



CF-GGA: a grouping genetic algorithm for the cell formation problem

EVELYN C. BROWN^{†*} and ROBERT T. SUMICHRAST[‡]

In manufacturing, the machine-part cell formation (MPCF) problem addresses the issues surrounding the formation of part families based on the processing requirements of the components, and the identification of machine groups based on their ability to process specific part families. Past research has shown that one key aspect of attaining efficient groupings of parts and machines is the block-diagonalization of the given machine-part (MP) incidence matrix. This paper presents and tests a grouping genetic algorithm (GGA) for solving the MPCF problem and gauges the quality of the GGA's solutions using the measurements of efficiency (Chandrasekharan and Rajagopalan 1986a) and efficacy (Kumar and Chandrasekharan 1990). The GGA in this study, CF-GGA, a grouping genetic algorithm for the cell formation problem, performs very well when applied to a variety of problems from the literature. With a minimal number of parameters and a straightforward encoding, CF-GGA is able to match solutions with several highly complex algorithms and heuristics that were previously employed to solve these problems.

1. Introduction

Group technology deals with the formation of the machine groups and part families that make up the cells at a cellular manufacturing facility. Specifically, the machine-part cell formation (MPCF) problem addresses the issues surrounding the creation of part families based on component processing requirements, and the identification of machine groups based on their ability to process specific part families. When solving this problem, previous researchers have concluded that solution methodologies for the MPCF problem must focus attention on the block-diagonalization of the given machine-part (MP) incidence matrix. The best solutions to the MPCF problem are those that contain a minimal number of voids (zeros in the diagonal blocks) and a minimal number of exceptions (ones outside of the diagonal blocks).

The MPCF problem is NP-complete (Ballakur and Steudel 1987), and thus it is appropriate to examine heuristic solution approaches. A genetic algorithm (GA) is a heuristic solution technique that works by encoding a population of solutions to a given problem and manipulating these solutions through the use of operators such as crossover and mutation in an attempt to evolve superior solutions. Introduced by Emmanuel Falkenauer (1992), a grouping genetic algorithm (GGA) is a specialized GA that has been adapted to handle the specific structure of grouping problems. The

Revision received April 2001.

[†] Management Science & Information Technology, 1007 Pamplin Hall (0235), Virginia Tech, Blacksburg, VA 24061, USA.

[‡] Pamplin College of Business, 1044 Pamplin Hall (0209), Virginia Tech, Blacksburg, VA 24061, USA.

* To whom correspondence should be addressed. e-mail: evebrown@vt.edu

GGA can handle the structure of the MPCF problem, and is able to produce highly efficient solutions to a variety of sample problems from the literature.

In order to compare multiple solutions to the same MPCF problem, a measure of effectiveness must be selected. In this study, solutions are evaluated in terms of their grouping efficiency (Chandrasekharan and Rajagopalan 1986a) or their grouping efficacy (Kumar and Chandrasekharan 1990). Since both of these measurements are well known in the MPCF literature, it is possible to compare the GGA's performance to that of other documented methodologies using sample problems from previous research. Empirical tests demonstrate the GGA's ability to determine efficient solutions across a spectrum of MPCF problems.

Our investigation begins with the application of our cell formation grouping genetic algorithm (CF-GGA) to a small sample of MPCF problems from the literature. Once success with CF-GGA is achieved, simulated problems are generated in order to fine-tune the algorithm's key parameters. Using appropriate parameter settings, CF-GGA is then applied to numerous other problems from the literature (Carrie 1973, King 1980b, Chandrasekharan and Rajagopalan 1986a, 1986b, 1989). Based on grouping efficiency, the average percentage improvement of CF-GGA over ZODIAC (Chandrasekharan and Rajagopalan 1987) is 17%. Based on grouping efficacy, the average percentage improvement of CF-GGA, when compared with the small set of algorithms presented in Srinivasan (1994), is 43% above ZODIAC, 2.85% above GRAFICS, and 1.5% above Srinivasan's algorithm.

This paper is organized as follows. Sections 2 and 3 provide background on the MPCF problem and the GGA solution methodology. Section 4 details the development of CF-GGA. Section 5 reports the results when CF-GGA is applied to problems from the literature, and section 6 analyses CF-GGA's performance in comparison to other noted MPCF solution methodologies. Conclusions are presented in section 7.

2. Problem definition and literature review

According to Burbidge, group technology is 'an approach to the organisation of work in which the organisational units are relatively independent groups, each responsible for the production of a given family of products' (Burbidge 1979, p. 36). Although this definition is over 20 years old, the focus of group technology remains the same. One of the key issues is determining the best formation of the separate manufacturing cells so that production can flow as efficiently as possible. This is called the machine-part cell formation (MPCF) problem.

The machine-part cell formation problem includes the identification of parts that have similar processing requirements (a part family) and the identification of the set of machines that can process each family of parts. For cells to operate efficiently, all of the machines within a cell should be fully utilized and the amount of intercell traffic should be kept to a minimum.

In order to determine the utilization of machines and the intercell flow of parts, much research has focused on the machine-part incidence matrix. A member of this matrix, a_{ij} , has a value of 1 if machine i is required by part j , and a value of 0 otherwise. When the columns and rows are arranged in the order corresponding to the groups identified by a MPCF solution, the machine-part incidence matrix can be evaluated to determine the efficiency of the solution.

The work of Chandrasekharan and Rajagopalan (1986a) utilizes an efficiency measure that allows for the normalized weighting of machine utilization and intercell

traffic. Later work by Kumar and Chandrasekharan (1990) produces a measurement called efficacy that attempts to improve on the efficiency measure by incorporating the size of the MP matrix into its formula. Both of these measurements have been used by other researchers attempting to solve the MPCF problem (Srinivasan 1994, Joines *et al.* 1996).

The machine-part cell formation problem is formally defined by Burbidge (1963). His original work focuses on production flow analysis in seeking block-diagonalization of the machine-part incidence matrix. Burbidge also develops several heuristic methods for small problems, and this research area is extended by others.

Carrie employs numerical taxonomy in his attempt to improve upon the earlier component-machine matrix method of production flow analysis. In his words, 'numerical taxonomy provides algorithms for the study of similarities between objects in a quantitative manner (Carrie 1973, p. 400)'. Its three stages include preparing the data matrix, computing a similarity coefficient matrix, and performing cluster analysis. Carrie's algorithm applies these stages of numerical taxonomy twice, first to classify the components and then to classify the machines. The data of the MP matrix are examined in order to calculate the similarity matrix. Cluster analysis is then performed. The data files Carrie uses include details regarding the machines, and this allows for detailed examination of the processing requirements of the components. Carrie's research demonstrates a method for efficiently processing machine and component requirements for real examples from industry, rather than for hypothetical examples from past research. The three problems Carrie tests are employed by other researchers (Srinivasan 1994, Joines *et al.* 1996) who wish to test their methodologies on practical examples.

In the 1980s, King developed a rank order clustering (ROC) method that attempted to diagonalize the MP matrix. King's algorithm begins by interpreting, for each row, the pattern of entries in the MP matrix as a binary word. Next, the rows are ranked in order of decreasing binary value. If the row order and rank order of the MP matrix are the same, the algorithm terminates. If not, the rows are rearranged in decreasing rank order, and the columns are ranked in order of decreasing binary value. If the column order and rank order of the matrix are identical, the process stops; otherwise, the columns are rearranged in decreasing rank order and the entire process repeats. The algorithm will, 'in a finite number of steps, produce a matrix in which both the columns and rows are arranged in order of decreasing value when read as binary words' (King 1980b, p. 219). King points out that, although other approaches such as single linkage cluster analysis (McAuley 1972) and the bond energy method (McCormick *et al.* 1972) are capable of handling the MPCF problem, ROC demonstrates the advantage of being able to deal with exceptional elements and bottleneck machines more easily than the other methods. As an extension of ROC, King and Nakornchai (1982) developed ROC2 to handle better the limitations imposed by the storage and sorting techniques of ROC.

Chandrasekharan and Rajagopalan (1986b) present a modified version of the ROC algorithm, which they call MODROC. Later, these same two researchers attempted to improve upon earlier versions of the ideal seed method with their algorithm ZODIAC, which stands for zero-one data: ideal seed algorithm for clustering (Chandrasekharan and Rajagopalan 1987).

Srinivasan (1994) presents a clustering algorithm that utilizes minimum spanning trees to solve the MPCF problem. Efficacy is used to measure the goodness of solutions. Srinivasan compares his proposed methodology to the earlier techniques,

ZODIAC (Chandrasekharan and Rajagopalan 1987) and GRAFICS (Srinivasan and Narendran 1991). His work includes extensive testing on a variety of problems from the literature.

A combinatorial search approach is proposed by Vakharia and Chang (1997). Their heuristic methods are grounded in simulated annealing and tabu search. With the simulated annealing approach, the authors simultaneously investigate minimizing equipment investment and materials handling costs. Success is achieved for both randomly generated problems and problems from the literature. Their tabu search approach is viable, but the solution quality and computation time are inferior to those achieved by the simulated annealing approach.

Other researchers have examined the application of heuristic approaches such as simulated annealing and tabu search, including: Sofianopoulou (1997); Su and Hsu (1998); Lozano *et al.*, 1999. Another heuristic approach demonstrating success with the MPCF problem is genetic algorithms (Joines *et al.* 1996). A review of the applications of these types of solution techniques can be found in Venugopal (1999).

The recent shift toward heuristic solutions to the MPCF problem is due, in part, to the fact that the problem is NP-complete (Ballakur and Steudel 1987). With heuristic approaches showing promise in this area, our research focuses on the application of a grouping genetic algorithm to the MPCF problem.

3. Grouping genetic algorithms

3.1. Genetic algorithms

Genetic algorithms are heuristic search techniques that utilize analogies to natural selection and survival of the fittest. They employ a population of solutions, combining those solutions in specific ways in an attempt to form better solutions. Introduced by John Holland (1975), genetic algorithms have become a popular solution methodology for a variety of complex problems (Brown 1996).

The population of solutions with which a genetic algorithm (GA) works is comprised of encodings, known as chromosomes. Individual elements of the chromosomes are called genes. Based on the objective function of the problem at hand, each chromosome is evaluated and given a fitness score. As an example, consider a possible encoding for the problem of grouping five manufactured items into part families. The gene value at location i in the chromosome represents the group to which item i is assigned. The chromosome ACBBA represents a solution that places items one and five together in a group, items three and four together in a group, and item two in a group by itself. The fitness of this chromosome depends on the objective function to be optimized.

Using operators such as crossover and mutation, the GA combines pairs of chromosomes (parents) to produce new solutions (offspring) that seek to have the advantages of both parents. While crossover works to exploit promising areas of the search space, mutation attempts to recover lost material and explore new regions of the search space. Using a population of chromosomes, the goal of a GA is continually to produce offspring whose fitness values are better than the fitness values of their parents. Although optimality cannot be guaranteed, GAs have been shown to achieve a relatively high level of performance across a broad spectrum of problems (Alendar 1994).

The GA operators used and the details of these operators vary from implementation to implementation. One of the simplest and most commonly used operators is called single-point crossover. To illustrate this operator, consider the previously

described chromosome ACBBA and a similar chromosome BBAA. Under standard single point crossover (Goldberg 1989), a cross-point is first randomly chosen. The first offspring is generated by taking the section to the left of the cross-point from parent one and appending it with the section to the right of the cross-point from parent two. Similarly, the second offspring is created by taking the section to the left of the cross-point from parent two and appending it with the section to the right of the cross-point from parent one. Essentially, a crossing over of the two sections of each parent takes place to create two offspring. With the | symbol representing the randomly generated cross-points, the result of standard single-point crossover would be as follows:

parent one: ACB BA	offspring one: ACBAA
parent two: BBC AA	offspring two: BBCBA

Without further problem-specific information, we cannot be certain that the offspring produced with this crossover operator represent valid solutions.

Standard mutation occurs by altering the value of a randomly chosen element of a chromosome. To illustrate the mutation operator, consider the results from the earlier crossover example, where the offspring was ACBAA. If the fourth gene is randomly selected for alteration, one possible revised offspring would be ACBCA. This altered chromosome is likely to have a different fitness value than that of the original offspring. Again, the validity of the revised offspring chromosome becomes an issue, as the post-mutation chromosome is not guaranteed to represent a feasible solution. Further details regarding genetic algorithms may be found in Holland (1975), Goldberg (1989), and Michalewicz (1992).

3.2. GGA – background

The grouping genetic algorithm, introduced by Emanuel Falkenauer (1992), is a genetic algorithm designed to handle the special structure of grouping problems. Falkenauer developed GGAs after realizing several significant drawbacks of using classical GAs for grouping problems. Falkenauer (1998) points out that others have attempted to minimize the troubles associated with applying GAs to grouping problems by specializing the standard operators, but this still results in various shortcomings.

In his text, *Genetic Algorithms and Grouping Problems*, Falkenauer (1998) details what he perceives as the drawbacks that arise from applying the classical GA to grouping problems. He points out that the standard encoding scheme is highly redundant in the instance of a grouping problem. For example, chromosomes ABCB and BCAC both represent a solution that groups the second and fourth items together and places the first and third items in separate, solo groups. This type of repetition greatly detracts from the usefulness of the encoding scheme and is in conflict with the principle of minimal redundancy (Radcliffe 1991). If such a chromosome representation is employed, the size of the search space greatly increases, hindering the performance of the GA.

Falkenauer also points out problems arising from the application of classical GA operators to standard encodings of grouping problems. In his words, ‘the straightforward encoding leads to the highly undesirable effect of casting context-dependent information out of context under standard crossover’ (Falkenauer 1998, p. 87). The process of recombination, or crossover, can easily result in a solution chromosome

that shares none of the qualities of the parents that produced it. For example, applying single-point crossover to the pair of parents representing identical solutions to a grouping problem (AB|CB and BC|AC) results in offspring that may not even be valid solutions.

Just as there exist problems with applying the standard encoding scheme and crossover operator, problems may arise with the standard mutation operator as the GA nears a good solution (Falkenauer 1998). Standard mutation can become too disruptive by injecting new group values into a successful chromosome, greatly reducing the solution's fitness and possibly causing its removal from the population.

With these drawbacks in mind, Falkenauer's GGA proposes a new encoding scheme. When accompanied by revised crossover and mutation operators, this chromosome representation can help produce high quality solutions to a broad spectrum of grouping problems (Falkenauer 1998).

The GGA varies from the standard GA in several ways. First, the encoding scheme includes an augmented chromosome. The groups are encoded on a one gene for one group basis, and this separate encoding is appended to the standard chromosome. Second, the genetic operators work with the group section of the chromosome, altering the composition of the groups. This, in turn, leads to alteration of the main chromosome. Third, the GGA operators of crossover and mutation do not function in the same manner as the classical operators.

3.3. *GGA encoding*

Falkenauer's GGA encoding scheme includes an additional group section that lists the groups included in the main chromosome. This modification is necessary since, as Falkenauer points out, 'the cost function of a grouping problem depends on the groups, but there is no structural counterpart given in their (standard) chromosome' (Falkenauer 1998, p. 98). As an example, chromosome ABCB is appended with a group section such as BAC. Any ordering of the groups is valid for the group section of the chromosome, as long as all of the groups utilized in the chromosome are represented. Since chromosomes of equal size may be composed of various numbers of groups, Falkenauer's GGA allows for variable length chromosomes. The key aspect of this encoding scheme is that the additional group section is what the GGA operators manipulate.

3.4. *GGA operators*

Under standard crossover, portions of each parent chromosome are exchanged to form offspring. When dealing with grouping problems, standard crossover can result in an overlapping among the groups of the parents. In other words, an offspring chromosome may contain one or more groups that share the same contents. Since each item may only be assigned to one group, this implies that the resulting offspring are invalid. To avoid the creation of invalid chromosomes, Falkenauer (1998) proposes the following five-step crossover operator.

- Step* 1. Select two cross-points from the group portion of each parent chromosome.
- Step* 2. Inject the cross-section of the second parent at the first cross-point of the first parent.
- Step* 3. For all of those items now occurring twice, remove them from the groups they were members of in the first parent. This means that the groups coming

from the first parent may be altered and no longer contain the same objects they contained originally.

Step 4. If needed, apply local problem-dependent heuristics in order to adapt the resulting groups. These adaptations will depend upon the constraints and objective function of the problem.

Step 5. Apply steps 2 – 4 to the pair of parents with their roles reversed.

Whenever this type of crossover results in components or machines being in more than one group, the components and machines are left in the groups to which they belonged in the injected portion. In other words, the injected section takes precedence with this type of crossover. When there is duplication caused by crossover, there may also be displacement of parts and/or machines. If removal of duplicate parts leads to a group containing no machines or no parts, then the corresponding parts and machines are considered to be displaced. Determining appropriate problem-specific heuristics to replace these parts and machines is a vital step in the GGA's crossover. The details of CF-GGA's replacement heuristic are given in section 4.2.4.

The goal of the mutation operator is to explore new areas of the search space. One way to achieve this is randomly to select one gene from an offspring chromosome and alter it. This alteration may or may not improve the fitness of the solution chromosome. To keep promising chromosomes intact, the mutation operator is generally applied to only a small percentage of the genes in a population (DeJong 1975).

Classical mutation may cause dilemmas when applied to the standard encoding. If the GGA has been successful in grouping components and machines into groups that operate efficiently, then this type of alteration may be too disruptive, as it will move a component or machine to a different group. If a component or machine is randomly transferred into a group where it has no similarities with the other members, the quality of the solution may be greatly affected.

Falkenauer (1998) proposes three basic mutation strategies for his GGA. The first involves creating a new group, the second entails eliminating an existing group, and the third requires shuffling a few randomly chosen objects among their respective groups. With any of these choices, problem-dependent heuristics must be employed to determine the reassignments that must take place.

4. Developing a grouping genetic algorithm for the MPCF problem

4.1. Encoding of the chromosome

When designing CF-GGA, we selected an encoding that fits naturally with the MPCF problem. In particular, the chromosome representation consists of three sections: the component section, the machine section, and the group section. The following notation is used for chromosome representation in CF-GGA:

$$c_1 c_2 c_3 \dots c_P \mid m_1 m_2 m_3 \dots m_M, \mid g_1 g_2 g_3 \dots g_G,$$

where

- c_i group to which component i is assigned, $i = 1$ to P , $1 \leq c_i \leq G$,
- m_j group to which machine j is assigned, $j = 1$ to M , $1 \leq m_j \leq G$,
- g_k an existing group number, each listed once, $k = 1$ to G ,
 $G = \max\{m_1, m_2, m_3, \dots, m_M\}$,
- P number of components in the problem,
- M number of machines in the problem,

G number of groups in the solution, $G \leq \min(M, P)$.

In the component section, each c_i may take on any value in the set of groups $\{g_1, g_2, g_3, \dots, g_G\}$. Similarly, in the machine section, each m_j may take on any value in the set $\{g_1, g_2, g_3, \dots, g_G\}$. The group section consists of any ordering of the set of groups $\{g_1, g_2, g_3, \dots, g_G\}$.

Note that the lengths of the component and machine sections are both constant for all chromosomes and depend on the number of components and machines, respectively, in the problem. The group section, however, may vary in length, and depends upon the number of groups into which the components and machines are divided.

To understand the encoding better, consider the problem of grouping 11 parts and 6 machines into cells. Under the assumption of no outside problem constraints, one solution to this problem is the chromosome 2 3 1 4 3 3 2 4 1 1 4 | 4 1 2 3 3 1 | 4 1 2 3. In this chromosome, components 1 and 7 are assigned to group 2. In addition, machine 3 is part of group 2. Relating this more directly to the MPCF problem, a cell is designed with machine 3 processing components 1 and 7. Similarly, components 2, 5, and 6 are in a cell with machines 4 and 5; components 3, 9 and 10 are in a cell with machines 2 and 6; and components 4, 8 and 11 are in a cell with machine 1.

4.2. CF-GGA operators

4.2.1. Selection

The type of selection that CF-GGA employs is termed *rank-based roulette-wheel selection*. With a roulette-wheel strategy (Goldberg 1989), each chromosome is assigned a portion of an imaginary roulette wheel, based on the chromosome's calculated fitness value. Chromosomes that are more fit and thus have higher objective function values receive a larger proportion of the roulette wheel. A random number between 0 and 1 is generated and the chromosome occupying the portion of the roulette wheel covered by the randomly generated percentage is selected to be a parent. A problem arises with standard roulette-wheel selection if the difference between the best and worst fitness values is small. If this occurs, each chromosome receives approximately the same share of the roulette wheel, and selection cannot contribute to 'survival of the fittest'. A different problem occurs if a single solution is much better than the rest of the population. This solution may be selected too frequently at the exclusion of other chromosomes and effectively eliminate much of the population diversity.

One suitable adjustment is to use rank-based fitness values to assign proportions of the roulette wheel to each chromosome. In order to do this, Reeves' (1995) formula is employed to determine the new fitness score for each chromosome. First, the chromosomes are ranked in order from worst (rank of 1) to best (rank of M). Then, according to Reeves' formula, a chromosome of rank r is assigned a fitness score of $2r/(M(M+1))$, where M is the number of chromosomes ranked. An adjusted roulette wheel is then created based on the revised fitness scores, and selection follows the strategy described above.

To illustrate the differentiating power of the rank-based strategy, consider two chromosomes, `chrom_one` with a fitness score of 0.4 and `chrom_two` with a fitness score of 0.5. Under standard roulette-wheel selection, the roulette wheel has a total value of $0.4 + 0.5 = 0.9$. `Chrom_one` occupies $0.4/0.9 = 44.44\%$ of the wheel and `chrom_two` occupies $0.5/0.9 = 55.56\%$ of the wheel. If these same two chromosomes

cross-section always consists of at least one group. In this example, the cross-section of parent one consists of groups 1 and 4. For parent two, the cross-section consists only of group 3.

parent one: 3 | 1 4 | 2

parent two: | 3 | 2 1

To create the first offspring, the cross-section of parent two is inserted after the first crosspoint of parent one. The result is the following new chromosome. Note that the second 3 is underlined to signify that it is part of the inserted section, not part of the original parent.

offspring one: 3 3 1 4 2

Now the composition of each group of offspring one is listed, with braces used to separate the components listing from the machines listing. For example, group 3 includes components 1, 7, and 9, as well as machine 1.

group 3	{1, 7, 9}, {1}
group <u>3</u>	{2, 6, 10}, {1, 3}
group 1	{3, 4, 10}, {2, 4}
group 4	{2, 6, 11}, {3}
group 2	{5, 8}, {5, 6}

Note that five items now occur twice: component 2, component 6, component 10, machine 1 and machine 3. Following Falkenauer's steps for crossover, we now remove the duplicates from the groups they were in as a part of parent one. Thus, the injected section 3 remains intact and the other groups are subject to alteration. The result of this step is the following:

group 3	{1, 7, 9}	[machine 1 removed]
group <u>3</u>	{2, 6, 10}, {1, 3}	[unaltered]
group 1	{3, 4}, {2, 4}	[component 10 removed]
group 4	{11}	[components 2 and 6 removed, and machine 3 removed]
group 2	{5, 8}, {5, 6}	[unaltered]

The solution resulting from this process may be infeasible, as this example illustrates. As can be seen, components 1, 7, 9 and 11 are assigned to groups with zero machines. A necessary condition for feasibility is that each group contains at least one component and one machine. Infeasible solutions must be altered using a problem-specific heuristic.

For CF-GGA, the heuristic works to relocate components 1, 7, 9 and 11 into one of the remaining groups (3, 1 or 2 for this example). The cell into which a particular component is placed is not selected randomly. Rather, an attempt is made to insert each displaced component into the cell that contains the most machines that the component requires. The replacement heuristic utilizes the information contained in the MP matrix to place a component into the group containing the most machine(s) it needs. Ties are broken arbitrarily. Similarly, when crossover results in a group containing no components, those groups are eliminated and each displaced machine

is placed into the existing cell that contains the most components that need it, based on the information contained in the MP matrix. Again, ties are broken arbitrarily.

Since the order in which displaced components and machines are reassigned may have an impact on succeeding reassignments, the set of displaced items is randomly ordered before any reassignments are made. The steps of the replacement heuristic follow.

Replacement Heuristic for CF-GGA

- Step 1.* Create a randomly ordered list of all displaced components and machines.
- Step 2.* Determine what type of item is at the top of the list (component = go to step 3, machine = go to step 7).
- Step 3.* For displaced component j , examine column j of the original MP matrix and make a list of all rows i in that column which contain the value 1.
- Step 4.* Examine the existing groups that resulted from crossover (and may have been altered by this heuristic already) and determine which of them contains the most machines from the list created in step 3.
- Step 5.* Place displaced component j in that group (ties are broken arbitrarily).
- Step 6.* If displaced components or machines still exist, return to step 2, otherwise stop.
- Step 7.* For displaced machine i , examine row i of the original MP matrix and make a list of all columns j in that row which contain the value 1.
- Step 8.* Examine the existing groups that resulted from crossover (and may have been altered by this heuristic already) and determine which of them contains the most components from the list created in step 7.
- Step 9.* Place displaced machine i in that group (ties are broken arbitrarily).
- Step 10.* If displaced components or machines still exist, return to step 2, otherwise stop.

Use of this replacement heuristic helps to increase the machine utilization and decrease the intercell traffic. Since these are the two goals when attempting to achieve group efficiency, they fit logically as goals for the replacement heuristic.

Continuing with the example, to insert component 1, which has been displaced, we refer to the given MP matrix from figure 1. From this figure, we see that component 1 requires machines 5 and 6. This being the case, component 1 is inserted onto group 2, since group 2 currently contains both of the machines that component 1 needs. When replacing component 7, the MP matrix indicates component 7 requires machine 2 only. In this case, we place component 7 in group 1 because that is the group in which machine 2 resides. Lastly, the MP matrix shows that components 9 and 11 both require machines 3 and 4. We are forced to choose between group 3 and group 1 for insertion of each of these components. Each group is given equal weight, and a random number is generated to determine which cell each of these last two displaced components is assigned. Assuming that component 9 is assigned to group 3 and component 11 is assigned to group 1, then the groups are:

group <u>3</u>	{2, 6, 10, 9}, {1, 3}
group 1	{3, 4, 7, 11}, {2, 4}
group 2	{5, 8, 1}, {5, 6}

This leads to the resulting offspring one: $2 \underline{3} 1 1 2 \underline{3} 1 2 \underline{3} \underline{3} 1 \mid \underline{3} 1 \underline{3} 1 2 2 \mid \underline{3} 1 2$. Even though parent one consisted of four cells, the result of crossover here is an offspring with only three cells.

The process is repeated for the second offspring, with each of the steps carried out in the same manner. Offspring two is originally formed by inserting the cross-section of parent one at the first crosspoint of parent two. In our case, offspring two originates as: $\underline{1} \underline{4} 3 2 1$.

4.2.3. Mutation

The mutation operator explored in CF-GGA consists of eliminating one of the groups of a chromosome. The group to eliminate is chosen randomly, but intelligence is built into the reassignment heuristic. Each of the displaced components is placed into a new group using information obtained from the problem's original MP matrix. The placement is based on which existing group has the most machines that the displaced component requires. If a tie occurs, it is broken arbitrarily. Likewise, displaced machines are reassigned based on which existing group has the most components that need it. Again, ties are broken arbitrarily.

4.2.4. Replacement

CF-GGA uses a generational replacement strategy. This means that with the exception of the best solution, the entire current population is replaced with offspring at the end of each generation. With CF-GGA, the best chromosome from each generation is always passed to the next generation. The remainder of the chromosomes chosen to replace their predecessors are selected using the rank-based roulette-wheel strategy described in section 4.2.1.

4.3. CF-GGA Parameters

The values assigned to various genetic algorithm parameters can have a significant impact on the algorithm's performance. When designing CF-GGA, we employ additional experimentation in order to determine appropriate values for the parameters of population size, crossover rate, and mutation rate. A small set of simulated problems is created by randomly generating MP matrices of varying size and sparsity. Experimentation is performed using three different population sizes (50, 100, 200), two potential crossover rates (1.0, 0.6), and two possible mutation rates (0.0, 0.25). For sample problems with small MP matrices (10×10), most combinations of parameter values find the best solution in less than 10 generations. For larger problems (25×40 , 25×100 , 40×100), however, the combination of population size 100, crossover rate 1.0, and mutation rate 0.0 results in the highest quality solutions. The inclusion of the mutation operator greatly increases the run time of the algorithm without significant improvement in solution quality. For this reason, mutation is not utilized when CF-GGA is tested on problems from the literature (i.e. the mutation rate is set to 0.0).

4.4. Initialization and termination

4.4.1. Initial population

The initial population employed by CF-GGA is created by first randomly generating values for each gene in the machine section of the chromosome. These values are constrained to be between 1 and G , where G is the maximum number of groups for the problem, $G \leq \min(M, P)$. After creation of the machine section, random

values are then generated for the components section of the chromosome. These values are not only restricted to be between 1 and G , but must also come from the set of values assigned to the machine section. By assigning the initial values in this manner, all of the initial chromosomes are valid solutions for the MPCF problem. Because each value contained in the set $\{c_1, c_2, c_3, \dots, c_P\}$ is also contained at least once in the set $\{m_1, m_2, m_3, \dots, m_M\}$, there are no groups with zero components or zero machines in the initial population of chromosomes. By ensuring these problem constraints are satisfied, and assuming no other constraints exist, CF-GGA begins with an initial feasible population.

4.4.2. Termination criteria

The most straightforward stopping criteria for a GGA is the number of generations. Unfortunately, there is no a priori method for determining how many generations is enough. However, as Radcliffe points out, using a population size of s with a GA that runs for t generations requires st evaluations. If st is not significantly less than N , where N is the size of the search space, then one may as well employ enumeration to solve the problem (Radcliffe 1996). For this reason, and also because preliminary tests indicate CF-GGA is capable of finding high quality solutions quickly, the limit on the number of generations is set to 50. CF-GGA monitors the quality of its solutions and may also stop once t consecutive generations pass by with no improvement. For this research, t is set to 25. Thus, CF-GGA terminates after 50 total generations or after 25 generations with no improvement, whichever comes first.

4.5. Measures of effectiveness

4.5.1. Grouping efficiency

In order to gauge the performance of CF-GGA, two measures of effectiveness are employed. The first, grouping efficiency, is formally defined by Chandrasekharan and Rajagopalan. They develop this measure to 'provide a quantitative standard on a rational scale for comparing different solutions to the same problem' (Chandrasekharan and Rajagopalan 1986a, p. 456). For the MPCF problem, the quality of a solution depends on machine utilization and intercell movement. Many solution techniques attempt to diagonalize the MP matrix in an effort to maximize machine utilization and minimize intercell traffic simultaneously. When a matrix representation of a solution has zeros on the diagonal, this indicates a machine is not employed by one of the components in the group. When the solution matrix has ones off of the diagonal, this represents intercell movement. Chandrasekharan and Rajagopalan (1986a) propose a measurement of efficiency, η , that is based on these concepts.

The grouping efficiency, η , is calculated as a weighted average of η_1 and η_2 , using the formula:

$$\eta = q\eta_1 + (1 - q)\eta_2, \quad (1)$$

where $0 \leq q \leq 1$, and

$$\eta_1 = \frac{e_d}{\sum_{r=1}^k M_r N_r} \quad (2)$$

$$\eta_2 = 1 - \left[\frac{e_o}{mn - \sum_{r=1}^k M_r N_r} \right], \quad (3)$$

where

- e_d total number of ones in the diagonal blocks,
- e_o total number of ones in the off-diagonal blocks,
- k limiting number of groups,
- m number of machines (rows),
- n number of components (columns),
- M_r number of machines in the r th cell,
- N_r number of components in the r th cell.

The formula for η_1 expresses the ratio of the number of non-zero elements in the diagonal blocks to the total number of elements in the diagonal blocks. The closer η_1 is to 1.0, the more likely that the machine utilization in the cell is close to 100%.

The expression for η_2 represents the ratio of the number of zeros in the off-diagonal sections to the total number of elements in the off-diagonal sections. Values of η_2 that are close to 1.0 indicate minimal intercell traffic. Incorporation of the weighting factor q 'enables the analyst to alter the emphasis between utilization and intercell movement, depending on the specific requirements of the given problem' (Chandrasekharan and Rajagopalan 1986a, p. 457).

4.5.2. Group efficacy

The research of Kumar and Chandrasekharan (1990) develops another method for measuring the quality of different solutions to a given MPCF problem. Their measure, termed *grouping efficacy*, considers the 'number of operations' to be the number of ones in the original MP matrix. With this in mind, ψ is defined as the ratio of exceptional elements to the number of operations, and ϕ is defined as the ratio of voids to the number of operations. The formula for the grouping efficacy is given as:

$$\Gamma = \frac{1 - \varphi}{1 + \phi} = \frac{1 - e_o/e}{1 + e_v/e} = \frac{e - e_o}{e + e_v} = 1 - \frac{e_o + e_v}{e + e_v}, \quad (4)$$

where

- e number of operations,
- e_v number of voids,
- e_o number of exceptions.

Whereas the formula for grouping efficiency allowed for equal weighting of the ratio involving exceptional elements and the ratio involving voids (by setting $q = 0.5$), the formula for grouping efficacy does not allow for this. 'As the efficacy function is not symmetric with references to ψ and ϕ , the influence of exceptions and voids are not identical' (Kumar and Chandrasekharan 1990, p. 239). This is an intentional variation from the efficiency formula because these authors believe that, in real life, exceptional elements are much more significant than voids.

5. Application of a GGA to MPCF problems from the literature

Initially, CF-GGA is applied to a small set of randomly generated problems so that fine-tuning of algorithmic parameters can occur. Once the population size (100) and crossover rate (1.0) are set, CF-GGA is applied to a subset (6×11) of a small problem from the literature (Chandrasekharan and Rajagopalan 1986a). The optimal grouping efficiency score is obtained in only five generations.

CF-GGA is then applied to the entire small problem taken from Chandrasekharan and Rajagopalan (1986a). Again, for this small 8×20 problem, CF-GGA obtains the optimal presented in the literature very quickly (seven generations). To continue the initial testing, a larger problem (50×22) is used (Seifoddini and Tjahjana 1999). Although no grouping efficiency score is given for this problem, CF-GGA finds a solution with an efficiency of 0.966 after only 20 generations.

CF-GGA's success on these few problems from the literature provides confidence that the algorithm can perform well across a spectrum of example MPCF problems. Six data sets are taken from an article by Chandrasekharan and Rajagopalan (1989). For each of these data sets, the authors report the best grouping efficiency that their algorithm ZODIAC (Chandrasekharan and Rajagopalan 1987) obtains. CF-GGA achieves an efficiency score of 100% for the first data set, and this matches the score obtained by Chandrasekharan and Rajagopalan. For the remaining five data sets, however, CF-GGA outperforms ZODIAC in all cases. The efficiency scores reported for ZODIAC and those obtained with CF-GGA are given in table 1. In all cases, CF-GGA obtains its best grouping efficiency score in 21 generations or less.

With grouping efficiency as the measure of effectiveness, CF-GGA is very successful, obtaining solutions that are as good as, or better, than those presented in the literature. To test the algorithm further, the objective function measure is changed to grouping efficacy, but no other changes are made to the algorithm or its parameters. Srinivasan (1994) presents a table of grouping efficacy scores for a set of MPCF problems. We apply CF-GGA to several of these problems with the following results.

Table 2 presents the results for the six data sets (CR1-CR6) used previously from Chandrasekharan and Rajagopalan (1989), using the efficacy scores of ZODIAC, GRAFICS (Srinivasan and Narendran 1991) and a proposed algorithm, as they are presented in Srinivasan (1994). Also included in table 2 are the results when each of the four solution methods is applied to two data sets (C01, C02) from Carrie (1973), two other data sets (CRA, CRB) from Chandrasekharan and Rajagopalan (1986a, 1986b), and one data set (K01) from King (1980b). The column labelled Joines presents the results obtained by Joines *et al.* (1996) when applying his genetic algo-

Data set	Size	ZODIAC	CF-GGA
CR1	24×40	100%	100%
CR2	24×40	95.20%	97.54%
CR3	24×40	91.14%	96.47%
CR4	24×40	77.31%	95.69%
CR5	24×40	72.43%	95.60%
CR6	24×40	69.33%	95.70%

Table 1. Grouping efficiency comparison of ZODIAC and CF-GGA.

Data set	Size	ZODIAC	GRAFICS	Srinivasan (1994)	Joines	CF-GGA
CR1	24 × 40	100.00	100.00	100.00	NA	100.00
CR2	24 × 40	85.11	85.11	85.11	85.11	85.11
CR3	24 × 40	73.51	73.51	73.51	73.51	73.29
CR4	24 × 40	20.42	43.27	51.81	NA	48.98
CR5	24 × 40	18.23	44.51	44.72	NA	46.81
CR6	24 × 40	17.61	41.67	44.17	NA	44.14
C01	24 × 18	41.84	48.91	44.20	57.01	52.38
C02	20 × 35	75.14	75.14	75.14	77.91	77.91
CRA	8 × 20	85.25	85.25	85.25	85.25	85.25
CRB	8 × 20	58.33	58.13	58.72	NA	56.88
K01	16 × 43	53.76	54.39	54.44	NA	53.70

Table 2. Grouping efficacy comparison of four solution methodologies.

ithm to some of the problems listed. NA is inserted in instances where results from Joines *et al.* are not available.

6. Analysis of performance

Using the grouping efficiency measure, CF-GGA outperforms ZODIAC on five of the six data sets, and matches its performance on the sixth. The average percentage improvement of CF-GGA over ZODIAC is 17%. The solutions created by CF-GGA using efficiency as the objective function were analysed to gain an understanding of why they were superior to solutions created by ZODIAC.

With the efficiency measure, voids and exceptional elements are weighted equally by setting q to have a value of 0.5. However, the proportion of the MP matrix where voids may occur is much smaller than the proportion where exceptions may occur, in most situations. CF-GGA realizes this and finds optimal solutions by creating η_1 values very close to 1.0. It does this by ensuring the diagonal blocks contain as many ones as possible, without worrying about an increase in exceptional elements. Since the off-diagonal portion of the MP matrix is generally much larger than the diagonal portion, the number of exceptions will be divided by a larger denominator. This means that a single exceptional element has less of an impact on the efficiency rating than does a single void.

Kumar and Chandrasekharan recognized this shortcoming of the efficiency measurement and created grouping efficacy to gauge more accurately the solution quality for MPCF problems. With efficacy as the measure of effectiveness, CF-GGA shows an average percentage improvement of 43% over ZODIAC, 2.85% over GRAFICS, and 1.5% over the proposed algorithm of Srinivasan. Although not applied to the entire set of test problems, Joines' genetic algorithm outperforms CF-GGA by 8.1% on one data set (C01) and by 0.3% on another (CR3).

Joines' algorithm employs seven genetic operators, including two that are 'highly problem-specific crossover operators' (Joines *et al.* 1996, p. 75). With CF-GGA, the traditional grouping genetic algorithm crossover is the only operator utilized, indicating the potential robustness of CF-GGA.

Table 3 indicates that Joines, while performing slightly better than CF-GGA on two of five test problems, requires many more generations to reach solutions. CF-GGA's success in achieving high quality solutions so quickly can be attributed, in part, to the intelligence built in to the replacement heuristic.

Data set	Joines GA	CF-GGA	Improvement (Joines over CF-GGA)
CR3	1789	14	0.3%
C01	986	14	8.1%
C02	1785	11	0.0%
CRA	119	6	0.0%

Table 3. Number of generations comparison.

The crossover operator may disrupt a group, leaving it with zero machines or zero components. When this occurs, the entire disrupted group is eliminated and the displaced components or machines are reinserted into existing groups. By placing displaced machines with components that need them or displaced components with machines they require, the heuristic speeds up the work of the algorithm by moving it in the right direction through intelligent replacement.

7. Conclusions

A new solution methodology for the MPCF problem is developed. This methodology is based on a grouping genetic algorithm and employs a specialized replacement heuristic within the crossover operator. Although DeLit *et al.* (2000) discuss the use of the GGA for the MPCF problem, our work provides the first testing of this approach on these problems from the literature.

CF-GGA is tested using well-know problems from the literature. Its performance is evaluated using two popular objective function measurements. Using these problems and measures, CF-GGA produces higher quality solutions than ZODIAC, GRAFICS and Srinivasan's algorithm. Although solution quality is slightly inferior to Joines, the number of generations required by CF-GGA is dramatically lower.

The use of CF-GGA in conjunction with the efficiency measurement has lead to a deeper understanding of the deficiencies of this measurement. Although voids and exceptional elements may be weighted equally in the objective function formula, the fact that the off-diagonal portion of the MP matrix is so much larger than the on-diagonal portion leads to more weight being attached to voids. CF-GGA works to minimize voids and the result is a maximized efficiency score.

CF-GGA is a practical tool. It requires only three parameters and is not highly sensitive to the choice of values for these parameters. For this reason, CF-GGA is easy to set up and use. In addition, CF-GGA works with limited reliance on specialized operators developed for a narrow problem range. Another advantage of CF-GGA is its ability to be adapted to different performance measures. Altering the performance measure may be done without changing the steps or parameters of the algorithm.

References

- ALENDAR, J. T., 1994, *An Indexed Bibliography of Genetic Algorithms: Years 1957–1993*. Compiled by Jarmo T. Alendar (Finland: Art of CAD).
- BALLAKUR, A. and STEUDEL, H. J., 1987, A with-in cell utilization based heuristic for designing cellular manufacturing systems. *International Journal of Production Research*, **25**, 639–655.

- BROWN, E. C., 1996, Using the facility location problem to explore operator policies and constraint-handling methods for genetic algorithms. PhD Dissertation, University of Virginia.
- BURBIDGE, J. L., 1963, Production flow analysis. *Production Engineer*, **42**, 742–752.
- BURBIDGE, J. L., 1979, *Group Technology in the Engineering Industry* (New York: Wiley).
- CARRIE, A. S., 1973, Numerical taxonomy applied to group technology and plant layout. *International Journal of Production Research*, **11**, pp. 399–416.
- CHANDRASEKHARAN, M. P. and RAJAGOPALAN, R., 1986a, An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *International Journal of Production Research*, **24**, 451–464.
- CHANDRASEKHARAN, M. P. and RAJAGOPALAN, R., 1986b, MODROC: an extension of rank-order clustering for group technology. *International Journal of Production Research*, **24**, 1221–1233.
- CHANDRASEKHARAN, M. P. and RAJAGOPALAN, R., 1987, ZODIAC: an algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research*, **25**, 835–850.
- CHANDRASEKHARAN, M. P. and RAJAGOPALAN, R., 1989, GROUPABILITY: an analysis of the properties of binary data matrices for group technology. *International Journal of Production Research*, **27**, 1035–1052.
- DEJONG, K., 1975, An analysis of the behavior of a class of genetic adaptive systems. PhD Dissertation, University of Michigan.
- DELIT, P., FALKENAUER, E. and DELCHAMBRE, A., 2000, Grouping genetic algorithms: an efficient method to solve the cell formation problem. *Mathematics and Computers in Simulation*, **51**, 257–271.
- FALKENAUER, E., 1992, The grouping genetic algorithms - widening the scope of the GAs. *JORBEL – Belgian Journal of Operations Research, Statistics and Computer Science*, **33**, 79–102.
- FALKENAUER, E., 1998, *Genetic Algorithms for Grouping Problems* (New York: Wiley).
- GOLDBERG, D., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, MA: Addison-Wesley).
- HOLLAND, J. H., 1975, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications in Biology, Control, and Artificial Intelligence* (Ann Arbor, MI: University of Michigan Press).
- JOINES, J., CULBRETH, C. T. and KING, R. E., 1996, Manufacturing cell design: an integer programming model employing genetic algorithms. *IEEE Transactions*, **28**, 69–85.
- KING, J. R., 1980a, Machine-component group formation in group technology. *Omega*, **8**, 193–199.
- KING, J. R., 1980b, Machine-component grouping in production flow analysis: an approach using rank order clustering algorithm. *International Journal of Production Research*, **18**, 213–237.
- KING, J. R. and NAKORNCHAI, V., 1982, Machine-component group formation in group technology review and extension. *International Journal of Production Research*, **20**, 117–133.
- KUMAR, C. S. and CHANDRASEKHARAN, M. P., 1990, Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Research*, **28**, 233–243.
- LOZANO, S., ADENSO-DIAZ, B., EGUIA, I. and ONIEVA, L., 1999, A one-step tabu search algorithm for manufacturing cell design. *Journal of the Operational Research Society*, **50**, 509–516.
- McAULEY, J., 1972, Machine grouping for efficient production. *The Production Engineer*, **52**, 53–57.
- MCCORMICK, W. T., SCHWEITZER, P. J. and WHITE, T. W., 1972, Problem decomposition and data reorganization using a clustering technique. *Operations Research*, **20**, 993–1009.
- MICHALEWICZ, Z., 1992, *Genetic Algorithms + Data Structures = Evolution Programs* (New York: Springer-Verlag).
- RADCLIFFE, N. J., 1991, Forma analysis and random respectful combination. *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 222–229.
- RADCLIFFE, N. J., 1996, Personal communications.

- REEVES, C., 1995, A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, **22**, 5–13.
- SEIFODDINI, H. K. and TJAHHANA, B., 1999, Part-family formation for cellular manufacturing: a case study at Harnischfeger. *International Journal of Production Research*, **37**, 3262–3273.
- SOFIANOPOULOU, S., 1997, Application of simulated annealing to a linear model for the formation of machine cells in group technology. *International Journal of Production Research*, **35**, 501–511.
- SRINIVASAN, G., 1994, A clustering algorithm for machine cell formation in group technology using minimum spanning trees. *International Journal of Production Research*, **32**, 2149–2158.
- SRINIVASAN, G. and NARENDRAN, T. T., 1991, GRAFICS – a non-hierarchical clustering algorithm for group technology. *International Journal of Production Research*, **29**, 463–478.
- SU, C. T. and HSU, C. M., 1998, Multi-objective machine-part cell formation through parallel simulated annealing. *International Journal of Production Research*, **36**, 2185–2207.
- VAKHARIA, A. J. and CHANG, Y. L., 1997, Cell formation in group technology: a combinatorial search approach. *International Journal of Production Research*, **35**, 2025–2043.
- VENUGOPAL, V., 1999, Soft-computing-based approaches to the group technology problem: a state-of-the-art review. *International Journal of Production Research*, **37**, 3335–3357.