Pergamon

S0360-8352(96)00208-2

Computers ind. Engng Vol. 32, No. 1, pp. 169–185, 1997

0360-8352/97 $17.00 + 0.00

A TABU SEARCH APPROACH TO THE CELL FORMATION PROBLEM

NASSER ALJABER, WONJANG BAEK and CHUEN-LUNG CHEN

Department of Industrial Engineering, Mississippi State University, P.O. Drawer U, Mississippi State, MS 39762, U.S.A.

(Received 1 June 1996)

Abstract—The formation of machine cells and part families is a central issue in the design of cellular production systems In this paper, we propose a tabu search based approach to deal with this problem by modeling it as a "Shortest Spanning Path" problem with respect to both parts and machines. A comparison of the proposed method with some of the existing methods is presented. Our results revealed that the proposed approach possesses several advantages that make it capable of handling the problem addressed in this paper as it exists in real-world situations. Copyright © 1997 Elsevier Science Ltd

1. INTRODUCTION

Nowadays, manufacturers all over the world are coming under intense global competitive pressure to simultaneously improve manufacturing flexibility, product quality, and production costs. Consequently, the Group Technology (GT) approach to manufacturing is receiving a considerable attention among manufacturers whose production systems are batch-oriented, because GT promises widespread benefits when applied to such systems. Cellular manufacturing system (CMS) is the application of GT to manufacturing. In CMS, the entire production system is decomposed into production cells. Each one of these cells usually consists of a group of dissimilar machines dedicated to produce one or more group(s) of similar (part families). Vakharia [1] presented three major causes of the increased interest in CMS. First, the advent of modern manufacturing methods (especially the concept of Flexible Manufacturing Systems) is really based on the formation of manufacturing cells. Second, there exists an increasing demand for customized products, which are produced in smaller lot sizes than before. Thus, there seems to be an increased focus on finding new methods that have most of the strategic advantages of a job shop but also can provide some of the operational advantages of an assembly line. CMS seems to be just such a method. Third, production managers are starting to realize the importance of easing the boredom and repetitive nature of the worker tasks. In using the GT concept, multifaceted workers are an essential requirement and consequently this concept holds out some hope in motivating the work force.

The first problem that needs to be addressed when implementing a CMS is the problem of forming machine cells and their associated part families. This problem is referred to as the 'Cell Formation' Problem. The partitioning of machines into cells and parts into part families is essentially a clustering problem. The number of ways in which n objects can be decomposed into R subsets is given by the Stirling's number [2], $S(n, R) \approx R^n/R!$. This suggests that total enumeration of all feasible clustering alternatives in large size problems is impossible, even when large computers are available. Therefore, efficient heuristic methodologies are needed to cope with the problem under consideration as it exists in today's industrial applications where hundreds of part types are produced in most average size production plants.

There are many heuristic methods available in the literature to solve CF problems. Burbidge's Production Flow Analysis (PFA) [3] is one of the first approaches to the problem. In this approach, part-machine groups are formed by conducting manual analysis of part routings and machines availability for operations. The majority of existing CF methods are matrix based methods. In such methods, the problem is formulated as a part-machine matrix in which an entry $a_{ij} = 1$ or 0,

depending on whether or not part j visits machine i. In matrix based methods, clustering is made by permuting the rows and columns of the 0–1 matrix until blocks of nonzero elements are formed around the main diagonal. The most popular matrix arrangement based methods are: the Rank Order Clustering (ROC) [4], the Direct Clustering Algorithm (DCA) [5], and the Bond Energy Algorithm (BEA) [6]. Similarity measures between parts or machines are used by many cell formation methods as the basis for forming part-machine groups. The well known similarity coefficient based methods are the Single Linkage Clustering Algorithm (SLCA) [7] and the Average Linkage Clustering Algorithm (ALCA) [8]. Some CF methods [9–11] use similarity coefficients in conjunction with graph theory. In such methods, machines or parts are represented by vertices of a graph in which the edges connecting these vertices represent similarity between machines or parts. The graphs are decomposed into disconnected subgraphs to identify machine cells or part families.

There are a few CF methods based on meta-heuristic search techniques which include tabu search, simulated annealing, and genetic algorithms. Simulated annealing based algorithms [12–14] have appeared in the CF literature more often than the other two approaches. To our knowledge, [15] and [16] are the only papers that use genetic algorithms and tabu search, respectively, to solve CF problems. In [16], the CF problem is formulated as an integer programming model in which the objective is to minimize the total number of inter- and intracell moves. This model contains nine constraints which can be classified as: assignment, cell design, and machine capacity constraints. The TS based algorithm proposed in [16] was specifically designed to deal with the proposed mathematical programming formulation. Therefore, this algorithm cannot be adapted to matrix representation of CF problems.

The purpose of this paper is to develop a 'Tabu Search' based heuristic algorithm that can deal with matrix forms of CF problems. This algorithm identifies part-machine clusters, in a part-machine matrix, with the objective of minimizing the total number of intercell moves. The problem is solved by first solving two 'Shortest Spanning Path' problems, one for parts (columns) and one for machines (rows). Then, the resulting spanning paths for parts and machines are decomposed into subgraphs that represent machine groups and part families, respectively. The objective of minimizing the total number of intercell moves is achieved, indirectly, through minimization of distances between machines that belong to the same machine group (machine subgraph) and parts that belong to the same part family. The remainder of this article is organized as follows: a brief overview of tabu search techniques is presented in Section 2. Section 3 presents the proposed method along with some experimental results for selection of tabu search parameter settings which will be used by the algorithm. Section 4 discusses the practicality of the proposed approach. Section 5 presents some conclusions and directions for future research.

2. BRIEF OVERVIEW OF TABU SEARCH TECHNIQUE

A tabu search based heuristic is regarded as a "higher level" heuristic for solving combinatorial optimization problems as it is designed to guide simpler local search procedures to escape the trap of local optimality. Many successful applications of tabu search for obtaining optimal or near-optimal solutions to a variety of problems are reported in the literature. For details on some of these applications refer to the pioneering work by Glover [17, 18]. In general, tabu search is useful to find a near optimal, or possibly optimal solutions to problems which are of the type [19]:

minimize $c(x)$
subject to $x \in X$.

Where $c(x)$ is any function of a discrete variable x, and X is the set of feasible solutions. A step of tabu search starts with the current feasible solution $x \in X$ to which a function $m \in M(x)$, that transforms x into x', is applied to generate a new feasible solution ($x' = m(x)$). This transformation is called a *move* and $\{x' : x' = m(x);\ x, x' \in X;\ m \in M(x)\}$ is called the *neighborhood* of x.

In order to avoid backtracking to a local optimum, some intelligence is incorporated in the search process by using a memory structure that forbids or penalizes certain moves that would return to recently visited solutions. This memory is called 'tabu list' and the moves it contains are called 'tabu moves'. The suitable size (cardinal) of this list varies from one type of problem to another.

An application of tabu search is generally characterized by:

1. Initial solution
2. Type of moves applicable to a feasible solution x
3. Size of neighborhood
4. Tabu list size
5. Stopping criteria.

The generic procedure of tabu search techniques as outlined by Taillard [19] is as follows:

1. Start with any feasible solution x_0, an empty tabu list T. Let $x^* = x_0$, $c^* = c(x_0)$ and $k = 0$ (x^* is the best solution found up to now and c^* is the value of the objective function for this solution).
2. In $M(x_k)$ choose m, a move transforming x_k that minimizes $c(m(x_k))$ and that is not forbidden by the elements of T. The move can be chosen by complete or partial examination of $M(x_k)$. Let $x_{k+1} = m(x_k)$.
3. If $c(x_{k+1}) < c^*$, let $c^* = c(x_{k+1})$ and $x^* = x_{k+1}$.
4. If $|T|$ = a specified number S, remove the oldest element of T; add the element t defined by m and x_{k+1}. Increment k by 1.
5. If the stopping condition (optimum reached, k larger than a fixed limit, etc.) is not satisfied, go back to (2).

3. APPLICATION OF TABU SEARCH TO CF PROBLEMS

3.1. Modeling the CF problem

In order to apply tabu search technique to the cell formation problem, this problem needs to be modeled in a way that makes tabu search applicable to it. One way of doing that is to model it as a 'Shortest Spanning Path' (SSP) Problem as suggested by Slagle *et al.* [20] for general clustering problems.

3.1.1. The SSP problem. The SSP problem is a special form of the well known combinatorial optimization problem called Traveling Salesman Problem (TSP). The TSP can be stated as follows [11]:

Given a graph $G(V, E)$, where V is the set of vertices and E is the set of edges connecting these vertices with length D_{ij} for each $(i, j) \in V$, find a cycle C that is incident to all $v \in V$ which minimizes:

$$\sum_{(i,j) \in C} D_{ij}.$$

The only difference between the two problems (i.e. TSP and SSP) is that the TSP requires a return to the starting vertex from the last one while SSP does not.

Tabu search can be adapted to the SSP problem by treating it as a permutation problem in which the objective is to find the permutation that gives the shortest spanning path, i.e.,

min. DIST(Ω)
St. Ω:Feasible Permutations of $1, \ldots, n$

where

n = number of vertices in a graph

DIST(Ω) = distance corresponding to a given permutation (sequence) Ω. This distance is calculated as:

$$\text{DIST}(\Omega) = \sum_{i=1}^{n} \sum_{j=1}^{n} D_{ij} X_{ij} \quad \text{for } i \neq j$$

where

$X_{ij} = 1$, if i adjacently preceded j, otherwise $X_{ij} = 0$

D_{ij} = distance between two vertices (parts or machines) i and j.

Distance between parts (or machines) can be measured using one of the distance measures available in the literature. For a recent review of distance and similarity measures refer to [21]. Recognize that similarity measures can be converted to distance measures and vice versa. This is usually made by subtracting their computed values from their upper bounds. For instance, the Jaccard similarity measure [7] can be converted to a distance measure by subtracting it from its upper bound which is 1, i.e.,

$$D_{ij} = 1 - S_{ij} = 1 - \frac{C_{ij}}{T_i + T_j - C_{ij}}$$

where

S_{ij} = computed value of the Jaccard similarity measure between parts (or machines) *i and j*.
C_{ij} = number of parts common to both machines (or number of machines common to both parts) i and j.
T_l = total number of parts processed by machine (or total number of machines required by part) l; $l = i$ or j.

3.1.2. The relationship between CF and SSP problems. In matrix based cell formation methods, part machine grouping is made by rearranging rows and columns of the 0–1 matrix so that the nonzero elements (the 1s) are grouped into a number of blocks (clusters). Rows and columns that constitute boundaries of a block represent a machine group and its associated part family, respectively. Forming this blocks is accomplished by moving some of the nonzero elements from their positions in the initial matrix to some other positions so that the elements that constitute the same block are gathered in close proximity around the center of their block. Since positions of nonzero elements in the final matrix are not known before solving the problem, some criteria must be used to guide the repositioning process.

In our approach, repositioning of the nonzero elements for the purpose of forming blocks is dealt with by solving two SSP problems, one for rows and one for columns. In these problems rows or columns correspond to a set of vertices in a graph, and distances between them correspond to length of edges connecting the vertices. Solving SSP problems for row and column graphs results in vertical and horizontal moves, respectively, made by the nonzero elements toward block formation. This can be explained as follows: by definition, the SSP problem is a problem of sequencing the vertices in a graph so that the length of the spanning path, as measured by the sum of distances between adjacent vertices, is minimal. In the context of the problem addressed here, distance between two vertices (rows or columns) is a function of the number of nonzero elements common to both vertices, i.e., the distance decreases as the number of common nonzero elements increases. This makes the horizontal boundaries of these submatrices easy to determine. Hence, by solving a SSP problem for one dimension, say rows, the rows that have a large number of common nonzero elements are expected to occupy neighboring positions in the SSP sequence of rows. Consequently, rearranging the rows in the matrix according to their SSP sequence would cause shifting of the nonzero elements along the vertical axis so that the elements that belong to highly similar rows would become closer to each other in the direction of the vertical axis. Such a result leads to a division of the matrix into a number of submatrices in which intra-submatrix rows are highly similar while inter-submatrix rows are highly dissimilar. Hence, machine groups, which correspond to sets of rows within the resulting submatrices, are identifiable. In the context of block formation, SSP sequencing of rows is only a half way toward complete formation of the blocks since it leads to the shifting of nonzero elements along the vertical axis only. Therefore, in order to identify the part families, the process of block formation needs to be completed by solving another SSP problem considering columns instead of rows.

Illustrative example

To see how cell formation and SSP problems are related, consider the 5 × 5 part-machine matrix presented in Fig. 1. The distance matrices, which were constructed using the Jaccard distance measure, for machines and parts are presented in Figs 2 and 3, respectively.

	P1	P2	P3	P4	P5
M1	1	0	0	1	0
M2	0	1	1	0	1
M3	1	0	0	0	0
M4	0	1	1	0	0
M5	0	0	0	1	0

Fig. 1. The initial 0–1 matrix.

	M1	M2	M3	M4	M5
M1	-	1	0.5	1	0.5
M2		-	1	0.33	1
M3			-	1	1
M4				-	1
M5					-

Fig. 2. Distance matrix for machines.

	P1	P2	P3	P4	P5
P1	-	1	1	0.67	1
P2		-	0	1	0.5
P3			-	1	0.5
P4				-	1
P5					-

Fig. 3. Distance matrix for parts.

As we mentioned earlier, we need to obtain SSP sequences for both machines and parts:

(i) Obtaining SSP sequence for machines

Let Ω_m represent the sequence of machines in the 0–1 matrix. In the initial matrix:

$$\Omega_m = 1\text{–}2\text{–}3\text{–}4\text{–}5 \Rightarrow X_{12} = X_{23} = X_{34} = X_{45} = 1, \quad \text{other } X_{ij}\text{s} = 0.$$

The corresponding distance will be:

$$\text{DIST}(\Omega_m) = D_{12}X_{12} + D_{23}X_{23} + D_{34}X_{34} + D_{45}X_{45} = D_{12} + D_{23} + D_{34} + D_{45} = 1 + 1 + 1 + 1 = 4$$

—Swap machines 1 and 2 to obtain a new sequence:

$$\Omega_m = 2\text{–}1\text{–}3\text{–}4\text{–}5; \quad \text{DIST}(\Omega_m) = D_{21} + D_{13} + D_{34} + D_{45} = 1 + 0.5 + 1 + 1 = 3.5$$

—Swap machines 1 and 4 to get:

$$\Omega_m = 2\text{–}4\text{–}3\text{–}1\text{–}5; \quad \text{DIST}(\Omega_m) = D_{24} + D_{43} + D_{31} + D_{15} = 0.33 + 1 + 0.5 + 0.5 = 2.33.$$

	P1	P2	P3	P4	P5
M2	0	1	1	0	1
M4	0	1	1	0	0
M3	1	0	0	0	0
M1	1	0	0	1	0
M5	0	0	0	1	0

Fig. 4. The 0–1 matrix with machines ordered.

	P5	P2	P3	P4	P1
M1	0	0	0	1	1
M2	1	1	1	0	0
M3	0	0	0	0	1
M4	0	1	1	0	0
M5	0	0	0	1	0

Fig. 5. The 0–1 matrix with parts ordered.

The 0–1 matrix corresponding to the best machine sequence found so far is presented in Fig. 4. By inspecting this matrix we can see that the objective function (DIST(Ω_m)) cannot be minimized any further.

(ii) Obtaining SSP sequence for parts

Let Ω_p represent the sequence of parts in the part-machine matrix. In the initial matrix:

$$\Omega_p = 1\text{–}2\text{–}3\text{–}4\text{–}5; \quad \text{DIST}(\Omega_p) = D_{12} + D_{23} + D_{34} + D_{45} = 1 + 0 + 1 + 1 = 3.0$$

—Swap parts 1 and 5 to get:

$$\Omega_p = 5\text{–}2\text{–}3\text{–}4\text{–}1; \quad \text{DIST}(\Omega_p) = D_{52} + D_{23} + D_{34} + D_{41} = 0.5 + 0 + 1 + 0.67 = 2.17/$$

The 0–1 matrix shown in Fig. 5 which corresponds to the new parts' sequence (5–2–3–4–1), indicates that there is no further improvement that can be achieved by generating a new sequence. The 0–1 matrix in which machines and parts are ordered according to their SSP sequences is shown in Fig. 6. There are two clusters appearing in this matrix with no intercell moves. The first cluster contains machine set {2, 4} and part family {2, 3, 5}. The other cluster contains machine set {1, 3, 5} and part family {1, 4}. Note that the example problem was solved optimally by making few number of moves which were decided upon simply by inspecting the 0–1 matrix. However, if a larger problem (say 10 × 10) is to be solved, then, efficient search procedure such as tabu search is required.

3.2. Developing the TS based algorithm

As it has been mentioned earlier, there are some parameters that affect the performance of tabu search. The best parameter settings for tabu search differ from one type of problem to another. This section presents the specific characteristics of the proposed TS algorithm and the results of the experiments that we carried out to select the best combination of parameter settings that suit the type of problem under consideration.

3.2.1. Initial solution. The initial solution in tabu search applications can be either a random or a heuristic solution. Our algorithm uses the sequences in which the parts and the machines are given in the initial part-machine matrix (i.e., machine in row $(i) = i$, part in column $(j) = j$) as initial solutions to the problems of sequencing parts and machines, respectively.

	P5	P2	P3	P4	P1
M2	1	1	1	0	0
M4	0	1	1	0	0
M3	0	0	0	0	1
M1	0	0	0	1	1
M5	0	0	0	1	0

Fig. 6. The solution matrix.

Table 1. Results of experiments

Size of neighborhood		Whole												Partial											
Type of move		Insert				Swap				API				Insert				Swap				API			
Tabu list size		5	7	9	11	5	7	9	11	5	7	9	11	5	7	9	11	5	7	9	11	5	7	9	11
No. of parts	5	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	3.0	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8
	10	5.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	6.1	6.3	6.1	6.6	5.6	5.6	5.6	5.6	5.6	5.6	5.6	6.2	6.6	6.2	5.6	6.2
	15	8.9	9.7	8.9	9.7	10.1	8.9	9.9	11.3	11.6	10.5	11.1	11.9	8.9	8.9	8.9	10.1	8.9	8.9	9.0	10.4	10.3	9.8	10.3	10.8
	20	11.2	11.2	12.6	13.2	12.3	12.8	13.2	12.8	14.1	14.6	15.2	15.5	11.2	11.8	11.2	11.2	11.2	11.6	11.2	12.8	13.5	14.1	13.0	14.1
	25	14.9	14.9	15.8	14.2	15.6	16.1	15.0	16.1	17.7	17.7	17.3	18.4	14.2	14.2	14.2	15.3	14.9	14.4	15.3	15.8	16.2	16.7	17.2	16.2
	30	17.6	16.4	17.3	16.4	17.6	17.1	17.6	18.1	20.6	21.1	21.6	22.2	16.4	17.3	16.9	16.4	16.6	17.5	17.1	17.5	19.8	19.8	19.2	20.3
	35	19.2	19.8	20.4	20.9	20.1	19.6	20.5	20.1	23.5	22.7	24.0	23.3	18.7	18.1	18.1	18.1	19.7	18.7	19.3	20.1	22.2	21.7	22.2	22.8
	40	21.9	21.4	22.5	23.1	22.2	23.5	22.2	23.1	26.3	26.3	25.7	26.9	19.8	19.8	20.7	21.1	21.8	20.9	21.4	22.3	24.7	25.6	25.2	25.2
	45	24.5	23.9	24.5	24.8	24.5	23.9	25.1	25.6	30.1	29.6	29.6	30.9	22.7	22.7	22.7	23.9	23.0	23.6	23.0	24.2	28.3	28.8	23.3	29.2
	50	27.0	26.1	26.6	27.0	28.3	27.3	28.8	27.8	32.6	32.2	33.3	33.3	26.4	25.1	25.1	26.8	27.7	26.7	27.1	27.7	31.0	31.7	32.3	32.6
	55	29.4	29.4	29.9	30.6	30.8	30.2	30.8	31.4	36.6	35.7	35.7	36.2	37.1	28.5	29.0	28.5	30.1	29.8	29.3	30.2	35.4	34.8	34.3	35.9
	60	32.1	33.2	32.6	33.2	32.9	33.8	34.3	34.7	39.9	40.4	40.4	40.9	31.3	31.0	31.0	31.0	32.4	31.8	32.9	32.9	38.3	39.2	39.2	39.2
	65	34.9	34.5	35.4	35.4	36.2	35.5	36.2	36.7	42.8	42.2	42.8	43.4	32.9	32.9	33.5	34.1	34.3	33.7	34.3	34.7	41.3	40.2	40.8	41.7
	70	36.1	36.7	36.7	37.2	38.4	38.4	38.0	39.3	45.7	46.3	46.3	46.9	35.3	34.8	35.3	36.1	36.1	36.7	37.1	37.7	44.1	44.7	45.4	45.8
	75	39.4	40.1	40.7	41.3	42.3	41.6	42.3	42.8	50.9	50.9	50.4	51.6	38.6	39.5	39.1	40.2	41.1	41.1	40.6	41.7	48.9	48.4	49.1	49.5
	80	42.9	43.7	44.8	44.3	43.7	44.6	44.1	45.2	54.3	54.8	53.7	54.8	41.1	41.1	41.7	42.2	43.2	43.7	42.6	43.2	52.3	52.8	53.3	53.9
	85	45.7	45.7	46.9	47.3	46.7	46.2	47.4	47.7	57.9	57.0	57.6	58.4	44.1	43.6	44.5	43.6	45.4	44.8	45.6	46.1	55.9	56.9	56.4	56.4
	90	49.3	48.5	51.2	50.1	50.1	51.3	51.3	50.8	62.0	62.0	61.6	62.4	47.1	47.1	47.1	48.3	49.3	48.8	48.9	49.3	60.6	60.1	61.0	61.7
	95	52.4	52.8	53.1	53.0	53.9	53.4	53.9	54.4	65.0	64.1	64.5	65.6	50.7	50.2	51.3	50.2	52.6	53.2	52.6	63.5	63.5	63.1	62.4	63.5
	100	55.7	55.2	56.1	55.23	56.4	56.8	55.9	57.3	68.7	68.2	68.7	69.2	53.3	53.3	53.8	54.5	55.0	55.9	55.5	56.4	66.3	66.3	67.6	67.1
NBEST		4	4	3	4	2	3	2	2	1	1	1	0	14	16	11	8	4	3	3	1	1	1	2	1

NBEST = Number of times best solution is obtained.
API = Adjacent pairwise interchange.
Bq: Best result is obtained for the corresponding problem (No. of parts).

3.2.2. Basic neighborhoods (moves). For permutation problems, the neighborhood may be defined in several ways. The most common types of neighborhoods are as follows:

(1) Adjacent Pairwise Interchange

This type of neighborhood contains all those permutations σ' obtained from σ by exchanging two adjacent jobs placed at the ith and the $(i+1)$th positions, i.e.,

$$\sigma = \langle x_1, \ldots, x_i, x_{i+1}, \ldots, x_n \rangle \Rightarrow$$

$$\sigma' = \langle x_1, \ldots, x_{i+1}, x_i, \ldots, x_n \rangle; \quad \forall\, i \in \{1, \ldots, n-1\}$$

(2) Insert

This type of move contains all those permutations σ' obtained from σ by inserting the component in position j before component in position i, i.e.,

$$\sigma = \langle x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n \rangle \Rightarrow$$

$$\sigma' = \langle x_1, \ldots, x_{i-1}, x_j, x_i, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n \rangle; \quad \forall\, i, j \in \{1, \ldots, n\}, i \neq j, j-1$$

(3) Swap

This neighborhood contains all those permutations σ' obtained from σ by swapping the components in positions i and j, i.e.,

$$\sigma = \langle x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n \rangle \Rightarrow$$

$$\sigma' = \langle x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n \rangle; \quad \forall\, i, j \in \{1, \ldots, n\}, i \neq j.$$

3.2.3. Size of tabu list. Several applications of tabu search have found the *magic number* $7(\pm 2)$ to be a remarkably good choice for tabu list size [22]. This outcome has invited speculations as to whether the role of such a number in human short term memory is the result of a natural selection process related to human problem solving ability [22]. Obviously, characteristics of the solution space can affect the ideal tabu list size, but it may also be that problems typically encountered in practice favor a range of values that cluster around 7.

3.2.4. Size of neighborhood. Size of neighborhood determines how to examine the neighborhood before choosing a move leading to the next step. Two possible choices are:

(1) Whole Neighborhood

Examine the entire neighborhood and take the most successful move that is not tabu. This method needs more computation time before a move is made.

(2) Partial Neighborhood

Generally, partial neighborhood is defined as the search method in which acceptance of a move is made by examining only part of the entire neighborhood. Such broad definition suggests that partial neighborhood can be defined in many ways (see [23] for details). For example, examine the neighborhood and take the first improving solution, that is not tabu, as the candidate solution in the next search. Under this definition, which is adopted in this study, the size of the neighborhood that needs to be covered by the search process before a new move is accepted is unknown. Simply because the location of the next improving solution, in the solution space, is unknown.

3.2.5. Defining and storing the tabu moves. There are several different approaches in the literature for defining and storing the tabu moves. In our algorithm, however, we are adopting the most widely recognized approach. In this approach, the tabu moves are defined as the reversals of the latest S accepted moves. This means that the tabu list is manipulated like a FIFO queue in that each time a new move is posted to one end of the list, it pushes the oldest move out the other end. In tabu search, the trap of local optimality becomes escapable by making a sort of temporary hold

on repeating certain moves (i.e., recently accepted moves), so that the search process is given the chance to tackle uncharted regions in the solution space. Forbidding occurrence of the reversal of an undesired move is made to avoid coming back to the condition (solution) that has already led, and therefore might lead again, to the occurrence of this move. The reason for using recency based tabu lists can be explained as follows: in tabu search, the search process moves successively from one region to another in the solution space, and recency of a move reflects the likelihood that the search is being conducted in the region where this move belongs. Thus, in order to expand the diversity of the search (i.e., by moving, as quickly as possible, from the currently searched region to another region in the solution space), it is relevant to consider recent moves only.

3.2.6. Stopping criteria. The stopping criteria plays a major rule in determining the computation time and the quality of solution obtained by tabu search. The most commonly used stopping rules are [19]:

(1) Stop if the number of iterations made without improving the current solution exceeds a specified constant k_1
(2) Stop if the number of iterations exceeds a specified constant k_2.

The TS based algorithm proposed here uses rule (1) because it sounds more relevant to the search process since we do not have any idea beforehand about the number of iterations required to reach a good solution. Hence, using rule (2) might lead to termination of the search process even if rapid improvement is taking place, or it might result in unnecessary computation time if the best solution is obtained in the early stages of the search process.

3.2.7. Experimental results. In order to select the tabu search parameter settings that suit the type of problem addressed in this paper, a comparative experimentation was conducted. The experiments consist of 20 different size problems which were sampled from a uniform distribution with the number of columns (parts) ranging between 5 and 100, while the number of rows was set at 25. In these problems, tabu search was only used to perform SSP sequencing of parts. Recognize that the fashion in which the sequencing of parts is made, is exactly the same as that of the machines' sequencing. Hence, it is sufficient to make selection of TS parameter settings based on a study that is conducted on either one of these two dimensions. In our experimentations, tabu search was applied to each one of the 20 generated problems using each possible combination of the following parameter settings:

(1) Size of neighborhood:Partial and Whole
(2) Type of move:Insert, Swap, and Adjacent Pairwise Interchange (API)
(3) Size of tabu list:5, 7, 9, and 11.

This results in a total of 480 runs (i.e., $2 \times 3 \times 4 \times 20$ problems). In order to evaluate the goodness of solutions, values of the associated objective functions (DIST(Ω)) are used. The criteria for selecting the best combination of parameter settings is the number of times in which a combination gives the best result (the least objective function values) over the 20 generated problems. Throughout the experiments, stopping rule (1), which was presented earlier in this paper, is used with K_1 = number of parts.

Results of the experiments are presented in Table 1. The entries of this table are the computed values of the objective functions, using the Jaccard distance measure. According to these results, the most suitable combination of parameter settings for our problem is: Partial neighborhood size, Insert type of move, and tabu list size of 7. This combination was able to obtain the best results more frequently (16 times) than the rest of the combinations. There are two comments which are worth making here concerning our findings. First, superiority of the 'Partial' neighborhood over the 'Whole' neighborhood may be attributed to the perturbation (i.e., exploration of more new regions) introduced in the solution space by accepting every successful move that is not tabu. Second, the Adjacent Pairwise Interchange (API) type of move was notably inefficient with respect to goodness of its solutions, as indicated by its large objective function values in comparison with Insert type of move. A possible reason for such a finding could be the API's number of moves (cardinal), made in each iterations, which is only $(N - 1)$. This number is very small as compared to the cardinal of the Insert type of move which is $(N - 1)^2$.

3.3. The proposed CF method

The proposed cell formation procedure consists of the following three phases: (I) obtaining SSP sequences for parts and machines using tabu search, (II) constructing the solution matrix, and (III) determining the boundaries of the part-machine clusters. These phases are presented below.

PHASE I: the TS algorithm for sequencing machines and parts

The proposed TS algorithm is used to construct the SSP sequences for the parts and machines independently. This algorithm will be presented after we define the following notations:

n = number of parts (columns) in the part-machine matrix
m = number of machines (rows) in the part-machine matrix
$NIWI$ = number of iterations made without improving the current solution
$N_I(x)$ = the neighborhood defined by the insert type of moves
S = size of the tabu list
T = the set of tabu moves
x = sequence (permutation) of rows or columns
DIST(x) = distance corresponding to sequence x
k_1 = a specified number of iterations corresponding to Stopping rule (1)
$\langle x, y \rangle$ = a move that transforms a solution x into another solution y

(i) Constructing SSP sequence for the machines

Step 0: Input the part-machine matrix
Set: $k_1 = m$
$NIWI = 0$
$S = 7$
$T = \{\emptyset\}$
$x = x_0$ (hint: x_0 is the order of machines as given in the initial matrix)

Step 1: Construct the distance matrix for machines
Step 2: Calculate 'DIST(x)' and set: incumbent = DIST(x)
Step 3: Let $NIWI = NIWI + 1$
Step 4: Make a move $\langle x, x' \rangle \in N_I(x)$ with $\langle x, x' \rangle \notin \{T\}$
Step 5: Calculate 'DIST(x')'
Step 6: If (DIST(x') < incumbent) then
—update T by including $\langle x', x \rangle$
—set: incumbent = DIST(x')
—$x = x'$
—$NIWI = 0$
—go to step 3
Else
—If ($NIWI < K_1$) then
—go to step 3
Else
—stop

(ii) Constructing SSP sequence for the parts

—repeat all the above steps considering parts (columns) instead of machines (rows).

PHASE II: constructing the solution matrix

—construct the final matrix by rearranging the rows and columns of the initial matrix according to their SSP sequences obtained in Phase I.

(a)

MACHINES		PARTS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	$D_{i,i+1}$	
		PF(1)					PF(2)			PF(3)				PF(4)		PF(5)			PF(6)				
	1										1	1	1									0.25	
MG(1)	2									1	1	1	1									0.50	
	3								1	1	1	1	1	1	1			1				0.70	
MG(2)	4	1						1	1	1					1							0.50	H(1)
	5	1						1	1					1								0.60	
MG(3)	6						1	1	1													0.40	H(2)
	7				1	1	1	1	1													0.77	
	8	1	1	1	1	1															1	0.17	H(3)
	9	1	1	1	1	1																0.20	
MG(4)	10	1		1	1	1																0.33	
	11	1	1	1	1	1														1		0.29	
	12	1	1	1	1														1	1		0.75	
	13					1													1	1	1	0.00	H(4)
	14					1													1	1	1	0.40	
MG(5)	15																	1	1	1	1	0.33	
	16	1														1	1			1	1	0.71	
	17															1	1	1	1			0.25	H(5)
	18															1	1	1				0.25	
MG(6)	19														1	1	1	1				0.33	
	20												1	1	1	1	1	1				-	
$D_{j,j+1}$		5 0	2 0	1 7	3 8	8 6	5 0	2 0	6 7	5 0	0 0	0 0	6 0	5 0	6 7	2 0	2 9	8 0	2 9	4 3	-		
							V(1)			V(2)				V(3)		V(4)			V(5)				

(b)

MACHINES		PARTS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	$D_{i,i+1}$	
		PF(1)					PF(2)			PF(3)				PF(4)					PF(5)				
	1										1	1	1									0.25	
MG(1)	2									1	1	1	1									0.50	
	3								1	1	1	1	1	1	1			1				0.70	
	4	1						1	1	1					1							0.50	H(1)
	5	1						1	1					1								0.60	
MG(2)	6						1	1	1													0.40	
	7				1	1	1	1	1													0.77	
	8	1	1	1	1	1															1	0.17	H(2)
	9	1	1	1	1	1																0.20	
MG(3)	10	1		1	1	1																0.33	
	11	1	1	1	1	1														1		0.29	
	12	1	1	1	1														1	1		0.75	
	13					1													1	1	1	0.00	H(3)
	14					1													1	1	1	0.40	
MG(4)	15																	1	1	1	1	0.33	
	16	1														1	1			1	1	0.71	
	17															1	1	1	1			0.25	H(4)
	18															1	1	1				0.25	
MG(5)	19														1	1	1	1				0.33	
	20												1	1	1	1	1	1				-	
$D_{j,j+1}$		5 0	2 0	1 7	3 8	8 6	5 0	2 0	6 7	5 0	0 0	0 0	6 0	5 0	6 7	2 0	2 9	8 0	2 9	4 3	-		
							V(1)			V(2)				V(3)					V(4)				

Fig. 7(a). A 20 × 20 solution matrix obtained by the TSA.

Fig. 7(b). The resulting matrix after determining the groups boundaries.

PHASE III: determining boundaries of part-machine clusters

In this part of our paper, we present a heuristic procedure to determine boundaries of the part-machine clusters in TSA's solution matrices. The problem of determining the horizontal and vertical boundaries of the clusters is equivalent to the problem of finding the edges in the machines' and parts' spanning paths, respectively, that need to be eliminated in order to decompose these two paths into a number of subpaths (machine cells or part families), so that the overall length of these subpaths is minimized. The number of subpaths, into which the machines' and parts' spanning paths should be decomposed, is determined by the number of natural part-machine groups (clusters) that exist in the given data set. In ideal cases, where the groups are mutually independent, the number of groups can be determined easily by inspecting the solution matrix. However, in cases where high interactions exist between the groups, the number of natural groups that exist in the data set, is difficult to determine. In order to overcome this problem, our heuristic procedure starts by decomposing both the machines' and parts' spanning paths into a user-specified number of segments, UL, that represents an upper limit to the number of groups. Then, it reduces the number of groups iteratively, using maximum reduction in number of intercell moves as a criteria, until a specified stopping criteria is satisfied. Clearly, the value of UL has to be greater than or, at least, equal to the number of natural groups in the data set. Therefore, if a rough estimate, to the maximum number of groups, cannot be made by inspecting the solution matrix, then, UL can be assigned a feasible large value (e.g., UL = $\min\{n, m\}/2$). The algorithmic form of the proposed procedure will be presented after we defined the following notations:

UL = a user-specified upper limit to the number of part-machine groups.
ng = the current number of part-machine groups.
$PF(j)$ = part family J.
$MG(I)$ = machine group I.
$H(I)$ = the horizontal line that separates $MG(I)$ and $MG(I+1)$.
$V(j)$ = the vertical line that separates $PF(j)$ and $PF(j+1)$.
$S(I, J)$ = the submatrix formed by the intersection between row set I and column set J.
$NS(I, J)$ = the number of 1s in $S(I, J)$.
$X(I, J)$ = 1, if $MG(I)$ and $PF(j)$ are assigned to each other. Otherwise, $X(I, J) = 0$
$B = \{S(I, J) | X(I, J) = 1\}$. The elements of B are referred to as 'blocks'.
$\Delta(I, J)$ = the change (increase or decrease), in number of intercell moves, that results from breaking block $S(I, J) \in B$.

The proposed procedure

Step 0: Set ng = UL and $B = \{\emptyset\}$

Step 1: Determine the boundaries of ng machine groups (row sets) and ng part families (column sets). This is accomplished by drawing a horizontal line ($H(I)$) and a vertical line ($V(j)$) across each one of the longest $ng - 1$ edges in the machines' and the parts' spanning

Table 2. Results of comparisons

Data set	Size $M \times N$	Source of data set	$N1$	Original solution			TSA solution		
				Nc	*Ne*	MAX	*Nc*	*Ne*	MAX
1	12 × 19	[11]	75	3	23	4	3	22	4
2	14 × 24	[26]	58	4	4	9	4	2	9
3	16 × 43	[11]	126	5	29	5	5	27	5
4	20 × 35	[27]	152	4	39	7	4	35	7
5	24 × 40	[28], DATA SET 1	131	7	0	5	7	0	5
6	24 × 40	[28], DATA SET 2	130	7	10	5	7	10	5
7	24 × 40	[28], DATA SET 5	130	7	49	6	7	46	5
8	24 × 40	[28], DATA SET 6	131	6	54	6	7	52	6
9	24 × 40	[28], DATA SET 7	131	8	64	10	9	61	7
10	30 × 40	[29]	152	5	21	8	5	18	8

$N1$ = Number of '1' elements in the 0–1 matrix.
Nc = Number of cells (clusters).
Ne = Number of intercell moves.
MAX = Number of machines in the largest cell.

paths, respectively. This results in partitioning of the solution matrix into $(ng)^2$ submatrices.

Step 2: Assign each part family to one machine group, and vice versa. This is done by solving the linear assignment problem presented below. This problem can be solved by using the Hungarian algorithm. For details on this algorithm, the reader is referred to [24].

Maximize

$$\sum_{I=1}^{UL} \sum_{J=1}^{UL} NS(I, J)X(I, J)$$

Subject to

$$\sum_{J=1}^{UL} X(I, J) = 1 \quad \text{for } I = 1, 2, \ldots, UL$$

$$\sum_{I=1}^{UL} X(I, J) = 1 \quad \text{for } J = 1, 2, \ldots, UL$$

$$X(I, J) = 0 \text{ or } 1 \quad \text{for } I, J = 1, 2, \ldots, UL$$

Step 3: Iteratively, reduce the number of groups until the specified stopping criteria (see (3) below) is reached. In each iteration, the number of groups is reduced, by one group, by breaking the block, among the currently existing blocks, that results in the largest reduction in number of intercell moves. Breaking a block, $S(I, J) \in B$, means that the assignment that was made between $MG(I)$ and $PF(j)$ is broken, so that the machines in $MG(I)$ are included in an adjacent machine group (i.e., in $MG(I + 1)$ or $MG(I - 1)$) and the parts in $PF(j)$ are included in an adjacent part family. This means that a block is broken by eliminating one of its bordering 'horizontal–vertical' line combinations. Hence, for each block, at most, two horizontal lines ($H(I)$ and $H(I - 1)$) and, at most, two vertical lines ($V(j)$ and $V(j - 1)$) have to be checked in order to find the best candidates (one vertical and one horizontal) for elimination. The stopping rule, used here, is to stop when the number of intercell moves cannot be reduced, any further, by breaking any one of the remaining blocks. The algorithmic form of the proposed procedure is as follows:

(1) For each $S(I, J) \in B$, compute: $\Delta(I, J) = \Delta H(I) + \Delta V(j) - NS(I, J)$, where:

$$\Delta H(I) = \max\{NS(I, J^+(I)), NS(I, J^-(I))\}$$

$$\Delta V(j) = \max\{NS(I^+(j), J), NS(I^-(j), J)\},$$

where:

$$I^+(j) = I | S(I, J + 1) \in B; \text{ and } I^-(j) = I | S(I, J - 1) \in B$$

$$J^+(I) = J | S(I + 1, J) \in B; \text{ and } J^-(I) = J | S(I - 1, J) \in B$$

(2) Identify: $\Delta(I^*, J^*) = \text{maximum } \{\Delta(I, J) | S(I, J) \in B\}$.

(3) If $\Delta(I^*, J^*) \leqslant 0$, go to 6. Otherwise, break the block $S(I^*, J^*)$ by eliminating the line combination $H^* - V^*$, that maximizes the reduction in number of intercell moves. H^* and V^* are determined as follows:

If $NS(I^*, J^+(I^*)) > NS(I^*, J^-(I^*))$, let $H^* = H(I^*)$. Otherwise, let $H^* = H(I^* - 1)$. Also, if $NS(I^+(j^*), J^*) > NS(I^-(j^*), J^*)$, let $V^* = V(j^*)$. Otherwise, let $V^* = V(j^* - 1)$.

(4) Break the assignment that was made between $MG(I^*)$ and $PF(j^*)$. Combine $MG(I^*)$ with the adjacent machine group that was separated from $MG(I^*)$ by the horizontal line H^*. Similarly, combine $PF(j^*)$ with the adjacent part family that was separated from $PF(j^*)$ by the vertical line V^*.

(5) Update the partitioned matrix and the set B in accordance with (4) and (5), let $ng = ng - 1$, and go to (1)

(6) Stop; the number of intercell moves cannot be reduced by breaking one of the remaining blocks.

EXAMPLE

In order to illustrate the proposed procedure for determining boundaries of the part-machine groups, consider the TSA's solution matrix presented in Fig. 7(a). A step by step illustration of the proposed procedure, using UL = 6, is given below.

Step 1: The longest 5 edges in the machines' spanning path are: (3, 4), (5, 6), (7, 8), (12, 13) and (16, 17), with length ($D_{i,i+1}$) of 0.70, 0.60, 0.77, 0.75, and 0.71, respectively. The longest 5 edges in the parts' spanning path are: (5, 6), (8, 9), (12, 13), (14, 15), and (17, 18), with length ($D_{j,j+1}$) of 0.86, 0.67, 0.67, 0.60, and 0.80, respectively. The partitioned matrix is presented in Fig. 7(a).

Step 2: In this step, the Hungarian algorithm gives the following part family-machine group assignments: 1-[$MG(4)$, $PF(1)$], 2-[$MG(5)$, $PF(6)$], 3-[$MG(6)$, $PF(5)$], 4-[$MG(1)$, $PF(3)$], 5-[$MG(3)$, $PF(2)$] and 6-[$MG(2)$, $PF(4)$]. The set $B = \{S(4, 1), S(5, 6), S(6, 5), S(1, 3), S(3, 2), S(2, 4)\}$.

Step 3: *Iteration* 1

(1) The computed values of $\Delta(I, J)$, for each $S(I, J) \in B$, are as follows:

$\Delta(1, 3) = -18$, $\Delta(2, 4) = +5$, $\Delta(3, 2) = -3$, $\Delta(4, 1) = -17$, $\Delta(5, 6) = -6$, and $\Delta(6, 5) = -8$. To illustrate, we show below how $\Delta(2, 4)$ was computed, $\Delta(2, 4) = \Delta H(2) + \Delta V(4) - NS(2, 4)$

$\Delta H(2) = \max\{NS(2, J^+(2)), NS(2, J^-(2))\}$
$J^+(2) = J | S(3, J) \in B = 2$; and $J^-(2) = J | S(1, J) \in B = 3$;
$\Rightarrow \Delta H(2) = \max\{NS(2, 2), NS(2, 3)\} = \max\{4, 1\} = 4$

$\Delta V(4) = \max\{NS(I^+(4), 4), NS(I^-(4), 4)\}$
$I^+(4) = I | S(I, 5) \in B = 6$; and $I^-(4) = I | S(I, 3) \in B = 1$;
$\Rightarrow \Delta V(4) = \max\{NS(6, 4), NS(1, 4)\} = \max\{3, 2\} = 3$
$\Rightarrow \Delta(2, 4) = 4 + 3 - 2 = 5$

(2) $\Delta(I^*, J^*) = \text{maximum}\{-17, -8, -8, -6, -3, +5\} = +5 = \Delta(2, 4) \Rightarrow I^* = 2$ and $J^* = 4$.

(3) Since $\Delta(2, 4) > 0$, then, we break block $S(2, 4)$ by eliminating the horizontal line H^* and the vertical line V^*. These two lines are determined as follows:

$$- NS(2, J^+(2)) > NS(2, J^-(2)) \Rightarrow H^* = H(I^*) = H(2).$$

$$- NS(I^+(4), 4) > NS(I^-(4), 4) \Rightarrow V^* = V(j^*) = V(4).$$

(4) Break the assignment that was made between $MG(2)$ and $PF(4)$. Combine $MG(2)$ with $MG(3)$, and combine $PF(4)$ with $PF(5)$.

(5) The partitioned matrix is updated in accordance with (4) and (5). The updated matrix is presented in Fig. 7(b). Set B is also updated as: $B = \{S(3, 1), S(2, 2), S(1, 3), S(5, 4), S(4, 5)\}$.

Iteration 2

The computed values of $\Delta(I, J)$ for $S(I, J) \in B$ are as follows: $\Delta(3, 1) = -15$, $\Delta(2, 2) = -5$, $\Delta(1, 3) = -9$, $\Delta(5, 4) = -11$, and $\Delta(4, 5) = -7$. None of these values is greater than zero, therefore, the procedure stops here, with $ng = 5$.

	P1	P2	P3	P4	P5	P6	P7	P8	P9
M1	3	1			3				
M2	1	3				1			3
M3			3				3	3	
M4		1	3	1				1	
M5	3			1	3			1	
M6		1				1			2
M7			3				3	1	
M8			3	1	1		1	3	
M9		3				2			3

Fig. 8. Non-binary matrix of Steudel and Ballakur.

4. PRACTICALITY OF THE PROPOSED APPROACH

Unlike most of the existing cell formation methodologies, the tabu search based approach proposed in this paper possesses several advantages which bring it closer to industrial reality. First, this method offers good quality solutions. Second, it can deal with cell formation problems with non-binary part-machine matrices. Third, it can handle large size problems efficiently.

4.1. Quality of solutions

In order to examine the effectiveness of the TSA, it has been applied to 10 data sets (problems) which were collected from the literature. The distance measure which was used by the TSA throughout the comparisons is the Jaccard distance measure [7], which was presented earlier in this paper. Table 2 presents the sources of the test problems, their solutions as reported in their sources, and their solutions as obtained by the TSA. The criteria, based on which the comparison is made, is the number of intercell moves. However, in order to ensure a fair comparison, the conditions (number of cells and maximum cell size) associated with the number of intercell moves, in a given solution, are reported because the number of intercell moves can be decreased or increased by manipulating these conditions (e.g., the number of intercell moves can be reduced by combining two or more cells). Obviously, when two clustering algorithms are to be compared with each other, a better clustering algorithm is supposed to give less number of intercell moves accompanied by the same or higher number of cells, and the same or less maximum cell size.

The results of the comparison indicate that the TSA was able to obtain better results than the original solutions 8 times out of 10, and it gave the same solutions as the original ones for the two remaining test problems. However, we believe that these results do not reflect superiority of the TSA as it exists, and if, larger size problems (e.g., 200 × 200) with somewhat ill structured matrices are used, then, superiority of the TSA will become more apparent.

4.2. Dealing with non-binary problems

There are many cell formation methods that cannot handle problems in which the entries of the part-machine matrix are non-binary (e.g., processing time, operations sequence, etc.). This property affects the practicality of such methods because real life applications involve many factors other than part routings. Our approach can be adjusted very easily to make it capable of handling non-binary matrices. This is achieved by using a distance measure that can accommodate binary data. Figure 7 presents the example problem presented by Steudel and Ballakur [25] where the entries of the part-machine matrix are the processing times of parts on machines. Figures 8 and 9 present the solutions obtained using the Steudel and Ballakur's dynamic programming procedure and the TSA, respectively. These two solutions are identical.

Deviation of the similarity measure proposed by Steudel and Ballakur from its upper bound, which is 2, was used by the TSA to measure distance between parts or machines. Hence, the distance measure used by the TSA is as follows:

$$D_{ij} = 2 - \left(\frac{Tc_i}{T_i} + \frac{Tc_j}{T_j}\right)$$

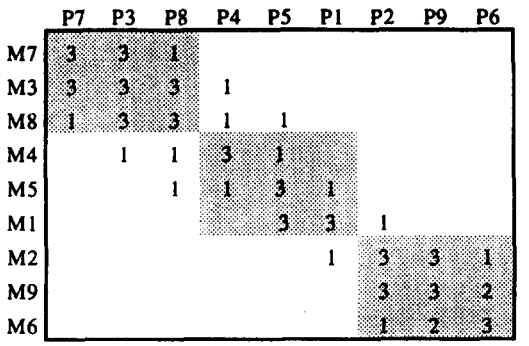

Fig. 9. Steudel and Ballakur's solution.

where:

D_{ij} = Distance between two machines (or two parts) i and j.
Tc_i = Total processing time on machine i required by all parts common to both machines i and j (or total processing time required by part i on all machines common to both parts i and j)
T_i = Total processing time on machine i (or total processing time required by part i).

4.3. Dealing with large size problems

Another factor that assures practicality of our tabu search approach is its ability to deal with large size problems efficiently. The proposed Tabu Search procedure was coded in Fortran and run on a SUN 4/490 computer system. Table 3 shows the CPU times required to solve five large size problems. To generate these problems, part-machine matrices with number of machines equal to 100 and number of parts ranging between 100 and 500, were deliberately designed to have perfect Block Diagonal Forms (BDF) with 25 equal size blocks (clusters). Then, the columns of each one of these matrices were randomly permuted to generate initial matrices.

The reported CPU times are the times required by Phase I of the proposed TS based method to re-obtain the original matrices (i.e., the matrices that have perfect BDF's). Note that the proposed method needs to be applied to the columns only sine the rows of the original matrices were not permuted. In the TS based algorithm (Phase I), Stopping rule (1) is used with k_T = number of parts. The results indicate that the TSA's CPU time requirements are very reasonable since a problem with 500 parts was solved optimally in less than 10 min.

5. CONCLUSIONS AND AVENUES FOR FUTURE RESEARCH

In this paper, a new machine-component clustering heuristic based on tabu search concept has been proposed. An investigation was conducted to select the tabu search parameter settings that suit the type of problem under consideration. The proposed heuristic was compared with some of the existing heuristics to cell formation. The results of the comparisons indicate that the proposed method is very efficient with respect to quality of its solutions. Although it requires somewhat larger CPU times as compared to simpler clustering algorithms, we believe that this could be justified by the quality of solutions it offers as well as its other features, which bring it closer to industrial reality, such as its ability to solve large size problems efficiently and to deal with non-binary part-machine matrices.

This study can be extended in the following two ways: (1) development of a new distance measure based on the specific objectives of CMS. For instance, one of the major objectives in adopting CMS is setup time reduction. This suggests the need for a distance measure that considers part tooling and setup time requirements. (2) The performance of the TS based heuristic, developed in this paper, can be enhanced by extending the study conducted in Section 3. Some of the factors that

Table 3. CPU time in seconds using the TSA

No. of parts	100	200	300	400	500
CPU time	19.7	58.6	162.1	319.5	503.3

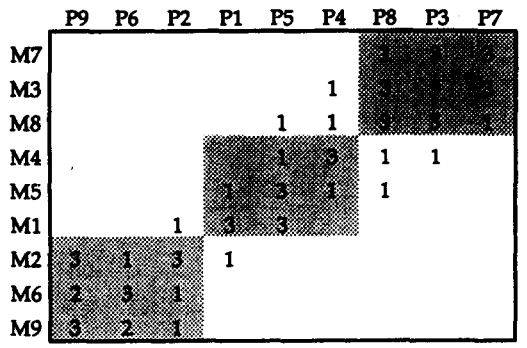

Fig. 10. Solution using the TSA.

need to be investigated, in extending this study, include diversification and intensification strategies, neighborhood decomposition strategies, and aspiration conditions.

REFERENCES

1. A. J. Vakharia. Methods of cell formation in group technology: a framework for evaluation. *J. Op Mgmt* **6**, 257–271 (1986).
2. H. Lee and A. D. Garcia. A network flow approach to solve clustering problems in group technology. *Int. J. Prod. Res.* **31**, 603–612 (1993).
3. J. L. Burbidge. A manual method for production flow analysis. *Prod. Engng* **56**, 34–38 (1977).
4. J. R. King. Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. *Int. J. Prod. Res.* **18**, 213–219 (1980).
5. H. M. Chan and D. A. Milner. Direct clustering algorithm for group formation in cellular manufacture. *J. Manufact. Syst.* **1**, 64–76 (1982).
6. W. T. McCormick, R. J. Schweitzer and T. W. White. Problem decomposition and data reorganization by clustering techniques. *Op. Res.* **20**, 993–1009 (1972).
7. J. McAuley. Machine grouping for efficient production. *Prod. Engng* **51**, 53–57 (1972).
8. H. Seifoddini and P. M. Wolfe. Application of the similarity coefficient method in group technology. *IIE Trans.* **18**, 271–277 (1986).
9. R. Rajagopalan and J. L. Batra. Design of cellular production systems—a graph theoretic approach. *Int. J. Prod. Res.* **13**, 567–579 (1975).
10. C. Y. Chen and S. A. Irani. Cluster first-sequence last heuristics for generating block diagonal forms for a machine-part matrix. *Int. J. Prod. Res.* **31**, 2623–2647 (1993).
11. D. G. Askin, S. H. Crosswell, J. H. Goldberg and A. J. Vakharia. A hamiltonian path approach to reordering the part-machine matrix for cellular manufacturing. *Int. J. Prod. Res.* **9**, 1081–1100 (1991).
12. W. H. Chen and B. Srivstava. Simulataed annealing procedures for forming machine cell in group technology. *Eur. J. Op. Res.* **71**, 100–111 (1994).
13. F. F. Boctor. A linear formulation of the machine-part cell formation problem. *Int. J. Prod. Res.* **29**, 343–356 (1991).
14. V. Venugopal and T. T. Narendran. Cell formation in manufacturing systems through simulated annealing: an experimental evaluation. *Eur. J. Op. Res.* **63**, 409–422 (1992).
15. V. Venugopal and T. T. Narendran. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Computers Ind. Engng* **22**, 469–480 (1992).
16. R. Logendran and P. Ramakrishna. Manufacturing cell formation in the presence of lot splitting and multiple units of the same machine. *Int. J. Prod. Res.* **31**, 657–693 (1993).
17. F. Glover. Tabu search, part I. *ORSA J. Computing* **1**, 190–206 (1989).
18. F. Glover. Tabu search, part II. *ORSA J. Computing* **2**, 4–32.
19. E. Taillard. Some efficient heuristic methods for flow shop sequencing problem. *Eur. J. Op. Res.* **47**, 65–74 (1990).
20. J. R. Slagle, C. L. Chang and R. C. Lee. A clustering and data-reorganizing algorithm. *IEEE Trans. Systems, Man Cybernetics* **5**, 126–128 (1975).
21. S. M. Shafer and D. F. Rogers. Similarity and distance measures for cellular manufacturing. Part II. An extension and comparison. *Int. J. Prod. Res.* **31**, 1315–1326 (1993).
22. F. Glover and H. J. Greenberg. New approaches for heuristic search: a bilateral linkage with artificial intelligence. *Eur. J. Op. Res.* **39**, 119–130 (1989).
23. F. Glover, E. Taillard and D. De werra. A user's guide to tabu search. *Ann. Ops Res.* **41**, 3–28 (1993).
24. N. Wu and R. Coppins. *Linear Programming and Extensions*. McGraw-Hill, New York (1981).
25. H. J. Steudel and A. Ballakur. A dynamic programming based heuristic for machine grouping in manufacturing cell formation. *Computers Engng* **12**, 215–222 (1987).
26. R. G. Askin and S. P. Subramanian. A cost based heuristic for group technology configuration. *Int. J. Prod. Res.* **25**, 101–113 (1987).
27. W. J. Boe and C. H. Cheng. A close neighbor algorithm for designing cellular manufacturing systems. *Int. J. Prod. Res.* **29**, 2097–2116 (1991).
28. M. P. Chandrasekharan and R. Rajagopalan. Groupability: an analysis of the properties of binary data matrices for group technology. *Int. J. Prod. Res.* **27**, 1035–1052 (1989).
29. M. T. Tabucanon and R. Ojha. ICERMA—a heuristic approach for intercell flow reduction in cellular manufacturing systems. *Mater. Flow* **4**, 189–197 (1987).