narmak, Tehran, Iran*)

†E-mail: hassan.rezazadeh@gmail.com

Received Apr. 15, 2008; Revision accepted Jan. 16, 2008; Crosschecked Jan. 8, 2009

Abstract: We extended an improved version of the discrete particle swarm optimization (DPSO) algorithm proposed by Liao *et al*.(2007) to solve the dynamic facility layout problem (DFLP). A computational study was performed with the existing heuristic algorithms, including the dynamic programming (DP), genetic algorithm (GA), simulated annealing (SA), hybrid ant system (HAS), hybrid simulated annealing (SA-EG), hybrid genetic algorithms (NLGA and CONGA). The proposed DPSO algorithm, SA, HAS, GA, DP, SA-EG, NLGA, and CONGA obtained the best solutions for 33, 24, 20, 10, 12, 20, 5, and 2 of the 48 problems from (Balakrishnan and Cheng, 2000), respectively. These results show that the DPSO is very effective in dealing with the DFLP. The extended DPSO also has very good computational efficiency when the problem size increases.

Key words: Dynamic facility layout problem (DFLP), Particle swarm optimization (PSO), Optimization, Heuristic method
doi:10.1631/jzus.A0820284 **Document code:** A **CLC number:** TP273

INTRODUCTION

Cost reduction is one of the major strategies for a manufacturing organization to adopt to stay in business under global market competition. It has been estimated that between 15% and 70% of total manufacturing operating expenses can be attributed to material handling, and that an effective facility layout can reduce these costs by at least 10%~30% (Tompkins *et al*., 1996).

Static facility layout problem (SFLP) is a plan for arranging the physical facilities within an area to achieve cost reduction objectives (Armour and Buffa, 1963). The most common objective considered is the minimization of material handling costs. Material handling costs are determined based on the amount of materials that flow between the facilities and the distance between the locations of the facilities. In an environment where material-handling flow does not change over a long time, a static layout analysis would be sufficient. However, due to the competitive and volatile market conditions today's manufacturing firms operate in a dynamic environment (Shore and Tompkins, 1980). To stay competitive and operate efficiently in such an environment, manufacturing firms must adapt their facility layouts to market fluctuations. Therefore, this paper investigates the dynamic facility layout problem (DFLP) based on multi-period planning horizons (Rosenblatt, 1986). During these horizons, the locations of facilities in the layout may change. The DFLP extends the SFLP by considering the changes in material-handling flow over multiple periods and the costs of rearranging the layout.

The remainder of this paper is organized as follows. In Section 2, we review the relevant literature for the DFLP. Section 3 presents the mathematical formulation for the DFLP. Section 4 describes the solution methodology based on the PSO procedure. The computational results are reported in Section 5 and conclusions are given in Section 6.

LITERATURE SURVEY

There have been a number of previous studies dealing with the DFLP. Rosenblatt (1986) used dynamic programming (DP) to solve the DFLP. Urban (1993) proposed a heuristic algorithm similar to CRAFT (Armour and Buffa, 1963), so-called steepest-descent pairwise exchange heuristic for the SFLP, which considers forecast windows. Lacksonen and Enscore (1993) surveyed the static and dynamic facility layout problems in varying area by mathematical programming approaches. Conway and Venkataramanan (1994) used a genetic algorithm (GA) for the DFLP. Balakrishnan and Cheng (2000) improved the GA of (Conway and Venkataramanan, 1994) to solve the DFLP. Kaku and Mazzola (1997) used a heuristic tabu search algorithm for the problem. Balakrishnan *et al*.(2000) presented two heuristic algorithms, which improved Urban's steepest-descent pairwise exchange heuristic algorithm. The first algorithm uses Urban's algorithm to generate solutions for the DFLP. The solutions generated for each forecast window are improved using a backward-pass pairwise exchange algorithm, and the best solution is selected. The second algorithm combines Urban's algorithm with DP. Baykasoglu and Gindy (2001) developed a simulated annealing (SA) algorithm for the DFLP. Using the test problems from (Balakrishnan and Cheng, 2000), they showed that the SA performed better than the developed GAs. In an erratum, Baykasoglu and Gindy (2004) corrected their computational results reported in (Baykasoglu and Gindy, 2001). Balakrishnan *et al*.(2003) presented a hybrid GA for the DFLP. McKendall and Shang (2006) presented a hybrid ant system (HAS) for the DFLP. Also, McKendall *et al*.(2006) presented an SA heuristic to solve the DFLP. A complete survey of the different approaches for the DFLP can be found in (Balakrishnan and Cheng, 1998).

By considering future changes in the design step, some authors addressed another aspect of the layout flexibility, so-called robust layout, which is related to handling the uncertainty of production scenarios over time without any external action. Kulturel-Konak (2007) attempted a more comprehensive study of the relevance of robust facility layout design issues under uncertainty.

PROBLEM FORMULATION

In this section, we have formulated the mathematical model for the DFLP that was adopted by Balakrishnan *et al*.(1992). The assumptions are described as follows:

(1) Equal-sized facilities and locations are considered.

(2) Shapes and dimensions of the shop floor are not restricted.

(3) The number of periods in the planning horizon is known.

(4) Distances between the facilities are determined a priori.

Indexing sets

i, j are indices for facilities, i, j=1, 2, …, M, $i\neq j$; h, l are indices for facility locations, h, l=1, 2, …, M, $h\neq l$; t is the index for periods, t=1, 2, …, P.

Parameters

M is the total number of locations and facilities; N_{p} is the swarm size, i.e., the number of particles; P is the total number of periods; f_{tik} is the flow cost for unit distance from facility i to k in period t; d_{tjl} is the distance from location j to l in period t; A_{tijl} is the cost of shifting facility i between locations j and l in period t.

Decision variables

The decision variables X_{tij} and Y_{tijl} of the model are defined as follows:

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

$$Y_{tijl}=\begin{cases}1, & \text{if facility } i \text{ is shifted between locations } j \text{ and } l\\ & \text{at the beginning of period } t,\\ 0, & \text{otherwise.}\end{cases}$$

Mathematical model

The quadratic assignment problem (QAP) for the DFLP is presented as follows:

$$\min Z=\sum_{t=1}^{P}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{M}\sum_{l=1}^{M} f_{tik}d_{tjl}X_{tij}X_{tkl}+\sum_{t=2}^{P}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{l=1}^{M}A_{tijl}Y_{tijl}, \tag{1}$$

s.t.

$$\sum_{i=1}^{M} X_{tij}=1,\quad j=1, 2, ..., M,\ t=1, 2, ..., P, \tag{2}$$

$$\sum_{j=1}^{M} X_{tij}=1,\quad i=1, 2, ..., M,\ t=1, 2, ..., P, \tag{3}$$

$$Y_{tijl}=X_{(t-1)ij}X_{til},\quad i, j, l=1, 2, ..., M,\ t=2, 3, ..., P, \tag{4}$$

$$X_{tkl}, X_{tij}, Y_{tijl}\in\{0, 1\},\ i,\ j, l=1, 2, ..., M,\ t=2, 3, ..., P. \tag{5}$$

The objective function Eq.(1) minimizes the sum of the material flow and layout rearrangement costs during the planning horizon. Constraints Eqs.(2) and (3) ensure that each facility location is assigned to one facility and each facility is assigned to one facility location at each period, respectively. Constraint Eq.(4) adds the rearrangement costs to the material flow cost if a facility is shifted between locations in consecutive periods. Lastly, the restrictions on the decision variables are given in Eq.(5).

PARTICLE SWARM OPTIMIZATION

Introduction to PSO

PSO is one of the optimization techniques that belong to evolutionary computation techniques. PSO and its discrete version, DPSO, originally designed and developed by Kennedy and Eberhart (1995; 1997), are stochastic, population-based search algorithms and gained much attention. PSO is a kind of simulation of the movement and flocking of birds. In the PSO algorithm, the best member in the swarm impresses the social behavior of particles. The algorithm initializes the flock of particles randomly over the searching space. These particles move with a certain law and find the global best result after some iterations. At each iteration, each particle adjusts its velocity vector based on its momentum and its best solution (*pbest*) and the best solution of its neighbors (*gbest*), and then computes a new point to examine. The PSO has undergone many changes since proposed in 1995. A comprehensive study of various aspects of the PSO algorithm can be found in (Poli *et al*., 2007).

Discrete PSO

DPSO essentially differs from the original (or continuous) PSO in two characteristics. First, the particle is composed of the binary variable. Second, the velocity must be transformed into the change of probability, which is the chance of the binary variable taking the value 1.

Let $\boldsymbol{X}_i^t=(x_{i1}^t, x_{i2}^t, ..., x_{iD}^t)$ $(x_{id}^t\in\{0, 1\},\ d=1, 2, \ldots, D)$ be particle i with D bits at iteration t, where the D-dimensional vector of $\boldsymbol{X}_i^t$ being treated as a potential solution has a rate of change called velocity, denoted as $\boldsymbol{V}_i^t=(v_{i1}^t, v_{i2}^t, ..., v_{iD}^t)$, $v_{id}^t\in\mathbb{R}$. Let $\boldsymbol{P}_i^t=(p_{i1}^t, p_{i2}^t, ..., p_{iD}^t)$ and $\boldsymbol{P}_{\mathrm{g}}^t=(p_{\mathrm{g}1}^t, p_{\mathrm{g}2}^t, ..., p_{\mathrm{g}D}^t)$ be the local best (*pbest*) and global best (*gbest*) at iteration t, respectively (Shi and Eberhart, 1998).

As in the continuous PSO, the velocity of each particle is gained according to the following equation:

$$v_{id}^t = v_{id}^{t-1} + c_1r_1(p_{id}^t - x_{id}^t) + c_2r_2(p_{\mathrm{g}d}^t - x_{id}^t), \tag{6}$$

where c_1 and c_2, which are random numbers uniformly distributed in [0, 1], are the cognition learning factor and social learning factor, respectively. The values c_1r_1 and c_2r_2 determine the weights of the two parts, and their sum is usually limited to 4 (Kennedy *et al*., 2001).

By Eq.(6), each particle moves according to its new velocity. Recall that particles are represented by binary variables. For the velocity value of each bit in a particle, Kennedy and Eberhart (1997) claimed that the higher value is more likely to choose 1, while the lower value favors the 0 choice. Furthermore, they constrained the velocity values to the interval [0, 1] by using the following sigmoid function:

$$s(v_{id}^t)=\frac{1}{1+\exp(-v_{id}^t)}, \tag{7}$$

where $s(v_{id}^t)$ denotes the probability of bit x_{id}^t taking 1. To avoid $s(v_{id}^t)$ approaching 0 or 1, a constant $V_{\max}$ is used to limit the range of v_{id}^t, i.e., $v_{id}^t\in[-V_{\max}, +V_{\max}]$. In practice, $V_{\max}$ is often set at 4 (Kennedy *et al*., 2001).

Proposed DPSO algorithm

Liao *et al*.(2007) and Tseng and Liao (2008) proposed DPSO algorithms extended from (Poli *et* *al*., 2007) for solving flow shop scheduling problems. In this subsection, we extend an improved version of the DPSO algorithm proposed by Liao *et al*.(2007) to solve the DFLP. To produce solutions with good quality in less computational time, we use a local search scheme based on semi-annealing heuristic to find a better solution of *gbest*. The steps of the proposed DPSO algorithm are represented in the flow chart given in Fig.1. In the algorithm we generate N_p sequences (particles) at each iteration, and hence a total of $N_p k_{max}$ sequences are enumerated.

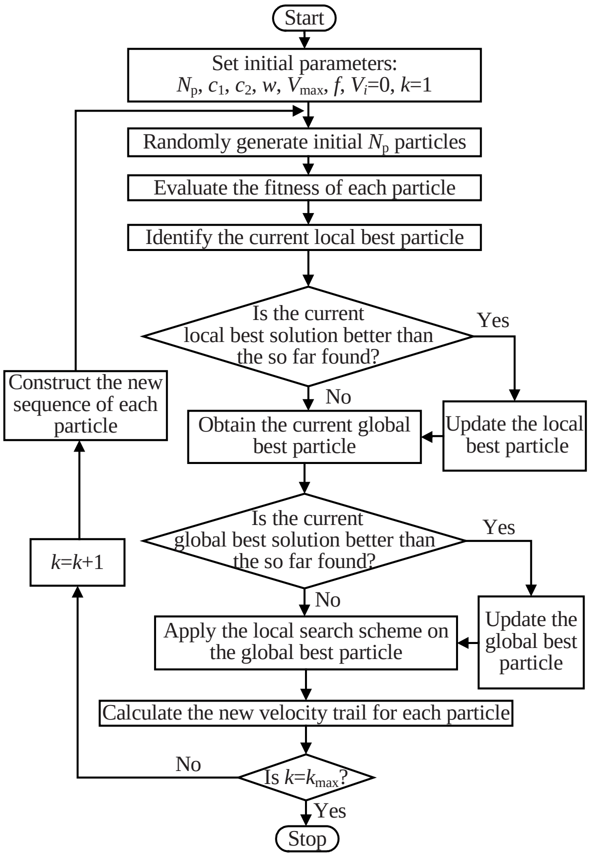

Fig.1 Steps of the proposed DPSO

Encoding

The most important issue in applying the PSO successfully is to develop an effective 'problem mapping' mechanism. The solution encoding of the proposed model involves the 0-1 binary integer decision variables X_{tij} enabling a randomly generated solution. Fig.2 illustrates a particle structure assuming P time periods (1, 2, …, P), M locations (1, 2, …, M) and M facilities (1, 2, …, M) in each period.

<table>
<tr><td colspan="16">Period 1</td></tr>
<tr><td colspan="16">Locations</td></tr>
<tr><td colspan="4">1</td><td colspan="4">2</td><td colspan="4">...</td><td colspan="4">M</td></tr>
<tr><td colspan="4">Facilities</td><td colspan="4">Facilities</td><td colspan="4">Facilities</td><td colspan="4">Facilities</td></tr>
<tr><td>1</td><td>2</td><td>...</td><td>M</td><td>1</td><td>2</td><td>...</td><td>M</td><td>1</td><td>2</td><td>...</td><td>M</td><td>1</td><td>2</td><td>...</td><td>M</td></tr>
</table>

⋮

<table>
<tr><td colspan="16">Period P</td></tr>
<tr><td colspan="16">Locations</td></tr>
<tr><td colspan="4">1</td><td colspan="4">2</td><td colspan="4">...</td><td colspan="4">M</td></tr>
<tr><td colspan="4">Facilities</td><td colspan="4">Facilities</td><td colspan="4">Facilities</td><td colspan="4">Facilities</td></tr>
<tr><td>1</td><td>2</td><td>...</td><td>M</td><td>1</td><td>2</td><td>...</td><td>M</td><td>1</td><td>2</td><td>...</td><td>M</td><td>1</td><td>2</td><td>...</td><td>M</td></tr>
</table>

Fig.2 Solution representation

Definition of a discrete particle

We define particle i at iteration t as $\boldsymbol{X}_i^t=(\boldsymbol{X}_{i1}^t, \boldsymbol{X}_{i2}^t, ..., \boldsymbol{X}_{iP}^t)$, where $\boldsymbol{X}_{ik}^t=(x_{ik11}^t, x_{ik12}^t, ..., x_{ik1M}^t, x_{ik21}^t, x_{ik22}^t, ..., x_{ikMM}^t)$, $x_{ikjl}\in\{0, 1\}$, k=1, 2, …, P, j, l=1, 2, …, M, and x_{ikjl}^t equals 1 if facility j of particle i is placed in the lth location of the sequence in period k, and 0 otherwise. For example, suppose the sequence of $\boldsymbol{X}_i^t$ is {(2314), (3421), …, (2143)}. By this definition, we have $x_{i121}^t=x_{i132}^t=x_{i113}^t=x_{i144}^t=1$ and all other $x_{ikjl}^t=0$ in the first period, $x_{i231}^t=x_{i242}^t=x_{i223}^t=x_{i214}^t=1$ and all other $x_{ikjl}^t=0$ in the second period, and $x_{iP21}^t=x_{iP12}^t=x_{iP43}^t=x_{iP34}^t=1$ and all other $x_{ikjl}^t=0$ in the Pth period (Fig.3).

		Location (l)												
		Period 1				Period 2				…	Period P			
		1	2	3	4	1	2	3	4		1	2	3	4
Facility (j)	1	0	0	1	0	0	0	0	1	…	0	1	0	0
	2	1	0	0	0	0	0	1	0	…	1	0	0	0
	3	0	1	0	0	1	0	0	0	…	0	0	0	1
	4	0	0	0	1	0	1	0	0	…	0	0	1	0

Fig.3 Definition of particle X_i^t for sequence {(2314), (3421), …, (2143)}

Velocity trail

After a period is selected, to move a particle to a new sequence, we define $\boldsymbol{V}_{ik}^t=(v_{ik11}^t, v_{ik12}^t, ..., v_{ik1M}^t, v_{ik21}^t, v_{ik22}^t, ..., v_{ikMM}^t)$, $v_{ikjl}^t\in\mathbb{R}$, where v_{ikjl}^t is the velocity value for facility j of particle i placed in the lth location in period k at iteration t. The velocity

$\boldsymbol{V}_{ik}^t$, called the velocity trail, is based on frequency-based memory (Onwubolu, 2002), which is often used in combinatorial optimization, e.g., the long-term memory of tabu search, to provide useful information that facilitates choosing preferred moves. A higher value of v_{ikjl}^t in the trail indicates that facility j is more likely to be placed in the lth location, while a lower value favors moving facility j out of the lth location. The new velocity trail of each particle is obtained from

$$v_{ikjl}^t = wv_{ikjl}^{t-1} + c_1 r_1 (p_{ikjl}^t - x_{ikjl}^t) + c_2 r_2 (p_{gkjl}^t - x_{ikjl}^t). \quad (8)$$

Here $\boldsymbol{P}_i^t = (\boldsymbol{P}_{i1}^t, \boldsymbol{P}_{i2}^t, ..., \boldsymbol{P}_{iP}^t)$; $\boldsymbol{P}_{ik}^t = (P_{ik11}^t, P_{ik12}^t, ..., P_{ik1M}^t, P_{ik21}^t, P_{ik22}^t, ..., P_{ikMM}^t)$ and $\boldsymbol{P}_{gk}^t = (P_{gk11}^t, P_{gk12}^t, ..., P_{gk1M}^t, P_{gk21}^t, P_{gk22}^t, ..., P_{gkMM}^t)$ ($P_{ikjl}^t, P_{gkjl}^t \in \{0, 1\}$, k=1, 2, …, P, j, l=1, 2, …, M) denote the *pbest* and *gbest* at iteration t, respectively. Also w is the inertia weight proposed by Shi and Eberhart (1998). A constant $V_{\max}$ is used to keep each component of the velocity trail in a constant range, i.e., $v_{ikjl}^t \in [-V_{\max}, +V_{\max}]$.

We now explain the meaning of ‘velocity trail’. For simplicity, suppose there exists only the social part in Eq.(8) and $c_2=r_2=1$. The sequence of $\boldsymbol{X}_i^t$ is assumed to be {(2314), (3421), …, (2143)}, the first period to be selected randomly, and the sequence of $\boldsymbol{P}_{g1}^t$ to be (1324). It is clear that $v_{i1jl}^t = p_{g1jl}^t - x_{i1jl}^t = 1, 0, -1$ (Fig.4). Value 1 intensifies the arrangement of facility j in the lth location, whereas −1 versifies such an arrangement. In the calculation, we can simply add $p_{g1jl}^t=1$ to the corresponding v_{i1jl}^t, subtract $x_{i1jl}^t=1$ from v_{i1jl}^t, and leave others unchanged. v_{i1jl}^t is set as $-V_{\max}$ ($+V_{\max}$) if it is smaller (greater) than $-V_{\max}$ ($+V_{\max}$).

Facility (j) \ Location (l)	1	2	3	4
1	1	0	–1	0
2	–1	0	1	0
3	0	0	0	0
4	0	0	0	0

Fig.4 Resulting values of the velocity trail at the first period

The above example and Eq.(8) demonstrate that the velocity trail is gradually accumulated by the individual’s own experience and its companions’ experience. This social behavior of sharing useful information among individuals in searching for the optimal solution gives PSO an advantage over more classical meta-heuristics.

As in DPSO, the velocity trail values need to be converted from real numbers to the changes of probabilities by the following sigmoid function:

$$s(v_{ikjl}^t) = \frac{1}{1+\exp(-v_{ikjl}^t)}, \quad (9)$$

where $s(v_{ikjl}^t)$ represents the probability of x_{ikjl}^t taking the value 1. For example, $s(v_{i112}^t)$=0.2 in Fig.5 represents that there is a 20% chance that facility 1 of particle i will be placed in the 2nd location at the first period.

Facility (j) \ Location (l)	1	2	3	4
1	0.10	0.20	0.45	0.01
2	0.99	0.13	0.55	0.98
3	0.53	0.99	0.67	0.23
4	0.85	0.85	0.89	0.50

Fig.5 Changes of probabilities from the velocity trail at the first period

Construction of a particle sequence

In the proposed algorithm, each particle constructs its new sequence based on its changes of probabilities from the velocity trail selected period. In the conventional approach, a particle i starts with a null sequence in the selected period and places an unlocated facility j in location L (L=1, 2, ..., M) according to the following probability:

$$q_{ik}^t(j, L) = \frac{s(v_{ikjl}^t)}{\sum_{j\in F} s(v_{ikjl}^t)}, \quad (10)$$

where F is the set among the first f unlocated facilities present in the best sequence B obtained so far. When there are less than f facilities unlocated, all of these facilities are included. For example, suppose B={(2314), (3421), …, (2143)}, f=2, that the first period is selected randomly and $s(v_{i1jl}^t)$ is as given in Fig.5. We start with the null sequence at the first period and consider only the first two (f=2) unlocated facilities (i.e., facilities 2 and 3) to be placed in the first location. By Eq.(10), $q_{i1}^t(2, 1)$=0.99/(0.99+0.53) =0.6513, $q_{i1}^t(3, 1)$=0.53/(0.99+0.53)=0.3487. Suppose that facility 3 is selected based on the above probabilities. Then the unlocated facilities to be considered in the second location will be facilities 2 and 1.

Variant of the gbest model

For the neighborhood structure of particles in the social part, we introduce the gbest model but modify the approach of searching for $\boldsymbol{P}_{gk}^t$ in our algorithm. In the original PSO approach, $\boldsymbol{P}_{gk}^t$ is obtained from $\boldsymbol{P}_{ik}^t$ (i=1, 2, ..., N_p). Based on our computational experiments on the facility layout problem, we find that the approach that obtains $\boldsymbol{P}_{gk}^t$ from the local search scheme performs better. Although our approach spends more computation time on converging, it increases the probability of leaving a local optimum.

Local search scheme

The local search scheme is one of the main components in the DPSO algorithm. It is referred to as a local search, since it starts with a feasible solution and, using moves, it searches a neighborhood for another feasible solution with lower cost. The neighborhood of a solution is the set of all solutions that can be reached with a move. If a better solution is found, the current solution is replaced and the neighborhood search is started again. If no further improvement can be made, a local optimum is found. It means that there is no better solution in the neighborhood of the current solution.

We used a heuristic local search for *gbest* at each iteration based on a semi-annealing approach in which there is only an outside loop (instead of two inside and outside loops) controlled by three parameters T, α, and δ. Parameters T and α denote the temperature and cooling schedule similar to simulated annealing (SA) and δ indicates the relative fitness of recently obtained neighborhood. The computation time allocated to local search depends on the initial setting of above parameters. We found the best parameter setting as T=1, δ=0.1, and α=0.9. In optimistic status, if in the first iteration $Fit(Tmp\_X) \leq 0.5Fit(New\_X)$, then δ=1 and the do-until loop iterates only once (Tmp_X is null in first iteration and takes neighborhood solutions in the last iterations. New_X is a solution and Fit is the fitness function). In pessimistic status, if no better neighborhood is found, then the do-until loop iterates r times, where $r=\ln\alpha/\ln\delta\approx 22$. The proposed algorithm causes a high computation time that is not allocated to local search.

COMPUTATIONAL RESULT

This section presents the computational results of the proposed DPSO algorithm applied to the 48 test problems obtained from (Balakrishnan and Cheng, 2000). The proposed algorithm was programmed using C++ programming language, and the set of the test problems was solved on a PC with 2.4 GHz Pentium IV CPU.

In the preliminary experiment, the ranges of parameter values from (Liao *et al*., 2007) were tested, i.e., $N_p\in[5, 60]$, $c_k\in[1, 4]$, $w\in(0.8, 1.2)$, $V_{max}\in[3, 20]$, $f\in[0.2M, 0.8M]$. Based on the experimental results, all the best parameter settings for the proposed DPSO algorithm are given in Table 1. Each test problem was solved 10 times for the DPSO algorithm, and the best and average solutions were recorded. Also the average running time (in min) for each test problem is given. Tables 2~7 summarize the results obtained by the DPSO algorithm. The bold numbers give the best solution for each test problem. For each dataset, the results for the DPSO algorithm were compared with the results obtained by the GA presented by (Conway and Venkataramanan, 1994; Balakrishnan and Cheng, 2000; Balakrishnan *et al*., 2003), the DP presented by Erel *et al*.(2003), the HAS presented by McKendall and Shang (2006), and the SA presented by (Baykasoglu and Gindy, 2004; McKendall *et al*., 2006).

Table 1 Parameters used in this study

Parameter	Value	Parameter	Value
N_p	20	c_2	1.5
w	1	V_{max}	10
c_1	1.5	f	0.4M

Tables 2 and 3 show the results for the test problems where M=6, P=5 (Problems 1~8) and M=6, P=10 (Problems 9~16), respectively. The DPSO algorithm obtained the best solutions for all the 16 test problems. Since the proposed DPSO algorithm, SA, and HAS obtained the best solutions for all the 16 test problems, these heuristics are the preferred choices for this set of 16 problems. However, GA, DP, hybrid simulated annealing (SA-EG), hybrid genetic algorithms NLGA and CONGA obtained the best solutions for 10, 12, 13, 5 and 2 problems, respectively. As a result, the proposed DPSO algorithm, SA, and HAS are the preferred heuristics for the test problems where M=6.

Tables 4 and 5 give the results for the test problems where M=15, P=5 (Problems 17~24) and M=15, P=10 (Problems 25~32), respectively. For the test problems where P=5 (Problems 17~24), the DPSO algorithm and SA-EG both obtained the best solution for 4 of the 8 problems. Also, SA, HAS, GA, DP, NLGA, CONGA obtained the best solution for 1, 1, 0, 0, 0, 0 of the 8 problems, respectively. Clearly, SA-EG and our proposed DPSO algorithm outperformed all of the other heuristics. Nevertheless, the objective function values of the best solutions obtained by the proposed heuristic algorithm were less than 0.8% above the best found solutions. In contrast, the proposed DPSO algorithm slightly outperformed SA-EG for P=10 (Problems 25~32) by obtaining 4 of the best solutions versus 3. SA, HAS, GA, DP, NLGA, CONGA obtained the best solution for 1, 0, 0, 0, 0, 0 of the 8 problems, respectively. As a result, the proposed DPSO algorithm and SA-EG are the preferred heuristics for the test problems where M=15.

Tables 6 and 7 give the results for the test problems where M=30, P=5 (Problems 33~40) and M=30, P=10 (Problems 41~48), respectively. For the test problems where P=5 (Problems 33~40), the DPSO algorithm and SA both obtained the best solution for 4 of the 8 problems. HAS, GA, DP, SA-EG, NLGA, CONGA obtained the best solution for 2, 0, 0, 0, 0, 0 of the 8 problems, respectively. Clearly, SA and our proposed DPSO algorithm outperformed all of the other heuristics. Nevertheless, the objective function values of the best solutions obtained by the proposed heuristic algorithm were less than 0.5% above the best found solutions. In contrast, the proposed DPSO algorithm slightly outperformed SA for P=10 (Problems 41~48) by obtaining 5 of the best solutions versus 2. HAS, GA, DP, SA-EG, NLGA, CONGA obtained the best solution for 1, 0, 0, 0, 0, 0 of the 8 problems, respectively. As a result, the proposed PSO algorithm is the preferred heuristic for the test problems where M=30.

Table 2 Solution results for problems with M=6, P=5*[*]

Problem No.	CONGA	NLGA	SA-EG	DP	GA	HAS	SA	DPSO**		
								Average solution	Best solution	Deviation (%)
1	108976	**106419**	**106419**	**106419**	**106419**	**106419**	**106419**	**106419**	**106419**	0
2	105170	**104834**	**104834**	**104834**	**104834**	**104834**	**104834**	**104834**	**104834**	0
3	104520	**104320**	**104320**	**104320**	**104320**	**104320**	**104320**	**104320**	**104320**	0
4	106719	106515	**106399**	106509	106515	**106399**	**106399**	**106399**	**106399**	0
5	**105628**	**105628**	**105628**	**105628**	**105628**	**105628**	**105628**	**105628**	**105628**	0
6	105605	104053	**103985**	**103985**	104053	**103985**	**103985**	**103985**	**103985**	0
7	**106439**	106978	**106439**	106447	**106439**	**106439**	**106439**	**106439**	**106439**	0
8	104485	**103771**	**103771**	**103771**	**103771**	**103771**	**103771**	**103771**	**103771**	0

* For all the algorithms other than the proposed DPSO, the data are the best solution results; the bold numbers give the best solution for each test problem—The same for Tables 3~7. ** Average run time: 0.11 min

Table 3 Solution results for problems with M=6, P=10

Problem No.	CONGA	NLGA	SA-EG	DP	GA	HAS	SA	DPSO**		
								Average solution	Best solution	Deviation (%)
9	218407	214397	**214313**	**214313**	**214313**	**214313**	**214313**	**214313**	**214313**	0
10	215623	212138	**212134**	**212134**	**212134**	**212134**	**212134**	**212134**	**212134**	0
11	211028	208453	**207987**	**207987**	**207987**	**207987**	**207987**	**207987**	**207987**	0
12	217493	212953	212747	212741	212741	**212530**	**212530**	**212530**	**212530**	0
13	215363	211575	211072	211022	210944	**210906**	**210906**	**210906**	**210906**	0
14	215564	210801	**209932**	**209932**	210000	**209932**	**209932**	**209932**	**209932**	0
15	220529	215685	214438	**214252**	215452	**214252**	**214252**	**214252**	**214252**	0
16	216291	214657	**212588**	**212588**	**212588**	**212588**	**212588**	**212588**	**212588**	0

** Average run time: 0.23 min

Table 4 Solution results for problems with *M*=15, *P*=5

Problem No.	CONGA	NLGA	SA-EG	DP	GA	HAS	SA	DPSO**		
								Average solution	Best solution	Deviation (%)
17	504 759	511 854	481 378	482 123	484 090	**480 453**	**480 453**	481 534	**480 453**	0.00
18	514 718	507 694	**478 816**	485 702	485 352	484 761	484 761	484 124	482 568	0.78
19	516 063	518 461	487 886	491 310	489 898	488 748	488 748	487 125	**486 658**	−0.25
20	508 532	514 242	481 628	486 851	484 625	484 446	484 405	482 402	**480 359**	−0.03
21	515 599	512 834	**484 177**	491 178	489 885	487 722	487 882	487 125	486 658	0.51
22	509 384	513 763	**482 321**	489 847	488 640	486 685	487 147	485 904	485 637	0.69
23	512 508	512 722	**485 384**	489 155	489 378	486 853	486 779	485 986	485 462	0.02
24	514 839	521 116	489 072	493 577	500 779	491 016	490 812	489 423	**488 865**	−0.04

** Average run time: 1.38 min

Table 5 Solution results for problems with *M*=15, *P*=10

Problem No.	CONGA	NLGA	SA-EG	DP	GA	HAS	SA	DPSO**		
								Average solution	Best solution	Deviation (%)
25	1 055 536	1 047 596	982 298	983 070	987 887	980 351	979 468	979 254	**978 546**	−0.09
26	1 061 940	1 037 580	**973 179**	983 826	980 638	978 271	978 065	976 354	975 684	0.26
27	1 073 603	1 056 185	985 364	988 635	985 886	978 027	982 396	977 246	**976 382**	−0.17
28	1 060 034	1 026 789	974 994	976 456	976 025	974 694	972 797	973 265	**972 684**	−0.01
29	1 064 692	1 033 591	**975 498**	982 893	982 778	979 196	977 188	977 254	976 645	0.12
30	1 066 370	1 028 606	968 323	974 436	973 912	971 548	**967 617**	970 125	969 326	0.18
31	1 066 617	1 043 823	**977 410**	982 790	982 872	980 752	979 114	979 125	978 657	0.13
32	1 068 216	1 048 853	985 041	988 584	987 789	985 707	983 672	983 254	**982 964**	−0.03

** Average run time: 2.98 min

Table 6 Solution results for problems with *M*=30, *P*=5

Problem No.	CONGA	NLGA	SA-EG	DP	GA	HAS	SA	DPSO**		
								Average solution	Best solution	Deviation (%)
33	632 737	611 794	583 081	579 741	578 689	576 886	576 039	576 394	**575 684**	−0.06
34	647 585	611 873	573 965	570 906	572 232	570 349	**568 095**	571 369	570 365	0.40
35	642 295	611 664	577 787	577 402	578 527	576 053	**573 739**	576 146	575 698	0.34
36	634 626	611 766	572 139	569 596	572 057	566 777	566 248	567 429	**566 124**	−0.02
37	639 693	604 564	563 503	561 078	559 777	**558 353**	558 460	559 462	558 680	0.06
38	637 620	606 010	570 905	567 154	566 792	566 792	566 077	566 432	**565 894**	−0.03
39	640 482	607 134	571 499	568 196	567 873	**567 131**	**567 131**	568 432	**567 131**	0.00
40	635 776	620 183	581 614	575 273	575 720	575 280	**573 755**	575 648	574 369	0.11

** Average run time: 3.52 min

Table 7 Solution results for problems with *M*=30, *P*=10

Problem No.	CONGA	NLGA	SA-EG	DP	GA	HAS	SA	DPSO**		
								Average solution	Best solution	Deviation (%)
41	1 362 513	1 228 411	1 174 815	1 171 178	1 169 474	1 166 164	1 163 222	1 162 326	**1 161 124**	−0.18
42	1 379 640	1 231 978	1 173 015	1 169 138	1 168 878	1 168 878	1 161 521	1 162 463	**1 155 634**	−0.50
43	1 365 024	1 231 829	1 166 295	1 165 525	1 166 366	1 166 366	**1 156 918**	1 168 965	1 158 264	0.12
44	1 367 130	1 227 413	1 154 196	1 152 684	1 154 192	1 148 202	1 145 918	1 152 365	**1 144 872**	−0.09
45	1 356 860	1 215 256	1 140 116	1 128 136	1 133 561	1 128 855	1 126 432	1 136 985	**1 125 687**	−0.07
46	1 372 513	1 221 356	1 158 227	1 143 824	1 145 000	**1 141 344**	1 145 146	1 152 498	1 142 568	0.11
47	1 382 799	1 212 273	1 157 505	1 142 494	1 145 927	1 140 773	**1 140 744**	1 148 956	1 141 722	0.09
48	1 383 610	1 245 423	1 177 565	1 167 163	1 168 657	1 166 157	1 161 437	1 169 865	**1 160 658**	−0.07

** Average run time: 5.17 min

In summary, the proposed DPSO algorithm, SA, HAS, GA, DP, SA-EG, NLGA, and CONGA obtained the best solutions for 33, 24, 20, 10, 12, 20, 5, and 2 of the 48 problems, respectively. Therefore the proposed PSO algorithm performed better than all of the other heuristic algorithms for this dataset with respect to solution quality.

Since the varied heuristic algorithms in the literature use different computing systems, programming language compilers, coding techniques, etc., it is very difficult to compare computation time for them, and thus we did not make the comparison of computation time in this study.

CONCLUSION

We extended a discrete particle swarm optimization algorithm to solve the DFLP. A computational study was performed with the existing heuristic algorithms including the SA, HAS, GA, DP, SA-EG, NLGA and CONGA. These algorithms were applied to the 48 test problems from (Balakrishnan and Cheng, 2000). Computation results show that the proposed algorithm performs very well. Also, the proposed algorithm has very good computational efficiency when the problem size increases. The following recommendations are given for future research:

(1) The time-dimension comparison of the developed algorithms for the DFLP can be interesting.

(2) The DFLP can be modeled with production uncertainty consideration.

(3) For comprehensive manufacturing system design, the DFLP can also be combined with the other domains like virtual manufacturing cells.

Reference

Armour, G.C., Buffa, E.S., 1963. A heuristic algorithm and simulation approach to relative allocation of facilities. *Manag. Sci.*, **9**(2):294-300. [doi:10.1287/mnsc.9.2.294]

Balakrishnan, J., Cheng, C.H., 1998. Dynamic layout algorithms: a state-of-the-art survey. *Omega, Int. J. Manag. Sci.*, **26**(4):507-521. [doi:10.1016/S0305-0483(97)00078-9]

Balakrishnan, J., Cheng, C.H., 2000. Genetic search and the dynamic layout problem. *Comput. Oper. Res.*, **27**(6):587-593. [doi:10.1016/S0305-0548(99)00052-0]

Balakrishnan, J., Jacobs, R.F., Venkataramanan, M.A., 1992. Solutions for the constrained dynamic facility layout problem. *Eur. J. Oper. Res.*, **57**(2):280-286. [doi:10.1016/0377-2217(92)90049-F]

Balakrishnan, J., Cheng, C.H., Conway, G., 2000. An improved pair wise exchange heuristic for the dynamic plant layout problem. *Comput. Oper. Res.*, **27**(6):587-593. [doi:10.1016/S0305-0548(99)00052-0]

Balakrishnan, J., Cheng, C.H., Conway, D.G., Lau, C.M., 2003. A hybrid genetic algorithm for the dynamic plant layout problem. *Int. J. Prod. Econ.*, **86**(2):107-120. [doi:10.1016/S0925-5273(03)00027-6]

Baykasoglu, A., Gindy, N.N.Z., 2001. A simulated annealing algorithm for the dynamic layout problem. *Comput. Oper. Res.*, **28**(14):1403-1426. [doi:10.1016/S0305-0548(00)00049-6]

Baykasoglu, A., Gindy, N.N.Z., 2004. Erratum to a simulated annealing algorithm for the dynamic layout problem. *Comput. Oper. Res.*, **31**(2):313-315. [doi:10.1016/S0305-0548(03)00205-3]

Conway, D.G., Venkataramanan, M.A., 1994. Genetic search and the dynamic facility layout problem. *Comput. Oper. Res.*, **21**(8):955-960. [doi:10.1016/0305-0548(94)90023-X]

Erel, E., Ghosh, J.B., Simon, J.T., 2003. New heuristic for the dynamic layout problem. *J. Oper. Res. Soc.*, **54**(12):1275-1282. [doi:10.1057/palgrave.jors.2601646]

Kaku, B., Mazzola, J.B., 1997. A tabu-search heuristic for the plant layout problem. *INFORMS J. Comput.*, **9**(4):374-384. [doi:10.1287/ijoc.9.4.374]

Kennedy, J., Eberhart, R.C., 1995. Particle Swarm Optimization. Proc. IEEE Int. Conf. on Neural Networks, **4**:1942-1948. [doi:10.1109/ICNN.1995.488968]

Kennedy, J., Eberhart, R.C., 1997. A Discrete Binary Version of the Particle Swarm Algorithm. Proc. World Multi-Conf. on Systemics, Cybernetics and Informatics, p.4104-4109.

Kennedy, J., Eberhart, R.C., Shi, Y., 2001. Swarm Intelligence. Morgan Kaufmann, San Francisco, CA.

Kulturel-Konak, S., 2007. Approaches to uncertainties in the facility layout problems: perspectives at the beginning of the 21th century. *J. Intell. Manuf.*, **18**(2):273-284. [doi:10.1007/s10845-007-0020-1]

Lacksonen, T.A., Enscore, E.E., 1993. Quadratic assignment algorithms for the dynamic layout problem. *Int. J. Prod. Res.*, **31**(3):503-517. [doi:10.1080/00207549308956741]

Liao, C.J., Tseng, C.T., Luarn, P., 2007. A discrete version of particle swarm optimization for flowshop scheduling problems. *Comput. Oper. Res.*, **34**(10):3099-3111. [doi:10.1016/j.cor.2005.11.017]

McKendall, A.R.Jr., Shang, J., 2006. Hybrid ant systems for the dynamic facility layout problem. *Comput. Oper. Res.*, **33**(3):790-803. [doi:10.1016/j.cor.2004.08.008]

McKendall, A.R.Jr., Shang, J., Kuppusamy, S., 2006. Simulated annealing heuristics for the dynamic facility layout problem. *Comput. Oper. Res.*, **33**(8):2431-2444. [doi:10.1016/j.cor.2005.02.021]

Onwubolu, G.C., 2002. Emerging Optimization Techniques in Production Planning and Control. Imperial College Press, London.

Poli, R., Kennedy, J., Blackwell, T., 2007. Particle swarm optimization. *Swarm Intell*., **1**(1):33-57. [doi:10.1007/s11721-007-0002-0]

Rosenblatt, M.J., 1986. The dynamics of plant layout. *Manag. Sci*., **32**(1):76-86. [doi:10.1287/mnsc.32.1.76]

Shi, Y., Eberhart, R.C., 1998. A Modified Particle Swarm Optimizer. Proc. IEEE Congress on Evolutionary Computation, p.169-173.

Shore, R.H., Tompkins, J.A., 1980. Flexible facilities design. *IIE Trans.*, **12**(2):200-205.

Tompkins, J.A., White, J.A., Bozer, Y.A., Frazelle, E.H., Tanchoco, J.M.A., Trevino, J., 1996. Facilities Planning. Wiley, New York.

Tseng, C.T., Liao, C.J., 2008. A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Int. J. Prod. Res*., **46**(17):4655-4670. [doi:10.1080/00207540701294627]

Urban, T.L., 1993. A heuristic for the dynamic layout problem. *IIE Trans*., **25**(4):57-63. [doi:10.1080/07408179308964304]