Expert Systems with Applications 36 (2009) 11221–11232

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Shortest path based simulated annealing algorithm for dynamic facility layout problem under dynamic business environment

Ming Dong [a,*], Chang Wu [a], Forest Hou [b]

[a] Antai College of Economics and Management, Shanghai Jiao Tong University, 535 Fahua Zhen Road, Shanghai 200052, PR China
[b] Intel Products (Shanghai) Ltd., 999 Yinglun Road, Pudong, Shanghai 200131, PR China

ARTICLE INFO

Keywords:
Dynamic layout
Adding/removing machines
Shortest path algorithm
Simulated annealing algorithm

ABSTRACT

This paper studies a new kind of dynamic multi-stage facility layout problem under dynamic business environment, in which new machines may be added into, or old machines may be removed from the plant. We define this problem first on the basis of unequal area machines and continual presentation of layouts. Compared with nodes and arcs of the flow chart, we convert this problem into a shortest path problem by studying its cost function and machine adding/removing heuristic rules, and the corresponding mathematical model for this problem is established. An auction algorithm is proposed here to solve the shortest path problem. Finally, a shortest path based simulated annealing algorithm is presented to solve the optimization problem. Parameters of the SP based SA algorithm are discussed to improve the performance of the algorithm. Some cases are used to verify the proposed algorithm.

1. Introduction and literature review

Today's manufacturing facility needs to be responsive to the frequent changes in product mix and demand while minimizing material handling and machine re-location costs. The dynamic plant layout problem (DPLP) extends the static plant layout problem (SPLP) by considering the changes in material-handling flow over multiple periods and the costs of rearranging the layout (Balakrishnan, Cheng, Conway, & Laub, 2003; Kusiak & Heragu, 1987). Different from the traditional DPLP, this paper investigates a dynamic facility layout problem with a characteristic of adding/removing machines in each period. That is, not only changes in material-handling flow and re-locations of existing machines, but also the new machines' adding and old machines' removing should be considered. The motivation behind this study is that, under dynamic business environment, as continuously introducing a wide range of products whose demands are variable and lifecycles are short, static facility layout is vulnerable to dynamic changes in requirements of products or machines, which will cause material handling inefficiency, prohibitive re-location costs (including production shutdown) and poor plant operational performances (Benjaafar, Heragu, & Irani, 2002). With the rapid introduction of new products, new machines need to be installed in the plant, and old machines will be removed from the plant when some products are eliminated from the current product line. For example, in semiconductor industry under fast changing business environment, life cycles of products become very short and types and amounts of products vary very fast, new machines and old machines may need to be added into/removed from the plant in multi-stages.

A layout plan for the DFLP can be represented as a series of layouts, and each layout is associated with a period. Therefore, the total cost of a layout plan consists of the sum of the material handling costs for all periods and the sum of the rearrangement costs. As indicated by McKendall and Shang (2006a, 2006b), the DPLP is not just a series of SPLPs/quadratic assignment problems. In traditional dynamic layout problems, the "dynamic" is resulted from the changes in the flow-to matrix of the materials between processes. This paper deals with a new type DPLP in which the number of machines in different planning periods is changing due to machines' adding/removing in each planning period. This machine changing data for each period are forecasted and can be obtained from so called PTL (POR Tool List, i.e., Plan of Record Tool List). Rosenblatt (1986) developed a dynamic programming (DP) model and solution procedure with branch and bound scheme for determining an optimal layout for each of several pre-specified future planning periods. This model takes into consideration material handling cost as well as cost of re-locating machines from one period to the next. In the heuristic DP method only a few layouts from each period are included. This procedure is to select the layouts in a period randomly. However this is not very effective. A better method is to select some of the better quality layouts in each period to form part of the DP. The DPLP is a combinatorial problem for which the optimal solution can be found only for very small problems. Heuristic procedures such as genetic algorithm for the dynamic layout problem can be found in a number of papers

* Corresponding author. Tel.: +86 21 34206101.
E-mail address: mdong@sjtu.edu.cn (M. Dong).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.02.091

including Conway and Venkataramanan (1994), Kochhar and Heragu (1999), and Urban (1998). Urban (1998) proposed an approach using a steepest-descent pairwise exchange heuristic similar to CRAFT (Armour & Buffa, 1963). Conway and Venkataramanan (1994), and Balakrishnan and Cheng (2000) make use of genetic algorithms (GA) for the DPLP while Kaku and Mazzola (1997) use Tabu Search. Baykasoglu and Gindy (2001) have developed a simulated annealing algorithm (SA) for the DPLP. Using the test problems from Balakrishnan et al. (2003), they showed that their algorithm performed better than the GA's of Conway and Venkataramanan, and Balakrishnan and Cheng (2000). The parameter settings in their SA algorithm involve determining the initial temperature (the probability that in the neighborhood search, an inferior solution will be accepted), the rate at which the temperature decreases (the decrease in the acceptance probability of an inferior solution), and the number of iterations. Wu, Chung, and Chang (2008) proposed a hybrid simulated annealing algorithm with mutation operator to solve the manufacturing cell formation problem considering multiple process routings for parts, so that either the intercellular movements are minimized or the grouping efficacy is maximized, depending on the definition of the decision objective.

Many previous studies of dynamic facility layout optimization problem are based on the changes in the material flow matrix. The problem is modeled as a QAP or bay structure problem using discrete presentation of facility layouts, which is difficult to be applied into the real world. While the quadratic assignment formulation of the layout problem has been deeply investigated there are very few papers solving it for departments of unequal size (Dunker, Radons, & Westkämper, 2005). Dunker et al. (2005) presented an approach which can handle unequal sizes of departments, which in addition may change from one period to the next. Their algorithm combines genetic search and dynamic programming. They described a representation of layouts which can be used as genetic code.

The representation of a DPLP solution forms the basis for a mathematical model and significantly impacts the structure and efficiency of the applied optimization algorithms (Liu & Meller, 2007). With a discrete representation, the facility is represented by an underlying grid structure with fixed dimensions and all departments are composed of an integer number of grids. However, shapes of the machines are not concerned for discrete representation, so it's difficult to define the real locations of machines. By representing the layout in a discrete fashion, the DPLP is simplified, but at the penalty of eliminating many solutions from consideration (Bos, 1993). Continual representation means that the locations of machines are represented continuously with the coordinates (x, y) of the machines (Drira, Pierreval, & Hajri-Gabouj, 2007). In a continuous representation, department dimensions are not restricted to an underlying grid structure. Continuous representation is more accurate and realistic than discrete representation, and thus, is capable of finding the "real optimal" layout solution. However, the continuous representation also increases the complexity of the DPLP (Cui & Huang, 2006; Das, 1993; Liu & Meller, 2007).

While there exist some literatures on DPLP investigation, up to our knowledge, there are no research has been implemented for this dynamic layout problem with unequal size machines and machines' adding/removing in each period. Machines' adding/removing in each period increases the complexity of DPLP. The aim of this research is to develop an effective method for DPLP with machines' adding/removing in each period. In order to obtain realistic layouts, continuous layout representation is adopted in this study.

The remainder of this article is organized as follows. In Section 2, a shortest path based mathematical model is presented. The machines' adding/removing heuristic rules are presented in Section 3. In Section 4, a hybrid simulated annealing algorithm with shortest path solution algorithm and heuristic rule is developed to find the optimal solution. Section 5 shows the computational results on a real-world application problem, and Section 6 concludes the paper.

2. Mathematical model

The result of dynamic layout problem is a series of layouts from initial layout to the last stage layout. The solution procedure can be regarded as how to choose a path with shortest distance (or minimal total cost for DPLP). A flow graph of n-period dynamic layout problem is shown in Fig. 1, in which directions of arrows are consistent with the planning time horizon. Assume that there are m feasible layouts chosen by some heuristic rules in each period. In Fig. 1, nodes represent feasible layouts in each period and values of edges are associated costs from current layout to the layout in the next period. Let s_0 be the initial layout. $(s_{11}, s_{12}, \ldots, s_{1m})$ represents m feasible layouts chosen by some heuristic rules in planning stage 1 and $(s_{21}, s_{22}, \ldots, s_{2m})$ and $(s_{n1}, s_{n2}, \ldots, s_{nm})$ are m feasible layouts chosen by some heuristic rules in planning stage 2 and stage n, respectively. In order to construct a shortest path optimization problem, a virtual node s_e is added at the rear of the nth period. Assume that layouts in the nth period can be transferred into the virtual layout without any cost. Now the dynamic layout problem becomes a $s_0 \rightarrow s_e$ shortest path problem.

Define 0–1 variable μ_{ij} as follows:

$$\mu_{ij} = \begin{cases} 1, & \text{if node } i \text{ and } j \text{ are directly connected} \\ 0, & \text{else} \end{cases} \tag{1}$$

Then the mathematical model for the shortest path based dynamic layout problem becomes:

$$\min C_t = \sum_{i,j} \mu_{ij} e_{ij} \tag{2}$$

s.t.

$$\mu_{ij} \in \{0, 1\}, \quad \forall (i,j) \in E \tag{3}$$

$$\sum\nolimits_j \mu_{ij} = m, \quad \forall (i,j) \in E \text{ and node } i \in P \tag{4}$$

$$\sum\nolimits_i \mu_{ij} = m, \quad \forall (i,j) \in E \text{ and node } j \in Q \tag{5}$$

$$e_{ij} \geqslant 0, \quad \forall (i,j) \in E \tag{6}$$

$$\sum\nolimits_i e_{ij} = 0, \quad \text{if node } j \text{ is virtual node } s_e \tag{7}$$

where C_t is total cost and μ_{ij} is a 0–1 variable indicating the direct connection relationship between node i and j. e_{ij} represents the cost of the edge between node i and j, and m is number of feasible layouts chosen by some heuristics in each stage. E and Ω are sets of all the edges and nodes, respectively. Let P be the set of all the nodes except the virtual node and nodes at the last period, Q be the set of all the nodes except the initial node and nodes at the first period,

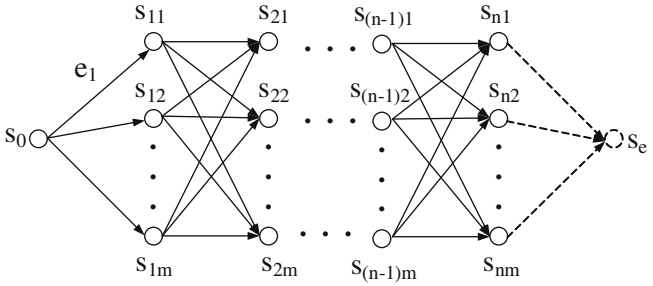


Fig. 1. A shortest path based representation of n-period dynamic layout problem.

and D be the set of all the nodes from the second stage to the $(n-1)$th stage. Then, we have $P \cup Q = \Omega$ and $P \cap Q = D$.

Constraint (3) means that there is at most one connecting edge between any two nodes. Constraint (4) indicates that, for every node i in P, there are m outgoing edges. Similarly, constraint (5) says that, for every node j in Q, there are m incoming edges. Constraint (6) means that values of edges are not less than 0. Constraint (7) says that the layouts in the last stage can be transformed into a virtual layout s_e without any cost.

When a plant has to change its layout, two kinds of cost may occur: static cost and dynamic cost. Static cost is the cost associated with the layouts within the planning periods that includes the material handing cost and the utilization of plant space. Dynamic cost is the transformation cost of facility layouts between different periods. Let e_{ij} be the static cost of s_j and $C(e_{ij})$ be the dynamic cost from s_i to s_j (see Fig. 2). For DPLP with machines' adding/removing in each period, the total cost consists of material handling cost C_m, area utilization cost C_a, re-location cost C_r, and shut-down cost C_s. Cost C_m and C_a are static costs. Material handling cost C_m occurs when material flows between machines in the production. It can be used to evaluate a static layout. When there are several identical machines, the distance between machines is calculated by the distance between the centroidal points of machine groups. Area utilization cost C_a is a penalty cost for unused space. The functions of C_m and C_a are given as follows, respectively:

$$C_m = \sum_{\substack{ij \in I \\ i<j}} w_1 X_{ij} p_{ij} d_m \left(|\bar{x}_i - \bar{x}_j| + |\bar{y}_i - \bar{y}_j| \right) \tag{8}$$

$$C_a = w_2 d_a \cdot \frac{\sum_{u=1}^{N_s} S_u}{S} \tag{9}$$

where w_1 and w_2 are weights of material handling cost C_m and area utilization cost C_a, respectively. $\bar{x}_i$, $\bar{y}_i$, $\bar{x}_j$ and $\bar{y}_j$ are coordinates of machines i and j, respectively. X_{ij} is a binary variable indicating whether or not node i and j are directly connected. p_{ij} represents the material amount between node i and j. d_m is material handling cost per unit and d_a represents the cost per unit area. N_s is total number of feasible spaces in the plant, S_u is area of uth free space and S is the total area of the plant.

C_r and C_s are dynamic costs. Re-location cost C_r occurs when a planning period begins. Shut-down cost C_s occurs when some machines have to be shutdown due to the machines' re-locating processes. In this situation, the production capacity will decrease. C_r and C_s can be expressed as follows:

$$C_r = \sum_{i \in M} w_3 Y_i m_i \tag{10}$$

$$C_s = \sum_{i \in M} [w_4 Z_i C_p (W_{io} - W_{iu}) t_i] \tag{11}$$

where w_3 and w_4 are weights of re-location cost C_r and shut-down cost C_s, respectively. Y_i is a binary variable indicating whether or not machine i is relocated and m_i is the re-location cost for machine i. Z_i is a binary variable indicating whether or not machine i is temporarily shutdown or its capacity decreases due to the re-location. C_p is the profit gained from a product. W_{io} andW_{iu} are capacities of the production system (in terms of amount of products produced per unit time) before and after machine$i$ is temporarily shutdown, respectively. t_i is the time duration of machine i being shutdown. Therefore, the total cost C becomes

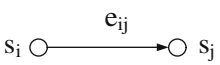


Fig. 2. Node and arc.

$$C = C_m + C_a + C_r + C_s \tag{12}$$

from which e_{ij} can be obtained.

3. Solution framework and heuristic procedures

3.1. Solution framework of shortest path based SA algorithm

In order to transform a dynamic layout problem with machines' adding/removing in each planning period into a shortest path based flow graph, two key issues need to be solved. The first one is how to generate nodes (layouts) and the second one is how to compute values of edges. The second issue has already been addressed through cost analysis above, and in the following, how to generate new nodes (layouts) will be studied.

A general solution framework of shortest path based SA algorithm for this new DPLP is illustrated in Fig. 3. The procedure consists of the following steps: in step 1, generate the initial solution S (S consists of layouts $L_0, L_{10}, \ldots, L_{n0}, L_e$ in different stages) from L_0. In step 2, construct the neighborhood of S by generating m feasible layouts at each period by using proposed heuristics rules. Construct a shortest path problem from the neighborhood of S and denote S' (S' consists of layouts $L_0, L'_{10}, \ldots, L'_{n0}, L_e$ in different stages) as the solution of the shortest path problem in step 3. In step 4, if the total cost of S (denoted as $C(S)$) is larger than the total cost of S' (denoted as $C(S')$), then update S by S'. Else if $\exp\{[C(S) - C(S')]/T_k\} > \text{random}[0, 1)$, then $S = S'$ (where T_k is the temperature at step k). In step 5, let $T_{k+1} = d(T_k)$ (where $d(T_k)$ is the temperature decreasing function) and $k = k + 1$. In step 6, repeat above computing procedure until temperature is below the fixed value T_c or other stop criteria are met.

3.2. Free-space searching rule

In this paper, free-space is defined as the space that can be used for placing machines. Removing machines will generate more free-space and adding machines will make use of some free-space. Therefore, free-space searching is a necessary step before adding/removing machines. This paper adopts continual representation for layouts, that is, locations of machines are represented with coordinates of the machines. Within the allowable free-space, machines can be placed in any position in the plant. With the fixed step length st, all free-space can be found by searching the space area of plant from the top-left corner to the bottom-right corner. Machines could move continuously within the free-space area.

Theorem 1. *Let ss be length of the shortest side of all the machines, if the fixed step length st is no more than ss, then no feasible space will be missed.*

Proof. Plant area A can be divided into two parts: used part U and unused part F, $U \cup F = A$ and $U \cap F = \emptyset$. When searching plant area A with fixed step st in some direction, suppose current search point is v ($v \in U$), then the next search point is $v + st$. If $(v + st) \in F$, unused space can be found; if $(v + st) \in U$, some unused space may be missed, that is, there may exist $(a, b) \in F$ and $(a, b) \in (v, v + st)$. Apparently, the longest length of (a, b) is st in the worst situation. That is, for a search space with fixed step st, in the worst case, some unused space with side-length st will be missed. Notice that free-space is defined here as space used for placing machines. Therefore, as long as $st \leqslant ss$, no feasible space will be missed.

For a searching point, if it is located inside some machines or out of the plant space, go to the next searching point. Otherwise, enlarge the rectangle around this point until all its four sides touch some machines. And this rectangle becomes a free-space. Fig. 4 shows an example about how to search for a free-space for placing machines. □

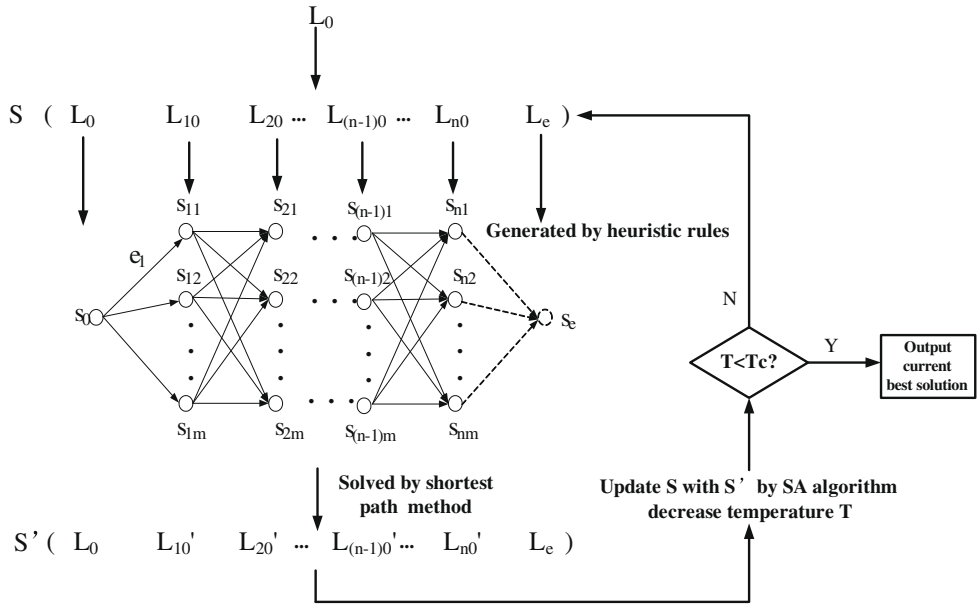


Fig. 3. The general solution framework of shortest path based SA algorithm.

3.3. Heuristics rules for machines' removing/adding

At each planning period, the information on machines that will be removed and added can be found from PTL. A general rule for machines' adding/removing is that, in order to make enough room for adding, the machines on the remove list will be removed first before adding any new machines.

3.3.1. Machines' removing rules

Removing procedure will have critical impact on the following machines' adding stage. If the removing method is appropriate then there should be as large as possible room near the locations where the new machines would be installed. The basic procedure for machines' removing rule is as follows. For a machine to be removed, identify its group (a set of machines with the same type). Searching for all the neighborhood machines nearby the group of machines, check if there is any machine that is same-type machine as those to be added into the plant in the next period, if so, remove them (see Fig. 5 as an example); if not, check if there is any machine that is of the same kind as those to be removed from the plant in the next period, if so, remove them. Otherwise, choose a machine that could increase most useful space. In Fig. 5, machine "3" is going to be added in the next period. So removing machine "A" next to machine "3" in the current period will generate more complete space in the future. The flowchart for machines' removing rule is given in Fig. 6.

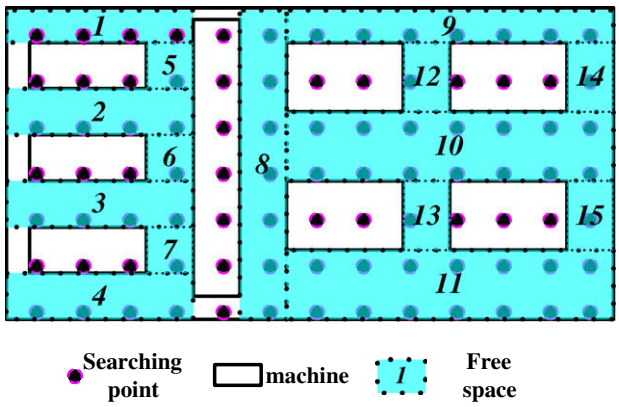


Fig. 4. An example of free-space searching.

3.3.2. Machines' adding rules

Once removing procedure is finished, new machines can be added into the plant. First, rank all the machines on the adding machines' list based on the areas of machines. When adding machines into the plant, there exist two strategies: (1) put it directly into a free-space, and (2) move other machines temporarily to make room for this new machine. The reason for temporary move is that there is not enough room or it could decrease total cost. For continual representation of layouts, the generation of new layout solutions is different, so we should decide which strategy to take.

The way for the first strategy is searching for a feasible space and adding machines into it (see Fig. 7 for illustration). In Fig. 7, if machine A is added into a free-space directly, we will have a layout as shown in Fig. 7b. However, in practice, the machines that have the same type will usually stay together. Therefore, the layout result (temporarily removing a machine 5 before adding machine A) in Fig. 7c is more suitable for industrial requirements.

The basic procedure for the second strategy is as follows. For the machine that is the same type as the new adding machine, searching for the space adjacent to this machine. In terms of the space size found, there are two cases: (1) If the new machine can be put into this space, remove the former one away and put the new one into it. The former one will be put into a free-space that is closest to other same-type machines (see Fig. 8). (2) Otherwise, choose a space that could at least satisfy the length requirement of one side and try to satisfy the requirement of its other side. Then all the machines located within these spaces should be removed and put the new one into the space. All the machines temporarily moved will be put into the nearest free-spaces according to the area-rankings of machines (see Fig. 9).

When new machines are added into the plant directly, only material handling cost C_m and area utilization cost C_a change, the changing cost is:

$$\Delta C^0 = \Delta C_m^0 + \Delta C_a^0 \tag{13}$$

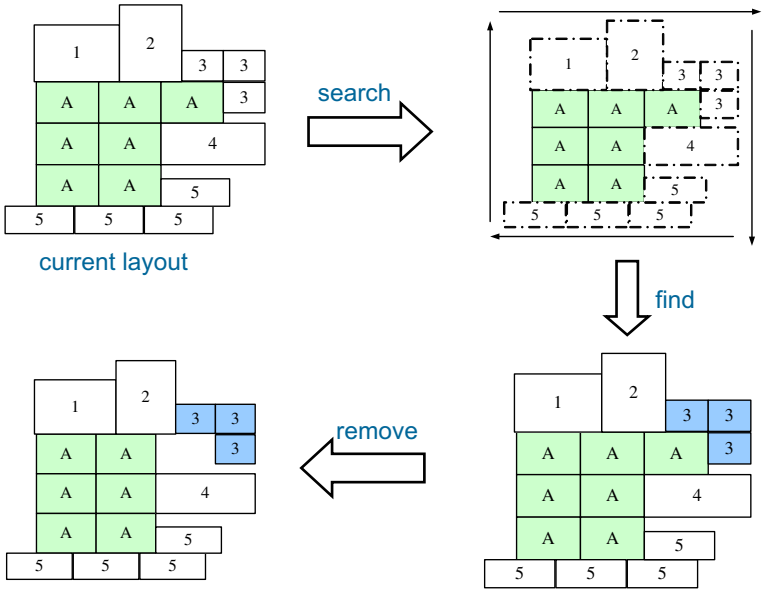


Fig. 5. An example of removing machines.

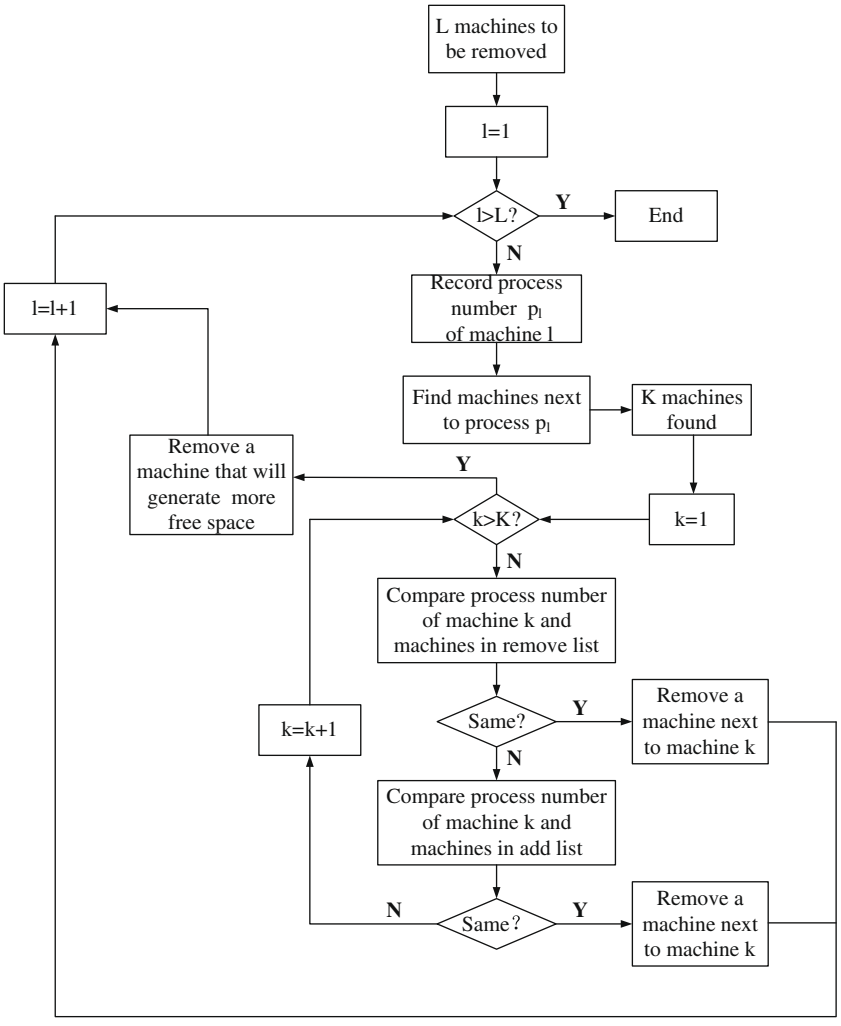


Fig. 6. Flowchart for machines' removing rule.

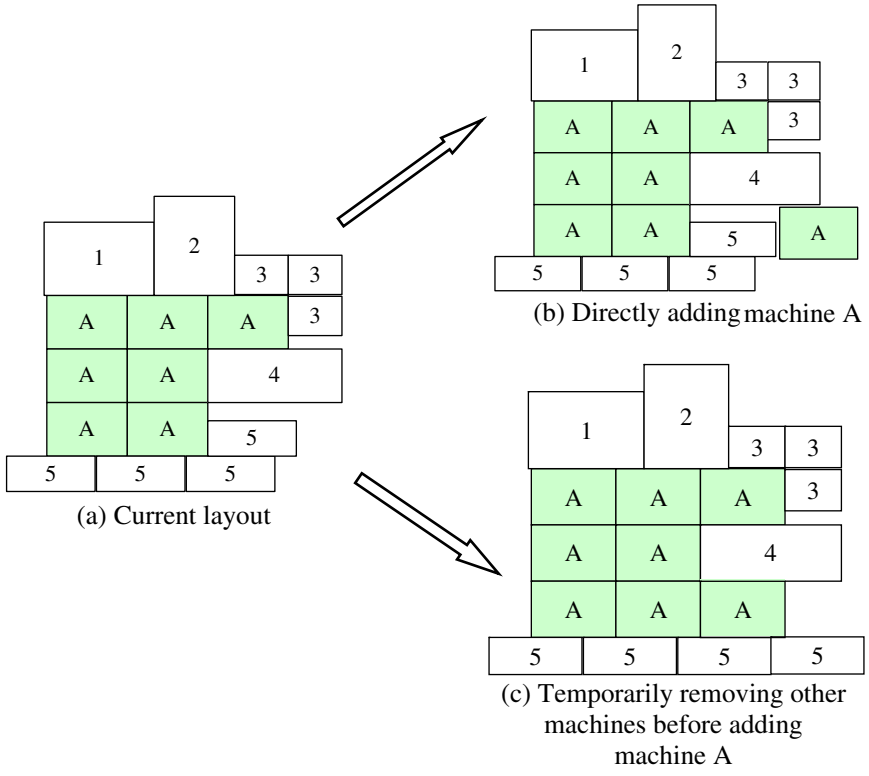


Fig. 7. An example of directly adding machines.

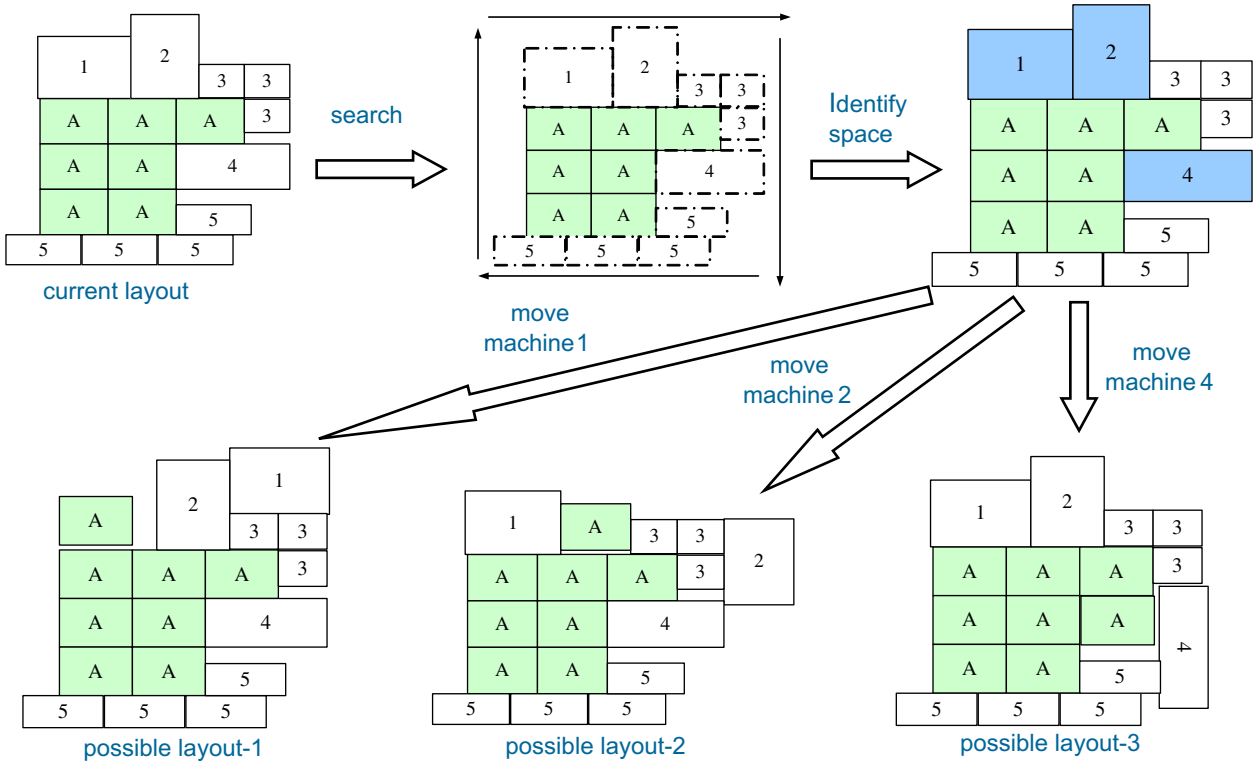


Fig. 8. Machines' adding rule: one machine' moving (case 1).

When temporarily removing other machines is necessary in order to put new machines into the plant, four cost may all change, in which re-location cost will increase and shut-down cost will not decrease. So the changing cost becomes:

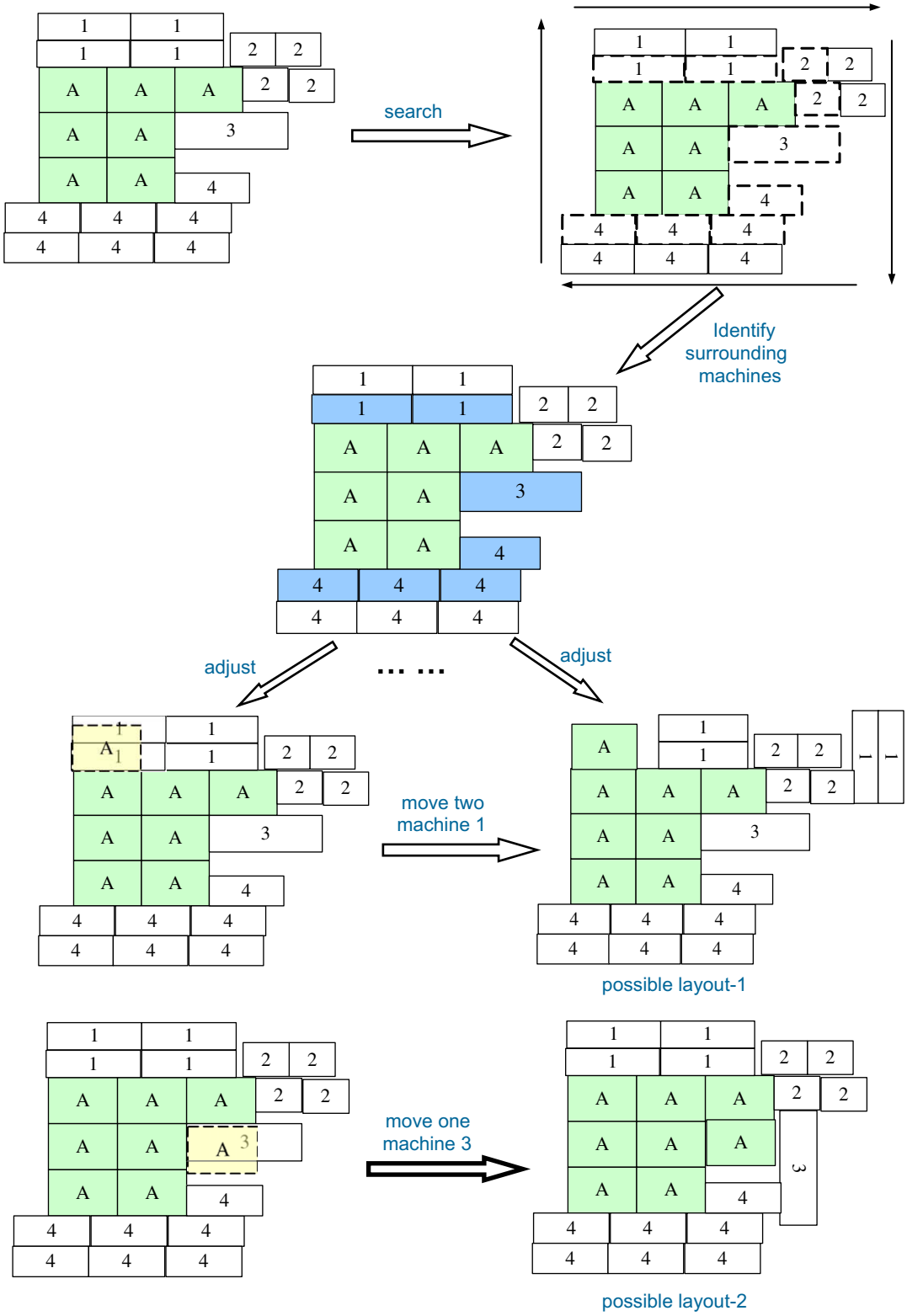


Fig. 9. Machines' adding rule: multiple machines' moving (case 2).

$$\Delta C^1 = \Delta C^1_m + \Delta C^1_r + \Delta C^1_s + \Delta C^1_a \tag{14}$$

When adding machines, randomly generate a direct putting strategy for this machine with the changing cost ΔC^0 and an indirect putting strategy (i.e., temporarily removing other machines) for this machine with the changing cost ΔC^1. According to greedy strategy, try to increase as little cost as possible, that is, choose the smaller one between ΔC^0 and ΔC^1 and its corresponding strategy on adding the machine into the plant. The flowchart for machines' adding rule is given in Fig. 10.

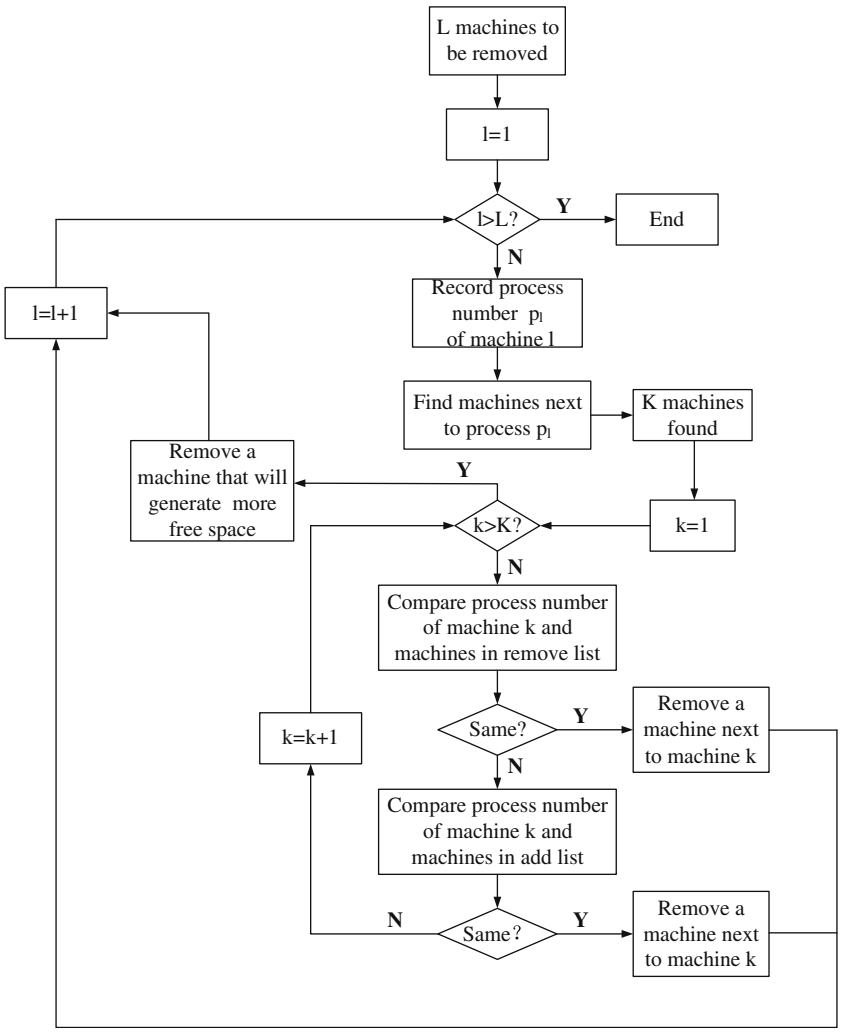


Fig. 10. Flow chart for machines' adding rule.

3.4. Auction algorithm for solving shortest path problem

Traditional shortest path algorithm is Dijkstra algorithm, but its performance is poor in large scale problems. Auction algorithm is a famous shortest path algorithm presented by Bersekas (1991), and it can be used to solve large scale problems. Bertsekas reduces the computational complexity of the algorithm to $O(p^2)$ (where p is number of nodes) in the auction algorithm (Bersekas, 1991). And for the no-looped auction algorithm, the computational complexity is reduced to $O(q)$ (where q is number of arcs, Bersekas, 1992).

Theorem 2. *In the flow graph of dynamic layout problem with m feasible layouts in each period, Auction algorithm with no-looped graph has lower computational complexity.*

Proof. Let m be number of feasible layouts in each period. Since values from nodes in the last period to virtual node s_e is 0, so the connections between nodes in the last period and virtual node s_e could be ignored. For a n-period dynamic layout problem, p is $(nm+1)$ and q is $(n-1)m^2+m$, let $a=p^2-q$, then we have $a=nm^2(n-1)+m(m-1)+2nm+1$. In real-world multi-stage dynamic layout problems, number of planning periods n is larger than 1 and number of feasible layouts within each period $m \geqslant 1$. Therefore, we have $a>0$. In this problem, $\forall p$ and q, $p^2>q$, so auction algorithm with no-looped graph has lower computational complexity (i.e., $O(q)$). □

Suppose when solving a shortest path problem, an intermediate route $U(s_0, s_1, \ldots, s_{k-1}, s_k)$ is obtained. According to the auction algorithm, there exists a node s_{k+1} connecting to s_k, and the route becomes the shortest path in the next round computation. Then we can add node s_{k+1} into route and it becomes $U(s_0, s_1, \ldots, s_{k-1}, s_k, s_{k+1})$. This is called adding operation. If there is no such shortest path exists, delete node s_k from route and it becomes $U(s_0, s_1, \ldots, s_{k-1})$. This is called subtracting operation.

According to the relaxation conditions of auction algorithm (E is the set of arcs):

$$\begin{cases} \pi_i \leqslant e_{ij} + \pi_j, \forall (i,j) \in E \\ \pi_i = e_{ij} + \pi_j, \forall (i,j) \in U \end{cases} \tag{15}$$

Assign values π for each node (π_i for node i, generally the initial $\pi_i = 0$). The shortest path U can be obtained by using adding and subtracting operations. The procedure of auction algorithm with no-looped graph is as follows (where A is the plant space):

Step 1: Start from the initial node s_o.
Step 2: When current node becomes k, if node k is virtual node s_e, go to step 6.
Step 3: If $\pi_k \geqslant \min_{(k,j)\in A}\{e_{kj}+\pi_j\}$, go to step 4; else if $\pi_k < \min_{(k,j)\in A}\{e_{kj}+\pi_j\}$, go to step 5.
Step 4: Add node j, then $\pi_j = \pi_k - e_{kj}$, go back to step 3.

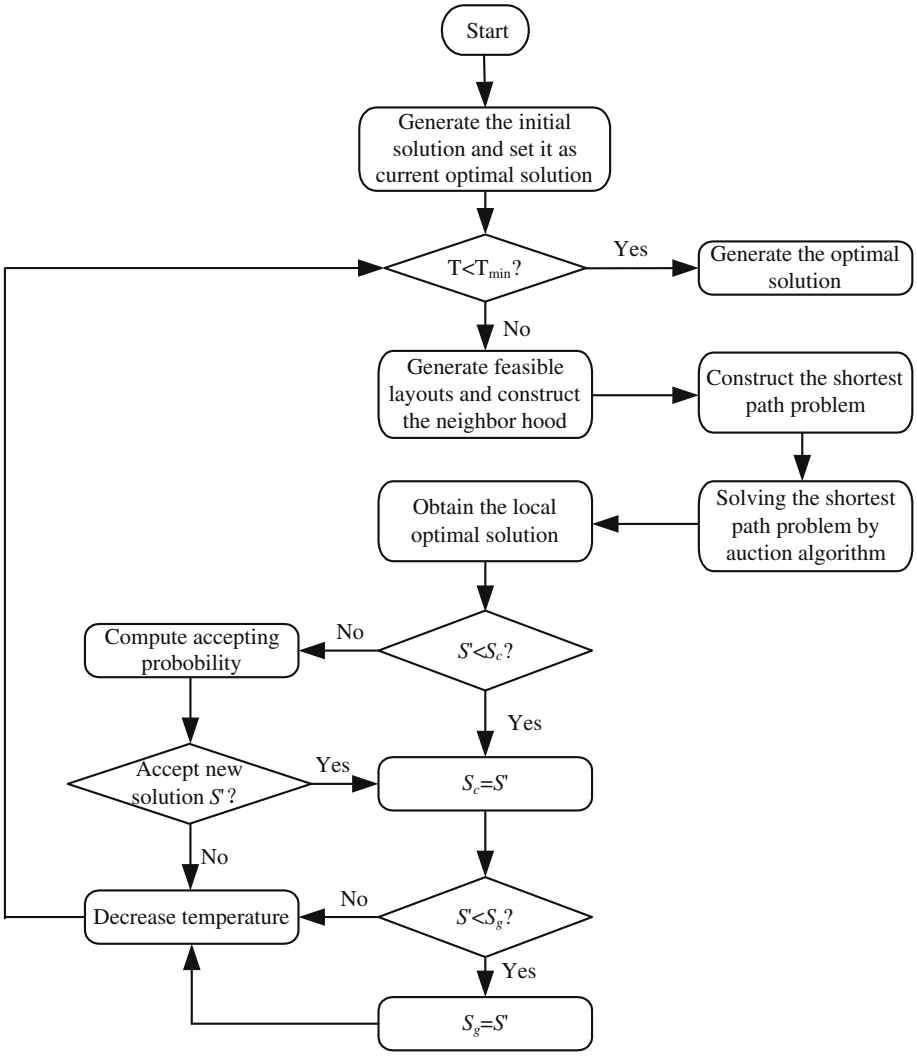


Fig. 11. Flow chart for shortest path based SA algorithm.

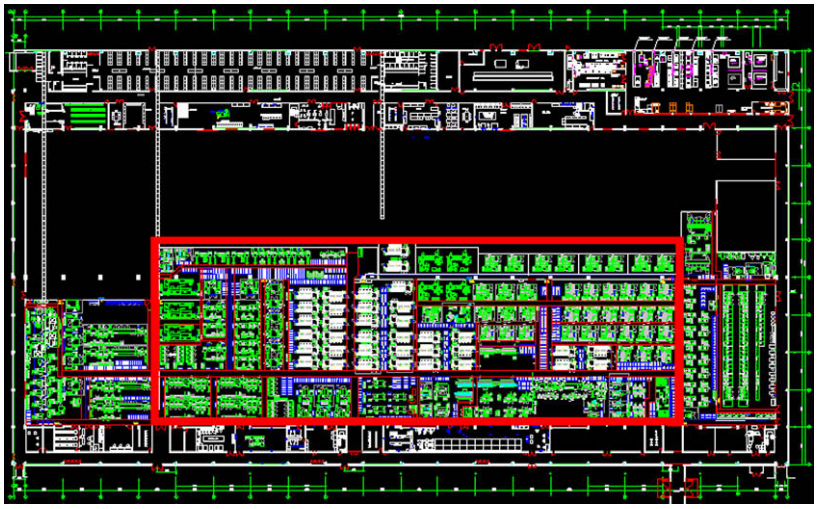


Fig. 12. The overall view of the studied plant layout.

Step 5: Subtract node j, then $\pi_k = \min_{(k,j)\in A}\{e_{kj} + \pi_j\}$, go back to step 3.

Step 6: Output current shortest path U and its length $C_t = \sum_{(i,j)\in U} e_{ij}$.

4. Very fast SA algorithm

In the DPLP with machines' adding/removing, there are infinite feasible layouts in each period. Therefore, auction algorithm for shortest paths combined with simulated annealing algorithm is used to solve this combinatorial optimization problem. In each iteration, m feasible layouts will be generated for a period and layouts at all periods are used to construct a shortest path problem. Auction algorithm is used to solve the shortest path problem and simulated annealing algorithm is used for searching optimal (or near optimal) solution in the solution space.

In simulated annealing algorithm, cooling scheduling is about how the temperature decreases. The basic requirement is that the temperature should decrease neither too fast which is not stable, nor too slowly which will result into low efficiency of the algorithm. Very fast SA has the same basic process as the traditional SA, and the difference between them lies in cooling schedule and accepting probability. Traditional SA can be extended to use any reasonable generating function $g(x)$, without relying on the principles underlying the ergodic nature of statistical physics (Ingber, 1989). The cooling schedule here is:

$$T_c = T_0 \alpha^{r-1}, \quad r = 1, 2, \ldots, L \tag{16}$$

where T_c is current temperature, T_0 is the initial temperature, α is the cooling coefficient (here, $\alpha = 0.9$), and r is the times of temperature decreasing.

Accepting probability is the probability that it accepts worse solutions. It allows the algorithm jumps out from local optimal solutions and searches for global optimal solutions. Generally, when temperature is high, this probability is high. As the temperature becomes lower, this probability gradually decreases to 0. And then the algorithm stops. Based on Boltzmann–Gibbs distribution, a new accepting probability is given as follows:

$$P = [1 - (1-h)\Delta E/T]^{1/(1-h)} \tag{17}$$

where $\Delta E = E(m') - E(m_c)$, $h \in R$. When $h \to 1$, this accepting probability turns into:

$$P = \exp(-\Delta E/T) \tag{18}$$

which is the accepting probability of the traditional SA algorithm.

The initial temperature T_0 should be neither too high nor too low. Here, the average value increased from several random changes is used to generate the initial temperature. Given initial temperature T_0, m new solutions are generated, in which there are a solutions that will decrease the objective function value and b solutions that will increase the objective function value. That is, $m = a + b$. Thus, we can get the initial acceptance probability of solutions χ as follows:

$$\chi = \frac{a + b[1 - (1-h)\overline{\Delta f}/T_0]^{1/(1-h)}}{a+b} \tag{19}$$

Let $a = k \cdot b$, and $\overline{\Delta f}$ be the average value increased from b solutions. Then the initial temperature becomes:

$$T_0 = \frac{1-h}{1 - [(k+1)\chi - k]^{(1-h)}} \cdot \overline{\Delta f} \tag{20}$$

Then, the algorithm procedure of shortest path based SA is as follows:

Step 1: Generate initial solution $S_0(L_0, L_1, \ldots, L_i, \ldots, L_n)$, set the times of temperature decreasing $r = 1$.
Step 2: Let current solution $S_c = S_0$ and the best solution $S_g = S_0$, its cost is $C(S_g)$.
Step 3: Use machines' adding/removing heuristic rules to generate m feasible layouts for each period.
Step 4: Use initial layout, feasible layouts in each period, and virtual node to construct a shortest path problem, then compute values of edges as follows: $e_{ij} = C = C_m + C_a + C_r + C_s$.
Step 5: Solve current shortest path problem with auction algorithm (denote U as the shortest path). That is, current optimal layout solution is $S'(L_0, L'_1, \ldots, L'_i, \ldots, L'_n)$ and the associated cost is $C(S') = \sum_{(i,j)\in U} e_{ij}$.
Step 6: Compute $\Delta E = C(S') - C(S_c)$. If $\Delta E < 0$, or{$\Delta E > 0$ and $random(0,1) < P(\Delta E)$}, then let $S_c = S'$ and $C(S_c) = C(S')$. Else if $C(S') < C(S_g)$, then the current best solution $S_g = S'$ and $C(S_g) = C(S')$.
Step 7: Let $r = r + 1$, compute new temperature $T_c = T_0 \ \alpha^{r-1}$. If $T_c < T_{\min}$, then stop the algorithm and generate the best solution S_g as the output. Otherwise, go back to step 3.

The flow chart of the above algorithm is given in Fig. 11.

5. Results and analysis

A real-world case of 4-period dynamic layout with 56 machines from semiconductor manufacturing industry will be used as an example to illustrate the proposed solution framework. The overall plant layout is given in Fig. 12 (the area inside the red lines is studied in this paper). The initial layout of the plant is given in Fig. 13. The planned tool changing list for each period is provided in Table 1.

Solving the above DPLP problem using the proposed solution approach, the facility layouts for different stages are provided in Fig. 14.

In the following, for DPLP with machines' adding/removing, the results of shortest path based SA algorithm are compared with those by SA alone (see Fig. 15).

It can be seen that, compared with the traditional SA, shortest path based SA not only has a better performance (average cost decreased by 3.42%), but also has a faster runtime (average runtime decreased by 52.3%). This makes it suitable for application in large scale DPLP problems.

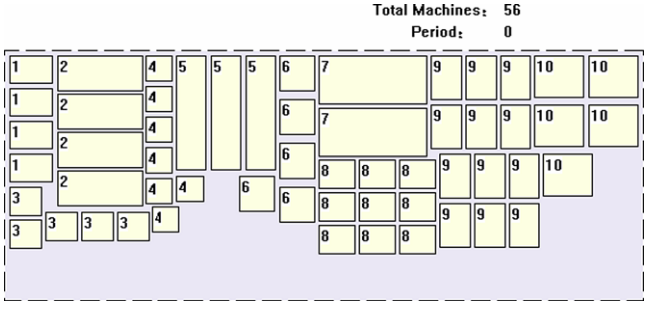


Fig. 13. Initial facility layout.

Table 1
Planned tool changing list for each period.

Stage	Machines added	Machines removed
1	Process_4 + 1	Process_2 − 1 Process_3-2
2	Process_5 + 1 Process_6 + 1	Process_8 − 3
3	process-1 + 1 process-4 + 1	Process_9 − 2
4	Process_3 + 3 Process_6 + 2 Process_10 + 1	Process_7 − 1

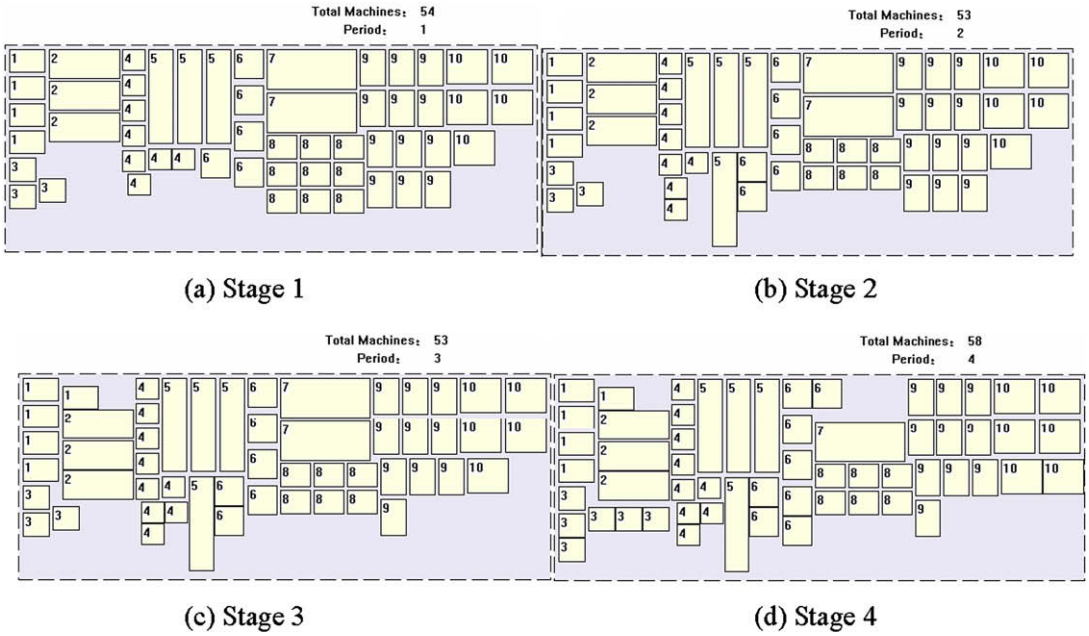
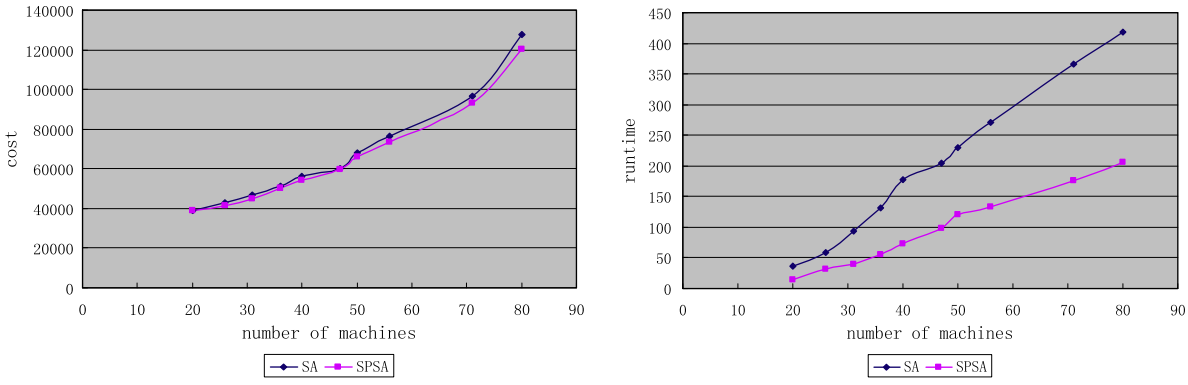


Fig. 14. Facility layouts of each period.



(a) Cost and scale of problem (b) Runtime and scale of problem

Fig. 15. Performance comparison between shortest path based SA and SA algorithm.

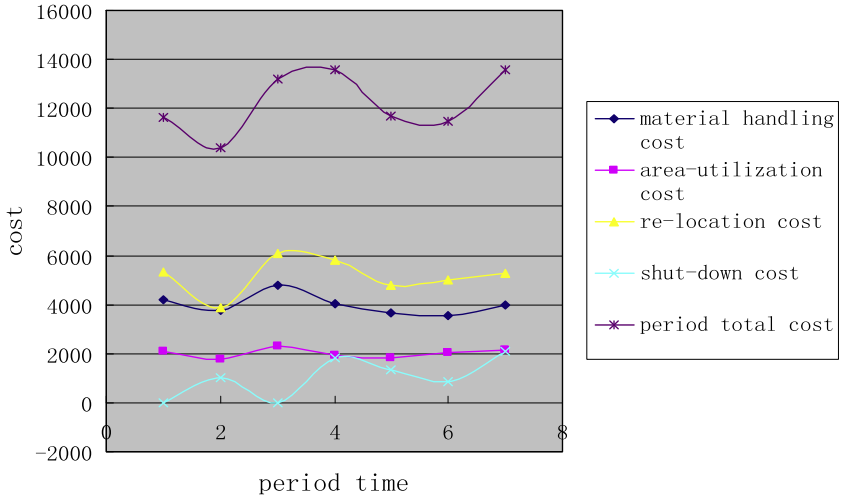


Fig. 16. Cost fluctuations in a case of 7-periods and 50-machines.

Another case of 7-period dynamic layout with 50 machines is studied for the cost fluctuation analysis. Fig. 16 shows how material handling cost, area utilization cost, re-location cost, shut-down cost, and periodic total cost fluctuate along the planning periods. It can be seen that shut-down cost could be zero and it varies rapidly. Two static costs (material handling cost and area utilization cost) have a similar trend and a relatively small fluctuation, while two dynamic costs (re-location cost and shut-down cost) have a relatively big fluctuation. This shows that the difficulty of DPLP with machines' adding/removing lies in how to reduce and smooth the re-location cost and shut-down cost of machines.

6. Conclusions

This paper studies a new kind of dynamic layout problem with machines' adding/removing at different planning periods. In the problem, machines have unequal sizes and are represented by continual coordinates. This paper proposes free-space searching rules, heuristic rules for adding/removing machines at different periods and the problem is transformed into a shortest path problem. An auction algorithm is provided to solve this shortest path problem. A general solution framework of shortest path based SA algorithm for this new DPLP is given. An industrial case of 4-period dynamic layout with 56 machines is employed to illustrate the proposed methodology and the results show that the proposed algorithm is efficient and effective.

This paper assumes that the machines' changing list is known and can be obtained from product demand forecasting. In the future research, this problem could be extended to consider uncertain machines' changing list. Besides, more actual requirements in the production such as aisles may be taken into consideration in the future research.

Acknowledgement

The work presented in this paper has been supported by grants from the National High Technology Research and Development Program ("863" Program) of China (2008AA04Z104) and Intel Products Ltd.

References

Armour, G. C., & Buffa, E. S. (1963). A heuristic algorithm and simulation approach to relative allocation of facilities. *Management Science, 9*(2), 294–300.

Balakrishnan, J., & Cheng, C. H. (2000). Genetic search and the dynamic layout problem: An improved algorithm. *Computers and Operations Research, 27*(6), 587–593.

Balakrishnan, J., Cheng, C. H., Conway, D. G., & Laub, C. M. (2003). A hybrid genetic algorithm for the dynamic plant layout problem. *International Journal of Production Economics, 86*(2), 107–120.

Baykasoglu, A., & Gindy, N. N. Z. (2001). A simulated annealing algorithm for dynamic facility layout problem. *Computers and Operations Research, 28*(14), 1403–1426.

Benjaafar, S., Heragu, S., & Irani, S. (2002). Next generation factory layouts: Research challenges and recent progress. *Interfaces, 32*(6), 58–76.

Bersekas, D. P. (1991). An auction algorithm for shortest paths. *SIAM Journal for Optimization, 1*(4), 425–447.

Bersekas, D. P. (1992). Modified auction algorithms for shortest paths, Technical Report, Laboratory for Information and Decision. Massachusetts Institute of Technology.

Bos, J. (1993). Zoning in forest management a quadratic assignment problem solved by simulated annealing. *Journal of Environmental Management, 37*(2), 127–145.

Conway, D. G., & Venkataramanan, M. A. (1994). Genetic search and the dynamic facility layout problem. *Computers and Operations Research, 21*(8), 955–960.

Cui, Y. D., & Huang, L. (2006). Dynamic programming algorithms for generating optimal strip layouts. *Computational Optimization and Applications, 33*(2–3), 287–301.

Das, S. K. (1993). A facility layout method for flexible manufacturing systems. *International Journal of Production Research, 31*(2), 279–297.

Drira, A., Pierreval, H., & Hajri-Gabouj, S. (2007). Facility layout problems: A survey. *Annual Reviews in Control, 31*(2), 255–267.

Dunker, T., Radons, G., & Westkämper, E. (2005). Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. *European Journal of Operational Research, 165*(1), 55–69.

Ingber, L. (1989). Very fast simulated re-annealing. *Mathematical and Computer Modeling, 12*(8), 967–973.

Kaku, B., & Mazzola, J. B. (1997). A tabu-search heuristic for the plant layout problem. *INFORMS Journal on Computing, 9*(4), 374–384.

Kochhar, J. S., & Heragu, S. S. (1999). Facility layout design in a changing environment. *International Journal of Production Research, 37*(11), 2429–2446.

Kusiak, A., & Heragu, S. S. (1987). The facility layout problem. *European Journal of Operational Research, 29*, 229–251.

Liu, Q., & Meller, R. D. (2007). A sequence-pair representation and MIP-model-based heuristic for the facility layout problem with rectangular departments. *IIE Transactions, 39*(4), 377–394.

McKendall, A. R., Jr., & Shang, J. (2006a). Hybrid ant systems for the dynamic facility layout problem. *Computers and Operations Research, 33*(3), 790–803.

McKendall, A. R., Jr., & Shang, J. (2006b). Simulated annealing heuristics for the dynamic facility layout problem. *Computers and Operations Research, 33*(8), 2431–2444.

Rosenblatt, M. J. (1986). The dynamics of plant layout. *Management Science, 32*(1), 76–86.

Urban, T. L. (1998). Solution procedures for the dynamic facility layout problem. *Annals of Operations Research, 76*(1), 323–342.

Wu, T. H., Chung, S. H., & Chang, C. C. (2008). Hybrid simulated annealing algorithm with mutation operator to the cell formation problem with alternative process routings. *Expert Systems with Applications, 36*(2P2), 3652–3661.