A parallel genetic algorithm for dynamic cell formation in cellular manufacturing systems

Published in 2008 in the International Journal of Production Research Vol. 46, 6389 -6413

Please cite this article as:

Defersha, F. M., and Chen, M., (2008) A Parallel Genetic Algorithm for Dynamic Cell Formation in Cellular Manufacturing Systems. International Journal of Production Research Vol. 46, 6389– 6413

The online version can be found at the following link:

http://dx.doi.org/10.1080/00207540701441962

A Parallel Genetic Algorithm for Dynamic Cell Formation in Cellular Manufacturing Systems

Fantahun M. Defersha and Mingyuan Chen*

Concordia University, Department of Mechanical and Industrial Engineering,
1455 de Maisonneuve W., Montreal, Quebec, Canada, H3G 1M8

Abstract

Instead of using expensive multiprocessor supercomputers, parallel computing can be implemented on a cluster of inexpensive personal computers. Commercial accesses to high performance parallel computing are also available on the pay-per-use basis. However, there are a limited report on the use of parallel computing in production research. In this paper, we present a dynamic cell formation problem in manufacturing systems solved by a parallel genetic algorithm approach. This method improves our previous work on the use of sequential genetic algorithm. Six parallel genetic algorithms for the dynamic cell formation problem were developed and tested. The parallel genetic algorithms are all based on the island model using migration of individuals but are differentiated by their connection topologies. The performance of the parallel genetic algorithm approach was evaluated against a sequential genetic algorithm as well as the off-shelf optimization software. The results are very encouraging. The considered dynamic manufacturing cell formation problem incorporates several design factors. They include dynamic cell configuration, alternative routings, sequence of operations, multiple units of identical machines, machine capacity, workload balancing, production cost and other practical constraints.

Keyword: *Cellular Manufacturing, Integer Programming, Genetic Algorithm, Parallel Computing*.

1 Introduction

In order to be successful in today's competitive manufacturing environment, managers must look for new approaches to facilities planning. Gupta and Seifoddini (1990) pointed out that one third of manufacturing companies in the US undergo major reconfiguration of production facilities every two years. The importance of good layout planning can also be seen from the fact that annually over 250 billion dollars are spent in the US alone on layouts that require planning and re-planning (Tompkins *et al*., 2003). Further, between 20% and 50% of the total costs within manufacturing are related to material handling. Effective and innovative facilities planning can

*For correspondence: mychen@encs.concordia.ca, Tel: (514) 848-2424 Ext. 3134; Fax: (514) 848-3175

reduce these costs by 10-30% (Tompkins *et al*., 2003). One effective approach to facilities planning is cellular manufacturing (CM). It is a production approach aimed at increasing production efficiency and system flexibility by utilizing the process similarities of the parts. It involves grouping similar parts into part families and the corresponding machines into machine cells. This results in the organization of production systems into relatively self-contained groups of machines such that each group undertakes an efficient production of a family of parts. Such decomposition of the plant operations into subsystems often lead to reduced paper work, reduced production lead time, reduced work-in-process, reduced labor, better supervisory control, reduced tooling, reduced setup time, reduced delivery time, reduced rework and scrap materials, and improved quality (Wemmerlöv and Johnson, 1997). In the last three decades, research in CM has been extensive and literature in this area is abundant. Comprehensive summaries and taxonomies of studies devoted to part-machine grouping problems were presented in Wemmerlöv and Hyer (1986), Kusiak (1987), Selim *et al*. (1998), and Mansouri *et al*. (2000). In most of the articles reviewed by these authors and those published in recent years, the cell formation problem has been considered under static conditions in which cells are formed for a single time period where product mix and demand are constant. In today's dynamic business environment, shorter time periods should be considered where the product mix and demand may vary from period to period. As a result, the best cell formation for one period may not be efficient for subsequent periods. A possible technique to counteract this problem is dynamic reconfiguration. To this end, there is a growing interest of research in developing models and solution procedures for dynamic cell reconfigurations over multiple time periods. Chen (1998) developed a mathematical model for dynamic reconfiguration in CM and proposed a decomposition approach to solve the model. The decomposed subproblems can be solved with less computational efforts, and dynamic programming is then employed to find a solution of the original problem. Balakrishnan and Cheng (2005) developed a dynamic programming approach to the multi-period CM reconfiguration problem similar to that for the general dynamic facility layout presented in Rosenblatt (1986). The method may be computationally prohibitive since for better results it requires large number of alternative static cell formations for each period. Defersha and Chen (2006a) developed a comprehensive model that incorporates several design factors in addition to dynamic cell configuration. These factors include alternative routings, lot splitting, sequence of operations, cell size limits, machine adjacency constraints, among others. The proposed model was solved using an off-the-shelf optimization package for small size problems. Later the authors developed a genetic algorithm to solve the comprehensive model efficiently Defersha and Chen (2006b). The use of genetic algorithm, simulated annealing and Tabu search for dynamic reconfiguration in CM were also reported in Wicks and Reasor (1999), Mungwattana

(2000), Tavakkoli-Moghaddam *et al.* (2005b), Tavakkoli-Moghaddam *et al.* (2005a), and Jeon and Leep (2006).

These search methods are generally able to find good solutions in reasonable amount of computing time. However, as they are applied to larger and complex problems, there is an increase in CPU time and computer memory to find adequate solutions. Moreover, in such large and complex problems, there can be a higher probability of for the search process being trapped in local optima. In such situations, the most promising choice is to use parallel implementations of the algorithms. We notice that in addition to using expensive multiprocessor supercomputers, parallel computing can be implemented on cluster of less expensive personal computers connected through a low cost network and using public domain software. Access to high performance parallel computing is nowadays available on pay-per-use basis, a business model called Utility Computing. In 1990, Baxter (Baxter, 1990) reported that commercial on-demand access to parallel computing is available at several government and private sector installations in US. In its November 2005 press release, IBM reported that it started commercial access to high performance computing in which clients worldwide can have on-demand access to over 5,200 CPUs (http://www-03.ibm.com/press/us/en/pressrelease/7949.wss). Tsunamic Technology Inc (http://www.clusterondemand.com/), Sun Microsystems Inc (http://www.network.com/), and 3Tetra (http://www.3tera.com/) are also examples of commercial access providers to parallel computing. Moreover, several universities and research institutes owe parallel computing facilities. Despite such availabilities of access to parallel computing, there are limited reports on the use of parallel computing in production research. For instance, out of the 178 papers reviewed in Chaudhry and Luo (2005) on the application of genetic algorithms in production and operations management, only 3 of them reported the use of parallel computing.

In this paper, we present a parallel genetic algorithm (PGA) for the design of dynamic cellular manufacturing systems which improves our previous work in Defersha and Chen (2006b) where a sequential genetic algorithm for solving the same problem was developed. To our knowledge, the use of PGA for static cell formation problem was reported only in Balakrishnan and Jog (1995) and has not been used for solving dynamic cell formation problems. As discussed in the literature, most mathematical models for static cell formation problem are NP-hard. Mathematical models for multiple period problems can be even more complex than their counterpart single period models due to the combinatorial nature of integer programming. To this end, powerful methods have to be developed to solve dynamic cell formation problems. In this paper, 6 parallel genetic algorithms are developed and tested. The parallel genetic algorithms are based on the island model using migration of individuals and differentiated by their connection topologies. The performance of the parallel genetic algorithm approach is evaluated

against a sequential genetic algorithm and a off-shelf optimization software. The results were very encouraging. The rest of this paper is organized as follows. Section 2 contains the problem description and its genetic representation. In section 3 we present a brief taxonomy of parallel genetic algorithms and the basic features of the class of PGA used to solve the dynamic CMS design problem. The important control parameters of this class of PGA are explained in Section 4. Numerical examples and computational results are in Section 5. Summary and conclusions are given in Section 6

2 Mathematical Model and Genetic Representation

As discussed before, the main objective of this work is to present a parallel genetic algorithm approach for dynamic CMS design. The CMS design problem is same as that addressed in our previous work (Defersha and Chen, 2006b) where problem and solution details were discussed. However, in order to provide a better comprehension of this paper, we describe the problem below and present its mathematical model and genetic representation.

2.1 Problem descriptions

Consider a manufacturing system consisting of a number of machines to process different parts. Each machine has a number of tools available on it and a part may require some or all of the tools on a given machine. A part may require several operations in a given sequence. An operation of a part can be processed by a machine if the required tool is available on that machine. If the tool is available on more than one machine type then the machines are considered as alternative routings for processing the part. The manufacturing system is considered for a number of time periods. One time period could be a month, a season, or a year. Each machine has a limited capacity expressed in hours during each time period. Machines can be duplicated to meet capacity requirements and to reduce or eliminate inter-cell movement. If additional machines are required in a given time period, the machines can be procured with certain limit. Assume that the demands for the part vary with time in a deterministic manner. Machines are to be grouped into relatively independent cells for each period with minimum inter-cell movement of the parts. In grouping the machines, it is also required that the workload of the cells should be balanced. Machines that cannot be located in a same cell due to technical and environmental requirements should be separated. To address this multiple time period cell formation problem, a mixed integer programming model is formulated. The objective of the model is to minimize machine maintenance and overhead cost, machine procurement cost, inter-cell travel cost, machine operation and setup cost, tool consumption cost, and system re-configuration cost for the

entire planning time horizon. The notations used in the model are presented below.

Indexes:

t Time index, $t = 1, 2, ..., T$,

i Part type index, $i = 1, 2, ..., I$,

j Index of operations of part i, $j = 1, 2, ..., J_i$,

k Machine type index, $k = 1, 2, ..., K$,

g Tool index, $g = 1, 2, ..., G$,

l Cell index, $l = 1, 2, ..., L$.

Input Data

$d_i(t)$ Demand for part i in time period t,

V_i Unit cost to move part i between cells,

B_i Batch size of part type i,

Φ_i Cost of subcontracting part i,

λ_{jig} A data equal to 1 if operation j of part i requires tool g; 0 otherwise,

δ_{gk} A data equal to 1 if tool g is available on machine; 0 otherwise,

h_{jik} Processing time of operation j of part i on type k machine in minutes,

w_{jik} Tool consumption cost of operation j of part i on machine type k,

μ_{jik} Setup cost for operation j of part i on type k machine,

P_k Procurement cost of type k machine,

H_k Maintenance and other overhead cost of type k machine,

O_k Operation cost per hour of type k machine,

C_k Capacity of one unit of type k machine,

LB_l Minimum number of machines in cell l,

UB_l Maximum number of machines in cell l,

I_k^+ Cost of installing one unit of type k machine,

I_k^- Cost of removing one unit of type k machine,

q $0 \leq q \leq 1$; a factor describing that the work load of a cell can be as low as $q \times 100\%$ from the average work load per cell,

M^∞ Large positive number,

S Set of machine pairs that cannot be placed in the same cell.

Decision Variables:

General Integer:

$N_{kl}(t)$ Number of type k machines assigned to cell l at the beginning of period t,

$y_{kl}^+(t)$ Number of type k machines added to cell l at the beginning of period t,

$y_{kl}^-(t)$ Number of type k machines removed from cell l at the beginning of period t.

Continuous:

$\eta_{jikl}(t)$ The proportion of the total demand of part i with its j^{th} operation performed by type k machine in cell l during period t,

$\bar{\eta}_i(t)$ The proportion of the total demand of part i to be subcontracted in time period t.

Auxiliary Binary Integer Variables:

$$r_{kl}(t) = \begin{cases} 1, & \text{if type } k \text{ machines are assigned to cell } l \text{ during time period } t, \\ 0, & \text{otherwise.} \end{cases}$$

$$p_{jil}(t) = \begin{cases} 1, & \text{if operation } j \text{ of part } i \text{ is processed in cell } l \text{ during period } t, \\ 0, & \text{otherwise.} \end{cases}$$

2.2 Objective Function and Constraints

Following the problem description and notations given in Section 2.1, the comprehensive mixed integer programming model for dynamic cellular manufacturing system design is presented below.

Objective:

Minimize

$$
\begin{aligned}
Z &= \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K} N_{kl}(t)\cdot H_k + \sum_{t=1}^{T}\sum_{k=1}^{K} P_k \cdot \left(\sum_{l=1}^{L} N_{kl}(t) - \sum_{l=1}^{L} N_{kl}(t-1)\right) \\
&+ \frac{1}{2}\sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{i=1}^{I}\sum_{j=1}^{J_i-1}\left(d_i(t)\cdot V_i \left|\sum_{k=1}^{K}\eta_{i,j+1,kl}(t) - \sum_{k=1}^{K}\eta_{ijkl}(t)\right|\right) \\
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{ijkl}(t)\cdot h_{ijk}(t)\cdot O_k \\
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{ijkl}(t)\cdot w_{ijk} \\
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J_i} \frac{d_i(t)\cdot\eta_{ijkl}(t)}{B_i}\cdot\mu_{ijk} \\
&+ \sum_{t=1}^{T}\sum_{l=1}^{L}\sum_{k=1}^{K}\left(I_k^{+}\cdot y_{kl}^{+}(t) + I_k^{-}\cdot y_{kl}^{-}(t)\right) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{I}\Phi_i\cdot d_i(t)\cdot\bar{\eta}_i(t) \qquad (1)
\end{aligned}
$$

Subject to:

$$d_i(t)\cdot\sum_{l=1}^{L}\sum_{k=1}^{K}\eta_{ijkl}(t) = d_i(t)(1-\bar{\eta}_i(t)) \qquad (2)$$

$$\eta_{ijkl}(t) \le \lambda_{jig}\times\delta_{gk} \qquad (3)$$

$$\sum_{k=1}^{K}\eta_{jikl}(t) = p_{jil}(t) \qquad (4)$$

$$\sum_{l=1}^{L} p_{jil}(t) = 1 \qquad (5)$$

$$C_k\cdot N_{kl}(t) - \sum_{i=1}^{I}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{ijkl}(t)\cdot h_{jik} \ge 0 \qquad (6)$$

$$\sum_{l=1}^{L} N_{kl}(t) - \sum_{l=1}^{L} N_{kl}(t-1) \ge 0 \qquad (7)$$

$$\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{ijkl}(t)\cdot h_{ijk} \ge \frac{q}{L}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J_i} d_i(t)\cdot\eta_{ijkl}(t)\cdot h_{ijk} \qquad (8)$$

$$LB_l \le \sum_{k=1}^{K} N_{kl}(t) \le UB_l \qquad (9)$$

$$N_{kl}(t) = N_{kl}(t-1) + y^{+}_{kl}(t) - y^{-}_{kl}(t) \tag{10}$$

$$N_{kl}(t) \leq M^{\infty} \cdot r_{kl}(t) \tag{11}$$

$$r_{kl}(t) \leq N_{kl}(t) \tag{12}$$

$$r_{k^1 l}(t) + r_{k^2 l}(t) \leq 1, \left(k^1, k^2\right) \in S \tag{13}$$

$$0 \leq \eta_{ijkl}(t) \leq 1 \tag{14}$$

$$0 \leq \bar{\eta}_{i}(t) \leq 1 \tag{15}$$

$$y^{+}_{kl}(t),\ y^{-}_{kl}(t),\ N_{kl}(t) \text{ are general integers and}$$

$$r_{kl}(t) \text{ is binary} \tag{16}$$

Model Objective Function: The objective function given in Eq. (1) comprises several cost terms. The first term is machine maintenance and overhead costs. The second term is machine procurement cost at the beginning of each period. In this cost term, $N_{kl}(t)$ stands for the number of type k machines assigned to cell l at the beginning of period t, with $N_{kl}(0) = 0, \forall k$. The third term of the objective function represents the inter-cell material handling cost. The forth, fifth, six and the seventh terms stand for machine operating cost, tool consumption cost, setup cost and machine relocation cost, respectively. The last term is the cost of subcontracting parts.

Model Constraints: The constraint in Eq. (2) ensures that if a part is not subcontracted, each operation of the part is assigned to a machine. Eq. (3) permits the assignment of an operation to a machine if and only if a tool required by the operation is available on that particular machine. Eqs. (4) and (5) allow the processing of operation j of part i in at most one cell in time period t. Eq. (6) guarantees that machine capacities are not exceeded. Eq. (7) implies that the number of type k machines used any period is greater than or equal to that of the previous period. This means that the model is not going to remove extra machines of any type if that type of machines happen to be in excess in a certain time period. The presence of extra machines in the system increases system flexibility and reliability by providing alternative routes during machine breakdown. Eq. (8) enforces workload balance among cells. Eq. (9) specifies the lower and upper bounds of cell sizes. Eq. (10) states that the number of type k machines in the current period in a particular cell is equal to the number of machines in the previous period, adding the number of machines being moved in and subtracting the number of machines being moved out of the cell. Eqs. (11) and (12) set the value of $r_{kl}(t)$ equal to 1 if at least one unit of type k machine is placed in cell l during period t or 0 otherwise. Eq. (13) ensures that machine pairs included in S are not placed in the same cell. Eqs. (14) and (15) limit the values of $\eta_{ijkl}(t)$ and $\bar{\eta}_{i}(t)$, respectively, within $[0, 1]$. Eq. (16) is the integrality constraint.

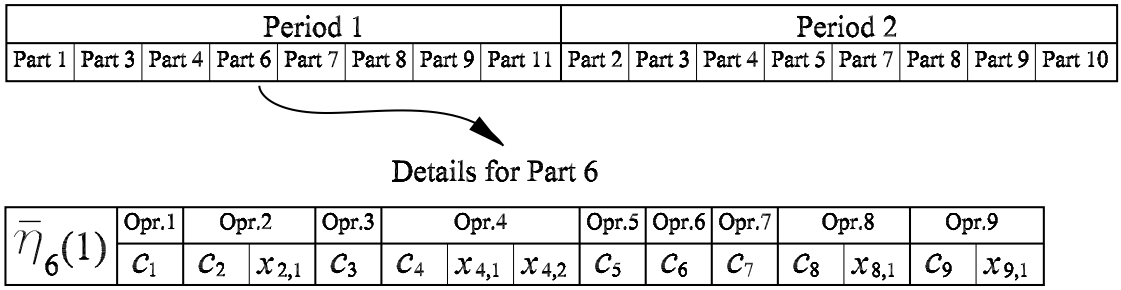


Figure 1: A chromosome structure for a two-planning period problem

2.3 The Chromosomal Encoding of a Solution

The chromosomal encoding of a solution is the first task in applying a genetic algorithm. In this research we developed a chromosomal representation of a solution which can satisfy some of the constraints of the model. Figure 1 illustrates a chromosome structure for a particular problem with two planning period and 11 part types. The variable $\bar{\eta}_i(t)$ takes a value in [0, 1] denoting the proposition of the total demand of part i subcontracted during period t. The variable c_j takes a value in $\{1, 2, ...L\}$ representing the cell in which operation j is performed. The x_j's assume values in [0, 1] and are used to calculate the proportions of the production volume among alternative routings. Operations without alternative routings do not have x's associated with them. In the chromosome structure, parts which do not have demand in a given period are not included in the segment of the chromosome representing the product mix for that particular period. For example parts 2, 5 and 10 shown in figure 1 do not appear in the first half of the chromosome since there is no such demand in period 1. The detailed representation of part 6 is shown in this figure. This part is assumed to have nine operations and the 2^{nd}, 8^{th} and 9^{th} operations have two alternative routings each. The 4^{th} operation has three alternative routings and the remaining operations have only one route each.

2.4 Decoding a Chromosome

The decision variables $\bar{\eta}_i(t)$ and $\eta_{jikl}(t)$ are determined by decoding a chromosome under consideration. The value of $\bar{\eta}_i(t)$ is directly read from the chromosome. For an operation with n alternative routings along machines $k^1, k^2, \cdots, k^n$, the values of $\eta_{jik^1l}(t), \eta_{jik^2l}(t), \cdots \eta_{jik^nl}(t)$ are determined using the sets of equations given in figure 2. In this set of equations the values of x_{j1}, x_{j2}, $\cdots, x_{j,n-1}$ are obtained from the chromosome and x_{jn} is set to 1. The subscript l takes the value c_j which is also obtained from the chromosome. As it was stated in Defersha and Chen (2006b), the chromosomal encoding and the decoding processes enable a randomly generated solution satisfying the constraints in Eqs. (2)–(5) of the mathematical model.

$$
\begin{aligned}
\eta_{jik^1l}(t) &= (1-\bar{\eta}_i(t)) \times x_{j1}, \\
\eta_{jik^2l}(t) &= (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times x_{j2}, \\
\eta_{jik^3l}(t) &= (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times (1-x_{j2}) \times x_{j3}, \\
\vdots &\quad \vdots \\
\eta_{jik^nl}(t) &= (1-\bar{\eta}_i(t)) \times (1-x_{j1}) \times (1-x_{j2}) \times \cdots \times (1-x_{j,n-2}) \\
&\quad \times x_{j,n-1} \times x_{jn}, \\
\& & \\
\eta_{jikl}(t) &= 0\,; \quad k \notin \{k^1, k^2, \cdots k^{n-1}\},\ l \neq c_j
\end{aligned}
$$

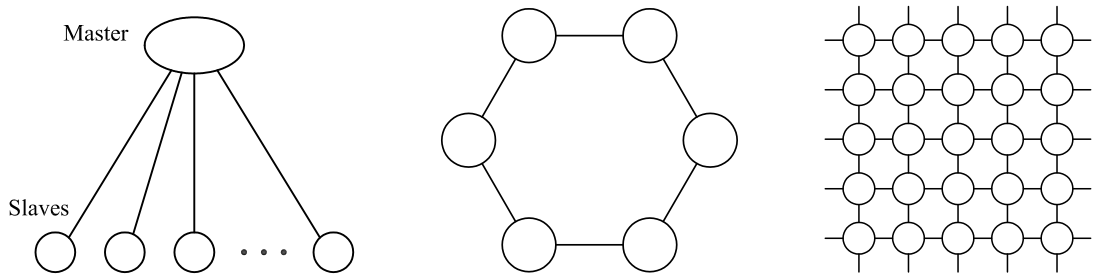
Figure 2: Equations required for decoding a chromosome

The continuous variables are determined by decoding a solution point under consideration as stated above. The corresponding integer variables $N_{kl}(t)$, $y^+_{kl}(t)$ and $y^-_{kl}(t)$ are determined using a problem specific heuristic discussed in Defersha and Chen (2006b). In that paper the genetic operators, constraint handling techniques, the steps and other several features of the sequential genetic algorithm were presented in details.

3 Parallel Genetic Algorithms

Sequential GAs (SGAs) have been shown to be very successful in many applications of different domains. However certain problems exist in some of the applications. One problem is that the fitness evaluation can be a very time-consuming process in particular, if the fitness value is to be determined by numerical simulations, for example. In other applications, the size of population needs to be very large and considerable size of computer memory is required to store the individuals. In addition, the SGAs search process may be trapped in a sub-optimal region of the search space as they search within a single population. These problems of the SGAs can be addressed with some form of parallel computation. Genetic algorithms can be parallelized in different ways. The detailed taxonomy can be found in Nowostawski and Poli (1999) and Cantú-Paz (2000). In those papers, we can recognize three major types of PGAs: (1) single-population master-slave PGAs, (2) multiple-population PGAs, and (3) fine-grained PGAs.

The master-slave PGA uses a single global population and the fitness evaluation is done by different processors as shown in Fig. 3-a. Furthermore, crossover and mutation operations may also be done in parallel. The nature of GA is not changed because the selection operation is done globally with the whole population using the master computer. This parallelism can



(a) Master-slave (b) Island model (c) Fine-grain

Figure 3: Classification of PGA In (a) and (b), each circle represent a processing element. In (c), each circle represent an individual in the population where preferable there is one processing element per each individual.

be very useful when the fitness evaluation is a time consuming process. The second approach of parallelizing GAs is an island-model PGA which uses multiple populations maintained by different processors (see Fig. 3-b). The subpopulations exchange individuals occasionally. This exchange of individuals is called migration. This class of PGAs are also called "coarse-grained" or "distributed" PGAs, because the communication to computation ratio is low, and they are often implemented on distributed memory computers or network of workstations. The third type of PGAs are fine-grained PGAs. They consist of a single spatially structured population (see Fig. 3-c). The population structure is usually a two dimensional rectangular grid, and there is one individual per grid point. Ideally, there should be one processor for each individual, so the evaluation of fitness is performed simultaneously for all the individuals. Selection and mating are restricted to a small neighborhood around each individual. The neighborhoods overlap, so that eventually the good traits of superior individuals can spread to the entire population. Fine grained PGAs are well suited for massively parallel SIMD computers, which execute the same single instruction on all processors. A comprehensive theoretical study of parallel genetic algorithm can be found in Cantú-Paz (2000).

In this research, we follow the island model PGAs for the design of dynamic cellular manufacturing systems. This class of PGAs has an appealing trait in that it often reduces the computational effort to solve the same problem as compared to SGAs, even on a single processor computer (Gordon and Whitley, 1993). This characteristic makes a difference with respect to other search algorithms in that island-model PGAs are not simple parallel versions of sequential algorithms. Thus they represent a new class of algorithms that search the solution space differently (Nowostawski and Poli, 1999). The reason for this can be found in the most significant characteristics of this class of parallelization (Alba and Troya, 2000): (1) their decentralized search, which allows speciation (different subpopulations evolve towards different solutions), (2) the larger diversity levels (many search regions are sought at the same time), and (3) exploitation inside these subpopulations, i.e., refining the better partial solutions found in each

subpopulations. In addition to these interesting characteristics, this class of PGAs can easily be implemented using public domain libraries such as MPI (Message Passing Interface) on a cluster of inexpensive computers connected by a slow network. These characteristics are the main motivating factors to consider island-model PGAs for designing dynamic cellular manufacturing systems. The psudocodes for SGA and island-model PGA are given in Fig. 4. The two psudocodes are essentially the same except for the communication routines of island-model PGA. Hence, the effort to convert SGA to island-model PGA is minimum. The complete algorithm of an island-model PGA consists of multiple copies of its sub-algorithm being executed in parallel where subpopulations exchange individuals periodically. This exchange of individuals is regulated by migration control parameters presented in the next section.

```
Initialize population
Repeat g = 1, 2, 3, ....
   Evaluate solutions in the population
   Perform competitive selection
   Apply genetic operators
Until convergence criterion satisfied
```

(a) SGA

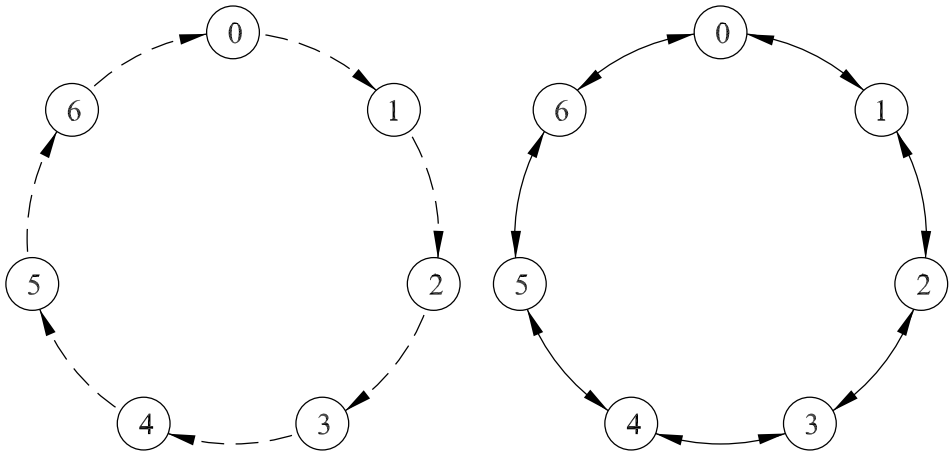
```
Initialize subpopulation
Repeat g = 1, 2, 3, ....
   Evaluate solutions in the subpopulation
   Perform competitive selection
   If it is time to communicate {
      Select migrants
      Send migrants to destinations
      Receive migrants from sources
   }
   Apply genetic operators
Until convergence criterion satisfied
```

(b) Island-model PGA sub-algorithm

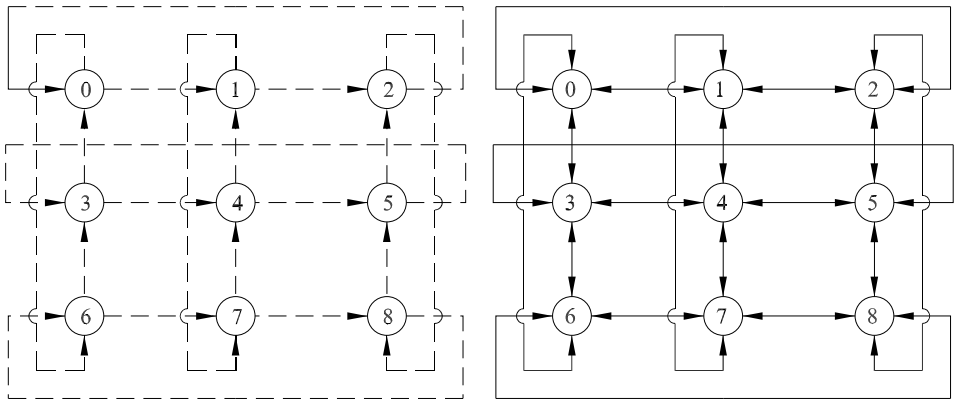
Figure 4: Psudocode for sequential and island-model parallel genetic algorithms.

4 Control Parameters of Island-Model PGA

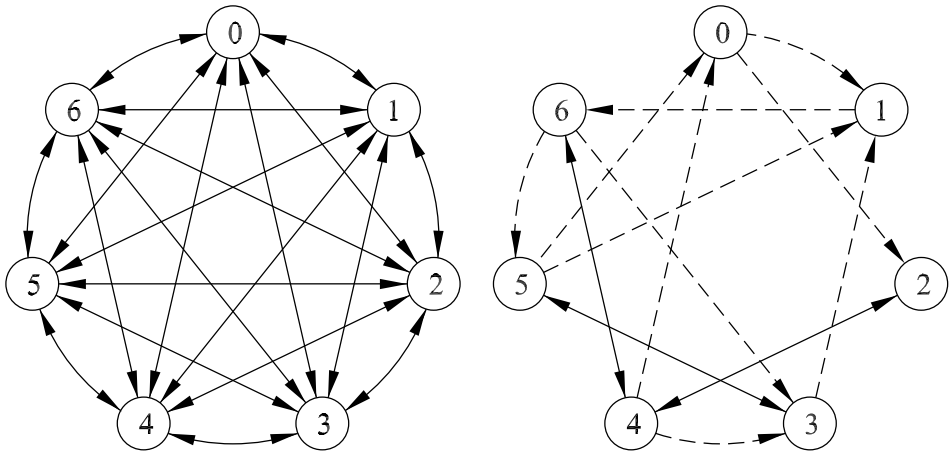
The island model PGA consists of several subpopulations which exchange individuals occasionally. This migration of individuals from one subpopulation to another is controlled by two parameters. The first is the topology that defines the connection among the subpopulations. The second parameter is the migration operator.



(a) Unidirectional ring (b) Bidirectional ring



(c) Unidirectional mesh (d) Bidirectional mesh

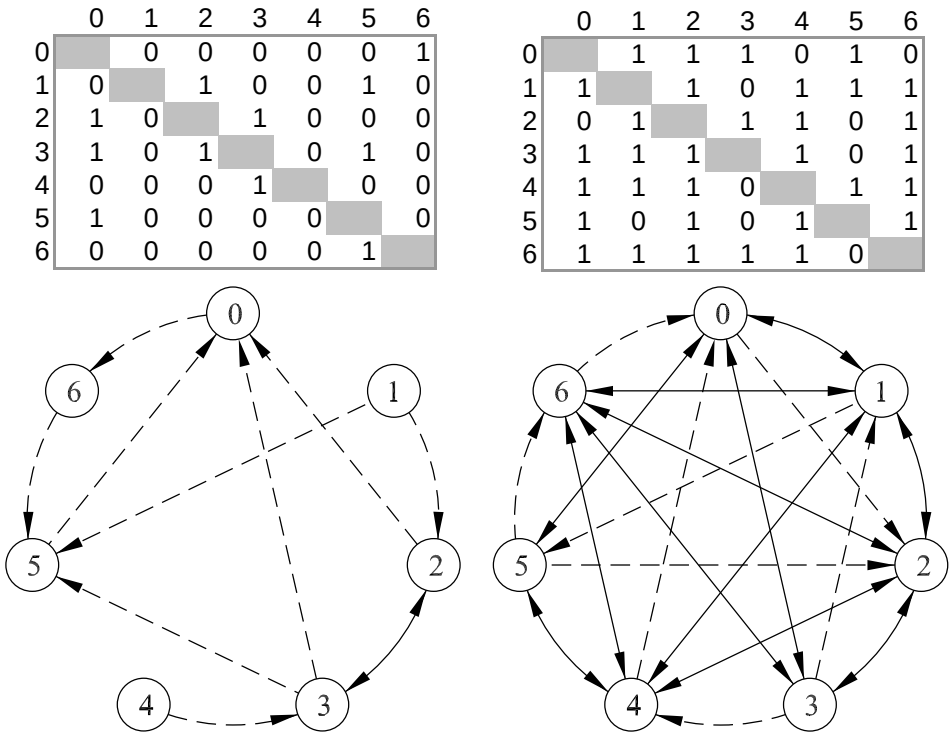


(e) Fully connected (f) Randomly connected

Figure 5: Connection topologies. *The doted lines represent one-way communications*

4.1 Connection Topology

Topology defines the connection among the subpopulations. In this paper we considers six different island model PGAs differentiated by their connection topologies as shown in Figure



(a) Generated for $\rho = 0.2$ (b) Generated for $\rho = 0.8$

Figure 6: Communication matrices and the corresponding topologies for different values ρ.

5. The first distinguishing property of a topology is the number of neighbors of each subpopulation. Unidirectional ring shown in Figure 5-a is a sparse topology as each subpopulation has only one neighbor. On the other hand, the fully connected topology shown in Figure 5-e is the densest topology. The second distinguishing feature of a topology is the migration route. The topologies in Figures 5-a to 5-e use fixed migration routes where the communication routes are known by each processor at the beginning of the computation and remain unchanged for the entire computation. The topology shown in Figure 5-f employs randomly generated migration routes for each communication epoch. In this topology, one of the processors will be designated to coordinate the communication in addition to evolving its subpopulation. This processor will randomly generate a communication matrix and broadcast it to other processors before each communication epoch. The communication matrix is a square matrix and its size equals to the number of processors. The entry $a_{i,j}$ of this matrix is binary number equal to 1 if migrants are sent from processor i subpopulation to that in processor j. The value of $a_{i,j}$ is determined using Eq. 17 where "rand()" is a random number generator and $\rho \in [0, 1]$ is a parameter controlling the density of communication topology. Figure 6 shows examples of communication matrices and their corresponding topologies. As can be seen from this figure, random topologies generated using lower values of ρ are sparser than those generated using higher values. The number of neighbors of a given subpopulation may change from epoch to epoch.

$$a_{i,j} = \begin{cases} 1 & \text{rand}() < \rho \text{ and } i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

4.2 Migration Operator

The migration operator controls the migration of individuals. This operator is composed of a number of attributes including: (1) the number of individuals undergoing migration, (2) the frequency of migration in numbers of generations and (3) the migration policy directing the type of individuals (best, according to fitness, random, etc.) from the source subpopulation to migrate to another subpopulation. It also direct the type of individuals (worst, random, etc.) to be replaced. The incoming individuals are combined with a subpopulation after selection and before the crossover and mutation operators are applied in the destination subpopulation.

5 Empirical Studies

The developed island-model PGAs were coded in C++ programming language using MPI message-passing library for communication. The codes were tested in a parallel computation environment composed of more than 800 computers each having one P4 processor (3.2 GHz, 2GB RAM). The test problems were run using up to 16 processors.

5.1 Performance Improvement through Parallelization

In order to demonstrate the performance improvement achieved through parallelization, we used the same example problem solved using the sequential genetic algorithm in our previous work in Defersha and Chen (2006b). In this research, the test problem was rerun using 1, 2, 4, 8, 12, and 16 processors applying the fully connected topology. Each processor uses separately seeded random number generator for the subpopulations in exploring different parts of the search space. Migration was taking place every 150 generations (migration frequency) and the number of migrants (the migration rate, MR) was determined in such a way that the total number of migrants joining a given subpopulation is about equal to 10% of its size. To meet this requirement in a fully connected topology, where each subpopulation receives M migrants from other subpopulations, M was determined such that $(N-1)\times \text{MR} \approx 0.1 \times P$ where N is the total number of subpopulations (or processors) and P is subpopulation size. Migrants were selected randomly from the source subpopulation and the individuals in a destination subpopulation were replaced randomly by the incoming migrants. A run was terminated after 120,000 generations.

Table 1: Comparison of LINGO, SGA and island-model PGA

	Objective function value in million									
		Test 1		Test 2		Test 3		Test 4		Test 5	
Time	LINGO	SGA	PGA	SGA	PGA	SGA	PGA	SGA	PGA	SGA	PGA
00:00:15	a	1.77717	1.77160	1.78740	1.76620	1.78492	1.78332	1.79666	1.78500	1.79058	1.76836
00:00:30	a	1.77335	1.76627	1.76618	1.76254	1.76558	1.76771	1.79343	1.77354	1.78241	1.76440
00:01:00	a	1.77133	1.76232	1.76491	1.76118	1.76411	1.76187	1.79094	1.76419	1.78074	1.76111
00:02:00	a	1.77064	1.76213	1.76467	1.76105	1.76215	1.76111	1.78993	1.75967	1.76727	1.76071
00:03:00	a	1.77061	1.76206	1.76466	1.76102	1.76204	1.76032	1.78990	1.75943	1.76629	1.76067
00:04:00	a	1.77056	1.76203	1.76465	1.76102	1.76184	1.76026	1.77294	1.75876	1.76615	1.76067
00:05:00	a	1.76672	1.76112	1.76448	1.76102	1.76182	1.75982	1.77244	1.75862	1.76582	1.76034
00:10:00	1.86962	1.76612	1.76112	1.76176	1.75981	1.76136	1.75973	1.77048	1.75828	1.76501	1.76031
00:12:00	1.81068	1.76612	1.76035	1.76176	1.75931	1.76135	1.75973	1.77041	1.75827	1.76501	1.76031
00:15:00	1.81068	1.76611	1.76035	1.76176	1.75931	1.76135	1.75973	1.77039	1.75826	1.76500	1.76031
00:20:00	1.81068	1.76611	1.75996	1.76176	1.75925	1.76121	1.75973	1.76936	1.75826	1.76499	1.76031
00:30:00	1.77401	1.76611	1.75996	1.76170	1.75882	1.76119	1.75973	1.76821	1.75826	1.76364	1.76031
00:45:00	1.76934	1.76501	1.75996	1.76170	1.75778	1.76069	1.75973	1.76815	1.75826	1.76297	1.75915
01:00:00	1.76934	1.76250	1.75996	1.76170	1.75778	1.76069	1.75973	1.76815	1.75825	1.76258	1.75915
01:30:00	1.76934	b	b	b	b	1.76069	1.75973	b	b	b	b
02:00:00	1.76934	b	b	b	b	b	b	b	b	b	b
03:00:00	1.76934	b	b	b	b	b	b	b	b	b	b
05:00:00	1.76865	b	b	b	b	b	b	b	b	b	b
10:00:00	1.76865	b	b	b	b	b	b	b	b	b	b
20:00:00	1.76865	b	b	b	b	b	b	b	b	b	b
25:00:00	1.76852	b	b	b	b	b	b	b	b	b	b

[a] A feasible solution was not found

[b] A termination criterion was met.

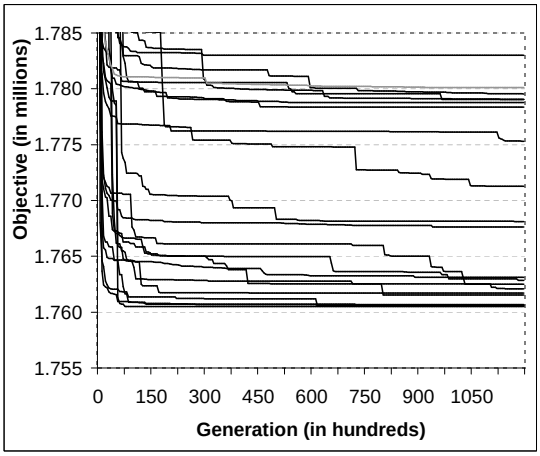
Table 1 shows a comparison of results from LINGO optimization software, SGA and a 16-processor island-model PGA. The genetic algorithms were run for 25 repetitions by varying the genetic parameters defined and explained in Defersha and Chen (2006b). The first five test runs were used as the basis of the comparison shown in Table 1. Under test 1, it can be seen that the solution found in 5 minutes using SAG and in just 30 seconds using PGA were better the solution found in more than 25 hours using LINGO. Under tests 1, 2, 4, and 5, the solutions found in 1 minute using PGA were better than those found in more than 1 hour using SGA. In test 3, the PGA took only 3 minutes to find a better solution than that found in more than $1\frac{1}{2}$ hours using SGA. In Figure 7, we further demonstrate the performance improvement achieved through parallelization by varying the number of processors. In Figures 7-a and 7-b are the convergence graphs from SGA and from the 16-processor PGA for the 25 repetitions. Figure 7-c shows the average convergence for these repetitions as we increase the number of processors. In Figure 7-a, we can see that there are several test repetitions that the SGA did not perform very well. From Figure 7-b, it is possible to see that the results from the 16-processor PGA converge very well for almost all of the genetic parameter settings. Figure 7-c shows that the average convergence curve improves as we increase the number of processors used in the computation. From these observations, it is quite clear that the parallel genetic algorithm is by far more efficient and robust than sequential implementation in solving the dynamic cell formation model.

5.2 Topology, Migration and Convergence

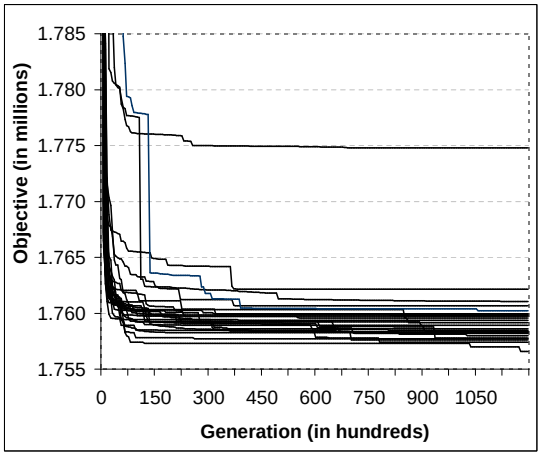
As discussed in the previous section, performance of the island model PGA is affected by connection topology, migration policy, migration frequency, and migration rate. We present an empirical study on the impact of these parameters on the convergence of island PGA approach in solving the dynamic cell formation problem.

5.2.1 Topology

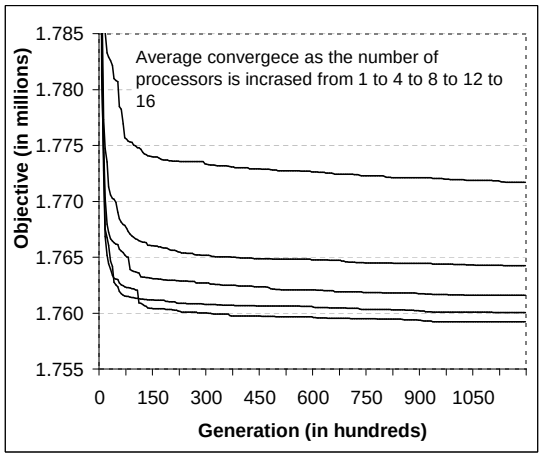
Intuitively, if a topology is densely connected, good solutions will spread fast to the subpopulations and may quickly take over the population. On the other hand, if the topology is sparsely connected, the subpopulations will be more isolated from each other and the full advantage of parallel computing may not be achieved. Figure 8 shows the convergence graphs of several island model PGAs with different connection topologies. Each curve represents an average convergence from 25 repetitions using different settings of genetic parameters and a particular connection topology. From this figure it can be seen that the randomly connected topology with $\rho = 0.5$ outperforms the other topologies. The parameter ρ devised in this research controls the



(a) Sequential GA (1 Processor)



(b) Parallel GA (16 Processors)



(f) Average Converagence

Figure 7: Convergence graphs of SGA and PGA using different number processors

density of the connectivity of the subpopulations. Lower values of ρ result in loosely connected topologies while higher values result in densely connected topology. Looking into Figure 8, it can be seen that the convergence graph of the randomly connected topology with $\rho = 0.2$ closely approximates that of the unidirectional ring which is the most sparse topology. With $\rho = 0.8$, it closely approximates that of the fully connected topology. Thus, the randomly connected topology with its control parameter ρ can alleviate the difficult problem of choosing a suitable topology out of several known topologies. Moreover, this topology has a randomness behavior which may be compatible with the random nature of genetic algorithm search processes.

5.2.2 Migration Policy

The migration policy determines which individuals migrate from the source subpopulation and which are replaced by migrants in the destination subpopulation. In this empirical study we consider three different migration policies: (1) random-replace-random, (2) best-replace-

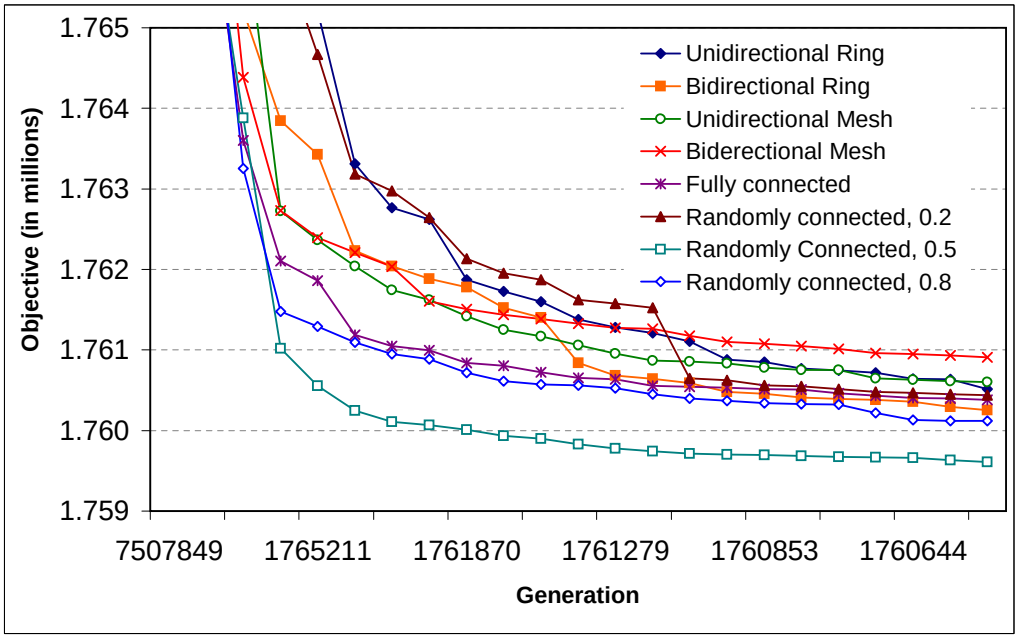
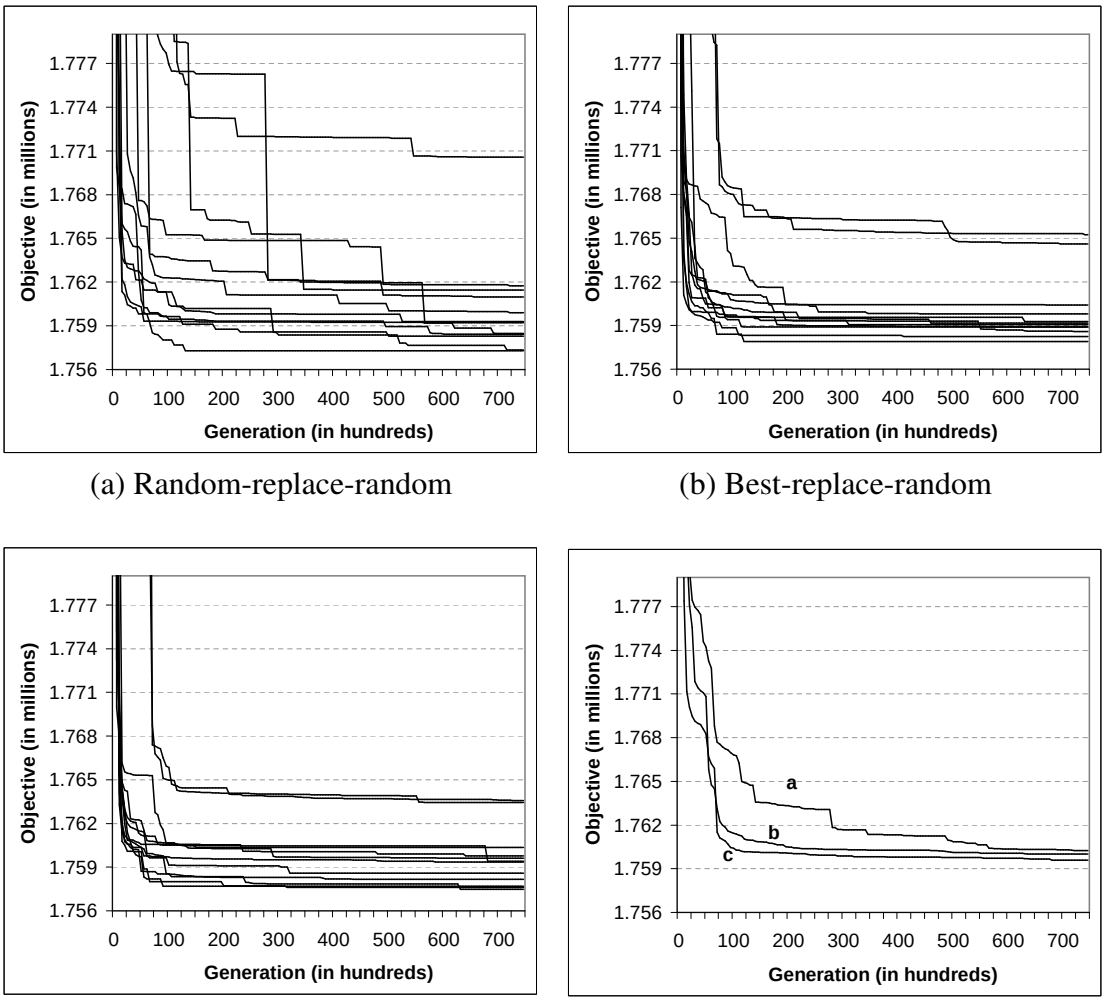


Figure 8: Average convergence graphs of 12-processor island model PGAs with different connection topologies

random, and (3) best-replace-worst. Figure 9 shows the convergence graphs of an island-model PGA for 12 repetitions using different genetic parameter settings and the three different migration policies. From these convergence graphs it can be seen that the island-model PGA is less robust to the genetic parameter settings when employing the first migration policy than the latter two policies. This can be seen from the fact that the individual convergence curves for the 12 repetitions in Figures 9-a are less alike to each other compared to those shown in Figures 9-b and 9-c. From Figure 9-d, we can see that the best-replace-worst migration policy slightly outperforms the other two migration policies. Hence, it may be considered as the first choice of migration policy in solving the dynamic cell formation model presented in Section 2.

5.2.3 Migration Frequency

This parameter of island-model PGA determines how often migrations occur. It is equal to the number of generations elapsed between each communication epoch. In this empirical study, we consider several settings of the migration frequency (MF) as shown in Figure 10. Each curve represents the average convergence for 12 repetitions using different genetic parameter settings and a particular MF value. As it can be seen from this figure, the average convergence graphs are similar for a wide range of migration frequencies from 60 to 2000. It can also be seen that for very small or very lager MF values, e.g., MF = 15 or 10000, the convergence behavior is very poor. This is understandable since with very small MF, relatively better solutions may spread quickly to other subpopulations and lead to premature convergence. As for very large



(a) Random-replace-random

(b) Best-replace-random

(c) Best-replace-worst

(d) Average convergence

Figure 9: Convergence graphs of a 12-processor island model PGA using different migrant policies

MF, the subpopulations will be too isolated to achieve the full advantage of parallel computing. Migration frequencies close to 300 generations can be recommended for solving problems presented in Section 2.

5.2.4 Migration Rate

The migration rate (MR) controls how many individuals migrate from each subpopulation to their respective destination subpopulations. In this empirical study, we consider several settings of MR from 2 to 40. When MR=2, the number of migrants received by each subpopulation in the fully connected 12-processor topology equals to 22 (2×11). When MR=40, this number is 440. In Figure 11-a, each curve represents an average convergence graph for 12 repetitions using different genetic parameter settings and a particular MF value. From this figure, it can be seen that the island model PGA converges to almost identical values of the objective function for

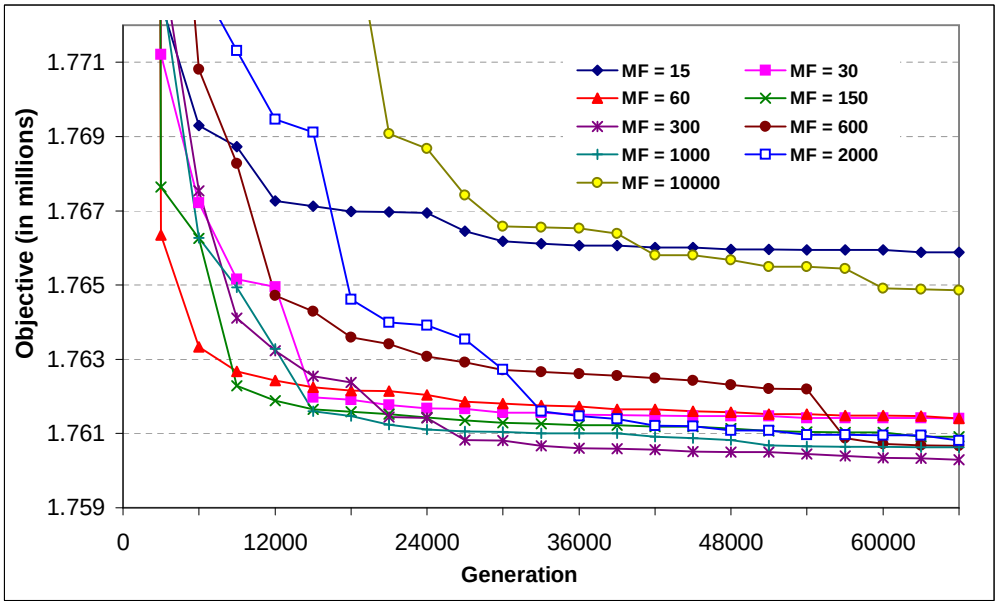
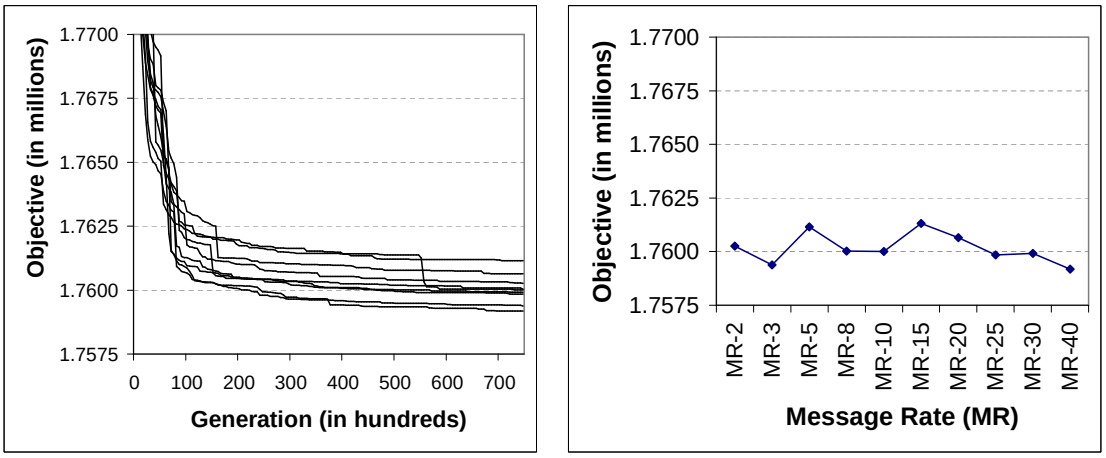


Figure 10: Average convergence graphs of a 12-processor island model PGA using different migration frequencies

a wide rang of MR. Figure 11-b shows the average final solution quality obtained for different values of MR. It can be seen that the solution quality varies without a clear trend as the migration rate is increased. These results show that the island model PGA is less sensitive to the migration rate in solving the dynamic cell formation problem presented in Section 2.



(a) Average Convergance (b) Average Final Solution

Figure 11: Average convergence graphs and final solutions using a 12-processor island model PGA for different values of migration rate MR

6 Summary and Conclusions

As it has been well established that most mathematical programming models for manufacturing cell formation problem in a single time period are NP-hard. Mathematical models for multiple period cell formation problems are normally much more difficult to solve due to the nature of combinatorial optimization. To this end, more powerful methods should be developed to solve dynamic cell formation problems. Some researchers recently proposed search methods based on genetic algorithm, Tabu search, simulated annealing, and other meta-heuristic search methods to solve these and similar problems. These search methods are generally able to find good solutions in reasonable amounts of computing time. However, as they are applied to larger and more complex problems, there is substantial increase in CPU time and memory required to find adequate problem solutions. Moreover, when these search methods are used to solving large and complex problems, there are higher probabilities that the search processes will be trapped in local optima. In such situations, the most promising choice is the implementation of the algorithms on parallel computing systems. Nowadays, such facilities are becoming more available to scientific computing as well as to industry applications while there have been limited reports on the use of parallel computing in production research. In this paper, we developed and tested different island model parallel GAs for solving dynamic manufacturing cell formation problems. We evaluated the performance of the parallel genetic algorithm against a sequential GA and an off-shelf optimization package. The parallel algorithm approach demonstrate substantial reductions of computing time and improves the search performances. The results found show the importance of using parallel genetic algorithms in solving dynamic cell formation problems where there are no reports on their use. We also evaluated the impacts of several parameters of the PGA on its performance in solving the dynamic cell formation model. These parameters include connection topology, migration policy, migration frequency and migration rate. The PGA with randomly connected topology outperforms the other PGAs having different topologies. This randomly connected topology proposed in this paper has a parameter to control the degree of connectivity and reduces the difficulty in choosing a suitable topology out of several known topologies. We plan to develop efficient parallel meta-heuristic algorithms and to solve other in production and operation problems. They include scheduling, facility layout, aggregate planning, inventory control, maintenance, supply chain management and capacity planning.

Acknowledgement: This research is supported by Discovery Grant from NSERC of Canada and by Faculty Research Support Fund from the Faculty of Engineering and Computer Science,

Concordia University, Montreal, Quebec, Canada. We also thank **RQCHP** – Réseau québécois en calcul de haute performance (http://www.rqchp.qc.ca/) – for its assistance in providing access to parallel computing facilities.

References

Alba, E. and Troya, J. M., 2000. Influence of the migration policy in parallel distributed gas with structured and panmictic populations. *Applied Intelligence*, **12**, 163–181.

Balakrishnan, J. and Cheng, C. H., 2005. Dynamic cellular manufacturing under multiperiod planning horizons. *Journal of Manufacturing Technology Management*, **16**, 516–530.

Balakrishnan, J. and Jog, P. D., 1995. Manufacturing cell formation using similarity coefficients and a parallel genetic tsp algorithm: Formulation and comparison. *Mathematical and Computer Modelling*, **21 (12)**, 61–73.

Baxter, F., 1990. Information technology and global changing science. Conference on Global Change: Economic Issues in Agriculture, Forestry and Natural Resources. Washington, D.C., November I9-21, 1990.

Chaudhry, S. S. and Luo, W., 2005. Application of genetic algorithms in production and operations management: a review. *International Journal of Production Research*, **43**, 4083–4101.

Chen, M., 1998. A mathematical programming model for system reconfiguration in a dynamic cellular manufacturing environment. *Annals of Operations Research*, **74**, 109–128.

Defersha, F. M. and Chen, M., 2006a. A comprehensive mathematical model for the design of cellular manufacturing systems. *International Journal of Production Economics*, **103**, 767–783.

Defersha, F. M. and Chen, M., 2006b. Machine cell formation using a mathematical model and a genetic-algorithm-based heuristic. *International Journal of Production Research*, **44**, 2421–2444.

Gordon, V. S. and Whitley, D., 1993. Serial and parallel genetic algorithms as function optimizers. Proceedings of the Fifth International Conference on Genetic Algorithms, edited by S. Forrest. Morgan Kaufmann, San Mateo, CA, pp. 177–183.

Gupta, T. and Seifoddini, H., 1990. Production data based similarity coefficient for machine componenet grouping in decisions in the desing of a cellular manufacturing systems. *International Journal of Production Research*, **28**, 1247–1269.

Jeon, G. and Leep, H., 2006. Forming part families by using genetic algorithm and designing machine cells under demand changes. *Computers & Operations Research*, **33**, 263–283.

Kusiak, A., 1987. The generalized group technology concept. *International Journal of Production Research*, **25**, 561–569.

Mansouri, S. A., Moattar Husseini, S. M., and Newman, S. T., 2000. A review of the modern approaches to multi-criteria cell design. *International Journal of Production Research*, **38**, 1201–1218.

Cantú-Paz, E., 2000. Efficient and accurate parallel genetic algorithms. Kluwer Academic Publishers, Massachusetts, USA,

Mungwattana, A., 2000. Design of cellular manufacturing systems for dynamic and uncertain production requirements with presence of routing flexibility. Ph.D. thesis, Virginia Polytechnic Institute and State University, Blackburg, VA.

Nowostawski, M. and Poli, R., 1999. Parallel genetic algorithm taxonomy. Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering System. Adelaide, Australia, pp. 88–92.

Rosenblatt, M. J., 1986. The dynamics of plant layout. *Management Science*, **32**, 76–86.

Selim, H. M., Askin, R. G., and Vakharia, A. J., 1998. Cell formation in groupt technology: Review, evaluation and direction for future research. *Computers & Industrial Engineering*, **34**, 2–30.

Tavakkoli-Moghaddam, R., Aryanezhad, M. B., Safaei, N., and Azaron, A., 2005a. Solving a dynamic cell formation problem using metaheuristics. *Applied Mathematics and Computation*, **170**, 761–780.

Tavakkoli-Moghaddam, R., Safaei, N., and Babakhani, M., 2005b. Solving a dynamic cell formation problem with machine cost and alternative process plan by memetic algorithms. *Lecture Notes in Computer Science*, **3777**, 213–227.

Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A., 2003. Facilities Planning. Wiley, New York, p. 10.

Wemmerlöv, U. and Hyer, N. L., 1986. Procedures for part-family/machine group identification problem in cellular manufacturing. *Journal of Operations Management*, **6**, 125–145.

Wemmerlöv, U. and Johnson, D. J., 1997. Cellular manufacturing at 46 user plants: implementation experiences and performance improvements. *International Journal of Production Research*, **35**, 29–49.

Wicks, E. M. and Reasor, R. J., 1999. Designing cellular manufacturing systems with dynamic part populations. *IIE Transactions*, **31**, 11–20.