Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 853586, 19 pages
http://dx.doi.org/10.1155/2013/853586

Research Article

A Slicing Tree Representation and QCP-Model-Based Heuristic Algorithm for the Unequal-Area Block Facility Layout Problem

Mei-Shiang Chang and Ting-Chen Ku

Department of Civil Engineering, Chung Yuan Christian University, 200 Chung Pei Road, Chungli 32023, Taiwan

Correspondence should be addressed to Mei-Shiang Chang; mschang@cycu.edu.tw

Received 4 July 2013; Revised 3 October 2013; Accepted 5 October 2013

Academic Editor: Yi-Chung Hu

The facility layout problem is a typical combinational optimization problem. In this research, a slicing tree representation and a quadratically constrained program model are combined with harmony search to develop a heuristic method for solving the unequal-area block layout problem. Because of characteristics of slicing tree structure, we propose a regional structure of harmony memory to memorize facility layout solutions and two kinds of harmony improvisation to enhance global search ability of the proposed heuristic method. The proposed harmony search based heuristic is tested on 10 well-known unequal-area facility layout problems from the literature. The results are compared with the previously best-known solutions obtained by genetic algorithm, tabu search, and ant system as well as exact methods. For problems O7, O9, vC10Ra, M11*, and Nug12, new best solutions are found. For other problems, the proposed approach can find solutions that are very similar to previous best-known solutions.

1. Introduction

Facility layout problems (FLPs) occur in many areas, including manufacturing cell design, hospital design, and service center design. Recognized as one of the most critical and difficult design tasks, FLP aims to find the optimal arrangement of a given number of nonoverlapping departments with unequal-area requirements within a facility, so as to minimize the costs associated with projected interactions between departments. The common objective is to minimize the total material workflow costs among departments. Tompkins et al. [1] demonstrated that such minimization can lead to a cost reduction of 10–30%. A solution to the FLP specifies the relative location and the dimensions of department in the facility. After a block layout has been accomplished, a detailed layout can be designed, including aisle structures, input/output points of departments, and department locations. A low cost block layout is a good starting point to complete a final facility layout design.

FLP was originally formalized by Armour and Buffa [2]. The department areas do not have to be the same, but each department should follow certain ratio constraints or minimum length constraints. A thorough survey of the FLP is given by Gau and Meller [3] as well as Drira et al. [4]. In short, recent areas of research in facility layout design have applied advanced several exact methods, heuristics, metaheuristics, and hybridization of different metaheuristics for solving the FLP. However, FLP is a very challenging nonlinear combinatorial optimization problem to be solved optimally. Exact approaches are limited as to the size of the problem that can be optimally solved due to computational intractability. As a result, many facility layout problems have focused on heuristic and metaheuristic approaches to find good solutions.

Metaheuristics can be distinguished into global search methods (tabu search and simulated annealing) and evolutionary approaches (genetic algorithm and ant colony algorithm). A very important problem when developing a metaheuristic is related to the coding of a candidate floor plan. The FLP can be represented by many schemes that define how departments can be arranged within a facility area. The slicing tree structure (STS) formulation, first defined by Otten [5], is a continual layout representation to code a candidate layout into a tree structure. By recursively dividing a rectangular area in either the horizontal or vertical direction, STS formulation allows departments to be located

in divided zones. STS formulation takes full advantage of shape and orientation flexibility, although this layout representation requires complex repairing procedures to ensure feasibility of the layout that it represents. Compared to the flexible bay structure (FBS) formulation, STS formulation is less restrictive than the FBS formulation. STS representation is a good choice for handling various placement constraints. However, the STS representation has received only limited attention from researchers.

A harmony search (HS) algorithm is a recently developed evolutionary metaheuristic. The HS algorithm proposed by Geem et al. [6] is a metaheuristic optimization algorithm based on the musical process of searching for a perfect state of harmony, such as jazz improvisation. It has been very successful in a wide variety of discrete optimization problems and has been successfully applied to many engineering and system optimization problems [7, 8]. Compared to other metaheuristics in the literature, the HS algorithm offers a simple concept, few parameters, easy implementation, and fewer mathematical requirements and does not require initial value settings of the decision variables [9]. However, it has not been previously applied to the unequal-area FLP.

Given the foregoing, we propose to solve the FLP using a harmony search based algorithm encoded by a slicing structure in order to overcome the simplifying assumptions of the FBS and not to suffer from algorithm parameters selection. Different from previous researches, spatial relationships of departments (relative department positions) inferred from an STS solution are used to determine department locations and shapes by solving a proposed quadratically constrained program (QCP) in this study.

The rest of this paper is organized as follows. Section 2 introduces the literature review and discusses its findings. In Section 3, the details of the authors' algorithm are described. In Section 4, the proposed algorithm is extensively tested on several problems from the literature. Promising results are obtained. The proposed algorithm can find the new best STS solutions for a few well-known FLP test problems. Section 5 concludes the paper.

2. Literature Review

2.1. Metaheuristic for Solving FLPs with STS. Metaheuristic approaches such as the tabu search (TS), the genetic algorithm (GA), and the ant system (AS) have been previously applied to FLPs with the STS.

2.1.1. Global Search Methods. Scholz et al. [10] present a tabu search algorithm with slicing tree representation and bounding curves. Through the use of bounding curves, geometric facility constraints are implicitly considered. Their tabu search incorporates four types of neighborhood searching performed on the slicing tree to find better solutions. They compared their algorithm with previous researches and showed obvious improvements.

2.1.2. Evolutionary Methods. Genetic algorithms are quite popular in solving FLPs with STS. Tam [11, 12] presented the concept of the STS to represent a layout as a chromosome of a character string. However, the facility's shape restrictions are not guaranteed by Tam's layout generation procedure. Tam and Chan [13] introduced an improved STS coding scheme and developed parallel genetic algorithms to solve FLPs with geometric constraints. The STS coding scheme uses binary digits to represent the tree structure, internal nodes, and external nodes. Gambler's Ruin method [14] is used to encode the slicing tree. Gau and Meller [3] presented a facility layout algorithm that iterates between a genetic algorithm with a slicing tree representation and a mixed-integer linear program (MILP) with a subset of the binary variables set via the GA, which is used to find STS solutions with feasible shaped departments. According to these STS solutions, relative department locations are used as the subset of the binary variables in the MILP formulation proposed by Meller et al. [15]. Then, the solution obtained by solving the MILP is used to generate an initial population of solutions for the GA. Al-Hakim [16] noted that the chromosome representation in Tam and Chan was not efficient. Al-Hakim demonstrated the difficulties in applying classical crossover and mutation operators for solving FLPs and modified the representation of Tam and Chan. A new preserving operation, referred to as transplanting, was introduced to ensure the coherence of an offspring. Wu and Appleton [17] developed a genetic algorithm to simultaneously determine a slicing block layout and its aisle structure. They use the STS to represent a layout as a chromosome consisting of three strings of characters: a facilities string, a cutting levels string that represents the aisles, and an orientation string. A repair operator makes sure that the chromosome represents a feasible layout. Shayan and Chittilappilly [18] presented a new genetic algorithm to solve FLPs represented by STS. By following Gambler's Ruin method, a two-dimensional chromosome is designed to represent the slicing tree without any need for repairing procedures to ensure that the chromosomes represent legal layouts after applying genetic operators.

Ant colony optimization has been recently applied for solving FLPs with STS. Komarudin and Wong [19] proposed an AS algorithm with slicing tree representation for solving FLPs. They use several types of local search to improve the search performance. Compared to Liu and Meller's [20] and Scholz et al.'s [10] approaches, the proposed algorithm can improve the results for six different problem instances.

2.2. Harmony Search. Harmony search is inspired by the natural musical performance process that occurs when a musician searches for a better state of harmony. In the HS algorithm, the solution vector is analogous to the harmony in music, and the local and global search schemes are analogous to musician's improvisations. HS is good at identifying the high performance regions of the solution space in a reasonable time but has difficulty performing a local search for numerical applications. Several variants of HS [21–25] have been proposed to improve the performance of HS. Proper balance between intensification and diversification can result in enhanced performance of HS.

Numerical comparisons demonstrate that evolution in the HS algorithm is faster than that in GA [6, 26, 27].

The main difference between GA and HS is that HS makes a new harmony vector from all existing harmony vectors in the harmony memory, while GA makes the new chromosome vector only from two of the existing chromosome vectors. Moreover, HS can independently consider each component variable in a vector while it generates a new harmony vector whereas GA cannot because it has to keep the structure of a gene [28].

3. A Self-Adaptive Harmony Search Algorithm with Quadratically Constrained Program for a Slicing Tree Structured Layout

Based on the HS algorithm, we develop an iterative optimization process of STS solutions. For each STS solution, the optimal locations and shapes of departments are determined by solving the proposed QCP; the fitness value of this STS solution is evaluated at the same time. Since harmony memory consideration rate (HMCR) and pitch adjustment rate (PAR) are related to the HS algorithm's ability to search for global exploration and local exploitation which are always twisted together in the search process, it is difficult to select the values for HMCR and PAR [25]. In order to prevent premature convergence, a self-adaptive harmony search algorithm (SHS) [9] introduces learning mechanisms to adjust HMCR and PAR. This algorithm is called the SHS algorithm with QCP for a slicing tree structured layout (SHS-QCP-STS).

3.1. Mathematical Models of a Facility Layout Problem. A mixed-integer nonlinear programming (MINP) model is proposed to address the optimal facility layout. The objective is to minimize the material workflow cost (MWC). As shown in (1), the material workflow cost functions are based on interdepartmental distances:

$$\min \sum_i \sum_j f_{ij} c_{ij} \left(d_{ij}^x + d_{ij}^y\right), \tag{1}$$

where f_{ij} is the workflow from departments i and j, c_{ij} is the cost per unit distance from i and j, d_{ij}^x is the rectilinear distance of the centroids from i and j on the x-axis, and d_{ij}^y is the rectilinear distance of the centroids from i and j on the y-axis.

The constraints of the model are grouped into five main categories.

(i) Area constraints are

$$w_i \cdot h_i = a_i \quad \forall i, \tag{2}$$

where w_i and h_i represent the width and the height of department i, respectively. The product of the width and the height must equal the area of department i, a_i.

(ii) Shape constraints are

$$\frac{\max\left(w_i, h_i\right)}{\min\left(w_i, h_i\right)} \le \alpha_i^{\max} \quad \forall i, \tag{3}$$

$$\min\left(w_i, h_i\right) \ge \rho_i^{\min} \quad \forall i, \tag{4}$$

where $\alpha_i^{\max}$ and $\rho_i^{\min}$ represent the maximum aspect ratio and the minimum side length, respectively. In order to avoid irregular department shapes, either (3) or (4) can be regarded as shape constraints.

(iii) Nonoverlapping constraints are

$$x_i + w_i \le x_j + W\left(1 - Z_{ij}^x\right) \quad \forall i, j, \tag{5}$$

$$x_j + w_j \le x_i + W\left(1 - Z_{ji}^x\right) \quad \forall i, j, \tag{6}$$

$$Z_{ij}^x + Z_{ji}^x \le 1 \quad \forall i, j, \tag{7}$$

$$y_i + h_i \le y_j + H\left(1 - Z_{ij}^y\right) \quad \forall i, j, \tag{8}$$

$$y_j + h_j \le y_i + H\left(1 - Z_{ji}^y\right) \quad \forall i, j, \tag{9}$$

$$Z_{ij}^y + Z_{ji}^y \le 1 \quad \forall i, j, \tag{10}$$

$$Z_{ij}^x + Z_{ji}^x + Z_{ij}^y + Z_{ji}^y \ge 1 \quad \forall i, j, \tag{11}$$

where x_i and y_i represent the x- and y-coordinates of the centroid of department i, respectively, W is the width of the facility along the x-axis, and H is length of the facility along the y-axis. The value of Z_{ij}^x equals 1 if department i is located on the left side of department j or zero otherwise. The value of Z_{ij}^y equals 1 if department i is below department j or zero otherwise. The functions of these constraints are as follows. For each pair of departments i and j, (5) through (7) force department i to be located on the left or right side of department j without overlapping in the x-axis direction. Equations (8) through (10) force department i to be above or below department j without overlapping in the y-axis direction. For each pair of departments i and j, (11) defines their relative positions: (1) department i is located on the left side of and below department j, (2) department i is located on the left side of department j, or (3) department i is below department j.

(iv) Definitional constraints are

$$x_i + 0.5w_i \le W \quad \forall i, \tag{12}$$

$$y_i + 0.5h_i \le H \quad \forall i, \tag{13}$$

$$x_i \ge 0.5w_i \quad \forall i, \tag{14}$$

$$y_i \ge 0.5h_i \quad \forall i, \tag{15}$$

$$d_{ij}^x \ge \left(x_i + 0.5w_i\right) - \left(x_j + 0.5w_j\right) \quad \forall i, j, \tag{16}$$

$$d_{ij}^x \ge \left(x_j + 0.5w_j\right) - \left(x_i + 0.5w_i\right) \quad \forall i, j, \tag{17}$$

$$d_{ij}^y \ge \left(y_i + 0.5h_i\right) - \left(y_j + 0.5h_j\right) \quad \forall i, j, \tag{18}$$

$$d_{ij}^y \ge \left(y_j + 0.5h_j\right) - \left(y_i + 0.5h_i\right) \quad \forall i, j. \tag{19}$$

Equations (12) through (15) ensure that the departments will be located within the boundaries of the facility along the x-axis and y-axis. Equations (16) through (19) linearize the absolute value term in the rectilinear distance function.

(v) Nonnegative constraints are

$$w_i \geq 0 \quad \forall i, \tag{20}$$

$$h_i \geq 0 \quad \forall i, \tag{21}$$

$$x_i \geq 0 \quad \forall i, \tag{22}$$

$$y_i \geq 0 \quad \forall i, \tag{23}$$

$$Z_{ij}^x \in \{0,1\} \quad \forall i, j, \tag{24}$$

$$Z_{ij}^y \in \{0,1\} \quad \forall i, j. \tag{25}$$

In general, MINP problems are more difficult to be solved than mixed-integer programming problems and nonlinear programming problems. We find that spatial relationships of departments known according to an STS solution can be used to replace binary variables Z_{ij}^x and Z_{ij}^y of this MINP model. Thus this MINP model can then be reformulated as a QCP model:

$$\min P_2 \left[\sum_i \sum_j f_{ij} c_{ij} \left(d_{ij}^x + d_{ij}^y \right) \right] + P_1 \left[\sum_i \left[x_i + 0.5 w_i - W \right]^+ \right] + P_1 \left[\sum_j \left[y_i + 0.5 h_i - H \right]^+ \right] \tag{26}$$

subject to the following.

(i) Area constraints

$$w_i h_i \geq a_i \quad \forall i. \tag{27}$$

(ii) Shape constraints: (3) or (4).

(iii) Nonoverlapping constraints:

$$y_i - 0.5 h_i = 0 \quad \forall i \in D_D, \tag{28}$$

$$x_i - 0.5 w_i = 0 \quad \forall i \in D_L, \tag{29}$$

$$y_i = y_j \quad \forall i = L_V(l),\ j = L_V(r), \tag{30}$$

$$h_i = h_j \quad \forall i = L_V(l),\ j = L_V(r), \tag{31}$$

$$x_i = x_j \quad \forall i = L_H(l),\ j = L_H(r), \tag{32}$$

$$w_i = w_j \quad \forall i = L_H(l),\ j = L_H(r), \tag{33}$$

$$x_i + 0.5 w_i \leq x_j - 0.5 w_i \quad \forall i \in S_V(l),\ j \in S_V(r), \tag{34}$$

$$y_i + 0.5 h_i \leq y_j - 0.5 h_j \quad \forall i \in S_H(l),\ j \in S_H(r), \tag{35}$$

where D_D represents the set of departments located on the bottom of the facility, D_L the set of departments located on the leftmost side of the facility, $L_V(l)$ and $L_V(r)$ the left and right leaf nodes of a vertical slicing, respectively, $L_H(l)$ and $L_H(r)$, the left and the right leaf nodes of a horizontal slicing, respectively, $S_V(l)$ and $S_V(r)$ the left and right subtrees of a vertical slicing, respectively, and $S_H(l)$ and $S_H(r)$ the left and right subtrees of a horizontal slicing, respectively.

According to an STS solution, the functions of these constraints are to define spatial relationships of departments and to arrange departments without overlapping. For department i at the bottom of the facility, (28) sets its y-coordinate of the bottom-left corner as equal to 0. For department i on the leftmost side of the facility, (29) sets its x-coordinate of the bottom-left corner as equal to 0. For departments i and j separated directly by a vertical cut, (30) and (31) define that their y-coordinates of the bottom-left corners and heights are equal. For departments i and j separated directly by a horizontal cut, (32) and (33) define that their x-coordinates of the bottom-left corners and widths are equal. For a vertical cut, (34) forces the x-coordinates of the right edges of departments located on the left subtree to be less than or equal to the x-coordinates of the left edges of departments located on the right subtree. For a horizontal cut, (35) forces the y-coordinates of the top edges of departments located on the left subtree must be less than or equal to the y-coordinates of the bottom edges of departments located on the right subtree.

(i) Definitional constraints: (14) through (19).

(ii) Nonnegative constraints: (20) through (23).

Here, $P_1[]$ and $P_2[]$ indicate that the absolute priority rule is applied to this objective function. This means that minimizing penalties of breaking the layout rule, (12) and (13), dominates minimizing material workflow costs. In this study, the sum of penalties and material workflow costs is regarded as the fitness value of an STS solution.

3.2. STS Solution Representation. In this study, the slicing tree structure proposed by Shayan and Chittilappilly [18] is used to represent an FLP. Each harmony vector consists of three parts: the node type codes, the node content codes, and the fitness value. For an FLP with n departments, there are $2(2n-1)+1$ codes. For example, a slicing tree and its corresponding slicing structure of an FLP with six departments are shown in Figure 1. We use the harmony vector (0-0-0-0-1-1-0-1-1-1-1) - (1-1-0-0-1-2-0-3-4-5-6) - (fitness value) to represent this layout solution.

(1) The first part is $2n-1$ binary numbers and records a blank slicing sequence. Here, 0 represents a cut and 1 otherwise. In total, there are $n-1$ cut nodes and n leaf nodes.

(2) The actual contents of these nodes are recorded in the second part. For the cut nodes, 1 denotes a horizontal cut, and 0 denotes a vertical cut. For the leaf nodes, an integer number represents that the department will be placed. The slicing structure of the FLP is confirmed.

(3) The third part is the fitness value of the harmony vector. The fitness value is evaluated by solving the proposed QCP model. Nonoverlapping constraints,

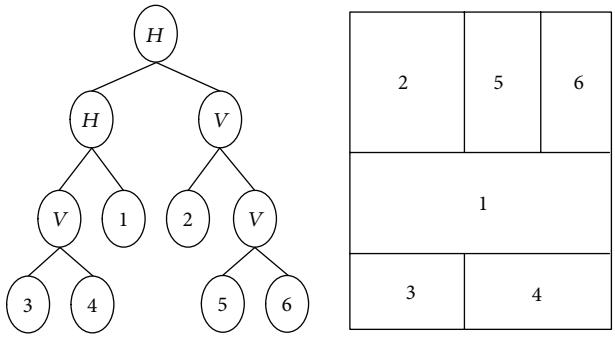


Figure 1: Slicing tree and slicing structure of FLP of 6 departments.

(28) through (35), are added according to the spatial relationships of a slicing tree. For example, the first cut is horizontal in Figure 1. The left leaf nodes of the root node are under the right leaf nodes of the root node. Set the y-coordinate of the top-left corners of departments 1, 3, and 4 as less than the y-coordinate of the bottom-left corners of departments 2, 5, and 6. After a vertical cut, department 3 is left to department 4. Set the x-coordinate of the bottom-left corners of department 4 as equal to the x-coordinate of the bottom-left corners of department 3 plus its width. The y-coordinates of the bottom-left corners are equal. Furthermore, departments 3 and 4 are at the bottom of the facility. Set the y-coordinate of the bottom-left corners of these two departments as equal to 0. Departments 1, 2, and 3 are on the leftmost side of the facility. Set the x-coordinate of the bottom-left corners of these three departments as equal to 0.

3.3. Harmony Memory. The harmony memory (HM) is a memory location where all solution vectors (sets of decision variables) and corresponding fitness values are stored. Because of characteristics of STS, we propose a regional structure of HM to memorize facility layout solutions.

First, HMS1 blank slicing sequences are generated. Then, the actual contents of each blank slicing sequence are randomly generated HMS2 times. This HM matrix, $[\mathrm{HM}]_{(\mathrm{HMS1}*\mathrm{HMS2})\times(4n-1)}$, is composed of the HMS1 harmony memory submatrix, $[hm_i]_{\mathrm{HMS2}\times(4n-1)}$, as stated in (36). Each HM submatrix is regarded as a regional memory:

$$[\mathrm{HM}]_{(\mathrm{HMS1}*\mathrm{HMS2})\times(4n-1)} = \begin{bmatrix} [hm_1]_{\mathrm{HMS2}\times(4n-1)} \\ \cdots \\ [hm_i]_{\mathrm{HMS2}\times(4n-1)} \end{bmatrix}_{\mathrm{HMS1}\times 1}. \tag{36}$$

Each HM submatrix, $[hm_i]_{\mathrm{HMS2}\times(4n-1)}$, is composed of the HMS2 harmony vectors with a common slicing sequence. The first element of the HM submatrix is the best harmony vector with the corresponding slicing sequence. In total, HMS1 $*$ HMS2 slicing trees of the FLP are stored in the HM matrix.

Taking an FLP with 6 departments as an example, the structure of HM matrix is shown in Figure 2. The size of the HM matrix is (HMS1 $*$ HMS2) $\times$ 23. Rows 1 to HMS2 of this HM matrix comprise HM submatrix 1, $[hm_1]$. It is shown that all blank slicing sequences of the HM submatrix are the same, but actual slicing contents of the counterparts are different.

3.4. New Harmony Improvisation. A new harmony vector, $hm_{ij}(1,\ldots,4n-1)$, is generated based on memory consideration (MC), random selection (RS), and pitch adjustment (PA).

3.4.1. Memory Consideration. In the memory consideration, if $r_1 \le$ HMCR, then the value for the new harmony vector is chosen from any value in the specified HM range. In order to fit the restrictions of a slicing tree structure, a blank slicing sequence cannot be chosen from any value in the harmony memory matrix.

MC Rule. For $l = 2n+1,\ldots,4n-2$, $j = 1,\ldots,$ HMS2, and $i = 1,\ldots,$ HMS1, if $r_1 \le$ HMCR, the lth code of the jth row of the harmony memory submatrix $[hm_i]$ is chosen from any other value in the corresponding column of this harmony memory submatrix. The correctness of the new harmony vector should be ensured. For leaf nodes, the actual slicing content codes are regenerated if departments have already been assigned to other leaf nodes. For example, if a new harmony vector (0-0-0-0-1-1-0-1-1-1-1) - (1-1-0-0-2-2-0-5-4-1-5) - (fitness value) is generated, it needs to be modified as (0-0-0-0-1-1-0-1-1-1-1) - (1-1-0-0-2-3-0-5-4-1-6) - (fitness value), because one department is not able to be assigned to two leaf nodes, and vice versa. In the original harmony vector, departments 2 and 5 are assigned twice, but departments 3 and 6 are not assigned. Replace department 2 with department 3 for the second assignment of department 2. Replace department 6 with department 5 for the second assignment of department 5:

$$\begin{aligned} &\text{if } r_1 \le \mathrm{HMCR}, \\ &hm_{ij}(l) \longleftarrow hm_{ij'}(l) \\ &\quad \in \{hm_{i,1}(l), hm_{i,2}(l), \ldots, hm_{i,\mathrm{HMS2}}(l)\}, \\ &\qquad j' \ne j,\ l \in \{2n+1,\ldots,4n-2\}; \\ &\quad j = 1,\ldots,\mathrm{HMS2};\quad i = 1,\ldots,\mathrm{HMS1}. \end{aligned} \tag{37}$$

3.4.2. Random Selection. Two random selection procedures are performed in this study. Note that RS rule 1 is valid only for generating an actual slicing content.

RS Rule 1. If $r_1 >$ HMCR, randomly generate an actual slicing content according to the given blank slicing sequence:

$$\begin{aligned} &\text{if } r_1 > \mathrm{HMCR}, \\ &\text{if } hm_{ij}(l-2n-1) = 0, \qquad hm_{ij}(l) \longleftarrow hm_{ij}(l) \in \{0,1\} \\ &\text{if } hm_{ij}(l-2n-1) = 1, \end{aligned}$$

Row 1:	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	1	2	0	3	4	5	6	Fitness		
Row 2:	0	0	0	0	1	1	0	1	1	1	1	0	1	1	0	1	4	1	2	6	3	5	Fitness		
...	...																						...		
Row HMS2:	0	0	0	0	1	1	0	1	1	1	1	1	1	1	0	4	3	0	1	6	5	2	Fitness		
...	...																						...		
Row (HMS1-1) * HMS2 + 1:	0	0	1	0	1	1	0	0	1	1	1	1	1	2	0	1	3	0	1	4	5	6	Fitness		
Row (HMS1-1) * HMS2 + 2:	0	0	1	0	1	1	0	0	1	1	1	0	1	1	1	5	2	1	1	6	3	4	Fitness		
...	...																						...		
Row (HMS1-1) * HMS2 + HMS2:	0	0	1	0	1	1	0	0	1	1	1	1	1	3	0	4	1	0	0	6	5	2	Fitness		

Figure 2: Structure of harmony memory matrix.

$$hm_{ij}(l) \longleftarrow hm_{ij}(l) \in \{1, \ldots, n\}$$
$$l \in \{2n+1, \ldots, 4n-2\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1}. \tag{38}$$

The correctness of the new harmony vector should also be verified to maintain a one-to-one relationship between departments and leaf nodes.

RS Rule 2. If $r_1 >$ HMCR, randomly generate a blank slicing sequence and correspondingly generate an actual slicing content:

$$\text{if } r_1 > \text{HMCR},$$
$$hm_{i,\circ}(l) \longleftarrow hm_{i,\circ}(l) \in \{0,1\}^c \quad l = 1, \ldots, 2n-1; \quad i = 1, \ldots, \text{HMS1},$$
$$\text{if } hm_{ij}(l-2n-1) = 0, \qquad hm_{ij}(l) \longleftarrow hm_{ij}(l) \in \{0,1\} \tag{39}$$

$$\text{if } hm_{ij}(l-2n-1) = 1,$$
$$hm_{ij}(l) \longleftarrow hm_{ij}(l) \in \{1, \ldots, n\}$$
$$l \in \{2n+1, \ldots, 4n-2\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1}. \tag{40}$$

Note that a blank slicing sequence is randomly generated by following the principles proposed by Shayan and Chittilappilly [18]. The correctness of the new harmony vector also should be checked to maintain a one-to-one relationship between departments and leaf nodes.

3.4.3. Pitch Adjustment. Within the new harmony vector, every component obtained by the memory consideration is examined to determine whether it should be pitch-adjusted. Three pitch adjustment procedures are performed in this study. Note that PA rule 1 is valid only for changing an actual slicing content.

PA Rule 1: A Leaf Node Switches with Another Leaf Node. If a pitch adjustment decision of actual slicing contents for $hm_{ij}(l)$ is yes, $hm_{ij}(l)$ is switched with $hm_{ij}(l+k)$, and the pitch-adjusted values of $hm_{ij}(l)$ and $hm_{ij}(l+k)$ become as follows:

$$\text{if } r_1 \le \text{HMCR}, \qquad r_2 \le \text{PAR},$$
$$\text{if } hm_{ij}(l-2n-1) = hm_{ij}(l-2n-1+k) = 1,$$
$$hm_{ij}(l) \longleftarrow hm_{ij}(l) = hm_{ij}(l+k), \tag{41}$$
$$l \in \{2n+1, \ldots, 4n-2\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1},$$
$$\text{if } hm_{ij}(l-2n-1) = hm_{ij}(l-2n-1+k) = 1,$$
$$hm_{ij}(l+k) \longleftarrow hm_{ij}(l+k) = hm_{ij}(l), \tag{42}$$
$$l \in \{2n+1, \ldots, 4n-2\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1},$$

where k is the neighboring index, $k \in \{\ldots, -2, -1, 1, 2, \ldots\}$.

Because the chosen nodes for pitch adjustment are leaf nodes, that is, $hm_{ij}(l-2n-1) = hm_{ij}(l-2n-1+k) = 1$, this operation of such a pitch adjustment is like switching between departments. The STS has not changed. The pitch adjustment occurs between each two leaf nodes. For example, there is an original harmony vector as follows:

(0-0-0-1-0-1-1-1-1) - (1-0-0-3-1-1-2-5-4) - (fitness value).

If the fifteenth position of the harmony vector is chosen to be adjusted and the neighboring index k is set as +3, we have an adjusted harmony vector:

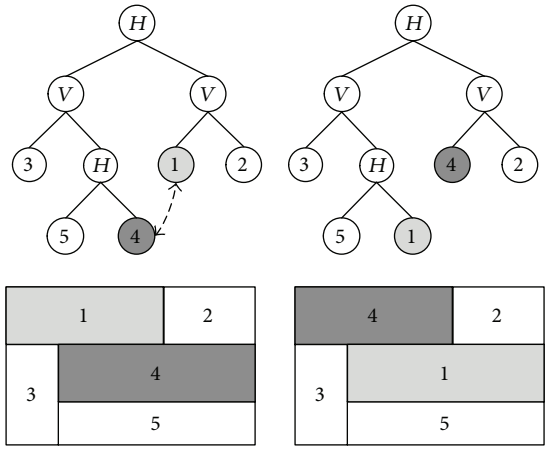
(0-0-0-1-0-1-1-1-1) - (1-0-0-3-1-4-2-5-1) - (fitness value).

Corresponding facility layouts are shown in Figure 3.

If a pitch adjustment decision of a blank slicing sequence for $hm_{ij}(l)$ stands, three cases will be considered.

PA Rule 2: A Cut Node Switches with a Leaf Node. If $hm_{ij}(l)$ is switched with $hm_{ij}(l+k)$, pitch-adjusted values of $hm_{ij}(l)$, $hm_{ij}(l+k)$, $hm_{ij}(l+2n-1)$, and $hm_{ij}(l+2n-1+k)$ become as follows:

$$\text{if } r_1 \le \text{HMCR}, \qquad r_2 \le \text{PAR},$$
$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 1,$$
$$hm_{ij}(l) \longleftarrow hm_{ij}(l) = hm_{ij}(l+k),$$



(a) Original layout (b) Adjusted layout

Figure 3: Pitch adjustment of two leaf nodes.

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \mathrm{HMS2}; \quad i = 1, \ldots, \mathrm{HMS1}, \tag{43}$$

$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 1, \quad hm_{ij}(l+k) \longleftarrow hm_{ij}(l+k) = hm_{ij}(l), \tag{44}$$

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \mathrm{HMS2}; \quad i = 1, \ldots, \mathrm{HMS1},$$

$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 1,$$

$$hm_{ij}(l+2n-1) \longleftarrow hm_{ij}(l+2n-1) = hm_{ij}(l+2n-1+k),$$

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \mathrm{HMS2}; \quad i = 1, \ldots, \mathrm{HMS1}, \tag{45}$$

$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 1,$$

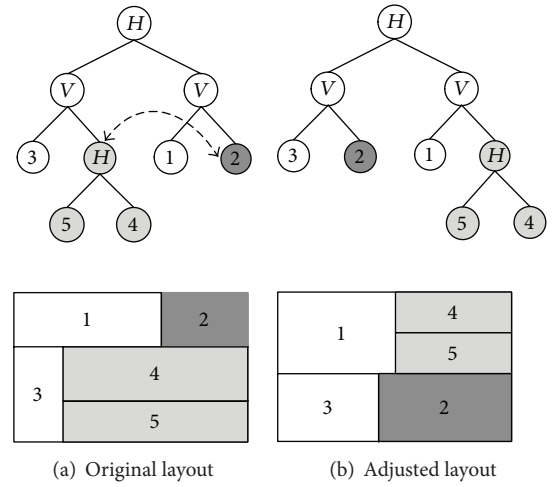
$$hm_{ij}(l+2n-1+k) \longleftarrow hm_{ij}(l+2n-1+k) = hm_{ij}(l+2n-1),$$

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \mathrm{HMS2}; \quad i = 1, \ldots, \mathrm{HMS1}, \tag{46}$$

where k is the neighboring index, $k \in \{\ldots, -2, -1, 1, 2, \ldots\}$. Note that two leaf nodes under the chosen cut node are moved together, and the STS is reconstructed. Except for pitch adjustment, the relation of a final cut and two leaf nodes cannot be broken.

For example, there is an original harmony vector:

(0-0-0-1-0-1-1-1-1) - (1-0-0-3-1-1-2-5-4) - (fitness value).



(a) Original layout (b) Adjusted layout

Figure 4: Pitch adjustment of one cut node and one leaf node at the same depths.

If the fifth position of the harmony vector is chosen to be adjusted and the neighboring index k is set as +2, we have the adjusted harmony vector:

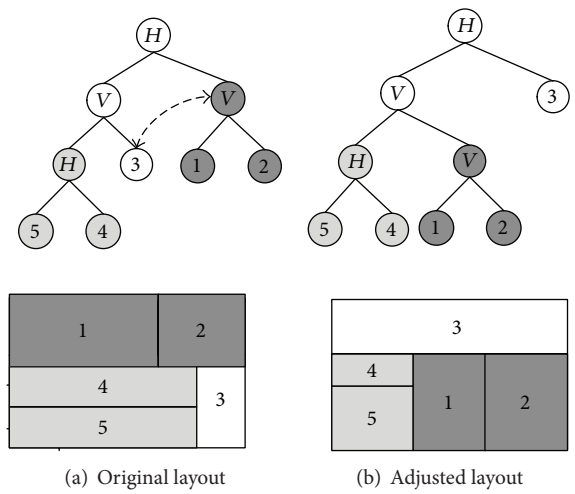
(0-0-0-1-1-1-0-1-1) - (1-0-0-3-2-1-1-5-4) - (fitness value).

Corresponding facility layouts are shown in Figure 4. In this example, the cut node and the lead node are at the same depth of the slicing tree.

Furthermore, we take a cut node and a lead node located at different depths of the slicing tree as an example. An original harmony vector is

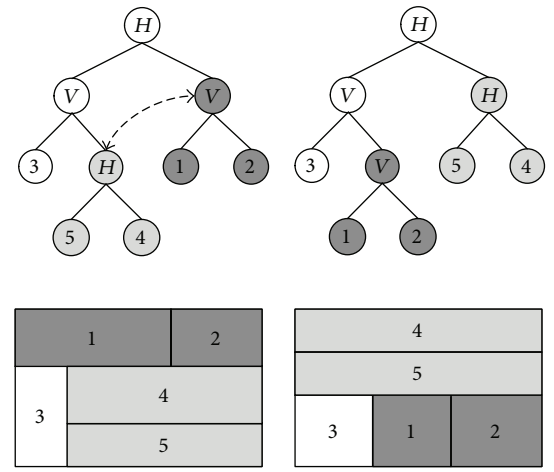
(0-0-0-0-1-1-1-1-1) - (1-0-0-1-3-1-2-5-4) - (fitness value).

The third position of the harmony vector is chosen to be adjusted, and the neighboring index k is set as +2. Note that the chosen cut node is moved from depth 1 to depth 2 of



(a) Original layout (b) Adjusted layout

FIGURE 5: Pitch adjustment of one cut node and one leaf node at different depths.



(a) Original layout (b) Adjusted layout

FIGURE 6: Pitch adjustment of two cut nodes at different depths.

the slicing tree. This means that the slicing sequence changes. Furthermore, the relation of a final cut and two leaf nodes is maintained. Then, we reconstruct the slicing tree to obtain the adjusted harmony vector:

(0-0-1-0-0-1-1-1-1) - (1-0-3-1-0-5-4-1-2) - (fitness value).

Corresponding facility layouts are shown in Figure 5.

PA Rule 3: A Cut Node Switches with Another Cut Node. If $hm_{ij}(l)$ is switched with $hm_{ij}(l+k)$, pitch-adjusted values of $hm_{ij}(l)$, $hm_{ij}(l+k)$, $hm_{ij}(l+2n-1)$, and $hm_{ij}(l+2n-1+k)$ become as follows:

$$\text{if } r_1 \le \text{HMCR}, \qquad r_2 \le \text{PAR},$$

$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 0,$$

$$hm_{ij}(l) \longleftarrow hm_{ij}(l) = hm_{ij}(l+k),$$

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1}, \tag{47}$$

$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 0,$$

$$hm_{ij}(l+k) \longleftarrow hm_{ij}(l+k) = hm_{ij}(l),$$

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1}, \tag{48}$$

$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 0,$$

$$hm_{ij}(l+2n-1) \longleftarrow hm_{ij}(l+2n-1)$$

$$= hm_{ij}(l+2n-1+k),$$

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1} \tag{49}$$

$$\text{if } hm_{ij}(l) + hm_{ij}(l+k) = 0,$$

$$hm_{ij}(l+2n-1+k) \longleftarrow hm_{ij}(l+2n-1+k)$$

$$= hm_{ij}(l+2n-1),$$

$$l \in \{2, \ldots, 2n-1\}; \quad j = 1, \ldots, \text{HMS2}; \quad i = 1, \ldots, \text{HMS1}, \tag{50}$$

where k is the neighboring index, $k \in \{\ldots, -2, -1, 1, 2, \ldots\}$. Because two cut nodes are chosen for pitch adjustment, the STS changes, despite whether these two cut nodes are at the same depth or at different depths. Two leaf nodes under the chosen cut node are moved together.

For example, consider the following original harmony vector:

(0-0-0-1-0-1-1-1-1) - (1-0-0-3-1-1-2-5-4) - (fitness value).

If the fifth position of the harmony vector is chosen to be adjusted and the neighboring index k is set as -2, these two cut nodes are at different depths of the slicing tree. We reconstruct the slicing tree, and the adjusted harmony vector is

(0-0-0-1-0-1-1-1-1) - (1-0-1-3-0-5-4-1-2) - (fitness value).

Corresponding facility layouts are shown in Figure 6.

We take two cut nodes at the same depth of the slicing tree as another example. The original harmony vector is

(0-0-1-0-0-1-1-1-1) - (1-0-3-1-0-5-4-1-2) - (fitness value).

The fourth position of the harmony vector is chosen to be adjusted, and the neighboring index k is set as $+1$. Two cut nodes are swapped. This means that the slicing sequence changes. The relation of a final cut and two leaf nodes is maintained. Then, we reconstruct the slicing tree, and the adjusted harmony vector is

(0-0-1-0-0-1-1-1-1) - (1-0-3-0-1-1-2-5-4) - (fitness value).

Corresponding facility layouts are shown in Figure 7.

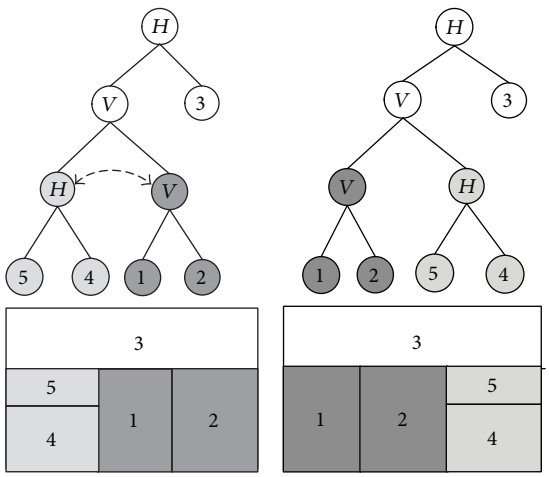


Figure 7: Pitch adjustment of two cut nodes at the same depth.

3.5. A Self-Learning Parameter Setting Mechanism. The parameters of the HS algorithm are harmony memory size (HMS), harmony memory consideration rate (HMCR), pitch adjusting rate (PAR), and number of improvisations (NI). In general, the values of HMS and NI are user-specified in order to address problems with different dimensions. Assume that the values of HMCR and PAR are normally distributed in reasonable ranges [9]. During a specified learning period, the HMCR (PAR) value associated with the generated harmony vector successfully replacing the worst harmony vector in the HM is recorded, and HMCR_m (PAR_m) is recalculated by averaging all recorded HMCR (PAR) values. With the new mean and the given standard deviation, a new HMCR (PAR) value is produced and used in the later iterations. As a result, a proper HMCR (PAR) value can be gradually learned. Detailed procedures are presented as follows.

Step 1. Initialize the learning period counter. Set $l = 0$.

Step 2. Randomly generate HMCR_n based on the corresponding normal distribution with a mean of μ_{HMCR} and a standard deviation of σ_{HMCR}.

Step 3. Randomly generate PAR_n based on the corresponding normal distribution with a mean of μ_{PAR} and a standard deviation of σ_{PAR}.

Step 4. Randomly generate number r_1, where $0 \le r_1 \le 1$.

Step 5. Update the learning period counter. Let $l = l + 1$.

Step 6. If $r_1 \le \text{HMCR}_n$, generate a new harmony vector based on memory consideration rule. Otherwise, go to Step 9.

Step 7. Randomly generate a number r_2, where $0 \le r_2 \le 1$.

Step 8. If $r_2 \le \text{PAR}_n$, generate a new harmony vector based on pitch adjustment rule. Otherwise, go to Step 10.

Step 9. Generate a new harmony vector based on random selection rule.

Step 10. Evaluate the fitness value of the new harmony vector generated is Step 6, 8, or 9.

Step 11. If the fitness value of the new harmony vector is better than the worst harmony vector in the specific range of HM, record this HMCR_n and PAR_n.

Step 12. Update the mean value of HMCR according to the current value and the recorded lists of HMCR_n.

Step 13. Update the mean value of PAR according to the current value and the recorded lists of PAR_n.

Step 14. If the length of the learning period l is less than LP, then go to Step 2.

3.6. Overall Algorithm. Because we adopt a regional structure of HM, two kinds of harmony improvisation are needed to develop. New harmony improvisation of HM submatrix is achieved by generating or changing actual slicing contents, that is, MC rule, RS rule 1, and PA rule 1. They are local searching mechanisms of regional HM. The best harmony vectors of each HM submatrix are basses of new harmony improvising of HM. RS rule 2, PA rule 2, and PA rule 3 can also be followed in this searching stage. Not only actual slicing contents but also slicing sequences are changed in these three procedures of improvisation. They are global searching mechanisms of the proposed algorithm. The flow chart of SHS-QCP-STS is given in Figure 8.

The overall procedure of the SHS-QCP-STS is given as follows.

Step 1. Specify parameter setting and initialization.

Step 1.1. Specify the SHS algorithm parameters: harmony memory size (HMS1× HMS2), initial mean value of harmony memory considering rate (μ_{HMCR}), initial standard deviation of harmony memory considering rate (σ_{HMCR}), initial mean value of pitch adjusting rate (μ_{PAR}), initial standard deviation of pitch adjusting rate (σ_{PAR}), length of the learning period (LP), and number of iterations (NI_1, NI_2).

Step 1.2. Initialize the fitness value of the current best solution. Set $z^* = \infty$.

Step 2. Initialize the harmony memory matrix.

Step 2.1. Initialize harmony counter 1. Set $m1 = 0$.

Step 2.2. Randomly generate a blank slicing sequence.

Step 2.3. Update harmony counter 1. Let $m1 := m1 + 1$.

Step 2.4. Initialize harmony counter 2. Set $m2 = 0$.

Step 2.5. Randomly generate the actual slicing contents based on the blank slicing sequence generated in Step 2.2.

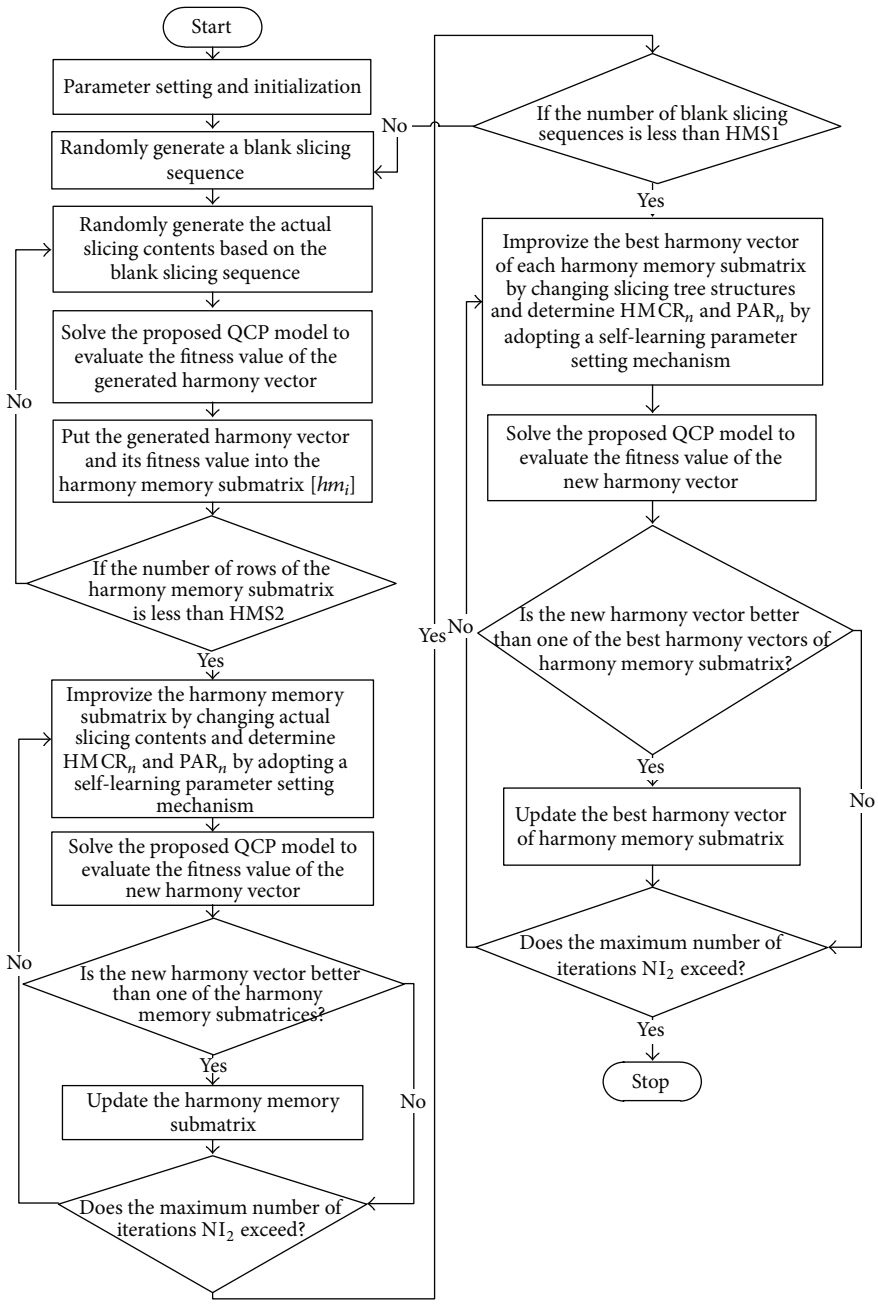


Figure 8: Flow chart of the proposed algorithm.

Step 2.6. Solve the proposed QCP model to evaluate the fitness value of the generated harmony vector.

Step 2.7. Update harmony counter 2. Let $m2 := m2 + 1$.

Step 2.8. If the number of harmony counter 2, $m2$, is less than HMS2, then go to Step 2.5.

Step 2.9. Sort Row $1 + (m1 - 1) \times$ HMS2 to Row $m1 \times$ HMS2 of the harmony memory matrix based on their fitness value.

Step 2.10. If the number of the harmony counter 1, $m1$, is less than HMS1, then go to Step 3.2.

Step 3. Improvize the initial harmony memory matrix by changing actual slicing contents.

Step 3.1. Initialize harmony counter 1. Set $m1 = 0$.

Step 3.2. Initialize the loop counter. Set $n = 1$.

Step 3.3. Applying the MC rule, RS rule 1, and PA rule 1, generate a new harmony vector by changing actual slicing contents based on the harmony memory submatrix $[hm_{m1}]_{\text{HMS2}\times(4n-1)}$, Row $1 + (m1 - 1) \times$ HMS2 to Row $m1 \times$ HMS2 of the harmony memory matrix. Herein,

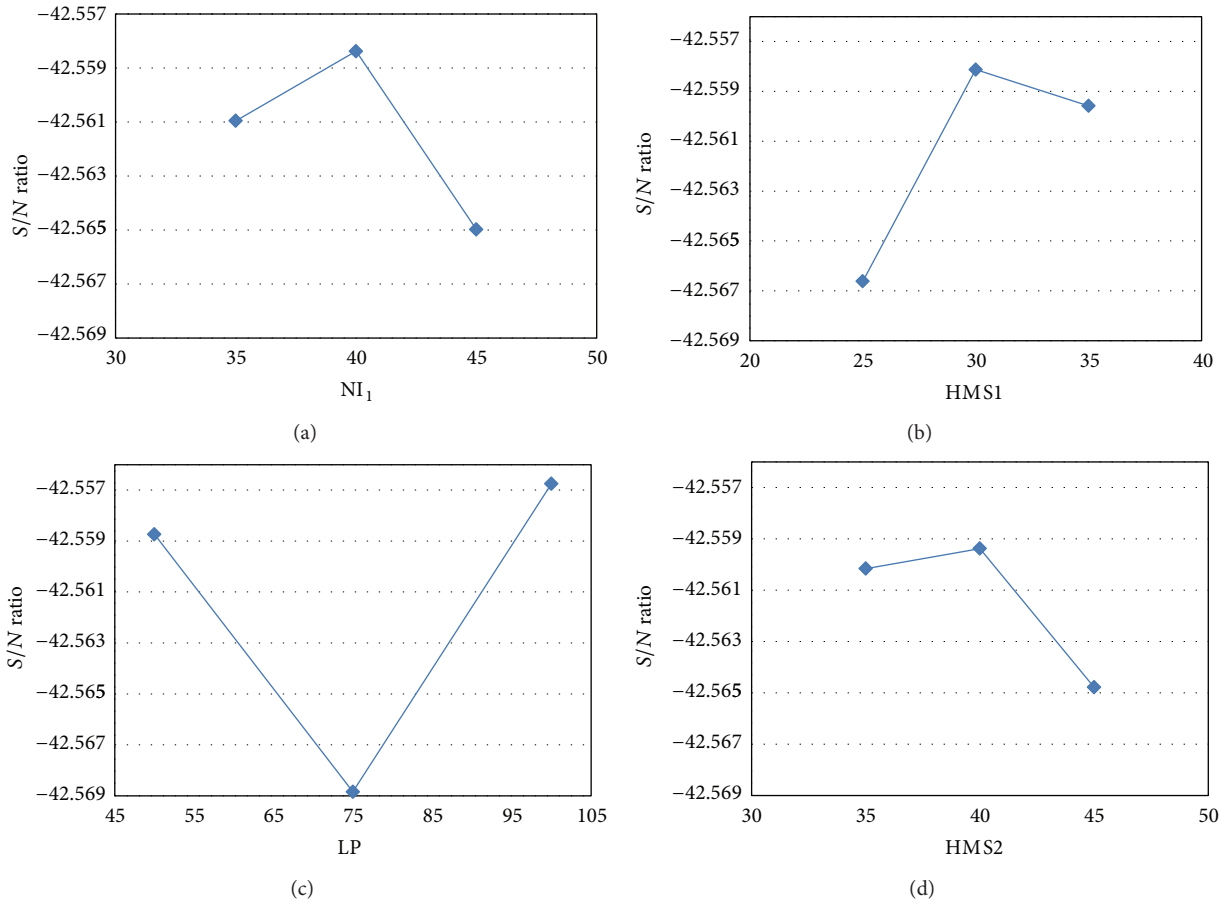


Figure 9: Level average responses for algorithm parameters by S/N ratios for problem O7.

Table 1: Problem set data.

Problem set	No. of dpt.	Problem data	Width	Height	Shape constraint
O7	7	[15]	8.54	13.00	$\alpha^{\max} = 4$
O8	8	[15]	11.31	13.00	$\alpha^{\max} = 4$
O9	9	[15]	12.00	13.00	$\alpha^{\max} = 5$
FO7	7	[15]	8.54	13.00	$\alpha^{\max} = 5$
FO8	8	[15]	11.31	13.00	$\alpha^{\max} = 5$
vC10Rs	10	[29]	51.00	25.00	$\rho^{\min} = 5$
vC10Ra	10	[29]	51.00	25.00	$\alpha^{\max} = 5$
M11*	11	[30]	6.00	6.00	$\alpha^{\max} = 5$
Nug12	12	[31]	4.00	3.00	$\alpha^{\max} = 5$
AB20	20	[2]	30.00	20.00	$\alpha^{\max} = 4$

$\alpha^{\max}$: maximum aspect ratio constraints.
$\rho^{\min}$: minimum side length constraints.

a self-learning parameter setting mechanism is adopted to determine HMCR_n and PAR_n.

Step 3.4. Solve the proposed QCP model to evaluate the fitness value of the new harmony vector.

Table 2: Parameter setting for problem O7.

Combination	NI_1	HMS1	HMS2	LP
1	35	25	35	50
2	35	30	40	75
3	35	35	45	100
4	40	25	40	100
5	40	30	45	50
6	40	35	35	75
7	45	25	45	75
8	45	30	35	100
9	45	35	40	50

Step 3.5. Update the harmony memory submatrix $[hm_{m1}]_{\text{HMS2}\times(4n-1)}$.

The new harmony vector, generated in Step 3.3, replaces the worst harmony vector in the harmony memory submatrix $[hm_{m1}]_{\text{HMS2}\times(4n-1)}$ only if its fitness value is better than that of the worst harmony vector.

Step 3.6. Sort the harmony memory submatrix $[hm_{m1}]_{\text{HMS2}\times(4n-1)}$ based on its fitness value. For the $m1$th

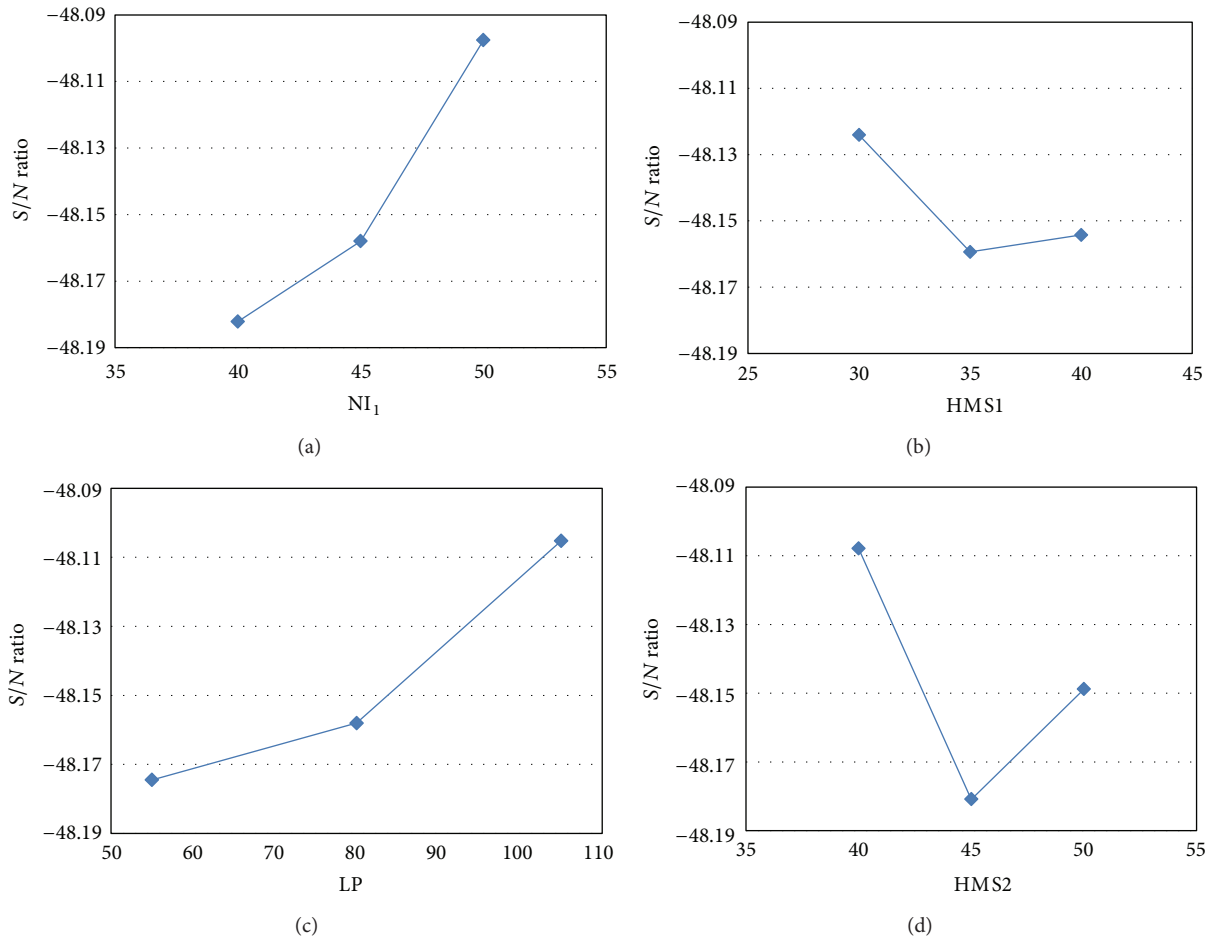


Figure 10: Level average responses for algorithm parameters by S/N ratios for problem O8.

harmony memory submatrix $[hm_{m1}]_{\mathrm{HMS2}\times(4n-1)}$, record the local-best harmony vector.

Step 3.7. Update the loop counter. Let $n := n + 1$.

Step 3.8. If the number of the loop n is less than NI_2, then go to Step 3.3.

Step 3.9. Update harmony counter 1. Let $m1 := m1 + 1$.

Step 3.10. If the number of harmony counter 1, $m1$, is less than HMS1, then go to Step 3.2.

Step 4. Improvize a new harmony.

Step 4.1. Initialize the loop counter. Set $n = 1$.

Step 4.2. Initialize harmony counter 1. Set $m1 = 1$.

Step 4.3. Applying one MC rule, two RS rules, and three PA rules, generate a new harmony vector based on the harmony memory matrix at intervals of HMS2 rows, that is, Row 1, Row 1 + HMS2, . . ., and Row (1 + (HMS1 − 1) × HMS2). These are the local-best harmony vectors recorded in Step 3.6. In addition, adopt a self-learning parameter setting mechanism to determine HMCR_n and PAR_n. Two RS rules are randomly selected. The same situation occurs with three PA rules.

Step 4.4. Solve the proposed QCP model to evaluate the fitness value of the new harmony vector.

Step 4.5. Update the harmony memory matrix at intervals of HMS2 rows, that is, Row 1, Row 1 + HMS2, . . ., and Row (1 + (HMS1 − 1) × HMS2).

The new harmony vector replaces the worst harmony vector only if its fitness value is better than that of the worst harmony vector.

Step 4.6. Update the current best harmony solution and the current worst harmony solution in the harmony memory matrix at intervals of HMS2 rows.

Step 4.7. Update the loop counter. Let $n := n + 1$.

Step 4.8. Update harmony counter 1. Let $m1 := m1 + 1$.

Step 4.9. If the number of harmony counter 1, $m1$, is greater than HMS1, set $m1 = 1$.

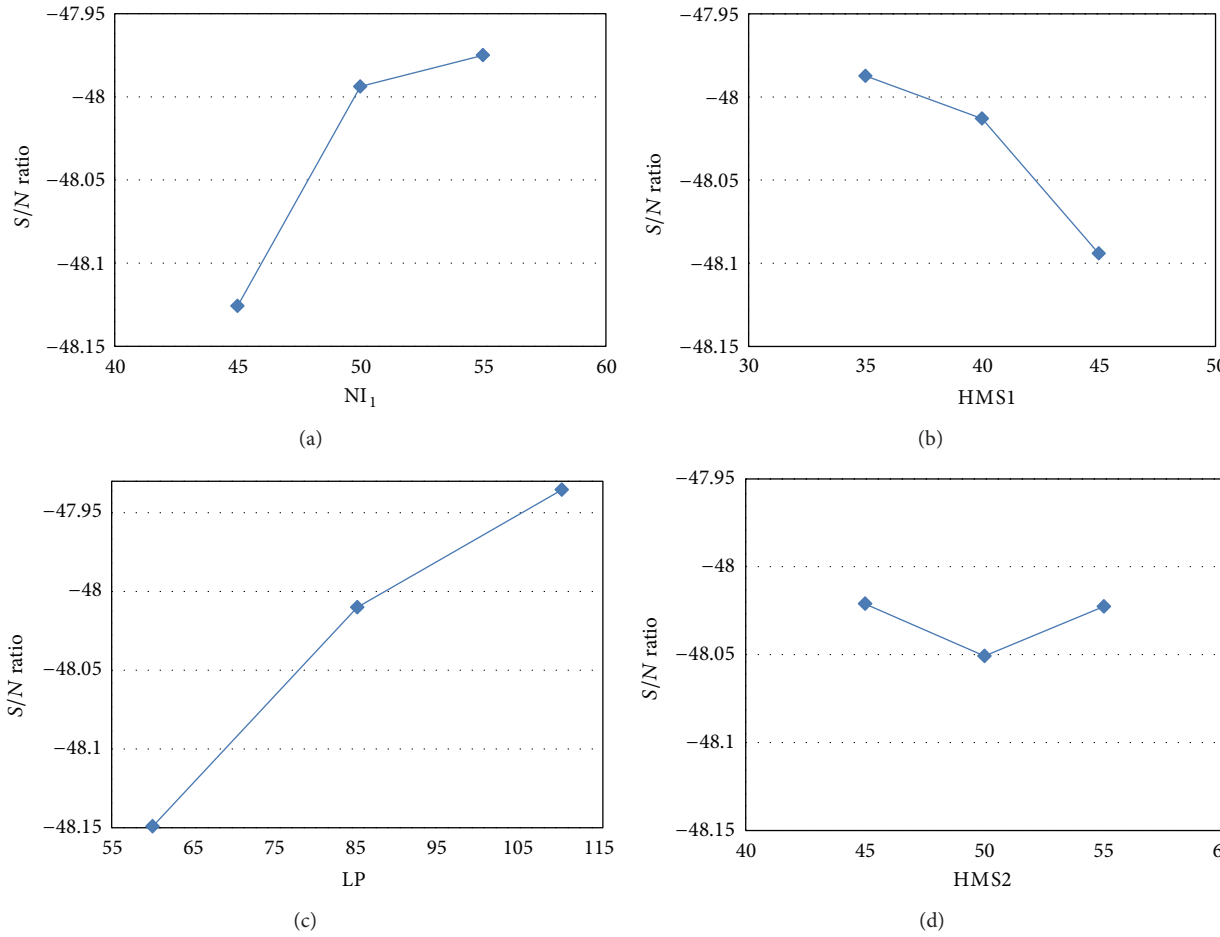


Figure 11: Level average responses for algorithm parameters by S/N ratios for problem O9.

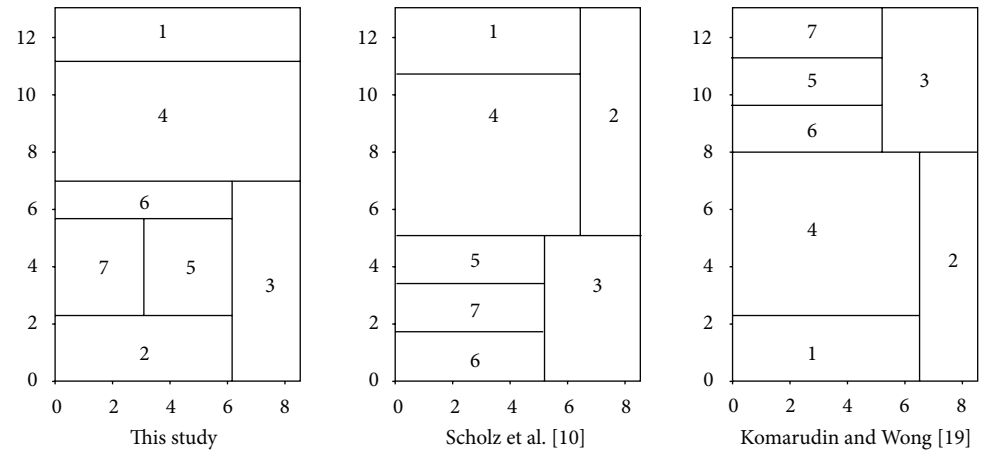


Figure 12: Best solutions found for problem O7.

Step 4.10. If the number of loop n is less than NI_1, then go to Step 4.3.

Step 5. Output the current best harmony solution and stop.

4. Numerical Results

4.1. Data and Implementation. To evaluate the effectiveness of the proposed algorithm, a series of computational experiments are performed, and testing results are compared

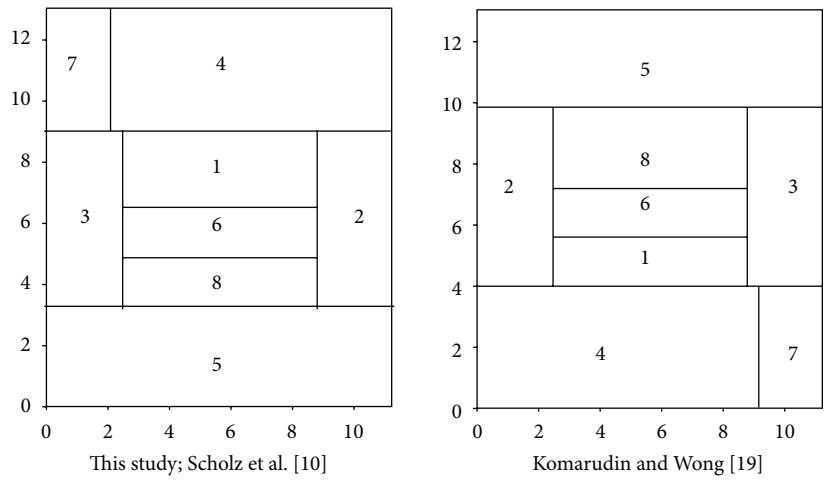


Figure 13: Best solutions found for problem O8.

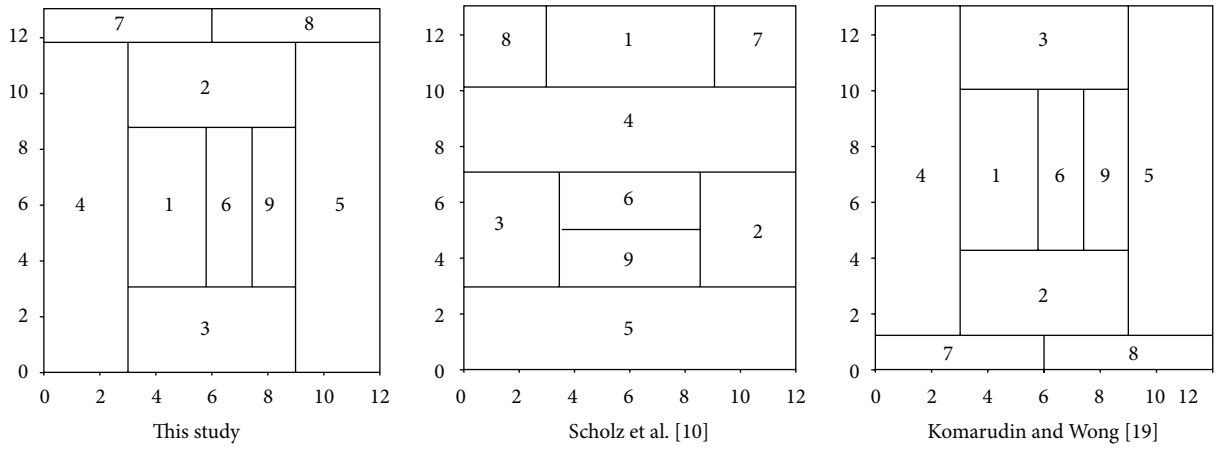


Figure 14: Best solutions found for problem O9.

Table 3: Parameter setting for problem O8.

Combination	NI_1	HMS1	HMS2	LP
1	40	30	40	55
2	40	35	45	80
3	40	40	50	105
4	45	30	45	105
5	45	35	50	55
6	45	40	40	80
7	50	30	50	80
8	50	35	40	105
9	50	40	45	55

Table 4: Parameter setting for problem O9.

Combination	NI_1	HMS1	HMS2	LP
1	45	35	45	60
2	45	40	50	85
3	45	45	55	110
4	50	35	50	110
5	50	40	55	60
6	50	45	45	85
7	55	35	55	85
8	55	40	45	110
9	55	45	50	60

to those found in the literature. Test problem sets are listed in Table 1. The original input data of the test problems can be found in the corresponding references [2, 15, 29–31]. Problems VC10Rs and VC10Ra are the versions of problem VC10 using the rectilinear distance metric. There is a fixed department in the original data set of M11; Meller [30] modified the original data set by relaxing the fixed department restriction and named the revised data sets as M11*.

The algorithm is coded with C++ and CPLEX 12.2 Full and tested using an AMD Athlon II X4 620 2.6 GHz CPU processor with 4 GB RAM. CPLEX 12.2 Full can handle problems that have quadratic constraints: QCP problems and mixed-integer quadratically constrained programming (MIQCP) problems.

All parameter values were determined based on previous research and pretuning conducted by this study.

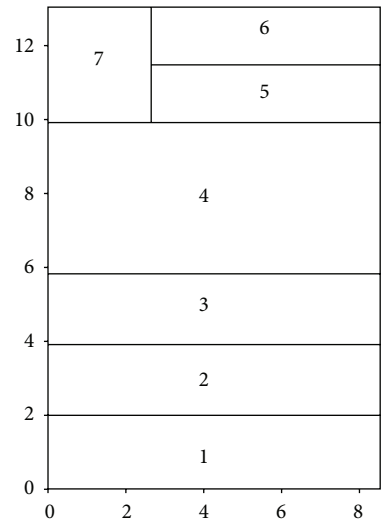


Figure 15: Best solutions found for problem FO7.

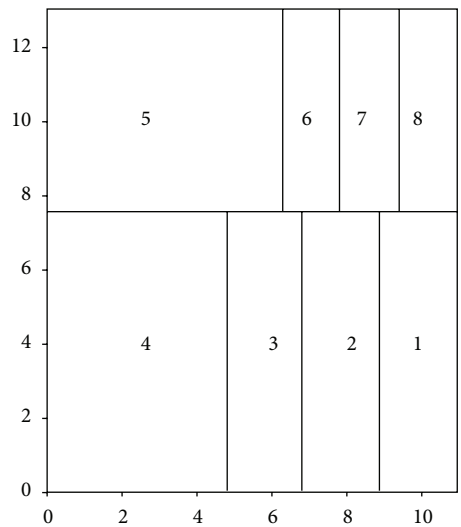


Figure 16: Best solutions found for problem FO8.

Table 5: Testing results of problem O7.

Combination	Best MWC	Worst MWC	Mean MWC	CPU Times (mins)
1	134.20	134.36	134.30	10.49
2	134.20	134.49	134.31	17.94
3	134.20	134.36	134.22	42.39
4	134.20	134.36	134.23	28.99
5	134.20	134.21	134.21	15.68
6	134.20	135.25	134.31	18.28
7	134.20	138.01	134.59	25.93
8	134.20	134.21	134.21	27.24
9	134.20	134.36	134.23	16.46

The algorithm was replicated 10 times. We set the initial mean value of HMCR and PAR to be equal to 0.98 and 0.3 and the initial standard deviation to be equal to 0.01 and 0.2, respectively. HMCR and PAR were dynamically adapted using learning mechanisms.

Table 6: Testing results of problem O8.

Combination	Best MWC	Worst MWC	Mean MWC	CPU times (mins)
1	244.72	261.00	255.56	17.07
2	255.24	260.52	258.15	27.36
3	243.18	260.54	255.78	39.97
4	243.18	260.51	254.90	40.03
5	246.31	261.67	257.02	23.96
6	243.18	267.12	255.07	27.73
7	244.73	261.00	253.75	38.79
8	243.18	260.51	251.97	40.62
9	243.18	260.58	256.06	23.77

Table 7: Testing results of problem O9.

Combination	Best MWC	Worst MWC	Mean MWC	CPU times (mins)
1	243.69	265.92	256.41	18.46
2	245.80	262.17	254.27	34.92
3	247.04	260.85	253.20	53.10
4	240.83	254.07	247.53	112.57
5	245.32	263.90	253.59	53.66
6	241.06	260.52	252.03	43.19
7	240.83	257.72	248.23	93.16
8	236.14	258.57	246.85	67.31
9	244.94	262.13	256.25	61.07

Table 8: Statistical data on results and computation time of the proposed algorithm.

Problem set	Best MWC	Worst MWC	Mean MWC	CPU times (mins)
O7	130.54	143.38	137.84	3.16
O8	243.18	260.54	255.78	4.00
O9	236.14	258.57	246.85	6.73
FO7	17.75	20.80	19.65	2.79
FO8	22.39	24.83	23.85	3.54
vC10Rs	20770.89	23715.31	22623.16	22.73
vC10Ra	19465.77	21342.42	22309.43	22.24
M11*	1189.95	1356.01	1295.09	37.02
Nug12	243.61	300.49	281.80	50.00
AB20	5151.86	6306.68	5682.54	334.12

4.2. Parameters Pretuning Results. Attribute level settings of four algorithm parameters (NI_1, HMS1, HMS2, and LP) were prespecified and are given in Tables 2, 3, and 4 for problems O7, O8, and O9, respectively. NI_2 is two times as many as

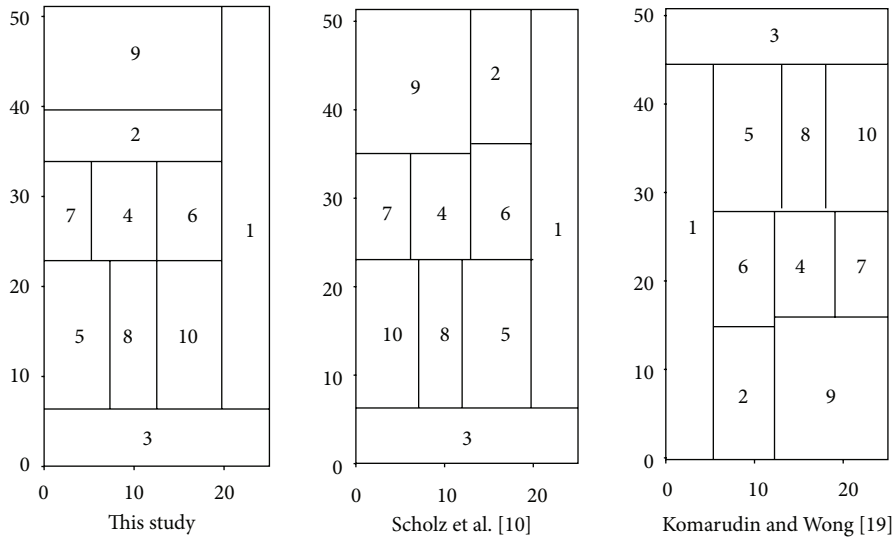


Figure 17: Best solutions found for problem vC10Rs.

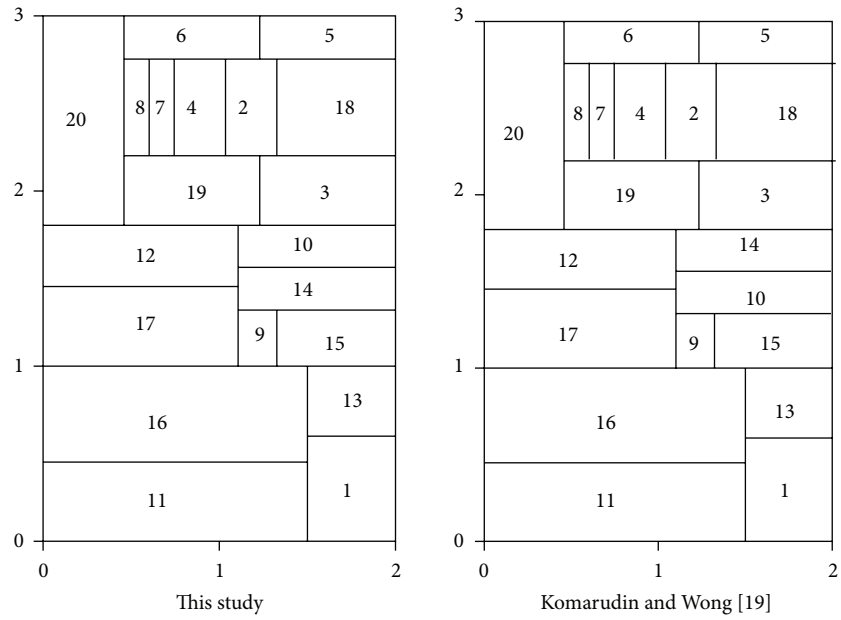


Figure 18: Best solutions found for problem AB20.

NI_1. The L9 combinations (4 factors, 3 levels) were generated by Taguchi's robust parameter design method using the orthogonal array method for each algorithm parameter.

The testing results of problems O7, O8, and O9 are given in Tables 5, 6, and 7. The level average response analysis is carried out by averaging the experimental results from three test runs corresponding to each level of each parameter. Figures 9, 10, and 11 give the results of level average response analysis by S/N ratio for problems O7, O8, and O9. It is found that the best factor level of NI_1 is 40 to 55, HMS1 is 30 to 35, HMS2 is 40 to 45, and LP is 100 to 110. The factor levels slightly increase as the problem size increases.

4.3. Testing Results. The statistical results of the proposed algorithm are summarized in Table 8. The proposed algorithm is robust since the difference between the best and worst solutions is relatively low. The computational time is acceptable for planning purposes. The best solutions found for problems are shown in Figures 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.

The results of the SHS-QCP-STS approach are compared to the results of mathematical programming approaches proposed by Sherali et al. [32] and Castillo et al. [33] as well as heuristic approaches proposed by Liu and Meller [20], Scholz et al. [10], and Komarudin and Wong [19]. We compare the results of SHS-QCP-STS to optimal solutions

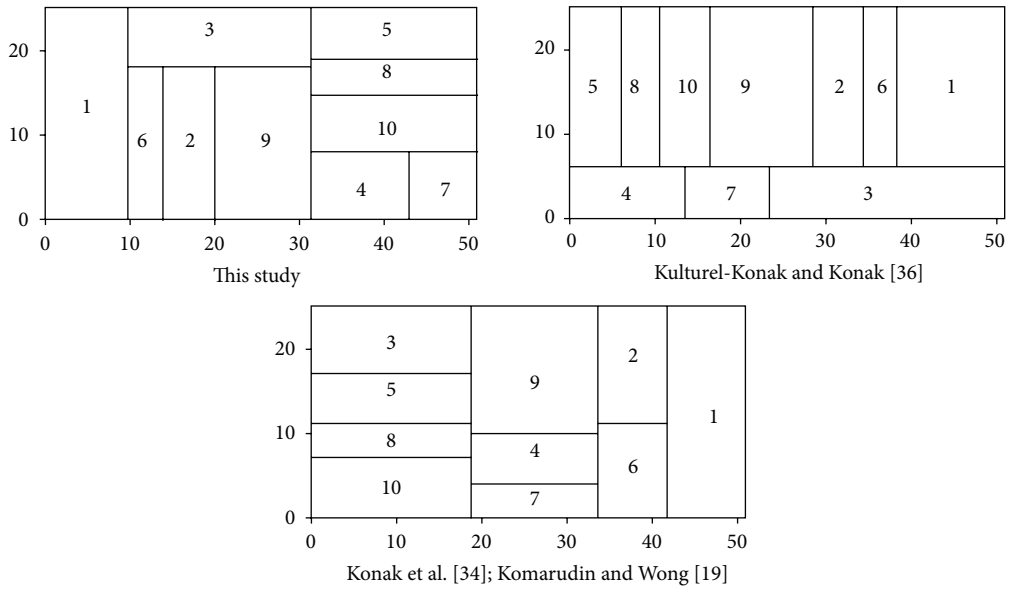


Figure 19: Best solutions found for problem vC10Ra.

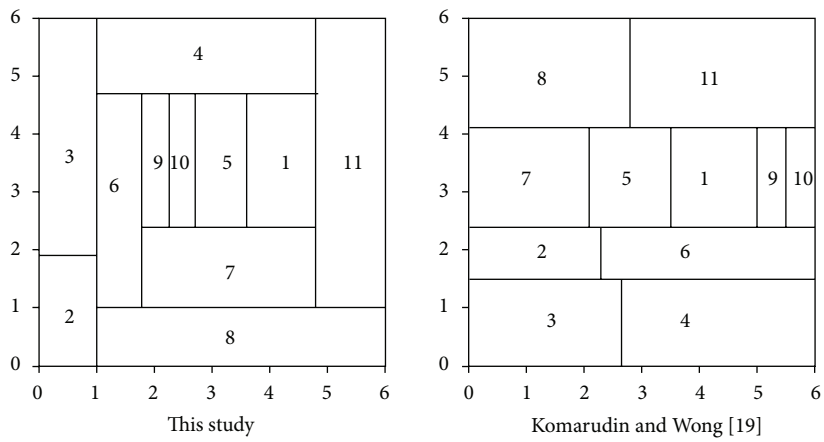


Figure 20: Best solutions found for problem M11*.

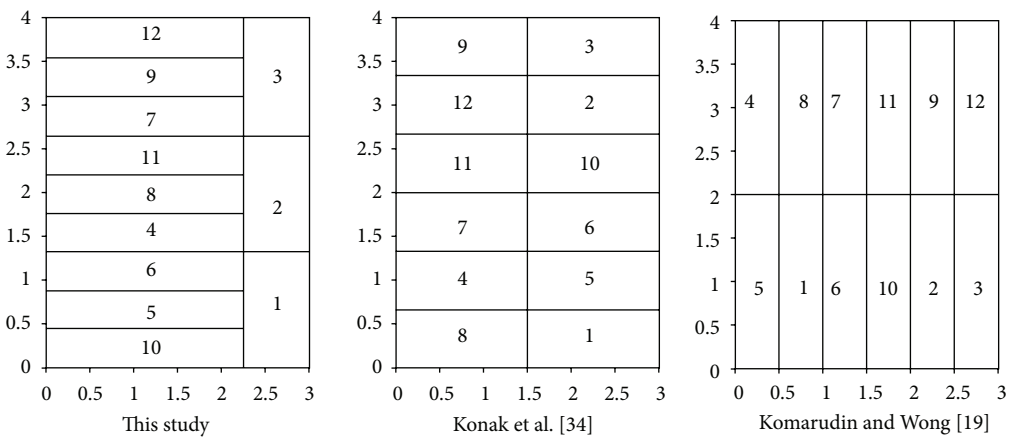


Figure 21: Best solutions found for problem Nug12.

Table 9: Comparisons of the best solutions with other approaches.

Prob. Set	Sherali et al. [32]	Castillo et al. [33]	Liu and Meller [20]	Scholz et al. [10]	Komarudin and Wong [19]	This Study
O7	131.64	131.58	131.63	132.00	131.68	**130.54**
O8	**242.89**	242.93	245.41	243.16	243.12	243.18
O9	**235.95**	236.14	246.26	239.07	236.14	236.14
FO7	—	**17.75**	**17.75**	—	—	**17.75**
FO8	—	**22.31**	**22.31**	—	—	22.39
vC10Rs	—	21297.98	19997	19994.10	**19967.60**	20,770.89
AB20	—	—	5668	—	**4972.56**	5151.86
Approach	MILP	MILP, MINLP	GA-MIP	Tabu-STS	AS-STS	SHS-QCP-STS

Table 10: Comparisons of the best solutions with other approaches.

Problem	Konak et al. [34]	Wong and Komarudin [35]	Kulturel-Konak and Konak [36]	This study
vC10Ra	21463.07	21463.10	20142.13	**19465.77**
M11*	1225	1204.15	—	**1189.95**
Nug12	265.5	262.0	—	**243.61**
Approach	MIP	AS-FBS	PSO-FBS	SHS-QCP-STS

obtained by a MINLP approach. As shown in Table 9, the best layouts obtained for O8, O9, FO7, and FO8 are the same as or similar to layouts which are found to be optimal by Sherali et al. [32] or Castillo et al. [33]. For O7, the SHS-QCP-STS found a new solution which improves by 0.16% over its previous best-known solution found by Castillo et al. [33]. For vC10Rs, the SHS-QCP-STS found a better solution found by the mathematical programming approach.

Next, we show that the SHS-QCP-STS approach is able to compete against other metaheuristic approaches. In Table 9, for O7, O8, O9, and FO7, SHS-QCP-STS can find better layouts than other metaheuristic approaches. Note that the best layout obtained for O8 is the same layout found by Komarudin and Wong [19] although SHS-QCP-STS reports slightly higher material workflow costs. For O7, the SHS-QCP-STS found the new best STS solutions which improve over the previous best-known STS solution by 0.79%. Furthermore, the best layout obtained for AB20 is very similar to previous best-known solutions except for changing department 10 with department 14, as shown in Figure 18.

In addition, the results of the SHS-QCP-STS approach are also compared to the results of mathematical programming approaches used by Konak et al. [34] as well as heuristic approaches used by Wong and Komarudin [35] and Kulturel-Konak and Konak [36]. As shown in Table 10, a new best STS solution (MWC = 19465.77) found by the SHS-QCP-STS significantly improved the best-known solutions of problem vC10Ra, with a 9.31% improvement for Konak et al. [34] and Wong and Komarudin [35] as well as 3.36% improvement for Kulturel-Konak and Konak [36]. These three layout results are illustrated in Figure 19.

For M11*, the SHS-QCP-STS found a new STS solution (MWC = 1189.95), shown in Figure 20, which is 1.18% better than the previous best-known solution solved by Wong and Komarudin [35]. A new best STS solution (MWC = 243.61) found by the SHS-QCP-STS significantly improved over the best-known solutions of Nug12, with an 8.28% improvement for Konak et al. [34] and a 7.02% improvement for Wong and Komarudin [35]. These three layout results are illustrated in Figure 21. Note that the new best STS solution is an asymmetric FBS solution.

5. Conclusion

In this research, a hybrid self-adaptive harmony search algorithm/quadratically constrained program approach is proposed for solving the unequal-area block layout problem with slicing tree structure, which is one of the layout representations in the block FLP. The proposed algorithm performs well and improves several problems compared to metaheuristics such as GA, TS, and AS as well as exact methods from the literature. The computational results show that the proposed self-adaptive harmony search based heuristic is effective and can find optimal or near-optimal solutions for most problems with fewer than 20 departments. However, one limitation of this study is the size of the layout, which should be expanded. Moreover, it would be interesting for future research to consider other practical layout restrictions such as fixed department location and fixed department size.

Acknowledgment

This work was partly supported by the National Science Council, Taiwan, under Grant NSC 99-2410-H-033-025-MY2.

References

[1] J. A. Tompkins, J. A. White, Y. A. Bozer, E. H. Frazelle, J. M. Tanchoco, and J. Trevino, *Facilities Planning*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1996.

[2] G. C. Armour and E. S. Buffa, "A heuristic algorithm and simulation approach to relative allocation of facilities," *Management Science*, vol. 9, no. 2, pp. 294–300, 1963.

[3] K.-Y. Gau and R. D. Meller, "Iterative facility layout algorithm," *International Journal of Production Research*, vol. 37, no. 16, pp. 3739–3758, 1999.

[4] A. Drira, H. Pierreval, and S. Hajri-Gabouj, "Facility layout problems: a survey," *Annual Reviews in Control*, vol. 31, no. 2, pp. 255–267, 2007.

[5] R. H. J. M. Otten, "Automatic floorplan design," *Proceeding of the 19th Design Automation Conference*, pp. 261–267, 1982.

[6] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[7] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Applied Mathematics and Computation*, vol. 199, no. 1, pp. 223–230, 2008.

[8] M.-S. Chang and W.-C. Yang, "Applying harmony search to the bulky waste recycling network design problem," in *Proceedings of the International Conference of Manufacturing Engineering and Engineering Management*, pp. 2280–2285, London, UK, July 2010.

[9] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation*, vol. 216, no. 3, pp. 830–848, 2010.

[10] D. Scholz, A. Petrick, and W. Domschke, "STaTS: a slicing tree and tabu search based heuristic for the unequal area facility layout problem," *European Journal of Operational Research*, vol. 197, no. 1, pp. 166–178, 2009.

[11] K. Y. Tam, "Genetic algorithms function optimization, and facility layout design," *European Journal of Operational Research*, vol. 63, no. 2, pp. 322–346, 1992.

[12] K. Y. Tam, "A simulated annealing algorithm for allocating space to manufacturing cells," *International Journal of Production Research*, vol. 30, no. 1, pp. 63–87, 1992.

[13] K. Y. Tam and S. K. Chan, "Solving facility layout problems with geometric constraints using parallel genetic algorithms: experimentation and findings," *International Journal of Production Research*, vol. 36, no. 12, pp. 3253–3272, 1998.

[14] R. Sedgewick and P. Flajolet, *An Introduction of Analysis of Algorithms*, Addison-Wesley, Reading, Mass, USA, 1996.

[15] R. D. Meller, V. Narayanan, and P. H. Vance, "Optimal facility layout design," *Operations Research Letters*, vol. 23, no. 3–5, pp. 117–127, 1998.

[16] L. Al-Hakim, "On solving facility layout problems using genetic algorithms," *International Journal of Production Research*, vol. 38, no. 11, pp. 2573–2582, 2000.

[17] Y. Wu and E. Appleton, "The optimisation of block layout and aisle structure by a genetic algorithm," *Computers and Industrial Engineering*, vol. 41, no. 4, pp. 371–387, 2002.

[18] E. Shayan and A. Chittilappilly, "Genetic algorithm for facilities layout problems based on slicing tree structure," *International Journal of Production Research*, vol. 42, no. 19, pp. 4055–4067, 2004.

[19] K. Komarudin and K. Y. Wong, "Applying ant system for solving unequal area facility layout problems," *European Journal of Operational Research*, vol. 202, no. 3, pp. 730–746, 2010.

[20] Q. Liu and R. D. Meller, "A sequence-pair representation and MIP-model-based heuristic for the facility layout problem with rectangular departments," *IIE Transactions*, vol. 39, no. 4, pp. 377–394, 2007.

[21] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[22] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643–656, 2008.

[23] Q.-K. Pan, P. N. Suganthan, J. J. Liang, and M. F. Tasgetiren, "A local-best harmony search algorithm with dynamic subpopulations," *Engineering Optimization*, vol. 42, no. 2, pp. 101–117, 2010.

[24] O. M. Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 49–68, 2011.

[25] P. Yadav, R. Kumar, S. K. Panda, and C. S. Chang, "An intelligent tuned harmony search algorithm for optimisation," *Information Sciences*, vol. 196, no. 1, pp. 47–72, 2012.

[26] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[27] M. S. Chang and W. C. Yang, "A stochastic network design of bulky waste recycling—a hybrid harmony search approach based on sample approximation," *International Journal of Applied Operational Research*, vol. 3, no. 3, pp. 15–40, 2013.

[28] Z. W. Geem, *Recent Advances in Harmony Search Algorithm*, vol. 270 of *Studies in Computational Intelligence*, Springer, 2010.

[29] D. J. van Camp, *A nonlinear optimization approach for solving facility layout problem [thesis]*, Department of Industrial Engineering, University of Toronto, Toronto, Canada, 1989.

[30] R. D. Meller, *Layout Algorithms for Single and Multiple Floor Facilities*, Department of Industrial and Operations Engineering, University of Michigan, 1992.

[31] C. E. Nugent, T. E. Vollman, and J. Ruml, "An experimental comparison of techniques for the assignment of facilities to locations," *Operations Research*, vol. 16, pp. 150–173, 1968.

[32] H. D. Sherali, B. M. P. Fraticelli, and R. D. Meller, "Enhanced model formulations for optimal facility layout," *Operations Research*, vol. 51, no. 4, pp. 629–644, 2003.

[33] I. Castillo, J. Westerlund, S. Emet, and T. Westerlund, "Optimization of block layout design problems with unequal areas: a comparison of MILP and MINLP optimization methods," *Computers and Chemical Engineering*, vol. 30, no. 1, pp. 54–69, 2005.

[34] A. Konak, S. Kulturel-Konak, B. A. Norman, and A. E. Smith, "A new mixed integer programming formulation for facility layout design using flexible bays," *Operations Research Letters*, vol. 34, no. 6, pp. 660–672, 2006.

[35] K. Y. Wong and K. Komarudin, "Using flexible bay structure representation and ant system algorithm solving facility layout problems," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5523–5527, 2010.

[36] S. Kulturel-Konak and A. Konak, "A new relaxed flexible bay structure representation and particle swarm optimization for the unequal area facility layout problem," *Engineering Optimization*, vol. 43, no. 12, pp. 1263–1287, 2011.