Moving Beyond the Parts Incidence Matrix: Alternative Routings and Operations for the Cell Formation Problem

Jeffrey A. Joines (jjoine@eos.ncsu.edu)
Russell E. King (king@eos.ncsu.edu)
C. Thomas Culbreth(culbreth@eos.ncsu.edu)

Furniture Manufacturing and Management Center
Department of Industrial Engineering
North Carolina State University
Raleigh, NC 27695-7906
Ph: (919) 515-5186 Fax: (919) 515-1543

This research was sponsored, in part,
by Office of Naval Research Grant N00014-90-J-1045 and
National Science Foundation Grant DDM-9215432.

Abstract

A genetic algorithm (GA) utilizing an integer-based representation is presented to assist in the design of cellular manufacturing (CM) systems. It allows for a variety of evaluation functions and the selective incorporation/exclusion of design constraints during cell formation. In this paper, the basic GA model is extended to consider alternative operations and machine redundancy as well as completely fixed alternative routes. The approach is demonstrated on data from the literature and is shown to be an effective cell design tool. An industrial case study is described where the clustering problem was solved using the genetic search approach. The resulting cell design has been implemented.

1 Introduction

In cellular manufacturing, the manufacturing system is decomposed into several manageable subsystems, or groups, by aggregating similar parts into part families and dissimilar machines into cells [17]. The ideal cell (1) is independent, i.e., part family(s) are completely produced within the cell; (2) has balanced setups; and (3) requires minimal backtracking. The result is simplified scheduling, control, and implementation of automation. Cellular manufacturing provides the benefits of a mass production system for a discrete part, batch production system [2, 3] including reduced setup times, work in progress, throughput time, and material handling as well as encouraging improved product quality [1, 8, 23, 29]. Employee/worker benefits include worker flexibility, importance of social group, reduced frustration, and improved status and job security[5].

A common objective for the designers of cellular manufacturing systems is to create a set of autonomous manufacturing units that eliminate the intercell movement of parts. In the area of manufacturing cell formation, techniques in the literature for partitioning the part/machine incidence matrix into machine cells and associated part families are varied and extensive [13, 25]. Techniques based on process routings can be categorized according to the type of algorithm employed to cluster the data, e.g., array-based clustering, hierarchical and non-hierarchical cluster analysis, mathematical programming methods, graph theoretic approaches, artificial intelligence techniques, and other heuristics.

Mathematical programming approaches to the cell formation problem are nonlinear or linear integer programming problems. These formulations suffer from three critical limitations. First, because of the resulting nonlinear form of the objective function, most approaches do not concurrently group machines into cells and parts into families. Second, the number of machine cells must be specified a priori, affecting the grouping process and potentially obscuring natural cell formations in the data. Third, since the variables are constrained to integer values, most of these models are computationally intractable for realistically sized problems [19].

Joines et al. [10] developed a stochastic solution technique using a genetic algorithm (GA) to solve integer programming formulations of the cell design problem. GAs are an effective and flexible optimization tools that can solve a large class of problems by exploring all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations. Also, unlike many other optimization techniques, they do not make strong assumptions about the form of the objective function.

The GA approach to cell design offers several advantages over existing techniques. It simultaneously groups the parts and machines into families and cells and assigns them to each other without visual analysis of the solution. The model also offers the ability to constrain the number of permissible cells or part families, selectively. Most clustering algorithms cannot identify all naturally occurring clusters and find solutions with a constrained number of clusters. The cell designer, at least initially, might specify an unconstrained problem to identify the naturally occurring groups of parts and/or machines. Afterwards, practical limits on the number of cells arising from availability of floor space, maximum work team sizes, or excessive machine redundancy requirements can be imposed. The GA approach [10] was tested on data from the literature and shown to outperform other methods operating on just the part/machine incidence matrix. The algorithm has been improved in terms of computational efficiency and quality of solution through the incorporation of a local improvement heuristic as either an evaluation function or a genetic operator [12].

Most algorithms in the literature utilize only the part/machine incidence matrix which contains only a pre-specified, single fixed routing for each part. Relatively very few researchers have addressed the issue of utilizing alternative operations, alternative routings, or machine redundancy in determining cells/families. These single fixed routings are often optimized to maximize machine utilization in a functional layout rather than the benefits of cellular manufacturing. The existence of functionally similar machines or workcenters is not considered in a functional layout because parts can be routed or scheduled to the next available machine [21]. These functionally identical machines lead to alternative operations or routings which can be used to obtain better cell design and independence.

The p-median algorithm, a mathematical programming approach, was extended by Kusiak [18] to include complete fixed alternative routes, while Shtub [26] used a generalized assignment problem to solve a similar problem in an attempt to obtain better cell independence. Nagi et al. [21] developed a two-phase approach that minimizes the intercell traffic in the manufacturing system while considering part demand, capacity considerations, multiple fixed part routings, and multiple, functionally identical machines. Kang and Wemmerlov [14] developed a similarity index heuristic that determines machine cells first and then part families by considering multiple routes specified both in terms of operations and machines. This procedure also considers capacity when combining operations into routes. However, most of these heuristic methods lack the flexibility to form cells using a variety of evaluation measures or to handle constraints adequately. A majority of the methods that include alternative operations and multiple routes utilize mathematical programming, limiting their usefulness for solving large-scale cell formation problems.

To move toward a more satisfactory algorithmic result, other manufacturing information, such as setup time requirements, tooling and crew requirements, machine capacity, alternative routings, machine costs, intercell transfers, intracell and intercell layout, and reduced machine utilization needs to be included. The GA approach to cell design developed in [10] allows the system designer to substitute various types of evaluation functions, permitting alternative designs to be quickly generated and reviewed.

In this paper, the flexibility/extensibility of the GA approach with respect to its ability to incorporate new representations and other manufacturing information is explored. First, the GA model is extended to incorporate alternative operations and redundant machinery. Next, complete fixed alternative routings are considered. This reflects the practice in some manufacturing systems, such as furniture manufacturing, where a part can be manufactured either completely with a single multi-purpose CNC workstation or with a set of manual machines. In this case, there are multiple routes since the set of operations will either be done entirely on the CNC machine or on a set of manual machines. Finally, alternative operations, redundant machines, and fixed alternative routings are combined. This reflects the case where there may be several manual machines that can be substituted for one another or a set of interchangeable multipurpose CNC machines. In each case, the algorithm is demonstrated on data from the literature. Finally, the alternative operations model is applied to an industrial problem.

2 Cell Formation Using an Integer-Based GA

Consider an m machine and n part cell formation problem with $k_{\max}$ cells. Joines et al. [10] developed an integer programming model with the following variable declarations using set notation (see Appendix A for the classical variable definition).

$$\begin{aligned} x_i &= l, \quad \text{machine } i \text{ is assigned to cell } l \\ y_j &= l, \quad \text{part } j \text{ is assigned to part family } l \end{aligned}$$

Each part and machine variable is equal to the number of its assigned family or cell. Part families are assigned to the respective machine cell of the same number. A conventional integer programming solution technique cannot be employed because of the objective function's inability to decode this variable representation. However, for the GA, the objective function is a computer procedure (function) that can decode and evaluate a solution easily.

GAs maintain and manipulate a family, or population, of solutions in their search for an optimal or near optimal solution by implementing a "survival of the fittest" strategy. This provides an implicit as well as explicit parallelism that allows for the exploitation of several promising areas of the solution space at the same time. A more complete discussion of GAs, including extensions and related topics, can be found in [4, 20] among others.

For any GA, a chromosome representation is needed to describe each individual in the population of interest. Using our previous representation [10], the first m variables represent the machines while the last n variables are associated with the parts. Therefore, each individual is a vector of $m+n$ integer variables on the range of 1 to the maximum number cells or families ($k_{\max}$). This representation differs from the classical binary representation developed by Holland [7].

$$\text{Individual} \longrightarrow (\underbrace{x_1, x_2, \ldots, x_m}_{\text{machines}}, \underbrace{y_1, y_2, \ldots, y_n}_{\text{parts}})$$

The first step in a GA procedure is to initialize the population either randomly or by seeding. Once the initial population is randomly created, each individual is evaluated using the objective function to determine its fitness or value. Evaluation functions of many forms can be used in a GA, subject to the minimal requirement that the function can map the population into a partially ordered set.

In this study, the "grouping efficacy" measure was used as the evaluation criteria. Grouping efficacy was chosen because it forces block diagonal cell formation to occur in the part/machine incidence matrix and is useful for comparing solutions to those of other techniques found in the literature [16]. Grouping efficacy has a value of one when there are no exceptional elements and no voids and a value of zero if the number of exceptional elements equals the total number of operations. An exceptional element is a part operation which is performed outside the part's designated cell. Formally, grouping efficacy (Γ) is defined as

$$\Gamma = \frac{(1-\psi)}{(1+\phi)} = \frac{1-\frac{e_o}{e}}{1+\frac{e_v}{e}} = \frac{e-e_o}{e+e_v},$$

where e is the number of operations in the data matrix, e_v is the number of voids in the diagonal blocks, and e_o is the number of exceptional elements. Note, a weighted grouping efficacy proposed by Ng[22] can also be used. Rogers and Shafer [24] demonstrated that grouping efficacy with operation sequence information to be one of the best choices to use when "making static decisions because a trade off is made between having exceptional elements outside of the cells and the voids in the cells." This paper will include the operation sequences in various forms.

After the population (of size N) has been evaluated, a new population of N individuals is selected from the previous generation. The selection of individuals to produce successive generations plays an extremely important role. These individuals do not have to be distinct; that is, an individual in the population can be selected more than once. A probabilistic selection is performed where each individual is assigned a probability based upon its fitness such that the better individuals have an increased chance of being selected. However, all of the individuals in the population have a chance of being selected to reproduce. The GA used in these experiments uses a normalized geometric ranking scheme and employs the elitist model in which the best individual from the previous generation is included in the current one [10].

After the new population is selected, R parents are randomly chosen from the new population to produce children by applying mutation and crossover genetic operators [20]. Mutation operators tend to make small random changes in a single parent to form a child in an attempt to explore all regions of the state space. Crossover operators combine information from two parents to form two offspring that contain a "likeness" (a set of building blocks) from each parent. Joines et al. [10] modified six float operators developed by Michalewicz [20] to work with the integer representation: *uniform mutation, non-uniform mutation, multi-non-uniform mutation, boundary mutation, simple crossover, arithmetic crossover.* Two problem-specific genetic operators (*cell-swap crossover* and *cell-two-point crossover*) based on the proposed cell formation representation were also developed and shown to enhance the GA's performance in terms of computational efficiency and solution quality [10].

Michalewicz [20] and Joines et al. [10]. have shown that all of these genetic operators are useful. Each operator aids the search process differently (e.g., *arithmetic crossover* forces the solutions to the middle of search space while *boundary mutation* forces the solutions to the boundaries). Also, many of the operators work better at different times during the search process. For example, *uniform mutation* works better at the beginning of the search process allowing the GA to explore several regions of the search space (i.e., good global exploitation) while near the end of the searching, one needs more local searching capability. Therefore, *Non-uniform mutation* works similar to the uniform mutation at the start of the search but the distribution tightens as the search proceeds performing more local exploitation at the end of the search. For more detailed information on the modifications made to the standard float operators or the problem-specific operators, see Appendix B.4 and Joines et al. [10].

The GA moves from generation to generation until the termination criterion is met. The stopping criterion used in these experiments is the specification of the maximum number of generations. This allows the maximum number of (not necessarily unique) solutions that are evaluated to be preset. However, other termination strategies can be used. See Appendix B.1 for more details on the genetic algorithm used, as well as the various GA parameters.

3 Incorporation of Alternative Operations and Machine Redundancy

One objective is to form completely independent cells since the benefits of cellular manufacturing (reduced WIP, throughput time, etc.) are reduced when inter-cell movement exists. Most techniques try to minimize inter-cell movement as represented by the number of exceptional elements. However, the number of exceptional elements may not accurately reflect the level of inter-cell movement required. For example, if the inter-cell movement occurs in the middle of the operational sequence, two (not one) inter-cell movements will be required. The operational sequence can be extremely valuable in identifying the effect of inter-cell traffic and backtracking. To eliminate exceptional elements, the machine causing inter-cell movement can be duplicated or the parts routing sequence can be changed by subcontracting the part, redesigning the part, or using an alternative routing or operation. Routings are often generated to maximize machine utilization rather than maximize the effectiveness of group technology. Therefore, a fixed relationship between a part and a particular set of machines can be constraining since there may exist alternative machines for a specific operation which may lead to greater cell independence [14].

Chan and Milner [2] and King [15] eliminate exceptional elements by interactively duplicating machines and re-solving the problem. Exceptional elements are interdependent because actions used to eliminate one element may affect other elements in the incidence matrix. Therefore, deciding to duplicate a machine early in the cell design process may not lead to the best solution. Others have suggested interactive sessions after the initial cell formation to eliminate the cells by redesigning the part or assigning the operation to another machine. Neither of these is satisfactory since real problems are often large and complicated.

Most cell formation algorithms use the binary part/machine matrix, A, consisting of elements $a_{ij} = 1$, if part j requires processing on machine i, otherwise $a_{ij} = 0$. This format leads to fixed routing sequences with no machine substitutions. Several researchers have proposed the following representation, where the operational sequence is embedded in the matrix [21]:

$$a_{ij} = \begin{cases} k, & \text{the } k\text{th operation of part } j \text{ is required on machine } i \\ 0, & \text{no processing on machine } i. \end{cases}$$

Using this variable definition, alternative operations can be incorporated into the part/machine incidence matrix by specifying that several machines can perform the kth operation of a particular part. Because the evaluation function is independent of the decision rules, the GA provides the flexibility to interchange various objective functions without changing the algorithm. The evaluation function must take into account only the new part/machine incidence matrix representation. Specifying alternative operations has the advantage of allowing the algorithm to determine the most appropriate routing sequences in the context of minimizing intercell flows. It also allows one to determine the true effect of intercell movement, backtracking, and sequence dependent setup times since the actual sequence of operations is specified. See Appendix A.1 for the non-linear integer programming formulation using the classical assignment variables.

An example (taken from Nagi et al. [21]) is used to demonstrate the GA's ability to use this new incidence matrix representation that permits alternate operation substitution. The problem consists of 20 different parts and 20 workcenters. Workcenters 5 and 6 are interchangeable as are workcenters 17, 18, and 19 for all parts. For comparison, the Nagi problem was modified by removing all duplicate workcenters (machines 6, 18, and 19), and a standard binary part/machine incidence matrix was created as shown in Figure 1. The GA requires 136.25 generations on average for 10 replications to solve the problem. The solution, having a Γ of 0.7308 and 10 exceptional elements, is shown in Figure 2. Further evaluation using the operation sequence information in the part/machine matrix shown in Figure 3 reveals that the 10 exceptional elements actually generate 13 intercell movements. This illustrates the limitation of a fixed binary representations of the routings in accurately measuring intercell flows.

Next, the GA was used to solve the problem with the inclusion of redundant machines along with the operational sequences (shown in Figure 3). The algorithm required an average of 143.2 generations to reach a final Γ of 0.7952 over 10 replications. The solution (shown in Figure 4) contains only one exceptional element. The fourth operation of part 4 cannot be processed inside its cell (cell 4), and one intercell movement to either of three cells (cell 0, 1, or 3) is required. Thus, an improved cell formation was achieved by the GA through the consideration of alternate machines for selected operations and including redundant machiners.

4 Incorporation of Alternative Routings

Even though the model from the previous section can generate cells using dynamic routes, the formulation suffers from two distinct disadvantages. First, it forms cells considering alternative operations in which preferred sets of machines can not be specified and preserved through the clustering process. This can be important for manufacturers who possess a mix of manual machines and flexible CNC technology. To illustrate, consider a part with two valid machining sequences. One sequence employs several manually operated machines and the second sequence might consist of a single CNC workstation. The previous GA formulation of Section 3 considering alternative operations would allow the CNC workstation to be substituted for any of the individual machines in the manual sequence. In practice, it would be unlikely that a part that can be produced entirely on the CNC workstation would visit this workstation for only one operation while the other operations are performed on manually operated machines. It is more likely that the part would be machined entirely on the CNC workstation or produced exclusively using the manually operated machines. A second disadvantage of this model over complete alternative fixed routings is the ability not to assign priorities to certain routes. This would allow the cell designer to generate cells that balance work loads, achieve minimum investment in tooling costs, take full advantage of manpower skills, etc.

A generalized cell formation problem that can consider complete fixed alternative routings which overcomes these two disadvantages can be formulated from the classic IP formulation [10] by adding constraints to ensure that each part is assigned one and only one route. However, it is easily shown to be NP-hard since the original problem is NP-hard [27, 28]. The classical formulation must incorporate the following additional variable declarations and constraints to handle fixed routings. The constraints ensure that each part is assigned only one routing and the non-linear integer formulation [10] can easily be updated as shown in Appendix A.2.

$$z_{jh} = \begin{cases} 1, & \text{if part } j \text{ uses route } h, \\ 0, & \text{otherwise.} \end{cases}$$

$$\sum_{l=1}^{p_j} = 1 \quad j = 1, \ldots , m$$

$$p_j = \text{the number of alternative routes for part } j$$

The GA approach is extended by developing an integer programming-based GA with a new set of variable declarations that allow evaluation of fixed alternative routings. Specifically, let $z_j = h$, if part j uses route h. If there is only a single routing for part j, this variable is unnecessary and is not be included in the programming model. Notice that the number of variables has grown by, at most, a factor of n. With this new variable definition, a new chromosome representation can be developed easily by adding the z variable component to individuals in the population.

$$\text{Individual} \longrightarrow (\underbrace{x_1, x_2, \ldots, x_m}_{\text{machines}}, \underbrace{y_1, y_2, \ldots, y_n}_{\text{parts}}, \underbrace{z_1, z_2, \ldots, z_n}_{\text{route}})$$

Again, because of the flexibility of the GA, the algorithm remains unchanged and only the evaluation function is modified to accommodate this new representation. This formulation of the GA is also demonstrated on the Nagi problem. Figure 5 shows the original part/machine incidence matrix with the alternate route for each part. Notice that parts 16, 17, 18, and 19 have only one process plan. The GA then determines the appropriate values for only 56 different variables (20 machine, 20 part, and 16 route assignments). As in the previous model, there is only one exceptional element and intercell movement (part 4, shown in the solution shown in Figure 6). The same value Γ of 0.7952 was obtained after an average of 231.6 generations. For this problem, identical cell configurations are obtained because the complete alternative routings matrix (Figure 5) can be reduced to the operational sequence matrix in Figure 3. This would not be possible if the number of operations per route were different. For example, if part 1 could be produced on machines 5, 8, and 11 or machines 6 and 9, then these two alternate routes cannot be reduced to the form of the operation sequences matrix in Figure 3. Therefore, different cell configurations are possible.

5 Complete alternative routings with machine redundancy

Even though consideration of complete alternative routings offers the ability to specify a preferred set of machines or to assign priorities to certain routes, it can also be limiting. The operational sequences model offers advantages by allowing machine redundancy as well as determination of the true effect of intercell movement. These two models can be combined into one representation by allowing the complete alternative routings to contain their operation sequences as well.

Again, because of the flexibility of the GA, the algorithm remains unchanged and only the evaluation function is modified to accommodate the combination of the operation sequences with complete alternative routings. This formulation of the GA is demonstrated on a modified version of the Nagi problem. Figure 7 shows the original part/machine incidence matrix with the alternate routes for each part, where each alternate routes now contains the operation sequences. Some routes contain machine redundancies (i.e., certain operations of a route can be performed on several different machines). Again, parts 16, 17, 18, and 19 have only one route. As in Section 4, the GA determines the appropriate values for only 56 variables. This model was able to eliminate the exceptional element as shown in Figure 8 as well as obtain a slightly higher value Γ of 0.7975 in an average of 227.7 generations. Notice that the GA was able to determine the most appropriate route to use as well as dynamically build the operational sequence.

By combining machine redundancy with the complete fixed alternative routes, this model offers distinct advantages over previous models. While the model of the previous section can handle machine redundancy through the use of separate routes, the approach in this section is more computationally and state space efficient. Take the same example of a part that can be manufactured at a single CNC workstation or with a set of manual machines. As stated before, one does not want the GA to assign certain operations to the CNC workstation and the remaining operations to manual machines. Therefore, a preferred set of machines needs to be specified. However, there may be several manual machines that can be substituted for one another or a set of interchangeable multipurpose CNC machines. The GA has the capability of utilizing the most appropriate CNC workstation or set of manual machines within the most appropriate complete alterntive routing. Also, this model, unlike the complete alternative routings model, is able to determine the true effect of intercell movement.

6 Industry case study

The GA described in this paper has been applied to several industrial problems. One of the problems included a focused factory for a division of a large international company. This particular division manufactures and machines rotary unions. Due to product changes over the past few years, the current layout had become very inefficient in terms of material flow (i.e., distance traveled by parts) and space utilization. Management wanted an improved layout to be designed. In particular, management was interested in determining if changing the machinery area to a cellular layout would be beneficial. Since attempts to cluster the data manually were ineffective, the GA approach was used to cluster the data. The objective was to identify the machine cells and the various parts assigned to those cells. Based upon the clusters found, a layout was to be generated and evaluated.

The model from Section 3, which incorporates alternative operations and machine redundancy, was deemed the most appropriate for the initial clustering. This was due, in part, to the fact that the current part operations could be easily obtained from the MRP system and there were functionally identical machines for a few of the machine types. More detailed information on alternate operations and/or possible alternate routes could have been obtained at an additional cost; however, management felt that it was not important in the initial design.

The manufacturing system consisted of 20 unique machine types. For all but three, there was a single machine of a given type. For these three types, there were three machines of one type and two of the other two types, resulting in a total of 24 machines. It was determined that two of these machines were not being used by the current part population and were not anticipated to be used in the future. Therefore, only 22 machines were considered. The company identified 185 representative (high volume) parts.

Management determined that the maximum number cells they could manage was four. Since the solution generated by the GA with $k_{\max} = 4$ was binding (i.e., generated a four cell solution), an alternative solution with $k_{\max} = 3$ was also generated. Ten different replications were performed for both maximum number of cells. It took approximately 559 seconds on a Motorolla 604 Power PC to run all 10 replications for 2000 generations.

Subsequently, the company took the two alternatives and created layouts based on the cell/family configurations generated by the GA. Using projected future demand, the company determined, for each of the machines, the within-cell utilization percentage, which is defined as the assigned part demand for this particular machine divided by the total demand. Also, they looked at capacity issues of the redundant machines. This reflects the amount of within-cell demand satisfied versus intercell demand. Based on these percentages, the space reduction generated by the reconfiguration of the machines into cells, the savings in distance traveled, and a few other qualitative issues, the company chose the four cell solution. Also, the four cell layout was a better design for the current building. Therefore, no additional analysis needed to be performed. The original operations part/machine incidence matrix can be seen in Figure 9 while the four cell solution can be seen in Figure 10.

For the four cell solution, the average within-cell machine utilization was 92%, with the lowest within cell machine utilization being 68%. Based on the new layout, there was a projected 38.5% reduction in material flow (43,150 feet) per year. With the removal of two machines and the machine cell reconfiguration, there was a reduction in space utilization of 4.1% and 12.5%, respectively, for a total reduction of 16.6%. This space reduction will allow for expansion of the current product line. The cell design generated by GA was implemented by the company.

7 Conclusions

There is an extensive literature concerning optimization and heuristic methods for cell formation. However, the practical application of many techniques is restricted either by problem scale, by the representation of the input data, or by the limited discrimination of the evaluation measure used for cell formation. For example, most of these methods utilize only the information stored in the part/machine incidence matrix.

Joines et al. [10] developed a genetic algorithm approach to solve integer programming formulations of the manufacturing cell formation problem. It offers several advantages over previous techniques, e.g., the technique simultaneous forms machine cells and part families and is capable of determining the naturally occurring clusters as well as solutions with a constrained number of cells/families. Since the evaluation function is a computer procedure and independent of the decision rules used in the algorithm, the method conveniently allows the substitution of various evaluation functions. Alternative cell designs can be generated and reviewed quickly and multi-criteria objective functions can be employed.

This paper extends the capability of the original model. The examples demonstrated the GA's capability to form cells using a variety of part/machine incidence matrix representations. Specifically, these representations allow the GA to generate solutions that considered alternate operations (substitute machines) or functionally identical machinery. The GA was able to generate the routings for each part dynamically so as to maximize the effectiveness of using manufacturing cells. The next extension incorporated complete fixed alternative routing sequences for selected parts. This allows the cell designer to force certain operations to be performed on a particular set of machinery. The last extension combined the two previous models. This extension offers the advantage of allowing the cell designer to prioritize or weight certain routes as well as include machine redundancy information. Since the actual operation sequences are embedded in the representation, the true effect of intercell movement, backtracking, and sequence dependent setup times can be determined and, possibly, used in further extensions.

Joines et al. [10, 12] demonstrated the computational efficiency of the GA approach in solving large instances of the cell formation problem. The GA approach scales very well in terms the number of machines m, parts n, and cell/families $k_{\max}$. The operation sequences model only makes a slight change to the evaluation function while the complete fixed routings model adds additional variables as well. The computational time need to run each model on the various Nagi problem for a 1000 generations on a DEC 5000 is given in Table 1 along with the average number of generations needed to solve the problem. Notice that it takes only a few seconds to run. The GA has been run on several problems from the furniture industry where the number of machines generally exceeds a 100 and the number of parts exceeds 2000. The size of these problems generally excludes most other cell formation techniques.

Model	Simple	Alt. Operations	Alt. Routings	Combined
Avg. No. Gen.	136.25	143.2	236.1	227.7
Computational Time (seconds)†	5.96	6.70	7.71	8.56

†-time to initialize population and run 1000 generations

Table 1: Computational Time of the GA Approaches in CPU Seconds

These extensions show how various representations can easily be incorporated into the GA. The algorithm was able to generate better solutions in terms of both grouping efficacy and the number of intercell movements by just incorporating this simple additional information. Even with these simple additions and evaluation measure, the clustering process has been shown to be effective for real industry data sets. The efficiency of the algorithm will allow us to include more practical aspects of the group technology problem (i.e., machine capacity, volume demand, intercell layout, tooling and crew requirements, sequence dependent setup times, cell size, etc.) in order to obtain a more satisfactory algorithmic result [9].

References

[1] J.T. Black. Cellular manufacturing systems reduce setup time, make small lot production economical. *Industrial Engineering*, 29(10):36–48, 1983.

[2] H.M. Chan and D.A. Milner. Direct clustering algorithm for group formation in cellular manufacture. *Journal of Manufacturing Systems*, 1(1):65–74, 1982.

[3] C.-H. Chu. Cluster analysis in manufacturing cellular formation. *International Journal of Management Science*, 17(3):289–295, 1989.

[4] L. Davis. *The Handbook of Genetic Algorithms.* Van Nostrand Reingold, New York, 1991.

[5] G. M. Fazakerley. A research report on the human aspects of group technology and cellular manufacture. *International Journal of Production Research*, 14(1):123–134, 1976.

[6] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[7] J.H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 2nd edition, 1992.

[8] N.L. Hyer and U. Wemmerlöv. Group technology in the us manufacturing industry: A survey of current practices. *International Journal of Production Reseach*, 27(8):1287–1304, 1989.

[9] J.A. Joines. *Hybrid Genetic Search for Manufacturing Cell Design*. Ph.d. thesis, North Carolina State University, Raleigh, NC, December 1996.

[10] J.A. Joines, C.T. Culbreth, and R.E. King. Manufacturing cell design: An integer programming model employing genetic. *IIE Transactions*, 28(1):69–85, 1996.

[11] J.A. Joines and C.R. Houck. On the use of non-stationary penalty functions to solve constrained optimization problems with genetic algorithms. In *1994 IEEE International Symposium Evolutionary Computation*, pages 579–584, Orlando, Fl, 1994.

[12] J.A. Joines, R.E. King, and C.T. Culbreth. The use of a local improvement heuristic with a genetic algorithm for manufacturing cell design. Technical Report NCSU-IE Technical Report 95-06, North Carolina State University, 1995.

[13] J.A. Joines, R.E. King, and C.T. Culbreth. A comprehensive review of production-oriented manufacturing cell formation techniques. *International Journal of Flexible Automation and Integrated Manufacturing*, 3(3&4):225–265, 1996.

[14] S. Kang and U. Wemmerlöv. A work load-oriented heuristic methodology for manufacturing cell formation allowing reallocation of operations. *European Journal of Operational Research*, 69(3):292–311, 1993.

[15] J.R. King. Machine-component grouping formation in group technology. *International Journal of Management Science*, 8(2):193–199, 1980.

[16] K.R. Kumar and M.P. Chandrasekharan. Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Reseach*, 28(2):233–243, 1990.

[17] A Kusiak. Group technology: Models and solution approaches. In *First Industrial Engineering Research Conference*, pages 349–352, 1992.

[18] A. Kusiak and M. Cho. Similarity coefficient algorithms for solving the group technology problem. *International Journal of Production Reseach*, 30(11):2633–2646, 1992.

[19] H. Lee and A. Garcia-Diaz. A network flow approach to solve clustering problems in group technology. *International Journal of Production Reseach*, 31(3):603–612, 1993.

[20] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. AI Series. Springer-Verlag, New York, 3rd edition, 1996.

[21] R. Nagi, G. Harhalakis, and J.M. Proth. Multiple routings and capacity considerations in group technology applications. *International Journal of Production Reseach*, 28(12):2243–2257, 1990.

[22] S. Ng. Worst-case analysis of an algorithm for cellular manufacturing. *European Journal of Operational Research*, 69(3):384–398, 1993.

[23] R.D. Pullen. A survey of cellular manufacturing cells. *Production Engineer*, 56(9):431–454, 1976.

[24] D.F. Rogers and S.M. Shafer. *Planning, Design, and Analysis of Cellular Manufacturing Systems*, chapter Measuring cellular manufacturing performance, pages 147–165. Elsevier Science, 1995.

[25] H.M. Selim, R.G. Askin, and A.J. Vakharia. Cell formation in group technology: Review, evaluation and directions of future research. Technical Report Working Paper, University of Arizona, Tucson, 1994.

[26] A. Shtub. Modeling group technology cell formation as a generalized assignment problem. *International Journal of Production Reseach*, 27(5):775–782, 1989.

[27] V. Venugopal and T.T. Narendran. Cell formation in manufacturing systems through simulated annealing: An experimental evaluation. *European Journal of Operational Research*, 63(3):409–422, 1992.

[28] V. Venugopal and T.T. Narendran. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Computers in Industrial Engineering*, 22(4):469–480, 1992.

[29] U. Wemmerlöv and N.L. Hyer. Cellular manufacturing in the u.s. industry: a survey of users. *International Journal of Production Reseach*, 27(9):1511–1530, 1989.

	Parts																			
Machines	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0:		1	1	1	1															
1:						1		1	1	1										
2:											1	1	1							
3:																		1	1	1
4:						1	1			1										
5:	1	1	1	1		1	1	1	1	1										
7:											1	1	1							
8:	1		1		1															
9:														1	1	1	1			
10:											1		1							
11:	1	1	1	1	1															
12:																		1	1	
13:														1		1	1			
14:																		1	1	1
15:							1	1	1	1										
16:														1	1		1			
17:					1		1		1		1	1	1	1	1	1				

Figure 1: Original Matrix without Machine Redundancy

Family/Cell #	Parts	Machines
Family/Cell 0	5, 6, 7, 8, 9	1, 4, 5, 15
Family/Cell 1	10, 11, 12	2, 7, 10, 17
Family/Cell 2	0, 1, 2, 3, 4	0, 8, 11
Family/Cell 3	13, 14, 15, 16	9, 13, 16
Family/Cell 4	17, 18, 19	3, 12, 14

	Parts																			
Machines	5	6	7	8	9	10	11	12	0	1	2	3	4	13	14	15	16	17	18	19
1:	1		1	1	1															
4:	1	1			1															
5:	1	1	1	1	1				1	1	1	1								
15:		1	1	1	1															
2:						1	1	1												
7:						1	1	1												
10:						1		1												
17:		1		1		1	1	1					1	1	1	1				
0:										1	1	1	1							
8:									1		1		1							
11:									1	1	1	1	1							
9:														1	1	1	1			
13:														1		1	1			
16:														1	1		1			
3:																		1	1	1
12:																		1	1	
14:																		1	1	1

Figure 2: Solution to Nagi's Problem Without Machine Redundancy

Machines \ Parts	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0:		3	1	1	2															
1:						3		3	2	1										
2:											2	1	3							
3:																		2	1	2
4:						2	1			3										
5:	3	1	4	3		1	4	2	4	4										
6:	3	1	4	3		1	4	2	4	4										
7:											1	2	2							
8:	2		2		1															
9:														1	3	2	3			
10:											3		1							
11:	1	2	3	2	3															
12:																		1	2	
13:														3		3	2			
14:																		3	3	1
15:							2	1	1	2										
16:														4	2		1			
17:					4		3		3		4	3	4	2	1	1				
18:					4		3		3		4	3	4	2	1	1				
19:					4		3		3		4	3	4	2	1	1				

Figure 3: Original Part/Machine Operation Incidence Matrix from Nagi [21]

Family/Cell #	Parts	Machines
Family/Cell 0	10, 11, 12	2, 7, 10, 17
Family/Cell 1	5, 6, 7, 8, 9	1, 4, 5, 15, 18
Family/Cell 2	17, 18, 19	3, 12, 14
Family/Cell 3	13, 14, 15, 16	9, 13, 16, 19
Family/Cell 4	0, 1, 2, 3, 4	0, 6, 8, 11

Machines \ Parts	10	11	12	5	6	7	8	9	17	18	19	13	14	15	16	0	1	2	3	4
2:	2	1	3																	
7:	1	2	2																	
10:	3		1																	
17:	4	3	4																	4
1:				3		3	2	1												
4:				2	1			3												
5:				1	4	2	4	4												
15:					2	1	1	2												
18:					3		3													4
3:									2	1	2									
12:									1	2										
14:									3	3	1									
9:												1	3	2	3					
13:												3		3	2					
16:												4	2		1					
19:												2	1	1						4
0:																	3	1	1	2
6:																3	1	4	3	
8:																2		2		1
11:																1	2	3	2	3

Figure 4: Solution to the Operation Incidence Matrix from Nagi et al. [21]

Part #	P 0		P 1		P 2		P 3		P 4			P 5		P 6						P 7		P 8						P 9		P10			P11			P12			P13			P14			P15			P16	P17	P18	P19
Rout#	1	2	1	2	1	2	1	2	1	2	3	1	2	1	2	3	4	5	6	1	2	1	2	3	4	5	6	1	2	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	1	1	1
0:			1	1	1	1	1	1	1	1	1																																								
1:												1	1							1	1	1	1	1	1	1	1	1	1																						
2:																														1	1	1	1	1	1	1	1	1													
3:																																																	1	1	1
4:												1	1	1	1	1	1	1	1									1	1																						
5:	1		1		1		1					1					1	1	1		1		1		1		1	1																							
6:		1		1		1		1					1	1	1	1				1		1		1		1			1																						
7:																														1	1	1	1	1	1	1	1	1													
8:	1	1			1	1			1	1	1																																								
9:																																							1	1	1	1	1	1	1	1	1	1			
10:																														1	1	1				1	1	1													
11:	1	1	1	1	1	1	1	1	1	1	1																																								
12:																																																	1	1	
13:																																							1	1	1				1	1	1	1			
14:																																																	1	1	1
15:														1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1																						
16:																																							1	1	1	1	1	1				1			
17:										1					1			1						1	1					1			1			1			1			1			1						
18:									1					1			1					1	1								1			1			1			1			1			1					
19:											1					1			1							1	1					1			1			1			1			1			1				

M a c h i n e s

Figure 5: Original matrix for the Nagi problem with complete alternative routings

family/cell #	parts	machines
family/cell 0	13, 14, 15, 16	9, 13, 16, 17
family/cell 1	17, 18, 19	3, 12, 14
family/cell 2	5, 6, 7, 8, 9	1, 4, 5, 15, 19
family/cell 3	0, 1, 2, 3, 4	0, 6, 8, 11
family/cell 4	10, 11, 12	2, 7, 10, 18

part #	0	1	2	3	4	5	6	7	8	9
route #	2	2	2	2	3	1	6	2	6	1

part #	10	11	12	13	14	15	16	17	18	19
route #	2	2	2	1	1	1	-	-	-	-

P a r t s

	13	14	15	16	17	18	19	5	6	7	8	9	0	1	2	3	4	10	11	12
9:	1	1	1	1																
13:	1		1	1																
16:	1	1		1																
17:	1	1	1																	
3:					1	1	1													
12:					1	1														
14:					1	1	1													
1:								1		1	1	1								
4:								1	1			1								
5:								1	1	1	1	1								
15:									1	1	1	1								
19:									1		1						1			
0:														1	1	1	1			
6:													1	1	1	1				
8:													1		1		1			
11:													1	1	1	1	1			
2:																		1	1	1
7:																		1	1	1
10:																		1		1
18:																		1	1	1

M a c h i n e s

Figure 6: Solution for the Nagi problem with complete alternative routings

Part #	P 0		P 1		P 2		P 3		P 4			P 5		P 6				P 7		P 8				P 9		P 10		P 11		P 12		P 13		P 14		P 15		P16	P17	P18	P19
Rout#	1	2	1	2	1	2	1	2	1	2	3	1	2	1	2	3	4	1	2	1	2	3	4	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	1	1	1
0:			3			1		1			2				1										1																
1:													3					1	3				2			1				1											
2:		1			2		3									1						1					2		1		3	1									
3:				1					2																	2		2											2	1	2
4:													2				1								3					2				3		4					
5:	3		1					3					1		2		4		2		2		4		4	3				3											
6:	3		1			4		3					1		2		4		2		2		4		4	3				3											
7:					1											2						2					1		2		2	3									
8:	2					2					1																														
9:										2				1						2				1									1		3		2	3			
10:																											3				1	4									
11:	1		2			3		2			3										1																				
12:				2			2											2		1																			1	2	
13:										1		1		2										2									3				3	2			
14:				3			1		1									3																					3	3	1
15:																	2		1				1		2			1						2		2					
16:												2		3						3				3									4		2			1			
17:		2			3					3	4	3				3	3					3	3				4		3	4	4	2	2	1	1	1	1				
18:		2			3					3	4	3				3	3					3	3				4		3	4	4	2	2	1	1	1	1				
19:		2			3					3	4	3				3	3					3	3				4		3	4	4	2	2	1	1	1	1				

Machines

Figure 7: Original matrix with complete alternative routings and machine redundancy

family/cell #	parts	machines
family/cell 0	7, 8, 12	1, 4, 5, 15, 19
family/cell 1	0, 3	0, 6, 8, 11
family/cell 2	1, 17, 18, 19	3, 12, 14
family/cell 3	4, 5, 9, 13, 14, 15, 16	9, 13, 16, 18
family/cell 4	2, 6, 10, 11	2, 7, 10, 17

part #	0	1	2	3	4	5	6	7	8	9
route #	1	1	1	1	2	1	3	2	4	2

part #	10	11	12	13	14	15	16	17	18	19
route #	2	2	2	2	2	2	-	-	-	-

Parts

	7	8	12	0	3	1	17	18	19	4	5	9	13	14	15	16	2	6	10	11
1	3	2	1																	
4			2																	
5	2	4	3																	
15	1	1																		
19		3	4																	
0					1															
6				3	3															
8				2																
11				1	2															
3						1	2	1	2											
12						2	1	2												
14						3	3	3	1											
9										2		1	1	3	2	3				
13										1	1	2	3		3	2				
16											2	3	4	2		1				
18										3	3		2	1	1					
2																	2	1	2	1
7																	1	2	1	2
10																			3	
17																	3	3	4	3

Machines

Figure 8: Solution matrix with complete alternative routings and machine redundancy

Figure 9: The original part/machine incidence matrix for Real Data Set (22×148)

Figure 10: Integer GA solution for the 22×148 Real Data Set with $\Gamma = .5448$

Appendix A

The following assignment variables and constraints define the classical cell formation problem. This set of variables and constraints are replaced with the set notation variable sets defined in Section 2 but is used in the non-linear integer programming formulations of the next section.

$$\begin{aligned}
x_{il} &= \begin{cases} 1, & \text{if machine } i \text{ is assigned to cell } l, \\ 0, & \text{otherwise;} \end{cases} \\
y_{jl} &= \begin{cases} 1, & \text{if part } j \text{ is assigned to part family } l, \\ 0, & \text{otherwise;} \end{cases} \\
\sum_{l=1}^{k} x_{il} &= 1 \;\; i = 1, \ldots, m; \\
\sum_{l=1}^{k} y_{jl} &= 1 \;\; j = 1, \ldots, n; \\
k &= \text{the number of cells (families) specified;} \\
m &= \text{the number of machines;} \\
n &= \text{the number of parts;}
\end{aligned}$$

Recall that grouping efficacy is defined as follows.

$$\Gamma = \frac{(1 - \frac{e_o}{e})}{(1 + \frac{e_v}{e})} = \frac{e - e_o}{e + e_v} = \frac{e - (e - e_d)}{e + (D - e_d)} = \frac{e_d}{e + D - e_d}$$

where:

$e =$ the number of operations in the data matrix;

$e_v =$ the number of voids in the diagonal blocks;

$e_d =$ the number of operations in the diagonal blocks;

$e_o =$ the number of exceptional elements;

$D =$ the area covered by the diagonal blocks.

A.1 Nonlinear Formulation of Γ for the Alternative Operations Model

In order for the alternative operations problem to be modeled as a nonlinear integer programming problem, the following new variable definitions and constraint sets need to be added to the original definition of grouping efficacy defined in [10]. In this model, a particular operation may be performed by more than one machine. Therefore, the new variable (o_{jkl}) determines which operation is selected. The first set of constraints ensures that operation k of part j is assigned to exactly one cell. The second set of constraints makes sure at least one machine that can perform operation k has been assigned to the same cell.

$$o_{jkl} = \begin{cases} 1, & \text{if operation } k \text{ of part } j \text{ is assigned to cell } l \\ 0, & \text{otherwise} \end{cases}$$

$$\sum_{l=1}^{k_{\max}} o_{jkl} = 1 \qquad j = 1, \ldots, n, k = 1, ..., |O_j|$$

$$o_{jkl} \leq \sum_{i \in M_{kj}} x_{il} \qquad l = 1, \ldots ; k_{\max}; j = 1, \ldots, n, k = 1, ..., |O_j|$$

where

$$O_j = \text{the set of operations for part } j.$$
$$M_{jk} = \text{the set of machines that can perform operation } k \text{ for part } j$$

Using these new variables with the previously defined assignment variables, x_{il} and y_{jl}, grouping efficacy (Γ) can now be defined as follows:

$$\Gamma = \frac{\sum_{l=1}^{k_{\max}} \sum_{j=1}^{n} \sum_{k=1}^{|O_j|} y_{jl} o_{jkl}}{\sum_{j=1}^{n} |O_j| + \sum_{l=1}^{k_{\max}} [(\sum_{j=1}^{n} y_{jl})(\sum_{i=1}^{m} x_{il})] - \sum_{l=1}^{k_{\max}} \sum_{j=1}^{n} \sum_{k=1}^{|O_j|} y_{jl} o_{jkl}}.$$

A.2 Nonlinear Formulation of Γ for the Alternative Routings Model

In order to include complete fixed alternative routings into the model, the following new variables and constraint set were added to the original model. Also, the binary part/machine matrix, A, now consists of elements $a_{ijh} = 1$, if part j requires processing on machine i utilizing route h, otherwise $a_{ijh} = 0$. This format leads to alternative fixed routing sequences.

$$\begin{aligned} z_{jh} &= \begin{cases} 1, & \text{if part } j \text{ uses route } h \\ 0, & \text{otherwise} \end{cases} \\ \sum_{l=1}^{p_j} z_{jh} &= 1 \;\; j = 1, \ldots, n \\ p_j &= \text{the number of routes for part } j \end{aligned}$$

Using these new variables with the previously defined assignment variables, x_{il} and y_{jl}, grouping efficacy (Γ) can now be defined as follows.

$$\Gamma = \frac{\sum_{l=1}^{k_{\max}} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{h=1}^{p_j} x_{il} y_{jl} z_{jh} a_{ijh}}{\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{h=1}^{p_j} z_{jh} a_{ijh} + \sum_{l=1}^{k_{\max}} [(\sum_{j=1}^{n} y_{jl})(\sum_{i=1}^{m} x_{il})] - \sum_{l=1}^{k_{\max}} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{h=1}^{p_j} x_{il} y_{jl} z_{jh} a_{ijh}}$$

Appendix B

B.1 Genetic Algorithm

Genetic algorithms search the solution space of a function through the use of simulated evolution, i.e., the survival of the fittest strategy. In general, the fittest individuals of any population tend to reproduce and survive to the next generation, thus improving successive generations. However, inferior individuals can, by chance, survive and also reproduce. Genetic algorithms have been shown to solve linear and nonlinear problems by exploring all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals in the population [20]. A more complete discussion of genetic algorithms, including extensions and related topics, can be found in the books by Davis [4], Goldberg [6], Holland[7], and Michalewicz [20]. A genetic algorithm (GA) is summarized in Fig. 11, and each of the major components is discussed in detail below.

```
1. Supply a population $P_0$ of $N$ individuals and respective function values.
2. $i \leftarrow 1$
3. $P_i' \leftarrow selection\_function(P_i - 1)$
4. $P_i \leftarrow reproduction\_function(P_i')$
5. $evaluate(P_i)$
6. $i \leftarrow i + 1$
7. Repeat step 3 until termination
8. Print out best solution found
```

Figure 11: A Simple Genetic Algorithm

The use of a genetic algorithm requires the determination of six fundamental issues: chromosome representation, selection function, the genetic operators making up the reproduction function, the creation of the initial population, termination criteria, and the evaluation function. The rest of this section describes each of these issues.

B.2 Solution Representation

For any GA, a chromosome representation is needed to describe each individual in the population of interest. The representation scheme determines how the problem is structured in the GA and also determines the genetic operators that are used. Each individual or chromosome is made up of a sequence of genes from a certain alphabet. An alphabet could consist of binary digits (0 and 1), floating point numbers, integers, symbols (i.e., A, B, C, D), matrices, etc. In Holland's original design, the alphabet was limited to binary digits. Since then, problem representation has been the subject of much investigation. It has been shown that more natural representations are more efficient and produce better solutions[20]. One useful representation of an individual or chromosome for function optimization involves genes or variables from an alphabet of floating point numbers with values within the variables upper and lower bounds. Michalewicz[20] has done extensive experimentation comparing real-valued and binary GAs and shows that the real-valued GA is an order of magnitude more efficient in terms of CPU time. He also shows that a real-valued representation moves the problem closer to the problem representation which offers h igher precision with more consistent results across replications. [20]

In the representation of [10], the first m variables represent the machines while the last n variables are associated with the parts. Therefore, each individual is a vector of $m + n$ integer variables on the range of 1 to the maximum # of cells or families (k_{max})..

$$\text{Individual} \longrightarrow (\underbrace{x_1, x_2, \ldots, x_m}_{\text{machines}}, \underbrace{y_1, y_2, \ldots, y_n}_{\text{parts}})$$

This representation is modified to include the route variables as described in Section 4.

B.3 Selection Function

The selection of individuals to produce successive generations plays an extremely important role in a genetic algorithm. A probabilistic selection is performed based upon the individual's fitness such that the better individuals have an increased chance of being selected. However, all of the individuals in the population have a chance of being selected to reproduce into the next generation. An individual in the population can be selected more than once with all individuals in the population having a chance of being selected to reproduce into the next generation. There are several schemes for the selection process: roulette wheel selection and its extensions, scaling techniques, tournament, elitist models, and ranking methods [6, 20].

A common selection approach assigns a probability of selection, P_j, to each individual, j

based on its fitness value. A series of N random numbers is generated and compared against the cumulative probability, $C_i = \sum_{j=1}^{i} P_j$, of the population. The appropriate individual, i, is selected and copied into the new population if $C_{i-1} < U(0,1) \leq C_i$. Ranking methods only require the evaluation function to map the solutions to a totally ordered set, thus allowing for minimization and negativity. Ranking methods assign P_i based on the rank of solution i when all solutions are sorted. Normalized geometric ranking, [11], defines P_i for each individual by:

$$P[\text{ Selecting the } i\text{th individual }] = q\prime(1-q)^{r-1}; \tag{1}$$

where:

$$\begin{aligned} q &= \text{the probability of selecting the best individual,} \\ r &= \text{the rank of the individual, where 1 is the best.} \\ P &= \text{the population size} \\ q\prime &= \frac{q}{1-(1-q)^P} \end{aligned}$$

The GA used in these experiments uses a normalized geometric ranking scheme and employs the elitist model in which the best individual from the previous generation is always included in the current one [10].

B.4 Genetic Operators

Genetic Operators provide the basic search mechanism of the GA. The operators are used to create new solutions based on existing solutions in the population. There are two basic types of operators: crossover and mutation. Mutation operators tend to make small random changes in one parent to form one child in an attempt to explore all regions of the state space. Crossover operators combine information from two parents to form two offspring such that the two children contain a "likeness" (a set of building blocks) from each parent. The application of these two basic types of operators and their derivatives depends on the chromosome representation used.

Six float operators described by Michalewicz [20] were modified to work with the integer representation: *uniform mutation, multi-uniform-mutation, non-uniform mutation, multi-non-uniform mutation, boundary mutation, simple crossover, arithmetic crossover.*[10] Two problem-specific genetic operators (*cell-swap crossover* and *cell-two-point crossover*) based on the proposed cell formation representation were also developed and shown to enhance the GA's performance.[10] Let $\bar{X} = (x_1, x_2, \ldots, x_n)$ and $\bar{Y} = (y_1, y_2, \ldots, y_n)$ be two n-dimensional integer row vectors denoting individuals (parents) from the population. Each of these operators were used in the experiments of this paper and are described briefly below. Let a_i and b_i be the lower and upper bound, respectively, for each variable i. For the cell formation problem (i.e., the machine and part variables), the lower bound is 0 while the upper bound is equal to $k_{\max}$, the maximum number of cells/families and P_j, the number of routes for part j for the machine/part variables and route variables, respectively. These operators are applied a discrete number of times to the population (see Table 2). Parents are randomly selected from the population to undergo the various operators.

Uniform Mutation: Randomly selects one variable, j, and sets it equal to a truncated uniform random number, $\lfloor U(a_i, b_i) \rfloor$ where $\lfloor x \rfloor$ is the largest integer less than or equal to x.

$$x'_i = \begin{cases} \lfloor U(a_i, b_i) \rfloor, & \text{if } i = j \\ x_i, & \text{otherwise} \end{cases} \tag{2}$$

Multi-Uniform Mutation: Apply equation 2 to all of the variables in the parent

Non-Uniform Mutation: Randomly selects one variable, j, and sets it equal to an non-uniform random number based on equation 3. The new variable is equal to the old variable plus or minus a random displacement.

$$x'_i = \begin{cases} \lceil x_i + (b_i - x_i) f(G) \rceil & \text{if } r_1 < 0.5, \\ \lfloor x_i - (x_i + a_i) f(G) \rfloor & \text{if } r_1 \geq 0.5, \\ x_i, & \text{otherwise} \end{cases} \tag{3}$$

where

$$\begin{aligned} f(G) &= (r_2(1 - \tfrac{G}{G_{\max}}))^b, \qquad (4) \\ r_1, r_2 &= \text{ uniform random numbers between (0,1),} \\ G &= \text{the current generation,} \\ G_{\max} &= \text{the maximum number of generations,} \\ b &= \text{a shape parameter.} \\ \lceil x \rceil &= \text{the smallest integer greater than or equal to } x \end{aligned}$$

Multi-Non-Uniform Mutation: Apply equation 3 to all of the variables in the parent

Boundary Mutation: Randomly selects one variable, j, and sets it equal to either its lower or upper bound, where $r = U(0,1)$.

$$x'_i = \begin{cases} a_i, & \text{if } i = j, r < 0.5 \\ b_i, & \text{if } i = j, r \geq 0.5 \\ x_i, & \text{otherwise} \end{cases} \tag{5}$$

Simple Crossover: Generate a random number r from a discrete uniform distribution from 2 to $(m + n$-1) and create two new individuals ($\bar{X}'$ and $\bar{Y}'$) according to equations 6 and 7.

$$x'_i = \begin{cases} x_i, & \text{if } i < r \\ y_i, & \text{otherwise} \end{cases} \tag{6}$$

$$y'_i = \begin{cases} y_i, & \text{if } i < r \\ x_i, & \text{otherwise} \end{cases} \tag{7}$$

Arithmetic Crossover: Arithmetic crossover produces two complimentary linear combinations of the parents, where $r = U(0,1)$.

$$\bar{X}' = r\bar{X} + (1-r)\bar{Y} \tag{8}$$

$$\bar{Y}' = (1-r)\bar{X} + r\bar{Y} \tag{9}$$

To achieve the necessary integer representation of the variables, the following is performed.

$$\bar{X}' = (\langle ax_1 + by_1 \rangle, \langle ax_2 + by_2 \rangle, \langle ax_3 + by_3 \rangle, \langle ax_4 + by_4 \rangle, \langle ax_5 + by_5 \rangle) \tag{10}$$

where

$$\langle ax_i + by_i \rangle = \begin{cases} \lceil ax_i + by_i \rceil & \text{if } x_i > y_i \\ \lfloor ax_i + by_i \rfloor & \text{otherwise.} \end{cases}$$

Cell-Swap Crossover: The previous genetic operators work for any integer programming formulation. The next two operators only work with the cell formation representation. Let $\bar{X} = (x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n)$ and $\bar{W} = (w_1, w_2, \ldots, w_m, z_1, z_2, \ldots, z_n)$ be two $m+n$-dimensional cell formation individuals (parents)

$$x_i' = \begin{cases} x_i, & \text{if } i < m \\ z_i, & \text{otherwise} \end{cases} \tag{11}$$

$$w_i' = \begin{cases} y_i, & \text{if } i < m \\ w_i, & \text{otherwise} \end{cases} \tag{12}$$

Cell-Two-Point Crossover: Generate two random number r_1 and r_2 from a discrete uniform distribution from 2 to (m-1) and $(m+2)$ to ($m+n$-1), respectively and create two new individuals ($\bar{X}'$ and $\bar{Y}'$) according to equations 13 and 14.

$$x_i' = \begin{cases} clx_i, & \text{if } i < r_1 \\ w_i, & \text{if } i \leq m \\ y_i, & \text{if } m < i < r_2 \\ z_i, & \text{otherwise} \end{cases} \tag{13}$$

$$w_i' = \begin{cases} clw_i, & \text{if } i < r_1 \\ x_i, & \text{if } i \leq m \\ z_i, & \text{if } m < i < r_2 \\ y_i, & \text{otherwise} \end{cases} \tag{14}$$

B.5 Initialization:

The GA must be provided an initial population as indicated in step 1 of Fig. 11. The most common method is to randomly generate solutions for the entire population. However, since GAs can iteratively improve existing solutions (i.e., solutions from other heuristics and/or current practices), the beginning population can be seeded with potentially good solutions, with the remainder of the population being randomly generated solutions. The experiments in this paper used a random initial population with common random numbers across all methods for each replication.

B.6 Termination:

The GA moves from generation to generation selecting and reproducing parents until a termination criterion is met. The most frequently used stopping criterion is a specified maximum number of generations. Another termination strategy involves population convergence criteria. In general, GAs will force much of the entire population to converge to a single solution. When the sum of the deviations among individuals becomes smaller than some specified threshold, the algorithm can be terminated. The algorithm can also be terminated due to a lack of improvement in the best solution over a specified number of generations. Alternatively, a target value for the evaluation measure can be established based on some arbitrarily "acceptable" threshold. Several strategies can be used in conjunction with each other. The stopping criterion used in the experiments was a specified maximum number of function evaluations were performed.

B.7 Evaluation Functions:

Evaluation functions of many forms can be used in a GA, subject to the minimal requirement that the function can map the population into a totally ordered set. As stated, the evaluation function is independent of the GA (i.e., stochastic decision rules). The grouping efficacy measure described in Section 2 was used in these experiments. Modifications were made to the original computer function to incorporate the new models.

B.8 Genetic Algorithm Parameters

Table 2: Parameters Used in the Integer-Based Experiments

Parameter	Value
number of Boundary Mutation Operators	4
number of Uniform Mutation Operators	4
number of Multi-Uniform Mutation Operators	4
number of Non-Uniform Mutation Operators	4
number of Multi-Non-Uniform Mutation Operators	8
number of Cell Swap Crossover Operators	6
number of Cell Multi-point Crossover Operators	6
number of Arithmetic Crossover Operators	6
$k_{\max}$, the maximum permissible number of cells	†
q, the probability of selecting the best individual	0.08
$G_{\max}$, the maximum number of generations	1000
Population Size	80

† - Problem Specific