Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 181 (2007) 156–167
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

A genetic algorithm for cellular manufacturing design and layout

Xiaodan Wu [a], Chao-Hsien Chu [b,*], Yunfeng Wang [a], Weili Yan [c]

[a] School of Management, Hebei University of Technology, Tianjin 300130, PR China
[b] School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902, Singapore
[c] School of Electrical Engineering and Automation, Hebei University of Technology, Tianjin 300130, PR China

Received 22 October 2004; accepted 17 May 2006
Available online 27 July 2006

Abstract

Cellular manufacturing (CM) is an approach that can be used to enhance both flexibility and efficiency in today's small-to-medium lot production environment. The design of a CM system (CMS) often involves three major decisions: cell formation, group layout, and group schedule. Ideally, these decisions should be addressed simultaneously in order to obtain the best results. However, due to the complexity and NP-complete nature of each decision and the limitations of traditional approaches, most researchers have only addressed these decisions sequentially or independently. In this study, a hierarchical genetic algorithm is developed to simultaneously form manufacturing cells and determine the group layout of a CMS. The intrinsic features of our proposed algorithm include a hierarchical chromosome structure to encode two important cell design decisions, a new selection scheme to dynamically consider two correlated fitness functions, and a group mutation operator to increase the probability of mutation. From the computational analyses, these proposed structure and operators are found to be effective in improving solution quality as well as accelerating convergence.

Keywords: Genetic algorithms; Cellular manufacturing; Cell formation; Group layout

1. Introduction

CM, an important application of group technology, is an approach that can be used to enhance both flexibility and efficiency in today's small-to-medium lot production environment. In essence, a manufacturing system is decomposed into several manageable subsystems, named manufacturing cells. The design of a CMS includes (1) cell formation (CF) – grouping parts with similar design features or processing requirements into part families and associated machines into machine cells, (2) group layout – laying out machines within each cell (intra-cell layout) and cells with respect to one another (inter-cell layout), (3) group scheduling – scheduling parts and part families for production, and (4) resource allocation – assigning tools and human and materials resources [27]. Ideally, all of

* Corresponding author. Tel.: +65 6828 0268; fax: +65 6828 0919. On leave (August 2005–July 2006) from The Pennsylvania State University, USA. After August 1, 2006: 301K IST Building, University Park, PA 16802, USA. Tel.: +1 814 865 4446; fax: +1 814 865 6426.
E-mail addresses: xwu@hebut.edu.cn (X. Wu), chu@ist.psu.edu (C.-H. Chu), ywang@hebut.edu.cn (Y. Wang), wlyan@hebut.edu.cn (W. Yan).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2006.05.035

these decisions should be addressed simultaneously in order to obtain the best results [1]. However, due to the complexity and NP-complete nature of each decision and the limitations of traditional approaches, most researchers have only addressed these decisions sequentially or independently [5,6,13,14].

Genetic algorithms (GAs), developed by Holland [15], have been used extensively as an alternative method for solving numerical optimization problems in a wide variety of application domains including engineering, biology, economics, agriculture, business, telecommunications, and manufacturing [11,12,18]. As a general-purpose search method, a GA combines elements of directed and stochastic search for exploring and exploiting the search space to obtain good solutions. In contrast to other stochastic searches, GAs have the following unique features: implicit parallelism, population-based search, independence of gradient information, and flexibility to hybridize with domain-dependent heuristics. These features often make them a preferable choice over traditional heuristics. GAs seem to perform well for some problems, but not so well for others, especially when multiple objectives and constraints are considered. Determining the contributing behavior of premature and slow convergence and examining the factors (operators and parameters) that have significant impacts on GA performance is important.

Various explorations have been made to study the convergence behavior and/or process of GAs from different viewpoints. Notable developments include dynamic parameter encoding [23], migration and artificial selection [22], and cloning operations [2,17]. Since most real-world problems involve multiple objectives and constraints, researchers have devoted their efforts to exploring methods that can solve such problems. Typically, methods for handling multiple constraints include encoding, modifying genetic operators [19], repairing, and penalizing or rejecting [9,11,20]. Approaches to dealing with multi-objective problems include weighted sums, and Epsilon-constrained and sub-population approaches [9]. Very limited research has been performed on solving concurrent decisions with multiple constraints and highly correlated multi-objectives. Past researchers have studied the impact of various factors, including problem encoding, crossover, mutation, population size, crossover and mutation rates, and halting criteria, on different domains [7]. Some of these factors will be examined in this study since they are closely interrelated and their impacts may be problem dependent.

GAs have been effectively used to solve CF problems of CM [4,16,26] or facility layout problems [3,17,25]. But attempts to concurrently make these two decisions are limited. In this study, a hierarchical genetic algorithm (HGA) is developed to simultaneously form manufacturing cells and determine the group layout of a CMS. In order to manage the complicated and correlated decisions, a hierarchical chromosome structure, a dynamic selection strategy, and a group mutation operator are developed. The major objectives for the current study are:

(1) to develop a GA approach to solving the integrated CF and group layout problem in CM;
(2) to determine the impacts of various parameters, specifically the impact of crossover rate, mutation rate, population size, and the maximum number of generations, on the GA's performance.

2. Mathematical description of the problem

We considered many factors, such as routing (sequence), work load, machine capacity, demand, batch size, and layout type in the problem formulation (see Fig. 1). Routing is often presented in a machine/part/sequence matrix, where the value, a_{ij}, inside the matrix indicates the operational sequence of part j to be processed by machine i. Since machine layout type (e.g., in single row, U shape, multiple rows, or other configurations) has significant impact on part transfer cost, it needs to be considered in CF. The following notation is used in the model:

Indices

i	machine index; $i = 1,\ldots,m$
j	part index; $j = 1,\ldots,n$
k	cell index; $k = 1,\ldots,c$
p	machine position index; $p = 1,\ldots,mp$

Parameters

B_j	transfer batch size for part j
C_{jA}	unit intra-cell transfer cost of part j
C_{jE}	unit inter-cell transfer cost of part j
C_{jB}	unit intra-cell backtracking cost of part j

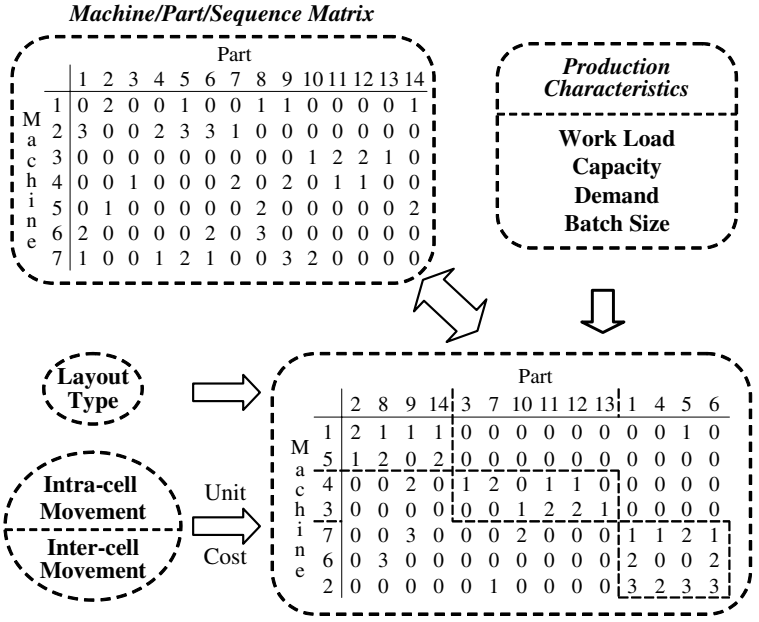

Fig. 1. The integrated cell design problem.

$C^j_{ii'}$ transfer cost of part j between machine i and i'

D_j demand quantity of part j

IC_k =1, if cell k is to be formed; 0, otherwise

M_i the position number of machine i (or the sequence of machine i to be laid out)

NC minimum number of cells to be formed

NM maximum number of machine types allowed in each cell

R_{ij} operation number done on part j using machine i

$f^j_{ii'}$ number of trips for moving part j between machine i and i'; $f^j_{ii'} = D_j/B_j$, if $R_{i'j} - R_{ij} = 1$; 0, otherwise

T_i the capacity of machine i

t_{ij} processing time of part j with machine i

SP set of pairs (i,j) such that $a_{ij} \geqslant 1$

Decision variables

X_{ik} =1, if machine i is to be assigned to cell k; 0, otherwise

Y_{jk} =1, if part j is to be assigned to cell k; 0, otherwise

Z_{ip} =1, if machine i is to be assigned to position p; 0, otherwise

U_{ijk} = 1, if $X_{ik} = 1$ and $Y_{jk} = 0$; 0, otherwise

V_{ijk} = 1, if $Y_{jk} = 1$ and $X_{ik} = 0$; 0, otherwise

Consequently, we obtain

$$C^j_{ii'} = \begin{cases} (M_{i'} - M_i)C_{jA} & \text{if } X_{ik}, X_{i'k} > 0, \quad M_{i'} > M_i \\ (M_i - M_{i'})C_{jB} & \text{if } X_{ik}, X_{i'k} > 0, \quad M_{i'} < M_i \\ |k - k'|C_{jE} & \text{if } X_{ik}X_{i'k} = 0, \quad X_{ik}X_{i'k'} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The problem can be formulated as

$$\text{Minimize} \quad \sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{i'=1}^{m} f^j_{ii'}C^j_{ii'} \quad \text{and} \tag{2}$$

$$\text{Minimize} \quad \sum_{k}\sum_{(i,j)\in sp}\frac{(U_{ijk} + V_{ijk})}{2} \tag{3}$$

$$\text{Subject to} \quad \sum_{k=1}^{c} X_{ik} = 1, \quad i = 1, \ldots, m, \tag{4}$$

$$\sum_{k=1}^{c} Y_{jk} = 1, \quad j = 1, \ldots, n, \tag{5}$$

$$\sum_{i=1}^{m} X_{ik} \leqslant \mathrm{NM} \times \mathrm{IC}_k, \quad k = 1, \ldots, c, \tag{6}$$

$$\sum_{k=1}^{c} \mathrm{IC}_k \geqslant \mathrm{NC}, \tag{7}$$

$$\sum_{i=1}^{m} Z_{ip} = 1, \quad p = 1, \ldots, mp, \tag{8}$$

$$\sum_{p=1}^{mp} Z_{ip} = 1, \quad i = 1, \ldots, m, \tag{9}$$

$$X_{ik} \sum_{j=1}^{n} Y_{jk} t_{ij} D_j / T_i \leqslant 1, \quad i = 1, \ldots, m, \tag{10}$$

$$X_{ik}, Y_{jk}, Z_{ip} = 0 \quad \text{or } 1. \tag{11}$$

Objective functions (2) and (3) are for minimizing the total cost of movement (both inter-cell and intra-cell) and exceptional elements (EEs) respectively. Here, an EE is defined as a part where some of its operations need to be performed outside the part's designated cell. Constraint set (4) is to ensure that each machine is only being assigned to one cell. Constraint set (5) is for ensuring that each part is only being assigned to one cell. Constraint set (6) is for preventing the assignment of more than NM machines to each cell. This constraint set is also used to prevent all machines from being assigned to a single cell. Constraint set (7) is for forcing at least NC cells to be formed. Constraint (8) is for restricting each position to accept only one machine. Constraint (9) is for ensuring that each machine is being assigned to only one position. Constraint (10) is for forcing machine workload to not exceed its capacity. The parameters used in the model can be estimated based on current shop floor configuration and past production data.

CF problems have been shown to be NP-complete [5,6,13,14]. The mathematical model, however, provides an important basis for designing GA applications. The objective functions serve as the fitness functions for evaluating the performance of chromosomes. The constraints are used to filter and repair unqualified chromosomes.

3. Applying a GA to a CMS design

The proposed GA involves the following steps – problem encoding, population initialization, fitness evaluation, selection, crossover, mutation, and replacement (see Fig. 2). GAs start by encoding the solution in a chromosome string containing several genes. The value of each gene is an allele. An initial population of chromosomes is then generated, usually in a random fashion, to be the initial solutions of the problem. A fitness function is selected and used to evaluate the relative performance, or fitness value, of the chromosomes. Based on the fitness value, the selection operator selects superior chromosomes for the new mating pool. The evolution is simulated using reproduction operators such as crossover, which mimics propagation; and mutation, which mimics random changes occurring in nature. Reproduction operators are used on the mating pool to generate new solutions, called offspring. After multiple generations of evolution, highly fit chromosomes often emerge that correspond to very good solutions to the problem.

3.1. Hierarchical chromosome structure

In order to encode CF information for both parts and machines and for machine layout information, a two-layer hierarchical scheme is proposed. In the first layer, a string of integer numbers is used to encode the CF results for machine and then part genes [26]. In the second layer, the allele of each gene represents the positional weight at which the machine is placed. These weights are then ranked to determine the layout sequence of the machines. The machine with the highest weight will be placed in the first position. The genes of the first layer were used to control the genes of the second layer in a hierarchical manner.

For illustration, consider a data set with 10 parts and seven machines to be classified into two manufacturing cells. In Table 1 we show a typical chromosome. The chromosome contains two layers of genes. In layer one, the chromosome contains 17 genes. The allele of each gene represents the cell number to which the machine or part belongs. For instance, machine 1 is assigned to cell #2, machine 2 is assigned to cell #1, and so on. As such cell #1 contains machines 2, 3, 4, and 5 and parts 1, 2, 3, 4, and 6 and cell #2 contains machines 1, 6, and 7 and parts 5, 7, 8, 9, and 10. In layer two, the chromosome contains seven genes. The positional weight for machine 1 is 3, the weight for machine 2 is 4, and so on. The weight matrix of this example is $\{3,4,2,5,1,6,7\}$. By ranking the weights, the machines will be laid out in the order of machines 7, 6, 4, 2, 1, 3, and 5. Combining the CF obtained from the control genes in layer 1, we obtain that the order to lay machines in cell #1 is machines 4, 2, 3, and 5 and machines 7, 6, and 1 for cell #2.

3.2. Dynamic selection strategy

A number of measures have been used to evaluate the performance of CF and machine layout. The popular metrics used are the number of EEs

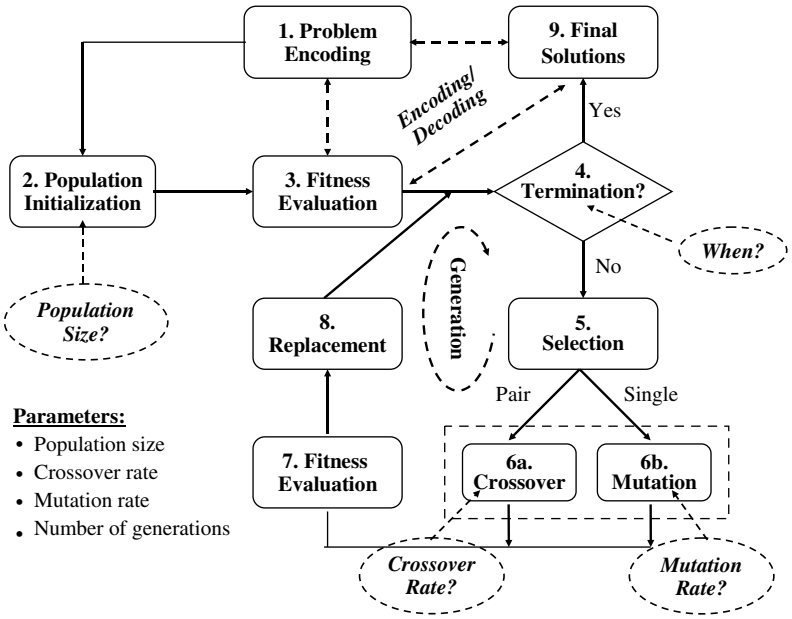

Fig. 2. A generic framework for GAs.

Table 1
Example of chromosome representation

	Machine							Part									
Genes	1	2	3	4	5	6	7	1	2	3	4	5	6	7	8	9	10
Locus	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Layer 1	2	1	1	1	1	2	2	1	1	1	1	2	1	2	2	2	2
Layer 2	3	4	2	5	1	6	7										

and total cost of movement [8]. In order to determine the optimal CF and layout decisions simultaneously, four approaches are used in selecting the fitness function:

(1) use a single objective as the fitness function;
(2) use linearly weighted sum of objective functions as the fitness function;
(3) use Pareto-based fitness assignment for the fitness function;
(4) use a dynamic assignment to switch between fitness functions.

There have been no general guidelines regarding which approach is the best [18]. Based upon schema concepts [15], for highly correlated fitness functions the dynamic switch assignment could be a better choice. A schema is a string of symbols taken from the alphabet $\{0,1,\#\}$ for a binary string. The symbol "#" indicates that we "don't care" what kind of attribute occurs at a given position, thus a schema can represent several bit strings. For example the schema #01# represents four strings: 0010, 0011, 1010 and 1011. Schema theorem was proposed to explain how GAs can be used to exploit in parallel many similarities contained in short, high-performance schemata and illustrate the necessity of finding the best solution by considering the effect of reproduction, crossover, and mutation for a particular schema [15].

The usefulness of schema theorem has been widely debated because it has some limitations [10,21]. In spite of that, the theorem provides general guideline on how GAs work. In this study, we place more emphasis on using correlated good schemata for improving solutions and exploring the idea behind the concept of schemata to increase the speed of convergence for lengthy chromosomes.

The logic of the dynamic selection strategy is given in Fig. 3. The main idea behind it is that if the best fitness value remains unchanged for a fixed number of generations (e.g., 10 generations), it is

```
1   Assume gen*: the generation to switch fitness function; Mgen: the max. allowed
    number of generations when there is no change of the best fitness value.
2   Set gen* ← M and Mgen ← 0
3   Set new generation: gen ← gen +1
4   Create new population
5   Do {
6     If (gen < gen*)
7   Select two chromosomes based on f1. Check if new fitness improved (< ε). If not,
    check if over allowed # of generations
8         Mgen ← Mgen +1
9          If (Mgen > gen*)
10            Set current generation to gen*
11           Mgen ← 0
12    If (gen > gen*)
13       Select two chromosomes based on f2. Check if new fitness improved (<ε)
14       If not, check if over allowed # of generations
15        Mgen ←Mgen +1
16        If (Mgen > gen*)
17            Stop
18       }
```

Fig. 3. Logic of dynamic selection.

time to switch to the other fitness function or stop the evolution process.

We now present the theory behind the dynamic selection logic.

Definition 1. For schema H, if the fitness value of the chromosome with H is always better than that of without H, then schema H is a good schema for the fitness function $f(X)$ and will be defined as $H^*(f)$, otherwise as $H^+(f)$. Thus, $f(H^*) \geqslant f(H^+)$.

Definition 2. If the correlation coefficient of the values of two functions is a positive value, then the two functions are positively correlated.

Theorem 1. *If two fitness functions f_1 and f_2 are positively correlated, then the good schema H^* for f_1 is also good for f_2. That is, $H^*(f_1) \subseteq H^*(f_2)$*

Proof. From Definition 1, $f_1(H^*) \geqslant f_1(H^+)$ if there exists $H^*(f_1)$.

Since f_1 and f_2 are positively correlated, $f_2(H^*) \geqslant f_2(H^+)$.

Thus, $\{H^*(f_1)\} \subseteq \{H^*(f_2)\}$. □

Since that total cost of movement consists of the costs of inter-cell and intra-cell movements and the inter-cell movement is implicitly determined by the EEs and unit inter-cell movement cost [8], there exists a positive correlation between total cost of movement and EEs.

For two highly correlated fitness functions f_1 and f_2, if we ignore f_2 at the first stage of evolution (that is, only consider the genes in the first layer), then the search space decreases by m^m. Similarly, if at the second stage we only consider the genes in the second layer, then the search space may be decreased by c^{m+n}, where c is the number of cells. Consequently, using a dynamic selection operator for GA evolution would significantly improve computational efficiency.

3.3. Crossover operators

Due to the unique hierarchical chromosome scheme used, a one-point crossover and its variation – partial mapped crossover (PMX) is used [11,12]. At first, a cut point is randomly selected over the whole chromosome. Then one-point crossover is used for layer 1 and PMX is applied to layer 2 to prevent from producing illegal chromosomes.

An example of the PMX crossover is given in Table 2. P1 and P2 is the chromosome pair selected for crossover. If only the one-point crossover is used, the reproduced chromosomes, C1′ and C2′, may contain some redundant alleles (e.g., 4 and 5) and miss some other necessary alleles (e.g., 1 and 6). These chromosomes are illegal because it means machines 2 and 6 will be assigned to the same location and there is no machine being assigned to position 1 or 6. To prevent these illegal chromosomes

Table 2
Numerical illustration of crossovers

Cut Point				↓													
Locus	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
P1	2	1	1	1	1	2	2	1	1	1	1	2	1	2	2	2	2
	3	4	2	5	1	6	7										
P2	1	1	1	2	2	1	2	1	1	2	2	1	2	1	2	1	1
	1	3	6	2	7	4	5										
C1’	2	1	1	1	2	1	2	1	1	2	2	1	2	1	2	1	1
	3	**4**	2	**5**	7	**4**	**5**										
C2’	1	1	1	2	1	2	2	1	1	1	1	2	1	2	2	2	2
	1	3	**6**	2	**1**	**6**	7										
C1	2	1	1	1	2	1	2	1	1	2	2	1	2	1	2	1	1
	3	**6**	2	**1**	7	4	5										
C2	1	1	1	2	1	2	2	1	1	1	1	2	1	2	2	2	2
	5	3	**4**	2	1	6	7										

from reproducing, we need to apply PMX to layer 2. In layer 2, for chromosome C1′ we first replace alleles 1, 6, and 7 with 7, 4, and 5, then we determine the mapping relationship ($5 \leftrightarrow 7 \leftrightarrow 1$ and $4 \leftrightarrow 6$), and finally we partially replace the illegal alleles with their corresponding mapping. For instance, in chromosome C1′, alleles 4 and 5 are illegal. We replace them with alleles 6 and 1 respectively. Similarly, in chromosome C2′, besides replacing alleles 7, 4, and 5 with 1, 6, and 7, we replace illegal allele 1 with 5 and allele 6 with 4. We thus obtain final legal children chromosomes C1 and C2.

3.4. Mutation operators

Mutation is designed to prevent premature convergence and to explore a new solution space. However, unlike crossover, the operation alters or mutates one or more genes within an individual chromosome rather than across a pair of chromosomes. We use a conventional exchange mutation operator and also propose a new variation called group mutation for exploration. Exchange mutation involves exchanging the allele of genes from two randomly chosen loci each time. Group mutation is for exchanging all genes' alleles of the two groups for layer 1 at one time. Our motivation for developing group mutation arises from the concept of schema that "*It is more difficult to find the schema* (H) *with long defining length* $\delta(H)$ *and larger number of fixed positions* [15]." An example of exchange mutation and group mutations is given in Table 3, where genes 3 and 7 are randomly chosen for mutations.

Let p_r denote the probability of each locus being selected, p_m the mutation probability, p_e the probability of using exchange mutation, and p_g the probability of using group mutation.

Theorem 2. *For CF problems, the probability of finding good schema using group mutation is higher than that using exchange mutation.*

Proof. If the length of the chromosome is l, then $p_r = 1/l$. Assume that there are q members in one group and r members in another cell. Then

$$p_e = (1/l)^q \cdot (1/l)^r \cdot p_m = l^{-(q+r)} p_m,$$

$$p_g = \frac{q}{l} \cdot \frac{r}{l} \cdot p_m = \frac{qr}{l^2} p_m = qr \cdot l^{-2} p_m,$$

$$\because \quad 1 \leqslant q < l \quad \text{and} \quad 1 \leqslant r < l, \quad (1/l)^{qr} < qr(1/l)^2$$

$$\therefore \quad p_e < p_g. \qquad \square$$

Therefore, theoretically, if we depend only on exchange mutation, when the length of the chromosome is long the chance of having an effective

Table 3
Numerical illustrations of mutations

Cut point			↓				↓										
Locus	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(a) Exchange mutation																	
Before	2	1	1	1	1	2	2	1	1	1	1	2	1	2	2	2	2
	3	4	2	5	1	6	7										
After	2	1	**2**	1	1	2	**1**	1	1	2	2	1	2	1	2	1	1
	3	4	**7**	5	1	6	**2**										
(b) Group mutation																	
Before	2	1	**1**	1	1	2	**2**	1	1	1	1	2	1	2	2	2	2
	3	4	**2**	5	1	6	**7**										
After	**1**	2	2	2	2	**1**	**1**	2	2	2	2	**1**	2	**1**	**1**	**1**	**1**
	3	4	**7**	5	1	6	**2**										

exploitation for finding the optimal solution in the near-optimal area is low, even though p_m may be set to 1. However, the group mutation can help to enhance the GA's ability of exploitation and make it converge rapidly to a promising region.

4. Experimental design and research hypotheses

We used two performance measures to evaluate the results: the value of the objective function (solution quality) and computation time. The solution quality was evaluated in terms of total movement cost and/or the number of EEs. The experiment was divided into two portions: effects of GA parameters and comparison of GA operators. The parameters considered for study are crossover rate (Pc), mutation rate (Pm), population size (Ps), and maximum number of generations (Mgen). The operators evaluated include dynamic selection (DS) and group mutation (GM). The other controlled factors include encoding, population initialization, crossover, and halting method.

We tested the performance using 10 medium-size data sets. Data set five was obtained from [6] and the remaining data sets were randomly generated using a uniform distribution. The characteristics and known solution of these data sets are summarized in Table 4. We also generated the operation sequences, part demand, and processing time for all data sets. The demand of each part was generated from a range of 0 to 50 and the processing time from 0 to 5. For each data set, we ran the test 10 times with different random number seeds. Since many clustering algorithms are known to be sensitive to the column and row orders in which the data is presented, we applied the replicated clustering approach [24] to enable a robust evaluation of the algorithm. We randomly reordered the generated data sets and these scrambled data sets are then solved by our proposed algorithm. We then compare the clustering results with the known solutions.

Table 4
Characteristics of data sets

Data	Size	Density (%)	EE	EE (%)
1	30×60	18.94	88	25.8
2	30×60	19.11	112	32.6
3	40×80	14.13	57	12.6
4	40×80	12.59	68	16.9
5	40×100	10.50	36	8.6
6	50×100	15.66	172	22.0
7	50×100	9.74	189	38.8
8	50×100	8.48	143	33.7
9	60×120	15.67	343	30.4
10	60×120	10.88	200	25.5

EE (%): percentage of EEs.

4.1. Effects of parameters

We considered two levels for each parameter obtained via literature and pilot tests. The values selected for each parameter are shown in Table 5. An experimental design with 16 cells was used to represent the combinations of these parameters. In total, there are 1600 data points for the experiment.

The following null hypotheses were formulated for this experiment:

H1: The mean value of solution quality is the same for the two crossover rates.
H2: The mean value of solution quality is the same for the two mutation rates.
H3: The mean value of solution quality is the same for the two population sizes.
H4: The mean value of solution quality is the same for the two maximum numbers of generations.

The corresponding alternative hypotheses, which can be formulated, for example, as H1′: The mean value of solution quality is different for the two crossover rates, were omitted to save space.

4.2. Comparison of GA operators

We contrasted the performance of the GA: (1) using roulette-wheel vs. dynamic selection strategy, (2) using exchange vs. group mutation, and (3) using dynamic selection vs. using a weighted-sum fitness function. In this study, we take equal weight for both objective functions. An experiment with six cells was used to evaluate all cases. In total, there were 600 ($10 \times 10 \times 6$) data points for the experiment. We used the best combination of parameters obtained from the previous phase for comparisons.

The following null hypotheses were formulated for this experiment:

Table 5
Experimental factors and their levels

Parameters	Levels	Values	
Crossover rate (Pc)	2	0.7	1.0
Mutation rate (Pm)	2	0.1	0.5
Population size (Ps)	2	50	100
Generations (Mgen)	2	100	200

H5: The mean value of solution quality is the same for the GA with roulette-wheel and dynamic evaluation.
H6: The mean value of computation time is the same for the GA with roulette-wheel and dynamic evaluation.
H7: The mean value of solution quality is the same for the GA with exchange and group mutation.
H8: The mean value of computation time is the same for the GA with exchange and group mutation.

The corresponding alternative hypotheses are similar to previous cases and again were omitted to save space.

5. Computational results and analysis

The GA and data generator were coded in Visual C++ and solved on a personal computer with an Intel Pentium IV (1300 MHz) processor to test our implementation.

5.1. Effects of parameters

The results of the experiment are shown in Table 6. The number in each cell is the average relative value for the 100 data points where, the relative value is equal to the absolute value of the differences between the calculated solution and the known solution divided by the known solution of each data set (see Table 4). In general, the smaller value the better result. The highlighted cells are the top five best combinations in average relative means, among which combination 16 produces the best solution for most data sets. Moreover, no matter which level of crossover and mutation rate was used, the proposed GA obtained more number of best solutions if Ps and Mgen were set to 100 and 200, respectively (combinations 4, 8, and 16).

The results of the analysis of variance (ANOVA) for solution quality using the four parameters are presented in Table 7. The results indicate that: all the factors except crossover rate are significant at the 0.001 level, thereby rejecting H2, H3, and H4; and among the parameters, mutation rate, which has the largest sum of square value, is the most important one, followed by population size, and then the stopping criterion. We also evaluated the two-way interactions among the four parameters. None of the interactions are significant. The mean values indicate that: (1) mutation rate 0.5 is better than 0.1; (2) population size 100 is better than 50; and (3) 200 generations is better than 100 generations. These results are consistent with those obtained from Table 6.

5.2. Comparison of GA operators

The ANOVA results of solution quality using the two new operators are presented in Table 8. The results indicate that: (1) the impact of both operators are significant at the 0.001 level, thereby reject-

Table 6
Summary: effect of parameters on solution quality

Index	Combination (Pc, Pm, Ps, Mgen)	Relative mean	Standard deviation	No. of known solution	Ranking by mean
1	0.7, 0.1, 50, 100	1.87	1.34	0	15
2	0.7, 0.1, 50, 200	1.63	1.06	0	12
3	0.7, 0.1, 100, 100	1.66	1.24	0	14
4	0.7, 0.1, 100, 200	1.47	1.03	**4**	8
5	0.7, 0.5, 50, 100	1.61	1.25	0	11
6	0.7, 0.5, 50, 200	1.38	0.74	0	6
7	**0.7**, **0.5**, **100**, **100**	**1.36**	0.80	1	**4**
8	**0.7**, **0.5**, **100**, **200**	**1.24**	0.50	**6**	**2**
9	1.0, 0.1, 50, 100	1.99	1.64	0	16
10	1.0, 0.1, 50, 200	1.63	1.09	2	13
11	1.0, 0.1, 100, 100	1.60	1.03	0	10
12	**1.0**, **0.1**, **100**, **200**	**1.37**	0.74	1	**5**
13	1.0, 0.5, 50, 100	1.60	0.92	0	9
14	1.0, 0.5, 50, 200	1.39	0.71	0	7
15	**1.0**, **0.5**, **100**, **100**	**1.24**	0.62	3	**3**
16	**1.0**, **0.5**, **100**, **200**	**1.12**	0.27	**4**	**1**

Table 7
ANOVA: effect of parameters on solution quality

	Df	Sum of square	Mean square	*F*-value	Significance
Main effects					
Crossover rate (Pc)	1	0.51	0.51	0.54	0.462
Mutation rate (Pm)	1	30.85	30.85	32.80	**0.000**
Population size (Ps)	1	24.56	24.56	26.12	**0.000**
# of generations (Mgen)	1	16.97	16.97	18.05	**0.000**
Two-way interactions					
Pc × Pm	1	0.26	0.26	0.28	0.598
Pc × Ps	1	1.59	1.59	1.69	0.193
Pc × Mgen	1	0.16	0.16	0.17	0.676
Pm × Ps	1	0.00	0.00	0.00	0.946
Pm × Mgen	1	0.72	0.72	0.76	0.382
Ps × Mgen	1	0.91	0.91	0.97	0.326
Error	1589	1494.34	0.94		
Total	1599				

Df: degree of freedom.

Table 8
ANOVA: effect of new operators on solution quality

Main effect	Mean square	DF	*F*-value	Significance
Dynamic selection (DS)	100.330	1	72.88	**0.000**
Group mutation (GM)	135.967	1	98.76	**0.000**
DS * GM	37.277	1	27.08	**0.000**

ing H5 and H7; and (2) dynamic selection operator can obtain better solution than group mutation; however, the results may be biased by their interaction (as the two operators are interacting with each other). In Fig. 4, we depict the convergence processes of the GA for data set #5. With conventional roulette-wheel selection strategy, the solution seems to converge to a total cost of $19,875 but the quality is far worse than the reference solution $3474 [6]. Using dynamic selection strategy but using conventional exchange mutation, the solution converged to $6665, which is significantly improved but is still worse than the reference solution. Using dynamic selection and group mutation, the solution converged to $2812 in 31 generations (about 24 seconds). This solution is better than the reference solution. This indicates the potential superior performance of our proposed group mutation operator and the dynamic selection strategy, which can play a very important role in fast convergence.

The ANOVA results of computational time using the two new operators are presented in Table 9. The results indicate that: only group mutation has significant impacts on the computational time (significant at the 0.001 level), thereby rejecting H8 and there is no interaction between the two operators. This implies that dynamic selection is a more efficient operator than group mutation.

Table 9
ANOVA: effect of new operators on computational time

Main effect	Mean square	DF	*F*-value	Significance
Dynamic selection (DS)	0.157	1	9.91	0.002
Group mutation (GM)	0.436	1	27.49	**0.000**
DS * GM	0.036	1	2.29	0.131

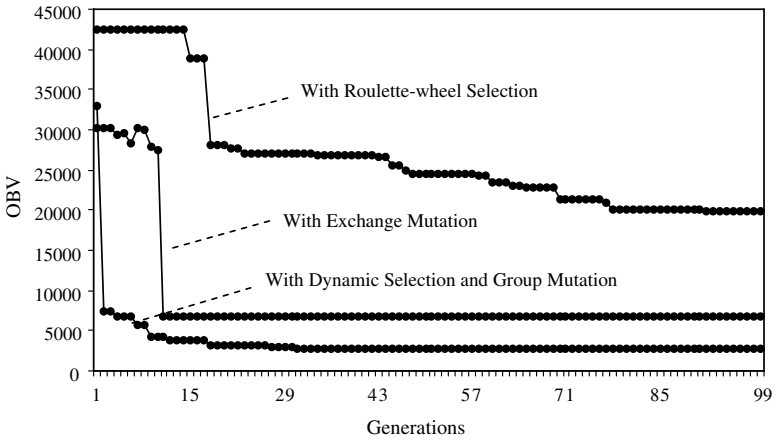

Fig. 4. Comparison of operators' performance.

Table 10
Paired *t*-test: dynamic evaluation and weighted sum

	Mean	SD	*t*-Value	Significance
(a) On solution quality				
GA with DS	12,411.072	34,066,011.913	1.984	0.000
GA with weighted sum	14,902.592	49,474,994.888		
(b) On computational time				
GA with DS	52.184	216.240	1.984	0.014
GA with weighted sum	50.978	231.447		

We also compared the performance between the proposed dynamic selection strategy and weighted-sum fitness function – one of the most popular approaches in handling multiple objectives. The results of the paired *t*-test are in Table 10. We found that: (1) the solution quality of the dynamic selection strategy is better than that of the weighted-sum fitness function; and (2) the computational times of the two approaches are not significantly different.

6. Discussion

Exploring the search space broadly and exploiting the best solution are two important principles in a GA study. The successful design of a GA depends on the balance of these two principles. For the crossover operator, we found that a crossover performed better if Pc was greater than 0.7, because a higher Pc expands the extent of exploration. For the mutation operator, a $Pm = 0.1$ performed better than $Pm = 0.5$. With more explorations, we found that a Pm with value from 0.001 to 0.01 produced better and consistent results. If Pm is too high, there will be much random perturbation and the offspring may lose resemblance to their parents. Furthermore, exchange mutation is for exploring the near-optimal area in the entire search space, while group mutation is for exploiting the best solution in the near-optimal space. The disturbance caused by the group mutations is far less than that of the exchange mutations. Thus, it is reasonable to set a comparatively high Pm.

Most previous GAs use the epsilon-constraint or weighted-sum approach to solve multi-objective problems. Thus, they involve only one of the important objectives at a time or must increase the importance of certain objectives via weights to improve evolution. We proposed a dynamic selection mechanism to determine suitable regions of the search space. Our method does not combine multiple objectives as a single objective but instead analyzes their correlations, simplifies and decomposes them into sub-problems, and then switches their focus when needed. As shown in Table 10, our proposed method performed better than the popular weighted-sum GA approach.

7. Conclusions

We have developed a two-layer chromosome structure to deal with problems that need to handle concurrent decisions. The hierarchical structure allows us to shorten the length of the chromosome; thus, it helps increase the probability of finding good schemata. It also helps reduce the number of active genes during operations and thus shorten the performance time. We have proposed a new dynamic selection strategy to deal with concurrent decisions that involve highly correlated objectives. We found that: without implementing the dynamic selection logic the GA would not properly converge to a good solution, and the dynamic selection strategy performs better than the popular weighted-sum approach without increasing computational time. We have also developed a new group mutation operator to increase the mutation probability. It can be used to further improve the solution quality for CF. However, if only use group mutation we may not be able to obtain good performance. Far better results are obtained when it was used with the dynamic selection strategy. Selecting proper values for GA parameters such as Pc, Pm, Ps, and Mgen is a critical decision. These parameters, except Pc, have more impacts on solution quality than computational time.

Several opportunities exist for further research. For instance, in this study we only compared the proposed dynamic evaluation strategy with the weighted-sum fitness method for a CMS. It would be interesting to see how well it performs compared with other multi-objective handling methods,

applied to other problem domains (e.g., scheduling or network design), and with different relationships between objectives (e.g., independent or low correlation). It may also be interesting to test the performance of group mutation in other clustering domains and find out whether its performance will be impacted by other GA operators such as crossover and selection method.

Acknowledgements

We gratefully acknowledge the valuable comments and suggestions from anonymous referees and the editor. The research was supported in part by the National Social Science Foundation, Hebei Natural Science Foundation and Doctoral Research Foundation of Education Department of China under the grant numbers 03BJY047, F2006000090 and B2004405, respectively.

References

[1] S. Alfa, M. Chen, S.S. Heragu, Integrating the grouping and layout problems in cellular manufacturing systems, Computer and Industrial Engineering 23 (1–4) (1992) 55–58.
[2] L. AL-Hakim, Discussion: A note on "a genetic algorithm approach for multiple criteria facility layout design", International Journal of Production Research 38 (4) (2000) 985–989.
[3] L. AL-Hakim, On solving facility layout problems using genetic algorithms, International Journal of Production Research 38 (11) (2000) 2571–2583.
[4] K.S. AL-Sultan, C.A. Fedjki, A genetic algorithm for the part family formation problem, Production Planning & Control 8 (8) (1997) 788–796.
[5] M. Bazargan-Lar, H. Kaebernick, A. Harraf, Cell formation and layout designs in a cellular manufacturing environment – a case study, International Journal of Production Research 38 (7) (2000) 1689–1709.
[6] M.P. Chandrasekharan, R. Rajagopalan, A multidimensional scaling algorithm for group layout in cellular manufacturing, International Journal of Production Economics 24 (1993) 65–76.
[7] H. Chou, G. Premkumar, C.H. Chu, Genetic algorithm for communication network design – an empirical study of the factors that influence performance, IEEE Transactions on Evolutionary Computation 5 (3) (2001) 236–249.
[8] C.H. Chu, Recent advances in mathematical programming for cell formation, in: A.K. Kamrani, H.R. Parsaei, D.H. Liles (Eds.), Planning, Design, and Analysis of Cellular Manufacturing Systems, Manufacturing Research and Technology, vol. 24, Elsevier, 1995, pp. 3–46.
[9] C.H. Chu, X.D. Wu, Genetic algorithms for manufacturing cell formation: A state-of-the-art review, Working Paper, School of Information Sciences and Technology, Pennsylvania State University, PA, 2002.
[10] D.B. Fogel, A. Ghozeil, Schema processing under proportional selection in the presence of random effects, IEEE Transactions on Evolutionary Computation 1 (4) (1997) 290–293.
[11] M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, Wiley Interscience Publication, MA, 1997.
[12] D.E. Goldberg, Genetic Algorithms: In Search, Optimization & Machine Learning, Addison-Wesley, Inc., MA, 1989.
[13] Y. Gupta, M. Gupta, A. Kumar, C. Sundaram, A genetic algorithm-based approach to cell composition and layout design problems, International Journal of Production Research 34 (2) (1996) 447–482.
[14] S.S. Heragu, S.R. Kakututi, Grouping and placement of machine cells, IIE Transactions 29 (1997) 561–571.
[15] J.H. Holland, Adaptation in Natural and Artificial System, Univ. of Michigan Press, Ann Arbor, MI, 1975.
[16] C.M. Hsu, C.T. Su, Multiobjective machine-component grouping in cellular manufacturing: A genetic algorithm, Production Planning & Control 9 (2) (1998) 155–166.
[17] A. Islier, A genetic algorithm approach for multiple criteria facility layout design, International Journal of Production Research 36 (6) (1998) 1549–1569.
[18] K.F. Man, K.S. Tang, S. Kwong, Genetic Algorithms: Concepts and Design, Springer, London, 1999.
[19] Z. Michalewicz, Genetic Algorithms + Data Structure = Evolution Programs, second ed., Springer-Verlag, New York, 1994.
[20] Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, in: Proceedings of the 4th Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 135–155.
[21] R. Poli, Why the schema theorem is correct also in the presence of stochastic effects, in: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, 2000, pp. 487–492.
[22] J.C. Poths, T.D. Giddens, S.B. Yadaw, The development and evaluation for an improved genetic algorithm based on migration and artificial selection, IEEE Transactions on Systems, Man, and Cybernetics 24 (1) (1994) 73–86.
[23] N.N. Schraudolph, R.K. Belew, Dynamic parameter encoding for genetic algorithms, Machine Learning 9 (1992) 9–21.
[24] N.C. Suresh, S. Kaparthi, Performance of fuzzy ART neural network for group technology cell formation, International Journal of Production Research 32 (7) (1994) 1693–1713.
[25] K.Y. Tam, S.K. Chan, Solving facility layout problems with geometric constraints using parallel genetic algorithms: Experimentation and findings, International Journal of Production Research 36 (12) (1998) 3253–3272.
[26] V. Venugopal, T.T. Narendran, A genetic algorithm approach to the machine component grouping problem with multiple objectives, Computer and Industrial Engineering 22 (1992) 469–480.
[27] U. Wemmerlöv, N.L. Hyer, Procedures for the part family/machine group identification problem in cellular manufacture, Journal of Operations Management 6 (1986) 125–147.