Baltzer Journals

Manufacturing Cell Formation by State-Space Search*

Subrata Ghosh[1, †] and Ambuj Mahanti[2,] and Rakesh Nagi[3, ‡] and Dana S. Nau[1 §]

[1] *University of Maryland, College Park, MD 20742*

[2] *Indian Institute of Management P. O. Box 16757, Calcutta 700027, India*

[3] *State University of New York at Buffalo, Buffalo, NY 14260*

This paper addresses the problem of grouping machines in order to design cellular manufacturing cells, with an objective to minimize inter-cell flow. This problem is related to one of the major aims of group technology (GT): to decompose the manufacturing system into manufacturing cells that are as independent as possible.

This problem is NP-hard. Thus, nonheuristic methods cannot address problems of typical industrial dimensions because they would require exorbitant amounts of computing time, while fast heuristic methods may suffer from poor solution quality.

We present a branch-and-bound state-space search algorithm that attempts to overcome both these deficiencies. One of the major strengths of this algorithm is its efficient branching and search strategy. In addition, the algorithm employs the fast Inter-Cell Traffic Minimization Method to provide good upper bounds, and computes lower bounds based on a relaxation of merging.

Keywords: Group Technology, Cellular Manufacturing, Machine Grouping, State-space search, Branch-and-bound algorithm.

*This work was supported in part by NSF Grants DDM-9201779, IRI-9306580, and NSFD EEC 94-02384 in the US, and the CMDS project (work order 019/7-148/CMDS-1039/90-91) in India. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

†Department of Computer Science, Email: subrata@cs.umd.edu.

‡Department of Industrial Engineering, Email: nagi@eng.buffalo.edu

§Department of Computer Science, Institute for Systems Research, and Institute for Advanced Computer Studies, Email: nau@cs.umd.edu.

1 Introduction

In designing shop layouts for manufacturing discrete parts, the conventional functional approach is to group resources in functionally similar areas. Due to changes in manufacturing philosophy, this approach is being replaced by the cellular approach, in which the production equipment is disaggregated into smaller subsystems called manufacturing cells. These cells are functionally autonomous, and contain most of the machines required to produce one or more families of parts with similar processing requirements. This concept of partitioning the manufacturing system into cells, and part types into part families based on the similarity of part manufacturing characteristics, is the manufacturing view of Group Technology (GT).

GT has had a profound impact on the realms of design, process planning, manufacturability evaluation and production planning. An established benefit from cellular manufacturing includes reduction in traffic of parts within the shop, which, in turn, reduces material handling efforts and costs (Kusiak and Heragu, 1987; Askin and Subramanian, 1987). In addition, since the machines within a cell are dedicated to a set of parts with similar processing requirements, set-ups can be shared to help reduce overall set-up time, reduce work-in-process inventories and queue times, flow-times and market response times. Furthermore, production planning and scheduling are aided by planning and scheduling for aggregates as opposed to individual parts. A survey of the benefits of cellular manufacturing can be found in Wemmerlöv and Hyder (1989), Ham, Hitomi and Yoshida (1985), Kusiak and Heragu (1987) and Willey and Dale (1977).

On the other hand, some disadvantages of cellular manufacturing include uneven distribution of workloads within cells and the disruptive effects of machine breakdowns (Askin and Subramanian, 1987), the cost of implementation, rate of change of product mix, inter-cellular operations and co-existence with non-cellular set-ups (Gallagher and Knight, 1986). A simulation study that compares the performance of GT and functional job-shops is presented in Flynn and Jacobs (1986).

1.1 Background and Motivation

Extensive research efforts have focused on the problem of aggregating machines to manufacturing cells. A review of these can be found in Kusiak and Chow, 1988, and Wemmerlöv and Hyder, 1986. The methods described in the literature can be broadly classified into four categories: (1) methods based on a part-machine incidence matrix that form cells and part families simultaneously (McCormick *et al.*, 1972; King, 1979, 1980; Chan and Miller, 1982; Chandrasekharan and Rajagopalan, 1987; Kusiak and Chow, 1987; Garcia and Proth, 1985, 1986; Khator and Irani, 1987; Askin *et al.*, 1991; Chen and Irani, 1993), (2) methods based on similarity coefficients and hierarchical clustering (McAuley, 1972; Carrie, 1973; Seifoddini and Wolfe, 1986; Leskowsky *et al.*, 1987), (3) methods based on material flow (Tabucanon and Ojha, 1987; Choobineh, 1988; Vakharia and Wemmerlöv, 1990; Harhalakis, Nagi and Proth, 1990; Harhalakis, Proth and Xie, 1990; Vohra *et al.*, 1990; Song and Hitomi, 1992; Okogbaa *et al.*, 1992), and (4) methods of the above categories that consider multiple criteria of practical significance, e.g., production costs, machining times, set-up times, queue times, utilization and capacities of machines, inter-cell flow and work-in-process (cost-based, Askin and Subramanian, 1987; tooling and processing times, Raja Gunasingh and Lashkar, 1989; functionally identical machines and workloads, Minis, Harhalakis and Jajodia, 1990; alternative routings and resource capacities, Nagi, Harhalakis and Proth, 1990, Logendran, Ramakrishna and Sriskandarajah, 1994; graph theoretic grouping and layout, Irani, Cohen and Cavalier, 1992; functionally identical machines, workloads and set-up families Harhalakis *et al.*, 1995; design constraints, Heragu and Gupta, 1994).

We focus our attention on the third category of material flow-based methods. Some of the reasons for this are: (i) material flow is perhaps the major consideration in the layout of manufacturing cells on the shop-floor, (ii) this method provides the basis for more comprehensive methods that include alternative process plans, functionally similar machines, and machine capacities. For instance, Nagi, Harhalakis and Proth (1990), presents a decomposition of a comprehensive problem into two-problems that are solved iteratively, one of which belongs to category (3), and Logendran, Ramakrishna and Sriskandarajah (1994) presents a division of a similar problem into two phases, the second phase employs category (3)-based method.

A further classification of the category (3) literature is based on the heuristic and non-heuristic nature of the solution algorithm. Among the heuristic methods, Tabucanon and Ojha (1987) have proposed a heuristic, ICRMA, for cell formation in order to minimize the inter-cell traffic of parts within the shop. Choobineh (1988) has suggested a method based on similarity coefficients, that considers operation sequences as well. Vakharia and Wemmerlöv (1990) have suggested a heuristic that uses duplicate machines to make the machine cells independent; machine loads are also considered. Harhalakis, Nagi and Proth (1990) present a two-step node aggregation Inter-Cell Traffic Minimization Method (ICTMM) for cell formation. Harhalakis, Proth and Xie (1990) have developed a heuristic based on simulated annealing, and compared it to ICTMM. Vohra *et al.* (1990) have suggested a network approach to this problem. Okogbaa *et al.* (1992) have proposed another heuristic for inter-cell flow reduction, and have performed comparison and simulation experiments on it. This heuristic facilitates formation of cells, and balances workload on identical machines. For an operation partition problem in assembly systems, which has a similar mathematical formulation, Ahmadi and Tang (1991) have proposed another simulated annealing algorithm that finds near-optimal solutions. The starting solution for this algorithm is chosen from two different Lagrangian heuristics which also provide lower bounds for this problem.[1]

In the non-heuristic methods, the work of Song and Hitomi (1992) is notable. They formulate the problem of cell formation to minimize the total number of parts produced in more than one cell as a quadratic assignment problem (QAP). It is solved using Lagrangian relaxation technique and the optimality conditions of quadratic programming, and a branch-and-bound algorithm is employed for optimal solutions.

As even the simplest formulation of the material flow-based cell formation is usually a combinatorial problem, nonheuristic methods cannot address problems of typical industrial dimensions. Greedy heuristic methods are faster, but they often return poor solution quality. The motivation of our work is to attempt to overcome the deficiencies of both these approaches. Of course, our work can benefit from the most recent, near-optimal heuristics (such as simulated annealing and genetic algorithms), and further enhance the solution quality or prove optimality.

1.2 Preliminary Discussion of the Problem

The problem of partitioning a set of machines having a specified traffic between each pair, to obtain manufacturing cells with no more than a desired number of machines in any cell, at minimum total inter-cell traffic, is NP-hard. Although this has not been proven formally in the literature, it can be done by a straightforward reduction of the clustering problem (Garey and Johnson, 1979) to the manufacturing cell optimization problem.

Since cell formation is part of designing a manufacturing system, and the system is expected to stay in place for a fairly long duration of time, longer solution times can be permitted in order to obtain better solutions. Even a small improvement in the suggested solution can accrue to significant saving in material handling costs over the life of the manufacturing system.

In this paper, we present a branch-and-bound state-space search algorithm that attempts to find good solutions in a reasonable amount of time. The algorithm starts with a good feasible solution which is considered as the upper bound, and tries to improve the solution through state-space search. The search is performed under time and memory constraints. In simple cases, i.e., problems of small size, the algorithm terminates successfully by finding the optimal solution—and in difficult cases, it can at least hope to improve upon the initial fea-

[1] Another significant difference between our work and that of Ahmadi and Tang is that they require that the number of nonempty components (in our terminology, the number of cells) be given as an input parameter, whereas our algorithm will consider all possible numbers of cells, in order to find which number is best.

sible solution. To provide the upper bound, the algorithm employs the Inter-Cell Traffic Minimization Method, a fast bottom-up aggregation heuristic presented in (Harhalakis *et al.*, 1990). The lower bound is derived based on a relaxation of merging. An efficient branching and search strategy constitutes a major strength of this algorithm.

The paper is organized as follows. The problem formulation is presented in section 2. The detailed branch-and-bound algorithm is presented in section 3. Section 4 is devoted to the numerical results obtained for the performance of the proposed algorithm. Finally, the conclusions are presented in section 5.

2 Problem Formulation

We consider a set $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ of m machines in a given manufacturing system. Each machine is recognized as unique, i.e., each work center is referred to by a different identification even if some work centers are functionally similar. We also consider a set $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of n part types to be manufactured. Each part type has associated with it a production routing which identifies the machines and sequence of operations to be used to manufacture it. Let $R_i = \{M_i^1, M_i^2, \ldots, M_i^{s_i}\}$ represent the routing of part p_i, where s_i is the number of operations, and $M_i^j \in \mathcal{M}$ is the machine on which the j'th operation is to be done, for $j = 1, 2, \ldots, s_i$. We ignore set-up and processing times, as we assume that the assignment of parts to machines has been performed *a priori* in a manner that respects machine capacity constraints; otherwise, the iterative approach suggested in (Nagi *et al.*, 1990) can be adopted. Let u_i be the production volume required for part type p_i in the chosen horizon, for $i = 1, 2, \ldots, n$. This information is the projected production requirement calculated on an average basis; either by long term production forecasts (in the case of new facilities), or by historical production information (in the case of existing facilities). We also introduce c_i as a cost factor for one unit of part type p_i, $i = 1, 2, \ldots, n$. The cost factor can be a combination of the following:

- *Material handling cost.* This depends on the size, shape, weight, or other attributes of a part. It can be based on the need of different types of material handling equipment, such as forklifts, cranes, and so forth.

- *Part cost.* The purpose of including this cost is to minimize the total monetary value of the work-in-process (WIP). Material movement is generally faster within a cell, rather than between different cells. Thus, a costly part critical to WIP, should be confined to a single cell.

For each $(p_i, M_j, M_k) \in \mathcal{P} \times \mathcal{M} \times \mathcal{M}$, we define q_{ijk} to be the number of times M_j follows M_k or M_k follows M_j in the routing R_i. Then for each pair $(M_j, M_k) \in$

$\mathcal{M} \times \mathcal{M}$, the *traffic* between machines M_j and M_k is defined as follows, where $t_{jk} = t_{kj}$ and $t_{jj} = 0$, for $i, j \in \{1, 2, \ldots, m\}$.

$$t_{jk} = \sum_{i=1}^{n} c_i u_i q_{ijk}. \tag{1}$$

We let N denote the maximal number of work-centers permissible in a cell (user defined). This number is derived from technical constraints and practical considerations: intra-cell transportation devices such as robots that cannot feed many machines, limitations on intra-cell buffers, ease of management and control, and so forth. A tight range for this constraint is usually decided upon by the company management, and depends on the specific manufacturing facility, the physical dimensions of the machines, operational flexibility of the machines, what types of intra/inter-cell transportation devices are used, the complexity of the parts to be manufactured, etc.

A *partition* of $\mathcal{M}$ is any set $\mathcal{C} = \{C_1, C_2, \ldots, C_w\}$ such that $C_i \cap C_j = \emptyset$ for $i, j \in \{1, 2, \ldots, w\}$ and $i \neq j$, and $\bigcup_{i=1}^{w} C_i = \mathcal{M}$. We let $\mathcal{U}$ be the set of all partitions $\mathcal{C}$ of $\mathcal{M}$ such that no member of $\mathcal{C}$ is larger than the maximum cell size N, i.e.,

$$\mathcal{U} = \{\mathcal{C} = \{C_1, C_2, \ldots, C_w\} \mid |C_i| \leq N, i = 1, 2, \ldots, w\}. \tag{2}$$

If C_i, C_j are two cells, then from equation (1) it follows that the traffic between C_i and C_j is

$$\mathcal{F}(C_i, C_j) = \sum_{M_r \in C_i, M_s \in C_j} t_{rs}.$$

Since $t_{jk} = t_{kj}$, it follows that $\mathcal{F}(C_i, C_j) = \mathcal{F}(C_j, C_i)$. The total inter-cell traffic for $\mathcal{C}$ is

$$\mathcal{F}(\mathcal{C}) = \sum_{i \neq j} \mathcal{F}(C_i, C_j). \tag{3}$$

The manufacturing cell optimization problem is the problem of finding the optimal partition in $\mathcal{U}$, i.e., the partition $\mathcal{C}^* \in \mathcal{U}$ such that

$$\mathcal{F}(\mathcal{C}^*) = \min_{\mathcal{C} \in \mathcal{U}} \mathcal{F}(\mathcal{C}). \tag{4}$$

3 Description of the Algorithm

State space search is commonly used for solving combinatorial optimization problems. Some well-known search algorithms are best-first branch-and-bound and depth-first branch-and-bound. However, as we explain below, these are not suitable for solving the manufacturing cell optimization problem.

$$
\begin{array}{l}
\qquad\quad S_0 \\
C_{0,1} = \{M_1\}, \text{ open} \\
C_{0,2} = \{M_2\}, \text{ open} \\
\quad\vdots \\
C_{0,m} = \{M_m\}, \text{ open}
\end{array}
$$

$$
\begin{array}{l}
\qquad\quad S_1 \\
C_{1,1} = \{M_1, M_2\}, \text{ open} \\
C_{1,2} = \{M_3\}, \text{ open} \\
\quad\vdots \\
C_{1,m-1} = \{M_m\}, \text{ open}
\end{array}
\qquad
\begin{array}{l}
\qquad\quad S_m \\
C_{m,1} = \{M_1\}, \text{ closed} \\
C_{m,2} = \{M_2\}, \text{ open} \\
\quad\vdots \\
C_{m,m} = \{M_m\}, \text{ open}
\end{array}
$$

$$
\begin{array}{l}
\qquad\quad S_2 \\
C_{2,1} = \{M_2\}, \text{ open} \\
C_{2,2} = \{M_1, M_3\}, \text{ open} \\
\quad\vdots \\
C_{2,m-1} = \{M_m\}, \text{ open}
\end{array}
\quad \cdots \quad
\begin{array}{l}
\qquad\quad S_{m-1} \\
C_{m-1,1} = \{M_2\}, \text{ open} \\
C_{m-1,2} = \{M_3\}, \text{ open} \\
\quad\vdots \\
C_{m-1,m-1} = \{M_1, M_m\}, \text{ open}
\end{array}
$$

Figure 1: The start state S_0 and its successors.

The main drawback of best-first branch-and-bound is the fact that it stores every node generated in its memory. As a result it runs out of memory very fast. Moreover, it expands every node with cost less than the cost of the optimal solution before terminating with a solution. Due to these two reasons and the fact that the search space generated by our problem is very large (Nagi *et al.*, 1990), best-first branch-and-bound can not solve any but very small problem instances.

Depth-first branch-and-bound does not suffer from any of the drawbacks of best-first branch-and-bound presented above. However, since depth-first branch-and-bound goes deep along one path, it may get stuck in a bad part of the search space in the available time and therefore return a very poor solution.

In this section, we describe a state-space search algorithm that is suitable for finding an optimal or near-optimal solution to the manufacturing cell optimization problem. This algorithm is basically an adaptation of the block-depth-first search (BDFS) algorithm of (Mahanti *et al.*, 1992). We describe how the search space is constructed, present a heuristic lower bound function for the states in the search space, and describe the algorithm for traversing the state space.

3.1 The Search Space

A *state* in the search space is a collection of machine cells (each containing a maximum of N machines), along with a matrix $[\mathcal{F}(C_i, C_j)]$ whose elements give the traffic between the cells.[2] In addition, each cell in the state is labeled *open* or *closed*; the purpose of this label is discussed below.

The *start state* S_0 is the state at which the search algorithm begins its search. As shown in Fig. 1, this state contains m cells $C_{0,1} = \{M_1\}, C_{0,2} = \{M_2\}, \ldots, C_{0,m} = \{M_m\}$, each consisting of a single machine. In the start state, every cell is marked as *open*, to indicate that it will be possible to merge it with other cells in states that are successors of S_0. A *goal state* is a state in which no further merging can take place, i.e., all cells are marked *closed*.

As shown in Fig. 1, S_0 has $m-1$ successor states that correspond to merging cell $C_{0,1}$ of S_0 with cells $C_{0,2}, C_{0,3}, \ldots, C_{0,m}$, respectively. S_0 has an additional m'th successor state, that corresponds to the decision not to merge $C_{0,1}$ with any other cell. The cells in this state are identical to the corresponding cells of S_0, except that the cell $C_{m,1}$ is marked *closed*.

More generally, suppose S is an arbitrary state, and let $C_1, \ldots, C_p$ be the cells in S. Then the successors of S are as follows:

- Let $C_i = \{M_{k_1}, \ldots, M_{k_p}\}$ be the first cell in S marked *open*. Let $C_j = \{M_{k'_1}, \ldots, M_{k'_{p'}}\}$ be any open cell in S such that $j > i$ and C_i's largest machine index k_p is less than C_j's smallest machine index k'_1.[3] For each such cell C_j, if $|C_i \cup C_j| \leq N$, then S has a successor S' that corresponds to merging C_i and C_j into a single cell $C_i \cup C_j$.

  S' contains one less cell than S, because in S', the cells C_i and C_j are replaced by a merged cell $C_i \cup C_j$. If the number of machines in $C_i \cup C_j$ is N, then $C_i \cup C_j$ is marked *closed*; otherwise it is marked *open*. For the new cell $C_i \cup C_j$, the traffic with other cells in S' is the sum of C_i's traffic and C_j's traffic in S; i.e.,

  $$\mathcal{F}(C_i \cup C_j, C_k) = \mathcal{F}(C_k, C_i \cup C_j) = \mathcal{F}(C_i, C_k) + \mathcal{F}(C_j, C_k).$$

  For all cells in S' other than the new cell $C_i \cup C_j$, the traffic is the same as it was in S. Thus, the total inter-cell traffic in S' is that of S, minus $\mathcal{F}(C_i, C_j)$ (the traffic between cell C_i and cell C_j).

[2] For efficient implementation of our algorithm, since the traffic matrix is symmetric (i.e. $\mathcal{F}(C_i, C_j) = \mathcal{F}(C_j, C_i)$ for all i, j), we represent each state by a triangular matrix of inter-cell traffics, a vector of the cardinalities of the cells, and some other bookkeeping information.

[3] The purpose of these conditions is to ensure that each possible state will appear exactly once in the search space.

- In addition to the above successors, S has a successor that corresponds to the decision not to merge C_i with any other cell. This state is identical to S except that C_i is marked *closed*.

It is easy to see that the search space defined above is a tree with maximum depth m. It can also be shown that the search space is complete, i.e. every feasible partition is one of the goal states in the search space defined.

The objective of our algorithm is to find a goal state with minimum inter-cell traffic. It is not always feasible to achieve this objective, because the problem is NP-hard and therefore requires exponential amount of time in the worst case. A more realistic objective is to find a good solution (not necessarily optimal) in the available time. Our search algorithm attempts to achieve this objective.

3.2 Heuristic

In this section, we present a heuristic lower bound function for the manufacturing cell optimization problem. This function is used for two purposes by our algorithm: for ordering states (as explained later), and for pruning states from the search space.

The lower bound function we present is based on the relaxation technique, which is a well known method for designing lower bound functions (Pearl, 1984). The basic idea is, given any state, to allow maximum possible merging for each cell and then take the remaining inter-cell traffic as the lower bound value. We state this more formally below.

Let S be any arbitrary state, and $C_1, \ldots, C_P$ be its cells. Then the traffic between cell C_i and cell C_j is $\mathcal{F}(C_i, C_j) = \mathcal{F}(C_j, C_i)$, so the total inter-cell traffic among the cells is $\sum_{i=1}^{P-1} \sum_{j=i+1}^{P} \mathcal{F}(C_i, C_j)$. If we can compute an upper bound $R(C_i)$ on the maximum possible amount of reduction in traffic that can be achieved by merging the cell C_i with other cells, then the quantity $\frac{1}{2}\sum_{i=1}^{P} R(C_i)$ will be an upper bound on the maximum possible reduction in traffic that can be achieved by merging cells in the state S. (The reason for multiplying the sum by $1/2$ is that in summing up these upper bounds, the traffic between each pair of cells C_i, C_j is counted twice: once in $R(C_i)$ and once in $R(C_j)$.) Therefore, the following is a lower bound on the cost of any solution achievable from S:

$$LB(S) = \sum_{i=1}^{P-1} \sum_{j=i+1}^{P} \mathcal{F}(C_i, C_j) - \frac{1}{2}\sum_{i=1}^{P} R(C_i).$$

The significance of LB being a lower bound is that whenever a state S in the search space is found with $LB(S)$ greater or equal to the cost of the currently known solution, it can be pruned without the possibility of losing the optimal solution.

Table 1
Traffic Matrix

	1	2	3	4	5	6
1	0	4	6	0	0	8
2	4	0	4	8	10	0
3	6	4	0	2	8	0
4	0	8	2	0	12	6
5	0	10	8	12	0	14
6	8	0	0	6	14	0

To compute the value $R(C_i)$, we use the procedure shown below. An intuitive explanation of this procedure is as follows. Since the cell cardinalities can be at most N, C_i can be merged with at most $N - |C_i|$ machines. By considering the cells in decreasing order of the ratio of cell traffic to cell cardinality, the procedure basically finds those cells which would reduce traffic the most if they were merged with C_i, and lets $R(C_i)$ be total amount of traffic reduction which could be obtained in this way.[4] The reason why $R(C_i)$ is an upper bound (rather than an exact value) is because it will not always be possible to merge those machines into a single cell.

PROCEDURE $R(C_i)$

```
r := 0, c := |C_i|            /* Initialize */
loop
    if there is no mergeable cell^5 C_j such that F(C_i, C_j) > 0, then return r
    else let C_j be the one that maximizes F(C_i, C_j)/|C_j|
    if c + |C_j| ≤ N then
        r := r + F(C_i, C_j)      /* accumulated reduction in traffic */
        c := c + |C_j|            /* accumulated number of machines */
        eliminate cell C_j from future consideration
    else
        r := r + F(C_i, C_j) * (N − c)/|C_j|; /* pro-rated traffic reduction */
        return r
repeat
```

3.3 Complexity of Heuristic Computation

In the procedure for computing $R(C_i)$, the loop is executed at most $N - |C_i|$ times. During one iteration of the loop, choosing C_j takes time $O(P)$, where P is the total

[4] The same approach has been used to find optimal solutions to the Knapsack problem, and upper bounds for the 0/1 Knapsack problem (Horowitz and Sahani, 1978).

[5] Two cells C_i and C_j are mergeable if $i \neq j$, both C_i and C_j are marked *open* and $|C_i| + |C_j| \leq N$.

number of cells. All other steps take time $O(1)$. Therefore, the procedure takes time $O(P(N - |C_i|))$. Finally, since the procedure is executed P times, the total time required to compute the second component of LB is $O(P^2(N - |C_i|)) = O(P^2N)$. The time required to compute the first component is clearly $O(P^2)$. Therefore, since $P \leq m$ where m is the total number of machines, the total time required to compute $LB(S)$ is $O(P^2) + O(P^2N) = O(P^2N) = O(m^2N)$.

3.4 An Example of Heuristic Computation

Consider a state S in which the number of cells is $P = 6$, and the maximum cell size is $N = 3$. For simplicity, assume all cells in S are marked *open*. Let the cell cardinalities be

$$|C_1| = 1,\ |C_2| = 2,\ |C_3| = 2,\ |C_4| = 1,\ |C_5| = 1,\ |C_6| = 2;$$

and let the traffic matrix $[\mathcal{F}(C_i, C_j)]$ be as shown in Table 1.
Then the procedure defined above will produce the following values:

$$R(C_1) = 8,\ R(C_2) = 8,\ R(C_3) = 8,\ R(C_4) = 16,\ R(C_5) = 19, R(C_6) = 14.$$

Thus,

$$LB(S) = \sum_{i=1}^{5} \sum_{j=i+1}^{6} \mathcal{F}(C_i, C_j) - \frac{1}{2} \sum_{i=1}^{6} R(C_i) = 82 - 36.5 = 45.5$$

3.5 Search Algorithm

Block-depth-first search (BDFS) is a search algorithm that is based on a novel combination of staged search and depth-first search. As a result, it has good features of both best-first and depth-first branch-and-bound and at the same time avoids the bad features of both. In this paper we describe an adaptation of BDFS for use in solving the cell-optimization problem. We use the following notation:

r: node generation rate (nodes per second)
MEM: amount of memory available (number of nodes)
T: amount of time available (in seconds)

BRANCH-AND-BOUND ALGORITHM

INPUT: Problem instance $(m, N, t_{ij}, 1 \leq i < j \leq m)$, MEM, r and T

Our algorithm has two main phases: (1) the forward phase, and (2) the backtracking phase. In the forward phase, it finds a good solution depending on the available time. In the backtracking phase, it finds successive improvements on the solution found in the forward phase, until the available time is completely exhausted.

FORWARD PHASE

In this phase, the algorithm explores the search tree, level by level. All nodes generated are stored in a linear list L of size MEM. The root node is assigned level 0 and stored. After this the algorithm runs iteratively, working on one level at each iteration. At iteration (level) i, it first estimates the number of nodes of level $i+1$ to be generated (using the available memory MEM, available time T, node generation rate r, and an estimate of the number of levels of the search tree yet to be generated) and then generates at most those many nodes by expanding nodes of level i. The expansion of a node means generation of all of its successors. The nodes of level i are expanded in increasing order of their lower bound values. An initial upper bound of the solution cost is obtained by running the ICTMM algorithm of (Harhalakis *et al.*, 1990). Any generated node with a lower bound value greater than the upper bound is discarded (pruned). The forward phase continues until a solution is found or the number of stored nodes in an iteration becomes zero. The number of nodes stored can become zero due to pruning. The details of the forward phase are given below:

Step 0. [Initialization]

create the start_node and store it in L.
current_level := 0.
solution_cost := ICTMM(). /* Initial upper bound */
rem_levels := m. /* Number of machines. */
rem_memory := MEM-1.
rem_time := $r * T - 1$.

Step 1. [Branching]

nodes_to_be_generated := $\min(\frac{\text{rem\_mem}}{\text{rem\_levels}}, \frac{\text{rem\_time}}{\text{rem\_levels}})$.
nodes_generated := 0.
while nodes generated < nodes_to_be_generated do
begin
  select the first unexpanded node n. if there is no such node then goto step 2.
  If n is a goal node, update solution_cost to total inter-cell traffic in n and goto step 4.
  expand n generating and storing all successors of n in L.
  nodes_generated := nodes_generated + number of successors of n.
  Mark n expanded.
end

Step 2. [Bounding and Ordering]

Compute the lower bound LB of every newly generated node in step 1.
Discard a node from L if its LB value is greater or equal to upper_bound.

If no new node is stored in L then goto step 4.
Sort the newly stored nodes in L in increasing order of their LB values.

Step 3. [Update]
rem_mem := rem_mem $-$ number of newly stored nodes in step 2.
rem_time := rem_time $-$ number of newly generated nodes in step 1.
rem_levels := rem_levels -1.
current_level := current_level $+1$.
goto step 1.

Step 4. [Termination]
Start backtracking phase

BACKTRACKING PHASE

After the completion of the forward phase, there may be some time left because the estimation of rem_levels and node generation rate may not be exact. That time is used in this phase to improve the solution found in the forward phase. This phase basically executes depth-first branch-and-bound (DFBB) starting at each unexpanded node. DFBB is performed in the reverse order i.e. from the last level generated down to level 1.

Step 1. [DFBB]
for each level from current_level down to 1 do
begin

select the first unexpanded node n.
perform a DFBB search starting at n, cutting off each generated path when its cost exceeds solution_cost. If a solution is found, then update solution_cost to the total inter-cell traffic of the solution state.
Mark n expanded.
rem_time := rem_time - nodes generated by DFBB(n).
if rem_time ≤ 0 then goto step 2.

end

Step 2. [Termination]
output the solution found.

4 Numerical Experiments

4.1 Problem generation

In order to perform numerical tests of our approach, problems of various degrees of complexity were constructed. In this section, we detail the generation of random

Table 2
Parameter values used in the experiments.

m	n	b	$poll$
15	30,45	3	10,20,30
20	40,60	3,4	10,20,30
30	60,90	4,5	10,20,30
50	100,150	6,8	10,20,30
100	200,300	10,20	10,20,30

problem data used in these tests. The predominant factors influencing problem complexity are (i) the number of machines, (ii) the maximal cell size, and (iii) intensity of traffic between pairs of machines, which is impacted by the number of parts and their routings. Thus, to incorporate the influence of these factors in the problem data, we used the following generation scheme.

1. Select the number of machines, m.

2. Select the number of parts, n.

3. Select the number of expected number of cells (or blocks), b; this is chosen such that the average number of machines per cell is between 5 to 10. b also corresponds to the number of part families.

4. Assign machines to cells at random, with at least $\lceil m/(2 \times b) \rceil$ machines per cell.

5. Assign parts to part families at random, with at least $\lceil n/(2 \times b) \rceil$ parts per family.

   At this stage, conceptually, a binary (0-1) matrix M is available. The rows correspond to parts, and the columns correspond to machines. An element m_{ij} of M is 1 if the part i and machine j belong to the same block, and 0 otherwise. Thus, a block diagonal matrix, D, can be constructed by permuting the rows and columns of M.[6]

6. Select an inside pollution factor, *inpoll*. This factor weakens the traffic between machines that belong to the same cell. At random, convert *inpoll* percentage of entries in the diagonal blocks of D from 1's to 0's.

[6] Note that we do not need to physically create either of these matrices. We simply point out that the information to construct these is available at this time, in order to make our approach more understandable to readers who are familiar with the incidence-matrix based methods for GT presented in Section 1.

Table 3
Results for $m = 15$ and $T = 300$

			BDFS				ICTMM
n	N	*poll*	Optimal solutions	Nodes generated	Nodes expanded	Solution cost	Solution cost
		10	5	20293	6123	47	48
40	7	20	5	38905	11767	61	63
		30	5	80680	28138	78	84
		10	5	8019	2418	68	71
60	7	20	5	63109	23973	99	110
		30	5	85326	30355	123	129

7. Select an outside pollution factor, *outpoll*. This factor introduces traffic between machines that do not belong to the same cell. At random, convert *outpoll* percentage of the entries outside the diagonal blocks of D from 0's to 1's.

8. At this point, we have the set of machines, S_i, that each part type p_i needs to visit in order to be manufactured. From this set, randomly generate the sequence in which each part type p_i must visit its machines.[7] This is the production routing R_i, as defined in section 2.

9. Set the cost factor c_i and production volume u_i of each part type p_i to unity.

The actual values of the parameters used in our experiments are shown in Table 2. In our experiments, we used *inpoll* equal to *outpoll*, hereby referred to as *poll*. For each set of parameters, five problem instances were generated, resulting in a total of 270 problem instances.

4.2 Results

To study the performance of the search algorithm described earlier, we implemented it, an ran it on all 270 of the problem instances described earlier. The value of the cell size limit (N) was set to $\lfloor m/(b-1) \rfloor$. The algorithm was run on a Sun 4 workstation, with MEM = 100,000 nodes, T = 300, 600, 900, 1200 and 7200 seconds, and m = 15, 20, 30, 50 and 100 machines, respectively.

The results of our experiments are listed in Tables 3–7. Each line of each table presents the results of running BDFS on five problem instances. The data include (i) the number of guaranteed-optimal solutions found, (ii) the average

[7] While generating the sequence, it would be possible to choose a machine for more than one operation, indicating *non-consecutive* operations on the same machine (Harhalakis *et al.*, 1990). However, we avoid this for the sake of simplicity.

number of nodes generated, (iii) the average number of nodes expanded, and (iv) the average solution cost found. For comparison, we have also listed the average solution cost found by ICTMM. It is noted that the CPU time for ICTMM was always less than 10 seconds. Items (ii) and (iii) are related to the tightness of the lower bound. Other informal computational experience with some other bounds that were attempted and the zero bound indicated the effectiveness our lower bound heuristic. It is recognized that experience is empirical and not based on worst/mean case analyses. However, the simplicity in ease of implementation as well as low order of complexity make it attractive for the cell formation problem. ¿From our analysis of the data, we note the following:

1. As can be seen from Table 3, for small problem instances BDFS returned provably optimal solutions. The solution returned by BDFS is provably optimal for all problem instances in which the algorithm terminates before its allotted search time T.

2. As expected, for large problems ($M \geq 30$) none of the solutions found by BDFS are guaranteed to be optimal. However, BDFS clearly improves the initial solution found by ICTMM.

3. As the pollution factor increases, the improvement of BDFS over ICTMM also increases. This is because the pollution factor determines the difficulty of the problem instance. The problem instances with low pollution factor are easy in general, because there is very little cross traffic between cells. However, as the pollution factor increases, the problems become increasingly difficult. Since ICTMM is a greedy algorithm, if fails to find good solutions for hard problem instances, and therefore the improvement of BDFS over ICTMM increases with an increasing pollution factor. For the same reason, for $m = 20$, BDFS finds optimal solutions for 15 out of 20 problem instances with pollution factor 10, as shown in Table 4. However, if the pollution factor is 30, none of the solutions are guaranteed optimal.

4. Finally, the relative improvement of BDFS over ICTMM (i.e., the percentage by which BDFS's solution quality is better than ICTMM's) does not remain the same with the problem size m. For example, the percentage improvement in solution quality for $m = 100$ is not as dramatic as for $m = 15$. This is primarily due to two factors: (1) the size of the traffic matrix and (2) the size of the search space. The first factor increases quadratically with problem size m, and the second factor increases exponentially with m. Therefore, the increase in the allowed search time T from 300 seconds to 7200 seconds does not compensate well for the increase in these two factors. We could not increase T any further because of time constraints.

Table 4
Results for $m = 20$ and $T = 600$

			BDFS				ICTMM
n	N	*poll*	Optimal solutions	Nodes generated	Nodes expanded	Solution cost	Solution cost
		10	5	33706	9311	66	66
40	10	20	0	179892	64591	118	132
		30	0	184074	68623	130	142
		10	5	28981	8974	103	110
60	10	20	1	151668	51653	174	191
		30	0	165634	61098	199	228
		10	4	99416	20411	98	100
40	6	20	0	284353	69877	140	145
		30	0	306171	79899	175	179
		10	1	142442	27667	149	151
60	6	20	0	267288	69906	217	225
		30	0	322659	87776	273	282

Table 5
Results for $m = 30$ and $T = 900$

			BDFS				ICTMM
n	N	*poll*	Optimal solutions	Nodes generated	Nodes expanded	Solution cost	Solution cost
		10	0	134669	40263	197	200
60	10	20	0	142127	42964	321	347
		30	0	151938	45750	391	418
		10	0	107024	32796	302	311
90	10	20	0	126375	37410	479	513
		30	0	137133	42528	613	648
		10	0	175254	43135	224	232
60	7	20	0	234083	58663	343	353
		30	0	295579	83450	433	450
		10	0	170769	41519	345	355
90	7	20	0	263312	76660	552	568
		30	0	301884	96141	679	702

Table 6
Results for $m = 50$ and $T = 1200$

			BDFS				ICTMM
n	N	*poll*	Optimal solutions	Nodes generated	Nodes expanded	Solution cost	Solution cost
		10	0	78410	19746	613	615
100	10	20	0	168820	61126	1036	1054
		30	0	166188	60243	1328	1352
		10	0	146533	30699	618	623
150	10	20	0	191452	49693	1021	1028
		30	0	226533	68826	1343	1350
		10	0	77048	21191	963	965
100	7	20	0	158419	61368	1595	1623
		30	0	162045	63030	1999	2036
		10	0	105168	19439	946	955
150	7	20	0	172360	44636	1542	1571
		30	0	173801	45801	2022	2068

Table 7
Results for $m = 100$ and $T = 7200$

			BDFS				ICTMM
n	N	*poll*	Optimal solutions	Nodes generated	Nodes expanded	Solution cost	Solution cost
		10	0	146514	26496	2583	2583
200	11	20	0	248543	71241	4284	4288
		30	0	271394	80699	5679	5704
		10	0	159705	40899	3920	3933
300	11	20	0	305343	94387	6532	6549
		30	0	345297	117084	8625	8792
		10	0	326279	28524	2257	2257
200	5	20	0	565396	88638	3993	4005
		30	0	640622	117751	5701	5709
		10	0	397616	45146	3480	3480
300	5	20	0	609223	105975	6050	6062
		30	0	556177	97010	8614	8638

5 Conclusion

Manufacturing cell formation for group technology is an important and well studied problem. In this paper, we have presented an efficient heuristic search algorithm for manufacturing cell optimization based on material flow. To guide this algorithm, we have developed a new lower bound function, based on a relaxation of the problem of merging machines into cells.

The heuristic search algorithm also employs the ICTMM heuristic (Harhalakis *et al.*, 1990) as an upper bound function. This improves the efficiency of the algorithm by allowing it to do pruning before the first solution is found—but the algorithm could equally be used without this upper bound function. On the other hand, more recent, high performance, non-deterministic search algorithms can also be employed to possibly improve on ICTMM's upper bound.

We have also presented the results of an extensive empirical study of our algorithm. The results indicate that our scheme is able to improve upon the existing ICTMM solution, and also finds provably optimal solutions for problems of small sizes, such as 15 to 20 machines. We envision that this improvement can impact significantly on the operations of the cellular manufacturing system over its entire life.

The primary benefits of our algorithm are that it runs with constrained time and storage resources. Given more execution time, it improves the solution quality or attempts to prove that its last solution found was indeed optimal. Since the manufacturing cell formation problem is a design level problem, computational efficiency is not as critical as the solution quality. Even small improvements at the design stage will result in significant savings in material handling costs over the life of the shop layout. Thus, the ability to prove optimality or potentially improve over other good heuristic solutions is the major contribution of this work.

Future work can be aimed at integrating this methodology with other realistic concerns of manufacturing cell formation, including alternative process plans, functionally similar machines, and workload distribution under resource capacities.

We hope that our results will encourage others to consider using heuristic search techniques to develop practical solutions to other industrial problems.

References

[1] R. H. Ahmadi, C. S. Tang, An operation partitioning problem for automated assembly system design, Operations Research, 39 (1991) 824-835.

[2] R. Askin, S. B. Subramanian, A cost-based heuristic for group technology configuration, International Journal of Production, 25 (1987) 101-113.

[3] R. Askin, S. H. Cresswell, J. B. Goldberg, A. Vakharia, A hamiltonian path approach

to reordering the part-machine matrix for cellular manufacturing, International Journal of Production Research, 29 (1991) 1081-1100.
[4] A. S. Carrie, Numerical taxonomy applied to group technology and plant layout, International Journal of Production Research, 11 (1973) 399-416.
[5] H. M. Chan, D. A. Milner, Direct clustering algorithm for group formation in cellular manufacturing,Journal of Manufacturing Systems, 1 (1982) 65-74.
[6] M. P. Chandrasekharan, R. Rajagopalan, ZODIAC - an algorithm for concurrent formation of part-families and machine-cells, International Journal of Production Research, 25 (1987) 835-850.
[7] C. Y. Chen, S. Irani, Cluster first-sequence last heuristics for generating block diagonal forms for a machine-part matrix, International Journal of Production Research, 31 (1993) 2623-2647.
[8] F. Choobineh, A framework for the design of cellular manufacturing systems, International Journal of Production Research, 26 (1988) 1161-1172.
[9] B. B. Flynn, F. R. Jacobs, A simulation comparison of group technology with traditional job shop manufacturing, International Journal of Production Research, 24 (1986) 1171-1192.
[10] C. C. Gallagher, W. A. Knight, *Group Technology Production Methods in Manufacture,* Ellis Horwood Limited, 1986.
[11] H. Garcia, J. M. Proth, Group technology in production management: the short horizon planning level, Applied Stochastic Models and Data Analysis, 1 (1985) 25-34.
[12] H. Garcia, J. M. Proth, A new cross-decomposition algorithm: the gpm. comparison with the bond energy method, Control and Cybernetics, 15 (1986) 155-164.
[13] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H. Freeman and Co., New York, 1979.
[14] I. Ham, K. Hitomi, T. Yoshida, *Group Technology - Applications to Production Management,* Kluwer-Nijhoff Publishing, 1985.
[15] G. Harhalakis, R. Nagi, J. M. Proth, An efficient heuristic in manufacturing cell formation for group technology applications, International Journal of Production Research, 28 (1990) 185-198.
[16] G. Harhalakis, J. M. Proth, X. L. Xie, Manufacturing cell design using simulated annealing, Journal of Intelligent Manufacturing, 1 (1990) 185-191.
[17] G. Harhalakis, T. Lu, I. Minis, R. Nagi, A practical method for hybrid-type production facilities," accepted in International Journal of Production Research, 1995.
[18] E. Horowitz, S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, MD, 1978.
[19] S. S. Heragu, Y. P. Gupta, A heuristic for designing cellular manufacturing facilities, International Journal of Production Research, 32 (1994) 125-140.
[20] S. A. Irani, P. H. Cohen, T. M. Cavalier, Design of cellular manufacturing systems, ASME Transactions of Engineering for Industry, 114 (1992) 352-361.
[21] S. K. Khator, S. A. Irani, Cell-formation in group technology: a new approach, Computers and Industrial Engineering, 12 (1987) 561-569.
[22] J. R. King, Machine-component grouping formation in group technology, OMEGA: The International Journal of Management Science, 8 (1979) 193-199.
[23] J. R. King, Machine-component grouping in production flow: an approach using rank order clustering, International Journal of Production Research, 18 (1980) 213-232.
[24] A. Kusiak, W. S. Chow, Efficient solving of the group technology problem, Journal of Manufacturing Systems, 6 (1987) 117-124.
[25] A. Kusiak, S. S. Heragu, Group technology, Computers in Industry, 9 (1987) 83-91.
[26] A. Kusiak, W. S. Chow, Decomposition of manufacturing systems, IEEE Journal of Robotics and Automation, 4 (1988) 457-471.
[27] Z. Leskowsky, L. Logan, A. Vannelli, Group technology decision aids in an expert system for

plant layout, *Modern Production Management Systems,* Elsevier Science Publisher, 1987.
[28] R. Logendran, P. Ramakrishna, C. Sriskandarajah, Tabu search-based heuristics for cellular manufacturing systems in the presence of alternative process plans, International Journal of Production Research, 32 (1994) 273-297.
[29] A. Mahanti, S. Ghosh, A. K. Pal, A high performance limited-memory admissible and real time search algorithm for networks, technical report, CS-TR-92-34. Dept. of Computer Science, University of Maryland, 1992.
[30] J. McAuley, Machine grouping for efficient production, The Production Engineer, 51 (1972) 53-57.
[31] W. T. McCormick, P. J. Schweitzer, T. E. White, Problem decomposition and data reorganization by a clustering technique, Operations Research, 20 (1972) 993-1009.
[32] I. Minis, G. Harhalakis, S. Jajodia, Manufacturing cell formation with multiple, functionally identical machines, Manufacturing review, 3 (1990) 252-261.
[33] R. Nagi, G. Harhalakis, J. M. Proth, Multiple routings and capacity considerations in group technology applications, International Journal of Production Research, 28 (1990) 2243-2257.
[34] O. G. Okogbaa, M. T. Chen, C. Changchit, R. L. Shell,Manufacturing system cell formation and evaluation using a new inter-cell flow reduction heuristic, International Journal of Production Research, 30 (1992) 1101-1118.
[35] J. Pearl, *Heuristics, Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1984.
[36] K. R Gunasingh, R. S. Lashkar, Machine grouping problem in cellular manufacturing systems – an integer programming approach, International Journal of Production Research, 27 (1989) 1465-1473.
[37] H. Seifoddini, P. M. Wolfe, Application of the similarity coefficient method in group technology, IIE Transactions, 18 (1986) 271-277.
[38] S. Song, K. Hitomi, GT cell formation for minimizing the intercell parts flow, International Journal of Production Research, 30 (1992) 2737-2753.
[39] M. T. Tabucanon, R. Ojha, ICRMA - a heuristic approach for intercell flow reduction in cellular manufacturing systems, Material flow, 4 (1987) 189-197.
[40] A. J. Vakharia, U. Wemmerlöv, Designing a cellular manufacturing system: a material flow approach based on operation sequences, IIE Transactions, 22 (1990) 84-97.
[41] T. Vohra, D. S. Chen, J. C. Chang, H. C. Chen, A network approach to cell formation in cellular manufacturing, International Journal of Production Research, 28 (1990) 2075-2084.
[42] U. Wemmerlöv, N. L. Hyder, Procedures for the part family/machine group identification problem in cellular manufacturing, Journal of Operations Management, 6 (1986) 125-148.
[43] U. Wemmerlöv, N. L. Hyder, Cellular manufacturing in the U.S. industry: a survey of user, International Journal of Production Research, 27 (1989) 1511-1530.
[44] P. C. T. Willey, B. G. Dale, Manufacturing characteristics and management performance of companies under group technology, *Proceedings of the 18th International Machine Tool Design and Research Conference*, 1977, p. 777.