10

Integrated Cellular Manufacturing System Design and Layout Using Group Genetic Algorithms

Michael Mutingi[1] and Godfrey C. Onwubolu[2]
[1]*National University of Singapore, Electrical & Computer Engineering*
[2]*School of Applied Technology, HITAL, Toronto*
[1] *Singapore*
[2]*Canada*

1. Introduction

Cellular Manufacturing System (CMS), an application of group technology philosophy, is a recent technological innovation that can be used to improve both productivity and flexibility in modern manufacturing environments (Signh, 93; Sarker and Xu, 1998). In practice, the essence of CMS is to decompose a manufacturing system into manageable autonomous subsystems (called manufacturing cells) so as to enhance shop-floor control, material handling, tooling, and scheduling. The decomposition process involves identification of part families with similar processes or design features and machine cells so that each family can possibly be processed in a single cell. In addition to this, machine layout within each cell is considered essential in order to improve efficiency and effectiveness of the overall production system. Consequently, setup times, work-in-process inventories, and throughput times are reduced significantly. The overall process of designing CMS involves the following four generic phases:

1. *Cell formation*: involves grouping of machines which can operate on a product family with little or no inter-cell movement of the products.
2. *Group layout*: includes layout of machines within each cell (intra-cell layout), and layout of cells with respect to one another (inter-cell layout).
3. *Group scheduling*: involves scheduling of parts for production
4. *Resource allocation*: assignment of tools, manpower, materials, and other resources

In general, the design of CMS includes three critical decisions, namely, cell formation, group layout, and scheduling. In the most ideal case, these criteria should be addressed simultaneously so as to obtain the best possible results (Kaebernick and Bazargan-Lari, 1996; Mahdavi and Mahadevan, 2008). However, due to the complex nature of the decision problem coupled with the limitations of conventional approaches, most of the cell formation studies have focused on these decisions independently or sequentially (Selim, 1998; Onwubolu and Mutingi, 2001). Most cell formation approaches proposed in literature use flow patterns of parts (sequence data) for cell design issues only. On the other hand, the layout designers did not consider the cell formation problem. Due to the fact that the sequential approach addresses the cell formation and the cell layout problem in a disjointed fashion, the quality of the final

solution is often compromised. In this chapter, an integrated approach to cell formation and layout design is presented, based on available sequence data. The GGA-based approach utilizes sequence data to identify machine cells as well as machine layout within each cell. In this view, the major objectives for this chapter are as follows:

- to develop a GGA based methodology for solving the integrated CMS design and layout problem using sequence data, or flow patterns.
- to develop relevant performance metrics to address the integrated cell formation and layout problem.
- to make a comparative analysis between the GGA approach and other well known algorithms found in literature.

The next section describes the cell formation and cell layout problem. Section 3 briefly explains the general GA framework. A GGA approach is presented in section 4. Section 5 provides the results and discussion. Finally, section 6 concludes this chapter.

2. Cell formation and layout problem

The cell formation problem (CFP) in CMS involves grouping of machines which can operate on a product family with similar manufacturing processes and features such that little or no inter-cell movement of products is involved. The overall objective of cell formation is to gain the advantages inherent in the philosophy of group technology. In assessing the quality of solutions, various objectives are considered. These objective functions include the following;

i. Minimization of inter-cell movements;
ii. Minimization of intra-cell movements;
iii. Maximization of utilization;
iv. Minimization of material handling costs, and
v. Minimizing cell work-load imbalances

The cell layout problem involves layout machine within each cell and layout of cells with respect to one another. Recently, researchers have made efforts to utilize interval data and ordinal data, consisting of process sequence data which identifies the order in which jobs are processed (Nair and Narendran, 1998; Won and Lee, 2001; Jayaswal and Adil, 2004). The application of sequence data in CMS has received little attention in the research community and in industry. Sequence data provides useful information on flow patterns of jobs in a manufacturing system. As such, sequence data is useful not only in identifying part family and machine groups but also the actual layout of machines within each cell, based on flow patterns. Earlier studies focused on the use of zero-one machine-component incidence matrix as the input data for the cell formation problem. However, the joint CFP and the layout problem are often treated independently in literature. In an attempt to jointly address the CF and the layout problem, solution methods from various researchers and practitioners often utilize a sequential approach. In this approach, cells are formed first, followed by intra-cell layout construction. Since the final solution is largely dependent on the initial cell formation, the quality of the final solution is often compromised.

The joint cell formation and layout problem is a new approach that seeks to identify manufacturing cells and the layout (sequence) of machines in the cells in an integrated manner. The whole aim of the approach is to avoid compromising the quality of solutions

with respect to cell formation and cell layout objectives. Therefore, this approach to the joint layout problem is of practical value. The basic cell formation problem is NP-complete, meaning that it has no known polynomial time algorithm due to its combinatorial nature (Kumar *et al*, 1986). It follows that the integrated cell formation and cell layout problem is highly computationally intractable. In this respect, the use of heuristic approaches such as simulated annealing, tabu search, and genetic algorithms, is quite appropriate. Simulated annealing is a probabilistic meta-heuristic method proposed in Kirkpatrick, Gelatt and Vecchi (1983) and in Cern (1985) for finding the global minimum of a cost function that may possess several minima. It works by emulating the physical process whereby a solid is slowly cooled down so that when its structure eventually frozen, this occurs at a minimum energy configuration. Tabu Search is a meta-heuristic local search algorithm created in Glover and McMillan (1986) for solving combinatorial optimization problems. It uses the concept of a local or neighbourhood search to iteratively move from one potential solution x to an improved solution x' in the neighbourhood of x, until some stopping criterion has been satisfied, usually an attempt limit or a score threshold (Glover, 1989; Glover, 1990).

3. Genetic algorithms

Genetic algorithm (GA), originated by Holland (1975), is a meta-heuristic approach based on evolutionary principles of natural selection and survival of the fittest. The GA methodology has been applied extensively in a wide range of combinatorial problems in engineering, business, manufacturing, agriculture, telecommunications and sciences (Gen and Cheng; Goldberg, 1989; Man *et al*, 1999). The method integrates the elements of stochastic and direct search to obtain optimal (or near-optimal) solutions within reasonable computation time. GA attempts to evolve a population of candidate solutions by giving preference of survival to quality solutions, whilst allowing some low quality solutions to survive in order to maintain a level of diversity in the population. This process enables GA to provide good solutions so as to avoid premature convergence. Each candidate is coded into a string of digits, called chromosomes. New offspring are obtained from probabilistic operators, mainly crossover and mutation. Comparison of new and old (parent) candidates is done based on a given objective or fitness function so as to retain the best performing candidates into the next population. In this process, characteristics of candidate solutions are passed from generation to generation through probabilistic selection, crossover, and mutation actuated in the population of candidate solutions.

The general GA framework can be represented as follows:

BEGIN
    Initialize *population* with random candidate solutions;
    Evaluate each candidate;
REPEAT
    Select parent chromosomes;
    Recombine pairs of parents;
    Mutate the resulting offspring;
    Evaluate new candidates;
    Select individuals for next generation
UNTIL (Termination condition is satisfied**)**

Figure 1 shows the general flow of the genetic algorithm.

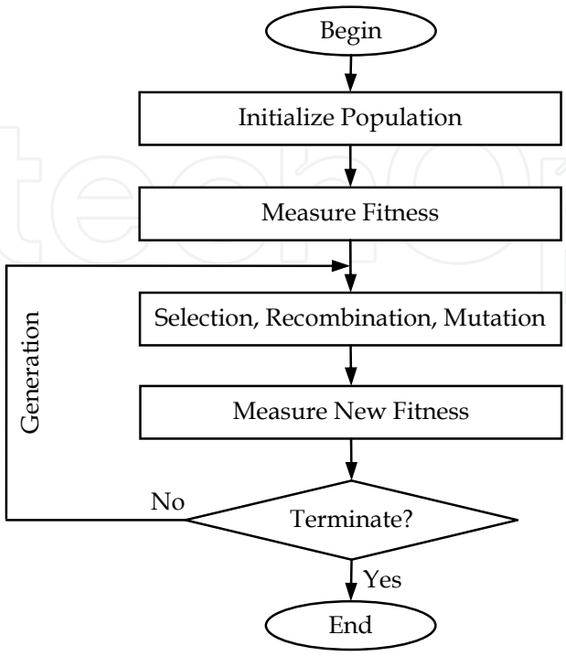


Fig. 1. Genetic algorithm framework

Genetic algorithm offers unique advantages over other stochastic searches, population-based search, including implicit parallelism, independence from gradient information, and flexibility to hybridization with other heuristics. Early applications of the GA approach to the cell formation problem include the work by Venugopal and Narendran (1992) based on minimization of cell load variation and inter-cell moves. Other applications were done by Gravel et al. (1998), and Hsu and Su (1998). However, Falkenauer (1992) realized several significant shortcomings of using classical GAs for grouping problems. Falkenauer (1998) pointed out that though attempts have been made to minimize the drawbacks associated with applying GAs to grouping problems by use of specialized genetic operators, this still result in various shortcomings. In this view, Falkenauer (1992) introduced a grouping genetic algorithm, designed to handle the special structure of grouping problems. Group genetic algorithm (GGA) is a modification of conventional GA designed specifically for clustering/grouping problems. In the next section, an enhanced GGA approach is proposed for the machine cell formation and layout problem.

4. A group genetic algorithm approach

Grouping genetic algorithm (GGA) combines specifically designed operators for grouping problems with the power of local search in order to refine new chromosomes generated. Therefore, GGA is a preferable approach over other heuristic and conventional approaches. The design of the proposed GGA for the joint cell design and layout problem is presented, based on its six main building blocks, namely:

i. Fitness/objective function
ii. Chromosome coding scheme
iii. Initial population generation
iv. Selection and recombination
v. Group genetic operators: crossover, mutation and inversion
vi. Genetic parameters

The next sections elaborate on these building blocks.

4.1 Objective/fitness function formulations

From the CMS design perspective, the existence of voids and exceptions should be minimized. In layout design, adjacency of machines in a cell is a key factor as it can reduce material handling costs significantly (Mahdavi and Mahadevan, 2008). From production planning perspective, the sequence in which machines are placed in cells may create unwanted reverse flows and skipping of workstations. For instance, a cell with machines 1 and 2 has two possible sequences (layouts), i.e., [1, 2] or [2, 1]. From Table 1, it can be seen that the cell layout [2, 1] has only one consecutive forward flow, while layout [1, 2] has four. From this analysis, layout [1, 2] is preferred.

	Parts										
	2	1	4	5	3	7	6	8	9	11	10
Machines											
2	2	2	3	2	2	2					
1	1	1	1	3	1	1					
5							1	1	1	2	1
3			2	1			3	2	2	3	2
4							2	3	3	1	3

Table 1. A typical solution for a cell formation problem

Ideally, a good objective function should be able to capture and evaluate the effects of the sequence of machines within each cell. A simplified way of evaluating the fitness of a cell layout is to express the objective function in terms of the number of consecutive forward flows. In this connection, Mahdavi and Mahadevan (2008) defined the cell flow index (CFI) and the overall flow index (OFI) for evaluating the performance of cell design and cell layout solutions.

The following notation is used in this model.

n number of parts in the system
m number of machines in the system
n_c number of parts in cell c
m_c number of machines in cell c
v_c number of voids in cell c
N_{fc} number of consecutive forward flows within cell c
S_{jk} machine-component matrix $[s_{jk}]$; $s_{jk} = 1$ if part k visits machine k, and 0 otherwise

In order to determine the average flow and overall flow performance measures, the total number of operations and the consecutive flows between a pair of machines are calculated. The total number of flows N_{flow} is:

$$N_{flow} = \sum_{k} \max_{j} s_{jk} - n \tag{1}$$

The total number of flows in each cell c is determined as follows:

$$N_{tc} = (n_c m_c) - v_c - n_c \tag{2}$$

4.1.1 Cell flow index (CFI)

The cell flow index for cell c, CFI_c is the ratio of the number of consecutive forward flows to the total number of flows within the cell. The cell average flow index is the weighted average of CFIs. This is further explained in the following expressions;

$$\text{CFI}_c = \frac{N_{fc}}{N_{tc}} \tag{3}$$

Therefore, the average cell flow index, ACFI is

$$\text{ACFI} = \left(\frac{1}{n}\right) \cdot \sum_{c} n_c \text{CFI}_c \tag{4}$$

It is clear from the above analysis that as the number of voids in the cell decreases and as the number of consecutive forward flows increases, the CFI measure increases. This indicates that the CFI represents the solution quality with respect to the number of voids and the intra-cell moves. Therefore, a combination of these performance measures ensures that the cell formation and layout are addressed jointly.

4.1.2 Overall cell flow index (OFI)

The OFI performance measure defines the ratio of the sum of consecutive forward flows in all the cells to the total number of the flows required to process all the parts. This can be expressed as follows;

$$\text{OFI} = \left(\frac{1}{N_{nflow}}\right) \cdot \sum_{c} N_{fc} \tag{5}$$

Expression (5) shows that the overall cell flow index defines the extent of inter-cell moves (exceptions); increasing values of OFI can be obtained by decreasing values of inter-cell moves. While the OFI points to the inter-cell movements, the ACFI addresses the intra-cell movements.

4.2 Solution encoding – chromosome representation

The GGA's performance strongly depends on the type of the coding scheme, that is, the chromosome (string) representations used. Effective coding schemes can improve the search

efficiency and quality. Most of the coding schemes in literature used strings of integer numbers to where the position of the number represents the machine and the value of the number identifies the cell number. For example, a typical chromosome (2 3 1 1 2 3 1 1 2) containing 9 machines represents a manufacturing system with 3 cells. Machines 1, 5, 6 and 8 are in cell 1, machines 2 and 3 are in cell 2, and machines 4, 7, 9 occupy cell 3.

Machine position:		1 2 3 4 5 6 7 8 9
Chromosome	:	2 3 1 1 2 3 1 1 2

The proposed GGA algorithm has an improved coding scheme, similar to the one proposed Filho and Tibert (2006). The coding scheme improves the utilization of the group structure by using a group structure for each feasible string based on three code schemes as shown in Figure 2. The first, code 1, is a string of size *m*, where *m* represents the total number of machines in the system. The second is a group structure upon which the genetic operators act, while the third represents the positions of the last nodes of each group.

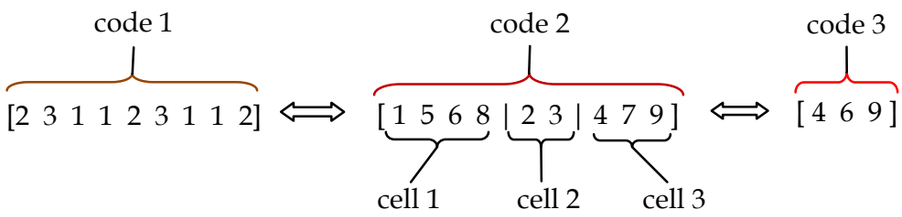


Fig. 2. Chromosome representation

It can be seen from code 3 in Figure 2 that cell 1 consists of the first four genes in code 2. Similarly, cell 2 is made up of the next two genes. Finally, cell 3 is composed of the last three genes in code2. Several features are enhanced in the implementation of the GGA structure, such as in formulation of objective/fitness functions, the genetic operators, chromosome repair and other genetic strategies.

4.3 Initial population

An initial population of the desired size, *popsize*, is randomly generated from the solution space. Consider a typical problem consisting of *m* machines and a predetermined number of cells, *v*. Assume that each cell comprises at least two machines. Then, the initial population is created according to the following procedure:

Repeat

1. For each cell *j* (*j*=1,...,*v*), randomly select two machines from the set of machines.
2. For the remaining (*n*-2*j*) unassigned machines, randomly assign a machine to a cell, until all machines are assigned.
3. Encode the chromosome using code 1 and add to the initial population.

Until (population size *popsize* is achieved).

In GGA application, the goal is to minimize some cost function which is usually mapped to a score function which is used to evaluate the generated chromosomes. A mapping procedure initially suggested by Goldberg (1989) is applied as follows;

$$f^i(t) = \begin{cases} f^i_{\max} - g^i(t) & \text{if } g_i(t) < f^i_{\max} \\ 0 & \text{if otherwise} \end{cases} \tag{6}$$

where, $g(t)$ is the objective function of a chromosome and $f_{\max}$ is the largest objective function in the current population.

4.4 Selection strategy

Several selection strategies have been suggested by Goldberg (1989), such as deterministic sampling, remainder stochastic sampling with/without replacement, stochastic tournament, and stochastic sampling with/without replacement. The remainder stochastic sampling without replacement has been found to be the most effective and is applied in this work (Goldberg, 1989). In this strategy, each chromosome i is selected and stored in the mating pool according to the expected count e_i calculated as,

$$e_i = \frac{f_i}{(1/popsize)\sum_{i=1}^{s} f_i} \tag{7}$$

Where, *popsize* is the desired population size and f_i is the score function value of the i^{th} chromosome.

Each chromosome receives copies equal to the integer part of e_i, that is, $[e_i]$, while the fractional part is treated as success probability of obtaining additional copies of the same chromosome into the mating pool.

4.5 Genetic operators

In this section, design issues relating to the development of the proposed GGA approach for the manufacturing cell design problem are defined. Unique crossover, mutation and inversion strategies are developed for the GGA algorithm.

4.5.1 Crossover

Crossover is a probabilistic evolutionary mechanism which seeks to mate chromosomes, chosen by the selection strategy, in order to produce a pool of new offsprings, called *selection pool*. It allows the algorithm to generate new solutions and to explore unvisited regions in the solution space. The proposed crossover, called group crossover operator, exchanges groups of genes of selected chromosomes. The crossover operation occurs with probability *prcoss* until the desired pool size, $poolsize = popsize \cdot pcross$, is obtained. The procedure for the group crossover operator is as follows:

Repeat

1. Generate a random integer number between 1 and (v-1), where v is number of cells. This number defines the crossover point.

2. Swap the groups to the right of the crossover point to generate two offspring.
3. Repair the offspring by eliminating any duplicated machines and introducing missing machines.

Until (selection *poolsize* is achieved).

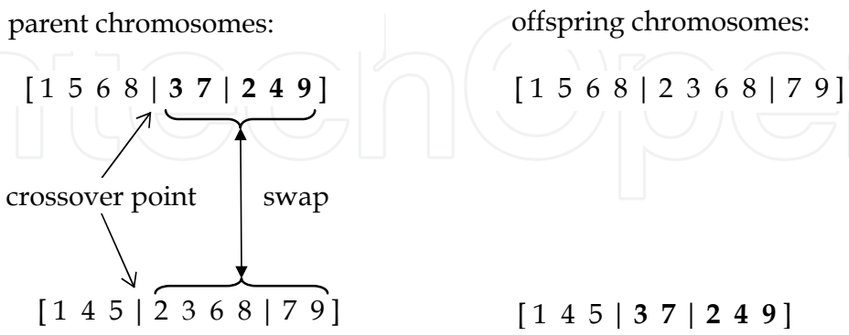


Fig. 3. Crossover operator

In the crossover process, some machines may appear in more than one cell, and some may be missing. Such offspring should be repaired. The repair procedure identifies duplicated machines and eliminates those to the left of the crossover point. Missing machines are inserted on the cell with the least number of nodes. Thus, the group representation scheme enhances the crossover operator by taking advantage of the group structure.

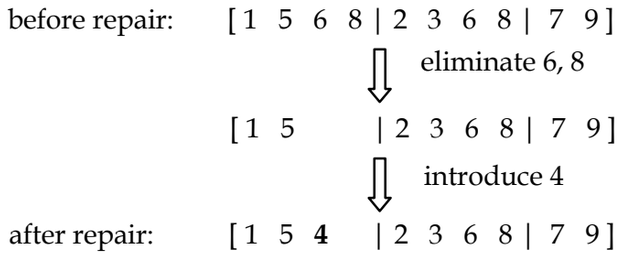


Fig. 4. Chromosome repair procedure

4.5.2 Mutation

The mutation operator is applied to every new chromosome in order to maintain diversity of the population and avoid premature convergence. Two mutation operators are proposed, namely *swap mutation* and *shift mutation*. The swap mutation operates by swapping genes between two randomly chosen groups in a chromosome (see Figure 5.). Its general procedure can be summarized as follows:

1. Randomly select two integer numbers from the set {1,2, ..., v}, where v is the number of cells or groups.
2. Randomly choose a gene from each group

3. Swap the selected genes, exchanging their values

swap

offspring chromosome	:	[1 5 4 | 2 3 6 8 | 7 9]
select group or cell	:	1 and 3
select genes or machines	:	genes **4** and **7**
mutated offspring	:	[1 5 **7** | 2 3 6 8 | **4** 9]

Fig. 5. Swap mutation

The *shift mutation operator* works by shifting the frontier between two adjacent groups by one step either to the right or to the left, as shown in Figure 6. Essentially, the number of nodes is increased in one group and simultaneously decreased in the other. The procedure for the mutation operator is summarized thus;

1. Generate a random integer number between 1 and (v-1). Let this number represent the chosen frontier.
2. Randomly choose the direction of shift: *right* or *left*.
3. Shift the frontier in the selected direction, thereby moving one node between adjacent groups.

offspring chromosome	:	[1 5 4 **|** 2 3 6 8 | 7 9]
select frontier, rand (1,2)	:	1 ↳ shift frontier
select direction	:	right
mutated offspring	:	[1 5 4 **2 |** 3 6 8 | 7 9]

Fig. 6. Shift mutation operator

4.5.3 Inversion operator

In order to curb premature convergence and control diversity level of the population, an inversion operator is designed. The inversion operator is applied, at a very low probability, on chromosomes selected by the selection strategy prior to crossover operation. Basically, the inversion strategy operates by rearranging the groups in the reverse order, for instance, the order of cells [1, 2, 3] is transformed to a new [3, 2, 1]. This procedure is further illustrated in Figure 7.

chromosome before inversion	:	[1 5 6 8 | 3 7 | 2 4 9]
chromosome after inversion	:	[2 4 9 | 3 7 | 1 5 6 8]

Fig. 7. Inversion operator

4.5.4 Diversification

In GGA application it is observed that as the iterations proceed, the solution space (population) converges to a particular solution. However, rapid loss of diversity and premature convergence may occur before an optimal solution is obtained; a problem called genetic drift. To track the diversity of the solution space, Grefenstette (1987) proposed an entropic measure H_i in a population of candidates. For each machine i, the H_i can be defined for GGA in this form;

$$H_i = \sum_{j=1}^{m} \frac{(n_{ij}/p) \cdot \log(n_{ij}/p)}{\log(m)} \tag{8}$$

Where n_{ij} is the number of strings in which machine i is assigned position denoted by j in the current population, p is the solution space size, and m is the number of machines. Divergence H is calculated as;

$$H = \sum_{i=1}^{m} H_i / m \tag{9}$$

As the iterations proceed, the divergence parameter H approaches zero. Thus, the diversity of the solution space can be monitored and controlled by applying the inversion operator till diversity improves to a preset value. In order to prevent loss of good solutions, a fraction (e.g., 0.2) of best performing solutions from the undiversified population is preserved. Performing candidate solutions from the diversified population are compared with those from the diversified population, preferring those that fair better. Thus, the best performing candidates are taken into the next generation.

4.6 The group genetic algorithm implementation

The structure of the proposed GGA for solving the integrated cellular manufacturing system problem was developed incorporating the group operators described in previous sections. A multi-objective approach is adopted in this application based on the two performance measures, ACFI and OFI. The overall GGA structure is now summarised as follows:

Step 1. *Input*: initial data input:

i. Select the typical initial GGA parameter values (see Table 2)
ii. Input the manufacturing data, with sequence data

Step 2. *Initial population*: create randomly, two initial populations, called old populations, *oldpop*1 *and oldpop*2.

Step 3. *Selection and recombination*: Select chromosomes using stochastic sampling without replacement

i. Evaluate strings by objective function, fitness function and expected count
ii. create two temporal population, *temppop*1 and *temppop*2 using the integer parts of expected count and fractional parts as success probabilities

Step 4. *Crossover/recombination*: Apply the group crossover to *temppop*1 and *temppop*2 to create a two selection pool populations, *spool*1 and *spool*2.

i. Select two candidates for crossover using remainder selection without replacement, one from *temppop*1 and another from *temppop*2.
ii. Apply crossover operator to the two strings
iii. If crossover is successful, apply inversion operator, otherwise go to step 5
iv. Apply repair mechanism if necessary

Step 5. *Mutation*: apply mutation operators to the two offspring and move them to new population

Step 6. *Replacement strategy*: Replace old populations with corresponding new populations

i. Compare corresponding chromosomes successively in each selection pool and old population
ii. Take the one that fares better in each comparison
iii. For the rest of the offspring, selection with probability 0.555

Step 7. *Diversification*: Diversify population by applying the mutation operator if mutation falls below a predetermined minimum

i. Calculate diversity H, of the population
ii. If the acceptable diversity H_a is such that $H<H_a$ then diversify until diversity is acceptable.
iii. re-evaluate chromosomes in terms of fitness functions, defined by ACFI and OFI

Step 8. *New population*: Check the current generation count, *gen* against maximum generation count *maxgen*.

i. If *gen* < *maxgen* then go to Step 3, otherwise stop
ii. Return the best solutions

GGA Parameter	Variable	Value
Number of generations	*maxgen*	Variable
Population size	*popsize*	10-40
Crossover probability	*crossprob*	0.4 – 0.7
Mutation probability	*mutprob*	0.02 – 0.3
Inversion probability	*invprob*	0.04 – 0.2
Chromosome size	*chrom*	Number of machines

Table 2. Typical GGA genetic parameters

Part families are identified based on the number of operations required by a part in a cell. Therefore, a part is assigned to a cell where it requires the maximum number of operations (or machines) for its processing.

5. Results and discussion

The proposed GGA approach was implemented in Java SE 7. An illustration of the GGA execution is first given. A comparative analysis on of the performance of the proposed approach with other algorithms is then presented based on computational analysis on known published data sets.

5.1 GGA computational analysis

This section first provides a numerical illustration obtained when executing the GGA algorithm on well known problem data sets in literature. The set of input data used in this illustration is found in Nair and Narendran (1998). The design and layout problem consists of 25 machines and 40 parts (a 25 x 40 problem). Figure 8 shows an illustration of the intermediate stages arrived at as the algorithm solves design and layout problem. The objective function represents the ACFI and the OFI objective values. The input number of cells used for the simulation run was four. The results of the simulation run show that the ACFI values increased from 20% to 68% after 40 iterations, while the OFI values rose from 21% to 42% after 25 iterations.

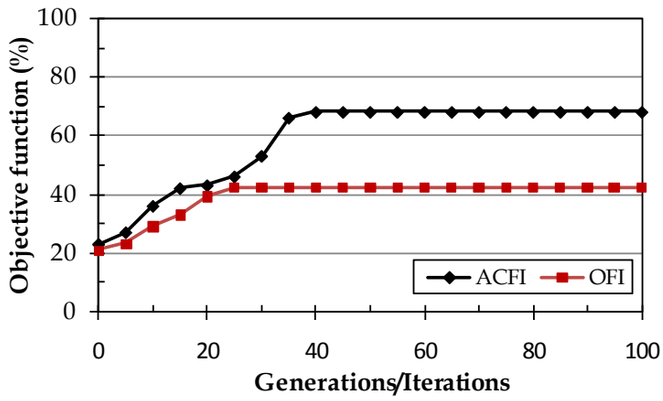


Fig. 8. GGA objective function for Nair & Narendran (1998) 25 x 40 problem

Further numerical experiments were carried out based on an 8 x 20 problem obtained from Nair and Narendran (1998), as shown in Table 3. With a typical set of input data for genetic parameters, the final solution from the GGA simulation run is an improved version of the Nair and Narendran (1998) problem. Table 4 provides the improved solution to the problem. Furthermore, a summary of the final improved version of the solution is provided in Table 5.

	Parts																			
	2	8	9	11	13	14	16	17	19	3	4	6	7	18	20	1	5	10	12	15
Machines																				
3	2	2	3	2	2	3	2	1	2											
1	1	1	1	3	1	1	1	3	1	2										
4										5	2	2	2	1	1			2		
7				1						3	3	3	3	4	4				2	
8										4	4	4	1	3	5					
2						2				1	1	1	4	2	2					
5								2				5				2	2	3	1	1
6			2												3	1	1	1	3	2

Table 3. Solution from Nair and Narendran (1998) – 8 x 20 problem

The machine cells obtained by the GGA approach are the same as those obtained from the CASE algorithm in Nair and Narendran (1998) and those obtained from the CLASS algorithm in Mahdavi and Mahadevan (2008). Similar to the results from the CLASS algorithm, GGA obtained an improved sequence of machines based on the use of sequence data, showing a remarkable improvement in the layout of machines within cells. In this respect, the GGA approach is effective when compared to well known algorithms in literature. Thus, the algorithm is able to simultaneously address the cell formation and the cell layout problems effectively within a reasonable computation time.

	Parts																		
	2	8	9	11	13	14	16	17	19	3	4	6	7	18	20	15	1	5	10	12
Machines																				
1	1	1	1	3	1	1	1	3	1	2										
3	2	2	3	2	2	3	2	1	2											
2						2				1	1	1	4	2	2					
4										5	2	2	2	1	1				2	
7				1						3	3	3	3	4	4					2
8										4	4	4	1	3	5					
6			2												3	2	1	1	1	3
5								2				5				1	2	2	3	1

Table 4. New solution of Nair and Narendran (1998) using GGA – 8 x 20 problem

Cell	Machines	Parts
C1	1, 3	2, 8, 9, 11, 13, 14, 16, 17, 19
C2	2, 4, 7, 8	3, 4, 6, 7, 18, 20
C3	6, 5	1, 5, 10, 12, 15

Table 5. Final improved solution from Nair and Narendran (1998) problem 8 x 20 problem

Cell No.	CASE Solution				CLASS Solution				GGA Solution			
	n_c	N_{fc}	N_{tc}	CFI%	n_c	N_{fc}	N_{tc}	CFI%	n_c	N_{fc}	N_{tc}	CFI%
1	9	1	9	11.1	9	5	9	55.6	9	5	9	55.6
2	6	7	18	38.9	6	9	18	50	6	9	18	50
3	5	1	5	20.0	5	2	5	40.0	5	2	5	40.0
N_{flow} = 41												
ACFI (%)				21.0				50.0				50.0
OFI (%)				22.0				39.0				39.0

Table 6. Comparative study of GGA, CASE and CLASS algorithms - 8 x 20 problem

In order to demonstrate the utility of the proposed GGA algorithm, a comparative study was done with GGA, CASE and CLASS algorithms. Table 6 provides the results of the comparative analysis. It can be seen from this analysis that though machine groups and part families are the same for the three algorithms, the ACFI and OFI differ with the CASE solution. However, the ACFI and OFI values of GGA are similar to those obtained from CLASS. This shows a remarkable improvement of the solution to the joint cell formation and layout problem.

The next section provides a comparative analysis of the performance of GGA approach and other algorithms found in literature.

5.2 Comparison of GGA with other algorithms

In order to gain more understanding on the effectiveness of the GGA, further comparative experiments were carried out based on data sets reported in literature including Tam (1988), Harhalakis et al. (1990), and Nair & Narendra (1998). Park and Suresh (2003) made a comparative study of known algorithms on sequence data. Algorithms such as fuzzy ART neural network and conventional clustering methods were compared. In addition to these algorithms, other approaches such as CASE designed by Nair and Narendran (1998) and CLASS originated by Mahdavi and Mahadevan (2008) are included in the comparative study. Therefore, the performance of GGA can sufficiently be analyzed based on these known data sets and algorithms. The results obtained in this comparative study are shown in Table 7.

Data set	Size	CLASS			Fuzzy Art			Hierarchical			GGA		
		Cells	ACFI	OFI	Cells	ACFI	OFI	Cells	ACFI	OFI	Cells	ACFI	OFI
1.	12 X 19	2	65%	50%	2	49%	36%	2	48%	45%	2	65%	50%
2.	20 x 20	4	65%	41%	4	42%	34%	4	42%	34%	4	69%	43%
3.	25 X 40	4	52%	34%	7	38%	27%	8	37%	22%	4	68%	42%
4.	08 x 20	3	50%	39%							3	50%	39%

Key: 1. Tam (1988); 2. Harhalakis et al. (1990); 3. Nair & Narendra (1998); 4. Nair & Narendra (1998);

Table 7. A comparison of GGA with other approaches

In all cases, the ACFI and OFI values obtained by GGA are much more preferable than those obtained from other algorithms. From this analysis, it can be seen that the utilization of sequence data in joint cell design and layout is important.

6. Conclusions

Integrated cellular manufacturing system design and layout is an important but hard and complex decision process that involves two main problems; cell formation and machine layout within each cell. Sequence data provides additional information on the dominant flow patterns in cells, which forms the basis for solving the integrated layout problem. However, sequence data has not been fully utilised in manufacturing cell design. The main

challenge, therefore, is the extension of the application of sequence data and the development of a robust meta-heuristic algorithm for solving the joint design and layout problem.

In this chapter, a GGA meta-heuristic approach was proposed to solve the integrated layout design problem based on sequence data. The proposed GGA meta-heuristic has unique enhanced features, including a group chromosome scheme, a group crossover operator, a group mutation operator, and a chromosome repair mechanism. The group operators enable the algorithm to reveal the group structure inherent in a data set, producing comparably good quality solutions. While crossover operator enhances exploration of unvisited points in the potential solution space, the mutation exploitation of the best solution in the near-optimal space. Although increasing the number of cells and/or machines may demand more iterations/generations before the algorithm converges to a good solution, the number of parts has no effect on the solution space when grouping machines. Moreover, the parallel mechanism of the approach gives the algorithm robustness and effectiveness over a variety of ill-structured input matrices. Thus, the algorithm is quite preferable in problem situations with a large number of parts.

Comparison with known algorithms in literature was done using known data sets. Apart from well-known performance measures, the average cell flow index was included as a performance parameter, which is a measure of the average magnitude of consecutive forward flows. This measure enabled the GGA approach to evaluate and solve the cell formation and layout design problem in an integrated fashion. The computational results in this study show the utility of the enhanced GGA approach.

Prospects for further research and application of the proposed GGA approach may be interesting. For instance, the group genetic algorithm can be extended to similar clustering problem domains, scheduling problems, as well as network design problems. Further research in these areas is worth exploring.

7. References

Cern, V. (1985). A thermodynamic approach to the travelling salesman problem: An efficient simulation. *Journal of Optimization Theory and Applications*, Vol. 45, 41-51.

Falkenauer E. (1992). The grouping genetic algorithms - widening the scope of the GAs. *Belgian Journal of Operations Research, Statistics and Computer Science*, Vol. 33, 79-102.

Falkenauer, E., 1998, *Genetic Algorithms for Grouping Problems,* New York, Wiley.

Filho, E.V.G., and Tibert, A.J. 2006. A group genetic algorithm for the machine cell formation problem. *International Journal of Production Economics.*

Gen, M. and Cheng, R. (1997). *Genetic Algorithms and Engineering Design*, Wiley Interscience Publication, MA, 1997.

Glover, F. (1989). Tabu Search – Part 1. *ORSA Journal of Computing*, Vol. 1 (2), 190-206.

Glover, F. (1990). Tabu Search – Part 2. *ORSA Journal of Computing*, Vol. 2, No.1, 4-32.

Glover, F., McMillan, C. (1986). The general employee scheduling problem: an integration of MS and AI. *Computers and Operations Research*, Vol. 13, No. 5, 563-573.

Goldberg, D. E. (1989). *Genetic Algorithms: In Search, Optimization & Machine Learning*, Addison-Wesley, Inc., MA, 1989.

Gravel, M., Nsakanda, A.L., Price, W., 1998. Efficient solutions to the cell-formation problem with multiple routings via a double-loop genetic algorithm. *European Journal of Operations Research*, Vol. 109, 286-298.

Grefenstette, J.J. (1987). Incorporating problem specific knowledge into genetic algorithms. In L. Davis, *Genetic and Simulated Annealing*, London: Pitman.

Holland, J. H. (1975). *Adaptation in Natural and Artificial System*, University of Michigan Press, Ann Arbor, MI, 1975.

Hsu, C.M., and Su, C.T. (1998). Multi-objective machine-component grouping in cellular manufacturing: a genetic algorithm approach. *Production Planning & Control*, Vol. 9, No. 2, 155-166.

Jayaswal, S. and Adil, G. K. (2004). Efficient algorithm for cell formation with sequence data, machine replications and alternative process routings. *International Journal of Production Research*, Vol. 42, 2419-2433.

Kaebernick H. and Bazargan-Lari, M. (1996). An integrated Approach to the Design of Cellular Manufacturing. *Annals of the CIRP*, Vol. 45, No. 1, 421-425.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., (1983). Optimisation by simulated annealing. *Science*, Vol. 220, 621-630.

Kumar, K. R., Kusiak, A. and Vaneli, A. (1986). Grouping parts and components in FMS. *European Journal of Operational Research*, Vol. 24387-24400.

Mahdavi, I. and Mahadevan, B. (2008). CLASS: An algorithm for cellular manufacturing system and layout design using sequence data. *Robotics and Computer-Integrated Manufacturing*, Vol. 24, 488-497.

Man, K. F., Tang, K. S. and Kwong, S. (1999). *Genetic Algorithms: Concepts and Design*, Springer, London, 1999.

Nair, G. J. and Narendran, T. T. (1998). CASE: a clustering algorithm for cell formation with sequence data. *International Journal of Production Research*, Vol. 36, 157-179.

Onwubolu, G.C. and Mutingi, M. (2001). A genetic algorithm approach to cellular manufacturing systems. *Computers & Industrial Engineering*, Vol. 39, 125-144.

Sarker B. R., and Xu, Y. (1998). Operation sequences-based cell formation methods: a critical survey. *Production Planning & Control*, Vol. 9, 771--783.

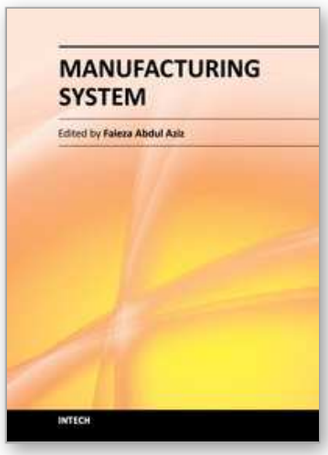
Selim, H. M., Askin, R. G., and Vakharia, A. J. (1998). Cell formation in group technology: Evaluation and directions for future research. *Computers & Industrial Engineering*, Vol. 34, No. 1, 3--20.

Singh N. (1993). Design of cellular manufacturing systems: an invited review. *European Journal of Operational Research*, Vol. 69, 284--291.

Suresh NC, Slomp J, KAparth S., 1999. Sequence-dependent clustering of parts and machines: a Fuzzy ART neural network approach. *International Journal of Production Research*, Vol. 37, 2793-816

Venugopal, V., Narendran, T.T., 1992. A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Computers and Industrial Engineering*, Vol. 22, No. 4, 469-480.

Won, Y. and Lee, K. C. (2001). Group technology cell formation considering operation sequences and production volumes. *International Journal of Production Research*, Vol. 39, 2755-2768.



Manufacturing System

Edited by Dr. Faieza Abdul Aziz

ISBN 978-953-51-0530-5
Hard cover, 448 pages
Publisher InTech
Published online 16, May, 2012
Published in print edition May, 2012

This book attempts to bring together selected recent advances, tools, application and new ideas in manufacturing systems. Manufacturing system comprise of equipment, products, people, information, control and support functions for the competitive development to satisfy market needs. It provides a comprehensive collection of papers on the latest fundamental and applied industrial research. The book will be of great interest to those involved in manufacturing engineering, systems and management and those involved in manufacturing research.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen