A Linear Programming Embedded Genetic Algorithm for an Integrated Dynamic Cell Formation and Lot Sizing Considering Product Quality

Published in 2008 in the European Journal of Operational Research Vol. 187, 46-69

A Linear Programming Embedded Genetic Algorithm for an Integrated Cell Formation and Lot Sizing Considering Product Quality

Fantahun M. Defersha and Mingyuan Chen*

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

Abstract

Production lot sizing models are often used to decide the best lot size to minimize operation cost, inventory cost, and setup cost. Cellular manufacturing analyses mainly address how machines should be grouped and parts be produced. In this paper, a mathematical programming model is developed following an integrated approach for cell configuration and lot sizing in a dynamic manufacturing environment. The model development also considers the impact of lot sizes on product quality. Solution of the mathematical model is to minimize both production and quality related costs. The proposed model, with nonlinear terms and integer variables, cannot be solved for real size problems efficiently due to its NP-complexity. To solve the model for practical purposes, a linear programming embedded genetic algorithm was developed. The algorithm searches over the integer variables and for each integer solution visited the corresponding values of the continuous variables are determined by solving a linear programming subproblem using the simplex algorithm. Numerical examples showed that the proposed method is efficient and effective in searching for near optimal solutions.

Keywords: Genetic Algorithm, Cellular Manufacturing, Production Planning, Product Quality

1. Introduction

Various manufacturing production planning and inventory control problems have been studied extensively by many researchers. Different models and methods developed to solve these

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

problems can be found in widely used textbooks of production engineering or manufacturing systems (Riggs 1981, Singh 1996). Inventory control models from simple EOQ to more complicated MRP, Kanban and CONWIP models have been developed and widely used in today's manufacturing industries. Mathematical programming is also a powerful tool for solving complicated production planning problems, when product structures with multi-item and multi-levels are considered. A review paper on mathematical programming models for Kanban and MRP systems is given in Price *et al.* (1994). Other mathematical programming models for MRP- or Kanban-based production planning have also been developed as shown in Bard and Golany (1991), Bitran and Chang (1987), Price *et al.* (1995), Herer and Shalom (2000), Clark (2003), Berretta and Rodrigues (2004), Grubbström and Huynh (2006). Many of the mathematical models and solution methods were developed to solve problems in general manufacturing or service industries and can be widely applied. Prominent manufacturing features such as production flexibility and manufacturing cell formation were often not considered in developing production planning models (Chen 2001). On the other hand, manufacturing systems analyses tend to study more specific system characters such as job sequencing and scheduling, alternative process plans, as well as different ways of forming cells and material handling (e.g. Burbridge (1989), Chen and Cheng (1995), Jamal (1993)). As pointed out in Arvindh and Irani (1994), an integrated approach should be pursued in manufacturing system analysis, since different aspects of a system are interrelated in many ways. In addition, a comprehensive model consisting of different aspects of the system can help one to understand the problem better. Integrated system approach can minimize the possibility of certain important aspects of the system being overlooked, while other issues are being studied.

One aspect of cellular manufacturing system that may be interrelated with production planning can be dynamic system reconfiguration. In most research articles, cell formation has been considered under static conditions in which cells are formed for a single time period with known and constant product mix and demand. In contrast, in a more realistic dynamic situation, a multi-period planning horizon is considered, where the product mix and demand in each period is different. This occurs in seasonally or monthly production. As a result, the cell configuration in one period may not be optimal in another period. To address this problem, several authors recently proposed models and solution procedures by considering dynamic cell reconfigurations over multiple time periods (e.g. Chen (1998), Wicks and Reasor (1999), Mungwattana (2000), Tavakkoli-Moghaddam *et al.* (2005b,a), Balakrishnan and Cheng (2005), Defersha and Chen (2006), Saidi-Mehrabad and Safaei (2006)). These methods assume that the production quantity is equal to demand in each planning period. In reality, however, production quantity may not equal the demand as it may be satisfied from inventory or by subcontracting. Thus

production quantity should be determined from production planning decisions in order to determine the number and type of machines to be installed in the system. However, in order to determine the production quantities in each planning period the number and type of machines to be installed in manufacturing cells should in turn be known because of capacity consideration. Thus dynamic cell formation problem and the production planning are interrelated and may not be solved sequentially. Moreover, production quantity of an item may also depend on the production quantity of other items because of assembly requirements. Thus product structure (bill-of-materials) should be considered. Several researchers (Garvin (1988), Porteus (1986), Kim and Hong (1999), Jaber and Bonney (2003)) also showed that there is a relationship between product quality (defect rate) and production run length (lot size) and this issue should also be considered in production planning.

Based on the above discussion, we propose a mathematical programming model for an integrated dynamic cell formation and a multi-item multi-level capacitated lot sizing problem considering the impact of lot size on product quality. This model is an extension of the model discussed in Chen (2001) where product structure (bill-of-materials), machine capacity, workload balancing, alternative routings and impacts of lot sizes on product quality were not taken into account. However, integrated models of this type may impose computational difficulties and may not be solvable using off-the-shelf optimization software even for small size problems. Thus, efficient heuristic methods are required to solve the proposed model for problems of larger sizes. In this paper, we develop a heuristic method based on genetic algorithm to solve the proposed model. The algorithm searches over the integer variables and uses the simplex algorithm in ILOG CPLEX (ILOG Inc. 2003) to solve a linear programming subproblem and determined the corresponding valuers of the continuous variables for each solution point visited. Finally, the general branch and cut algorithm in ILOG CPLEX is used as a post-optimizer to further improve the solution found by the genetic algorithm. The remainder of this paper is organized as follows. In Section 2, we discuss the impact of run length on product quality and present a detailed description of the proposed dynamic cell formation and lot sizing model. The components of the LP embedded genetic algorithm are presented in Section 3. Numerical examples are in Section 4 to illustrate the model and computational efficiency of the algorithm. Discussion and conclusion are given in Section 5.

2. Model Development

In this section, first we discuss the impact of run length on product quality as found in literature. Following this discussion, the problem description and the mathematical formulation

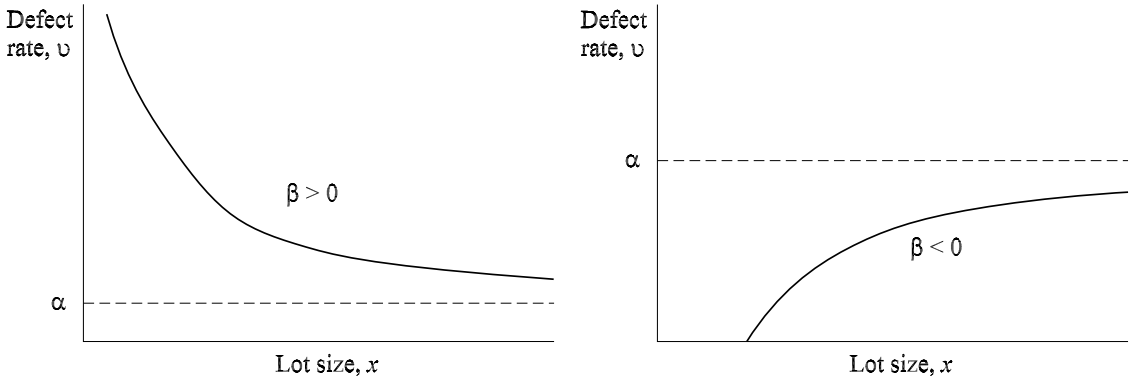
for an integrated dynamic cell formation and MRP lot sizing model is presented. The model incorporates the impact of run length on product quality. The choice of MRP as the production planning technique in the integrated model follows the recent report in Hyer and Wemmerlöv (2002) which stated most CMS users employ MRP in their production planning activities.

2.1. Impact of Run Length on Product Quality

Product quality has not been considered in most lot sizing models. At the same time, several managers pointed to a connection between run length and defect rates (Garvin 1988). Long runs often provide a stable environment–the opportunity to master required skills through repetition (Disruptive philosophy). As operators become familiar with the production process, defect rates normally fall, and eventually diminish to a minimum when technical limits are reached and opportunities for learning are exhausted. Garvin (1988) pointed out that this observation was strongly supported by statistical analysis and concluded that the differences in run length alone explained 50-70% of the variation in defect rates. Contrary to this observation, Porteus (1986), Huge and Anderson (1988), Li and Cheng (1994), Schonberger and Knod (1994), Kim and Hong (1999), and Jaber and Bonney (2003) showed that product quality is positively affected by reduced lot-sizes (JIT philosophy). Among them, Porteus (1986) and Kim and Hong (1999) modeled the situation where a production process deteriorates with a certain probability from an initially in-control state to an out-of-control state during the production. Huge and Anderson (1988) stated that without even working on quality improvement, defect rates improve proportionally with the reduction in lot sizes. Prompt identification of defects is the major reason frequently given for justifying this positive relationship between short run length and product quality. Smaller lots get used up sooner; hence defective parts are identified earlier (Schonberger and Knod 1994). This reduces scrap and rework and allows sources of problems to be quickly identified and corrected. Thus there is an incentive to produce smaller lots, and have a smaller fraction of defective units.

Urban (1998) proposed a simple relation between lot size x and defect rate υ given by the equation $\upsilon = \alpha + \beta/x$ where α and β are determined using linear regression from historical data. In his model, the number of defective items in a lot size of x is given by υx which is equal to $\alpha x + \beta$. The constant α is in $[0, 1)$. The other constant β is negative for JIT philosophy and positive for disruptive philosophy. The relationships of υ versus x for the two different philosophies are plotted in Figure 1. This equation can be used to represent the influence of run length on product quality in a production planning model under the JIT philosophy or the disruptive philosophy. In this paper, we use Urban's approach to incorporate the impact of run length on product quality in the integrated dynamic cell formation and lot sizing model

presented in the following sections.



(a) Disruptive philosophy (b) JIT philosophy

Figure 1. The impact of lot size on product quality (Urban (1998))

2.2. Problem Description

Consider a manufacturing system consisting of a number of machines to process different parts. A production lot of a part may be split into sub-lots to be processed along different alternative routes. In each route several operations are performed on different machines in a given sequence. In addition, we consider the manufacturing system in 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 have one or more identical copies to meet capacity requirements and reduce/eliminate inter-cell movement. Bill-of-material of finished products is given. The independent demands for the parts vary from period to period in a deterministic manner. In planning the production, the production volume of a part in a given period should be derived from (1) the number of finished parts in storage, (2) the level of its independent demand, (3) the level of subcontracting, (4) defect allowance, (5) the level of inventory at the beginning of the planning period and those to be carried over to the next period, and (6) the quantities of the parent parts in the products structure. The defect allowances are to be calculated following the JIT or disruptive philosophy. To determine manufacturing cells, we assume that the machines will be grouped into separate cells with minimum inter-cell movement of the parts and the subsequent system reconfiguration should be planned. In planning the production and (re)configuring the system, it is also required that (1) the workload of the cells should be balanced, (2) machine capacities should not be exceeded (3) machines that cannot be located in the same cell should be separated, (4) machines that cannot be separated should be co-located, and (5) the number of machines in each cell should be with the lower and upper limits. The overall objective is to minimize the system costs due to machine procurement,

inter-cell materials handling, machine operation, system reconfiguration, part subcontracting, setup, inventory holding, and replacement of defective parts for the entire planning horizon. The notations used in the model are presented below.

Model Parameters:

T	-	Number of planning period indexed by $t = 1, 2, \cdots, T$
I	-	Number of part type indexed by $i = 1, 2, \cdots, I$
R_i	-	Number of route of part i indexed by $r = 1, 2, \cdots, R_i$
J_{ri}	-	Number of operations of part i in route r indexed by $j = 1, 2, \cdots, J_{ri}$
K	-	Number of machine types index by $k = 1, 2, \cdots, K$
L	-	Number of cell indexed by $l = 1, 2, \cdots, L$
$d_i(t)$	-	Demand for part i in time period t
Γ_i	-	The set of immediate successor items to item i
$a_{ii'}$	-	The number of item i needed by one unit of item i', where $i' \in \Gamma(i)$
m_{jri}	-	Index of the machine type used to process operation j of part i along route r
λ_{jri}	-	Processing time in minutes of operation j of part i along route r
S_{ri}	-	Cost to set up route r of part i
W_i	-	Replacement cost of one defective item of type i
H_i	-	Unit inventory carrying cost per period for item i
P_k	-	Procurement cost of type k machine,
O_k	-	Operation cost per hour of type k machine
C_k	-	Per period capacity of one unit of type k machine
R_k^+	-	Cost of installing one unit of type k machine,
R_k^-	-	Cost of removing one unit of type k machine,
V_i	-	Unit cost to move part i between cells,
Θ	-	A set of machine pairs $\{(k^a, k^b)/k^a, k^b \in \{1, \cdots, K\}, k^a \neq k^b$, and k^a cannot be placed in the same cell with $k^b\}$
Ω	-	A set of machine pairs $\{(k^c, k^d)/k^c, k^d \in \{1, \cdots, K\}, k^c \neq k^d$, and k^c should be placed in the same cell with $k^d\}$
LB_l	-	Minimum number of machines in cell l
UB_l	-	Maximum number of machines in cell l
q	-	Inter-cell workload balancing factor
M	-	Large positive number,

Model Variables:

General Integer

$N_{kl}(t)$	-	Number of type k machines in cell l during 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

Binary

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

$z_{ri}(t)$ - $\begin{cases} 1, & \text{if route } r \text{ of part } i \text{ is set up for production during time period } t \\ 0, & \text{otherwise.} \end{cases}$

$\eta_{jril}(t)$ - $\begin{cases} 1, & \text{if } j^{th} \text{ operation in route } r \text{ of part } i \text{ is processed in cell } l \text{ during time period } t \\ 0, & \text{otherwise.} \end{cases}$

Continuous

$x_{ri}(t)$	-	The production sub-lot size of item i along route r in period t
$\bar{x}_i(t)$	-	The quantity of item i subcontracted in period t
$I_i(t)$	-	Inventory level of product i at the beginning of time period t
$df_{ri}(t)$	-	Defect allowance of part type i along its r^{th} route in period t
$e_{ri}(t)$	-	Auxiliary variable

2.3. Objective Function and Constraints

Following the problem description and notations given in the previous section, the integrated mathematical model for cellular manufacturing system design and production planning is presented below.

Minimize:

$$\begin{aligned} Z &= \sum_{k=1}^{K} P_k \cdot \left(\sum_{l=1}^{L} N_{kl}(T) - \sum_{l=1}^{L} N_{kl}(0) \right) \\ &+ \frac{1}{2} \sum_{t=1}^{T} \sum_{l=1}^{L} \sum_{i=1}^{I} \sum_{r=1}^{R_i} \sum_{j=1}^{J_{ri}-1} V_i \cdot x_{ri}(t) \left| \eta_{j+1,ril}(t) - \eta_{jril}(t) \right| \\ &+ \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{r=1}^{R_i} \sum_{j=1}^{J_{ri}} x_{ri}(t) \cdot \lambda_{jri} \cdot O_{m_{jri}} \\ &+ \sum_{t=1}^{T} \sum_{l=1}^{L} \sum_{k=1}^{K} \left(R_k^+ \cdot y_{kl}^+(t) + R_k^- \cdot y_{kl}^-(t) \right) \\ &+ \sum_{t=1}^{T} \sum_{i=1}^{I} \Phi_i \cdot \bar{x}_i(t) \\ &+ \sum_{t=1}^{T} \sum_{i=1}^{I} \sum_{r=1}^{R_i} S_{ri} \cdot z_{ri}(t) \end{aligned}$$

$$+ \quad \sum_{t=1}^{T} \sum_{i=1}^{I} H_i \cdot I_i(t)$$

$$+ \quad \sum_{t=1}^{T} \sum_{i=1}^{I} W_i \cdot \sum_{r=1}^{R_i} df_{ri}(t) \tag{1}$$

Subject to:

$$\sum_{l=1}^{L} \eta_{jril}(t) = z_{ri}(t) \quad ; \quad \forall(j,r,i,t) \tag{2}$$

$$x_{ri}(t) \leq M \cdot z_{ri}(t) \quad ; \quad \forall(r,i,t) \tag{3}$$

$$I_i(t-1) + \sum_{r=1}^{R_i} x_{ri}(t) + \bar{x}_i(t) - \sum_{r=1}^{R_i} df_{ri}(t) - I_i(t)$$

$$= d_i(t) + \sum_{i' \in \Gamma_i} a_{ii'} \cdot \left(\sum_{r=1}^{R_{i'}} x_{ri'}(t) + \bar{x}_{i'}(t) - \sum_{r=1}^{R_{i'}} df_{ri'}(t) \right) \quad ; \quad \forall(i,t) \tag{4}$$

$$df_{ri}(t) = \alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t) \quad ; \quad \forall(r,i,t) \tag{5}$$

$$\sum_{\forall(j,r,i)|m_{jri}=k} x_i(t) \cdot \eta_{jril}(t) \cdot \lambda_{jii} \leq C_k \cdot N_{kl}(t) \quad ; \quad \forall(k,l,t) \tag{6}$$

$$\sum_{l=1}^{L} N_{kl}(t) - \sum_{l=1}^{L} N_{kl}(t-1) \geq 0 \quad ; \quad \forall(k,t) \tag{7}$$

$$\sum_{i=1}^{I} \sum_{r=1}^{R_i} \sum_{j=1}^{J_{ri}} x_{ri}(t) \cdot \eta_{jril}(t) \cdot \lambda_{jri} \geq$$

$$\frac{q}{L} \sum_{l=1}^{L} \sum_{i=1}^{I} \sum_{r=1}^{R_i} \sum_{j=1}^{J_{ri}} x_{ri}(t) \cdot \eta_{jril}(t) \cdot \lambda_{jri} \quad ; \quad \forall(l,t) \tag{8}$$

$$N_{kl}(t) = N_{kl}(t-1) + y^{+}_{kl}(t) - y^{-}_{kl}(t) \quad ; \quad \forall(k,l,t) \tag{9}$$

$$LB_l \leq \sum_{k=1}^{K} N_{kl}(t) \leq UB_l \quad ; \quad \forall(l,t) \tag{10}$$

$$N_{kl}(t) \leq M^{\infty} \cdot r_{kl}(t) \quad ; \quad \forall(k,l,t) \tag{11}$$

$$r_{kl}(t) \leq N_{kl}(t) \quad ; \quad \forall(k,l,t) \tag{12}$$

$$r_{k^a l}(t) + r_{k^b l}(t) \leq 1 \quad ; \quad \left(k^a, k^b\right) \in \Theta, \quad \forall(l,t) \tag{13}$$

$$r_{k^c l}(t) - r_{k^d l}(t) = 0 \quad ; \quad \left(k^c, k^d\right) \in \Omega, \quad \forall(l,t) \tag{14}$$

$$y^{+}_{kl}(t), \;\; y^{-}_{kl}(t), \;\; N_{kl}(t) \in \{0,1,2,\cdots\} \quad ; \quad \forall(k,l,t) \tag{15}$$

$$\eta_{jril}(t), z_{ri}(t), \;\; r_{kl}(t) \in \{0,1\} \quad ; \quad \forall(j,r,i,l,t) \tag{16}$$

Model Objective Function: The objective function given in Eq. (1) comprises several cost terms. The first term is machine procurement cost in the entire planning horizon. In this cost term, $N_{kl}(0)$ stands for the number of type k machines available in the current job shop system. $N_{kl}(0) = 0, \forall k$ in the case of setting up a new system. The second term of the objective function represents the inter-cell material handling cost. The third, forth, and fifth terms stand for machine operating cost, machine (re)configuration cost and subcontracting cost. The last three terms are machine setup cost, inventory holding cost and replacement costs of defective parts.

Model Constraints: The constraint in Eq. (2) ensures that if a production route of a part is setup, an operation in that route will be assigned to a cell. Eq. (3) ensures that setup is performed (i.e. $z_{r,i}(t) = 1$) whenever there is production in period t (i.e. $x_{r,i}(t) > 0$). Eq. (4) is inventory balance constraint. It states that the beginning inventories together with the production and subcontracted quantities of each item in each period less the defect allowance and the inventory at the end of the period should meet the external demand and the dependent demand generated by its non-defective successor items. The constraint in Eq. (5) is required to determine the number of defective items under the disruptive philosophy. If the JIT philosophy is perused, the constraint in Eq. (5) will be replaced by the constraint in Eq. (17). In using the equation proposed in Urban (1998), the defective rate is zero for a production run length below certain value under JIT philosophy (see figure 1-b). In such cases, the right hand side of Eq. (17) will be negative and the constraint can be satisfied by assigning zero to $d_{ri}(t)$ and a positive value to the auxiliary variable $e_{ri}(t)$.

$$df_{ri}(t) - e_{ri}(t) = \alpha_{ri} \cdot x_{ri} + \beta_{ri} \cdot z_{ri}(t);\ \forall (r, i, t) \tag{17}$$

Capacity limitations of the machines are expressed in Eq. (6). Eq. (7) implies that the number of type k machines used in any time period is greater than or equal to that in 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. In this constraint, factor $q \in [0, 1)$ is used to determine the extent of the workload balance. If q is chosen close to unity, the allowable workload of each cell will be close to the average workload. This makes the workloads of the cells more or less equal. Eq. (9) 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. Eq. (10) specifies the lower and upper bounds of cell sizes. 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 Θ are not placed in the same cell. Eq. (14) is to ensure that machine pairs included in Ω should be placed in the same cell. Eqs. (15) and (16) are integrality constraint.

2.4. Linearizing the Model

The model presented above is a nonlinear programming model due to the nonlinear terms in the second term of the objective function, the capacity constraint (Eq. (6)) and the workload balancing constraint (Eq. (8)). These terms may be linearized in order to (a) solve the model using the general branch and cut algorithm for small size problems and (b) use such exact algorithm as a post-optimizer where the final good solution found by a heuristic method is used as the starting incumbent solution. The nonlinear term $x_{ri}(t)\,|\eta_{j+1,ril}(t) - \eta_{jril}(t)|$ in the second term of the objective function can be linearized in two steps. In the first step, we introduce a binary variable $a_{jril}(t)$ and replace $x_{ri}(t)\,|\eta_{j+1,ril}(t) - \eta_{jril}(t)|$ by the quadratic term $x_{ri}(t) \cdot a_{jril}(t)$ with the additional constraints given in Eqs. (18)–(20). In the second step, we replace $x_{ri}(t) \cdot a_{jril}(t)$ by a continuous variable $b_{jril}(t)$ with additional constraints given in Eqs. (21)-(23). Similarly, the quadratic terms $x_i(t) \cdot \eta_{jril}(t)$ in the capacity and the workload balancing constraints can be replaced by another continuous variable $c_{jril}(t)$ with the added constraints given in Eqs. (24)-(26).

$$\eta_{j+1,ril}(t) - \eta_{jril}(t) \leq a_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{18}$$

$$-\eta_{j+1,ril}(t) + \eta_{jril}(t) \leq a_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{19}$$

$$a_{jril}(t) \in \{0,1\} \quad ; \quad \forall(j,r,i,l,t) \tag{20}$$

$$b_{jril}(t) \geq x_{ri}(t) + M \cdot a_{jril}(t) - M \quad ; \quad \forall(j,r,i,l,t) \tag{21}$$

$$b_{jril}(t) \leq x_{ri}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{22}$$

$$b_{jril}(t) \leq M \cdot a_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{23}$$

$$c_{jril}(t) \geq x_{ri}(t) + M \cdot \eta_{jril}(t) - M \quad ; \quad \forall(j,r,i,l,t) \tag{24}$$

$$c_{jril}(t) \leq x_{ri}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{25}$$

$$c_{jril}(t) \leq M \cdot \eta_{jril}(t) \quad ; \quad \forall(j,r,i,l,t) \tag{26}$$

3. Linear Programming Embedded Genetic Algorithm

In order to efficiently solve the model presented in the previous section for large data set, we develop a linear programming embedded genetic algorithm (LPEGA). For a given solution point, the values of the integer variables are obtained by decoding the solution representation and using a problem specific heuristic. To compute the corresponding values of the continuous variables and the value of the objective function, a LP sub-subproblem is solved using the simplex algorithm in ILOG CPLEX (ILOG Inc. 2003). The main idea of embedding a simplex algorithm in a meta-heuristic is similar to that presented in Teghem *et al.* (1995). The advantage of embedding an LP subproblem in the genetic algorithm can be explained as follows. For a given integer solution, there may be infinite combinations of the values for the continuous variables. However, by solving the LP subproblem, values that optimally correspond to the integer solution can be obtained easily. It is also important to note that, the solution of the LP subproblem satisfies several constraints having continuous variables which otherwise may be difficult to satisfy by using genetic search alone.

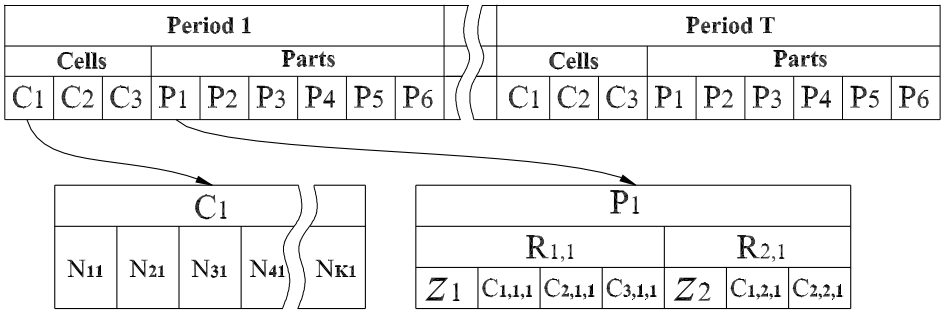


Figure 2. Solution representation

3.1. Chromosomal Encoding of a Solution

The chromosomal encoding of a solution is the first task in applying a genetic algorithm. The solution encoding of the proposed model involves the integer decision variables $N_{kl}(t)$, $\eta_{jril}(t)$, and $z_{ri}(t)$ enabling a randomly generated solution satisfy the constraint in (2). The constraints in (7), (9), (11), (12), and (13) are being taken care by a repair heuristic. Figure 2 illustrates a chromosome structure assuming 6 part types (P_1 to P_6) are to be precessed in 3 cells (C_1, C_2 and C_3) during T planning period. A segment corresponding to a given time period has two sub-segments: the first sub-segment, labelled "cells", represents the machine configurations and the second sub-segment, labelled "parts", represents the operation assignment of the parts to various cells. In this figure, C_1 in the cells sub-segment and P_1 in the parts sub-segment of

period 1 are shown in details. The gene N_{kl} in the chromosome takes a positive integer value. It is the number of type k machines installed in cell C_l in the period corresponding to its location in the chromosome structure. In the details for P_1, we assume that part type 1 has two alternative routes labelled $R_{1,1}$ and $R_{2,1}$ with three and two operations respectively. The gene Z_r takes an integer value in $\{0, 1\}$ to show whether route r is to be setup for production. The gene C_{jri} takes the value in $\{1, \cdots, L\}$ and represents the index of the cell in which operation j in route r of part i is to be processed.

3.2. Decoding a Chromosome

The values of decision variables $N_{kl}(t)$ and $z_{ri}(t)$ are directly read from the chromosome, whereas the decision variable η_{jril} is determined using Eq. (27). From this equation, it can be seen that the constraint in Eq. (2) can be satisfied by any randomly generated solution as shown in Eq. (28).

$$\eta_{jril}(t) = \begin{cases} z_{ri}(t) & ; \text{ If the subscritp } l = C_{jri} \\ \\ 0 & ; \text{ otherwise.} \end{cases} \tag{27}$$

$$\sum_{l=1}^{L} \eta_{jril} = \left(\sum_{\forall l \neq C_{jri}} \eta_{jril} \right) + \eta_{jri,c_{jri}} = 0 + z_{ri}(t) = z_{ri}(t) \tag{28}$$

3.3. Machine Assignment Repair Heuristics

The machine configuration decision $N_{kl}(t)$ directly obtained from the chromosome can be regarded as a preliminary value as the constraint in Eq. (7) and the machine separation constraint in Eq. (13) may be violated. A chromosome violating these constraints is repaired using machine assignment repair heuristic. Violation of the constraint in Eq. (7) can be repaired as follows. The heuristic first defines a new variable $M_{kl}(t)$ and set it equal to the preliminary values $N_{kl}(t)$ as obtained directly from the chromosome. After this step, the heuristic recalculates $N_{kl}(t)$ as follows. First, it sets the number of machines, $N_{kl}(1)$, in each cell for period 1 equal to $M_{kl}(1)$. The number of machines of each type installed in various cells for $t > 1$ are determined recursively as follows. Let $\widetilde{M}_k(t)$ and $\widetilde{N}_k(t)$ be equal to $\sum_{l=1}^{L} M_{kl}(t)$ and $\sum_{l=1}^{L} N_{kl}(t)$, respectively. If $\widetilde{N}_k(t-1) \leq \widetilde{M}_k(t)$, then the heuristic sets $N_{kl}(t) = M_{kl}(t)$. If $\widetilde{N}_k(t-1) > \widetilde{M}_k(t)$, then the heuristic first assigns those type k machines to the various cells to the level equal to $M_{kl}(t)$ and leaves the extra machines in the cells in which they were perviously installed.

The next repair is to correct the violation of the machine separation constraint given in Eq. (13) which is difficult to satisfy by using a penalty method. In order to perform this

repairing operation, the heuristic first arbitrarily forms a certain number of mutually exclusive sets of machines taken from Θ such that the machines in a given set can be placed in the same cell. Once these sets of machines are identified, a certain number of cells will be associated with each set. A cell will be associated with at most one set of machines while the set of machines can be associated with more than one cell. Finally, for a chromosome under repair, move out the machines in a given set from the cells that are not associated to this set and arbitrarily redistribute these machines among the cells associated to this particular set. This guarantees the fulfillment of the machine separation constraint. The heuristic has randomness behavior making it compatible with the genetic search. Once the decision variables $N_{kl}(t)$ are determined by decoding a chromosome and using the machine assignment heuristics, the configuration decision variables $y^{+}_{kl}(t)$ and $y^{-}_{kl}(t)$ can be determined using Eqs. (29) and (30) respectively. This last step satisfies the constraint in Eq. (9) of the model.

$$y^{+}_{kl}(t) = \begin{cases} N_{kl}(t), & \text{if } t = 1, \\ \\ max\{0,\ N_{kl}(t) - N_{kl}(t-1)\}, & \text{if } t > 1. \end{cases} \tag{29}$$

$$y^{-}_{kl}(t) = \begin{cases} 0, & \text{if } t = 1, \\ \\ max\{0,\ N_{kl}(t-1) - N_{kl}(t)\}, & \text{if } t > 1. \end{cases} \tag{30}$$

3.4. A Linear Programming Subproblem

The values of all the integer decision variables are obtained by decoding a chromosome and using the repair heuristic as explained in the previous two sections. This integer solution satisfies all the constraints involving only the integer variables, except the cell size and machine collocation constraints which are being taken care by the penalty method. The corresponding values of the continuous variables $x_{ri}(t)$, $\bar{x}_i(t)$, $I_i(t)$ and $df_{ri}(t)$ are determined by solving a linear programming subproblem given below. This LP subproblem is to minimize the sum of the inter-cell movement cost, operation cost, inventory holding cost and replacement cost of defective parts, subject to the constraints in Eqs. (3)–(6) and (8). In this LP subproblem, these constraints are renumbered as Eqs. (32)–(36).

Minimize

$$Z_{\text{LP}} \ = \ \frac{1}{2} \sum_{t=1}^{T} \sum_{l=1}^{L} \sum_{i=1}^{I} \sum_{r=1}^{R_i} \sum_{j=1}^{J_{ri}-1} V_i \cdot x_{ri}(t)\, |\eta_{j+1,ril}(t) - \eta_{jril}(t)|$$

$$
\begin{aligned}
&+ \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{r=1}^{R_i}\sum_{j=1}^{J_{ri}} x_{ri}(t)\cdot\lambda_{jri}\cdot O_{m_{jri}} \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{I}\Phi_i\cdot\bar{x}_i(t) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{I} H_i\cdot I_i(t) \\
&+ \sum_{t=1}^{T}\sum_{i=1}^{I} W_i\cdot\sum_{r=1}^{R_i} df_{ri}(t) \qquad (31)
\end{aligned}
$$

Subject to:

$$x_{ri}(t) \le M\cdot z_{ri}(t) \quad ; \quad \forall(r,i,t) \qquad (32)$$

$$
\begin{aligned}
&I_i(t-1) + \sum_{r=1}^{R_i} x_{ri}(t) + \bar{x}_i(t) - \sum_{r=1}^{R_i} df_{ri}(t) - I_i(t) \\
&= d_i(t) + \sum_{i'\,\in\,\Gamma_i} a_{ii'}\cdot\left(\sum_{r=1}^{R_{i'}} x_{ri'}(t) + \bar{x}_{i'}(t) - \sum_{r=1}^{R_{i'}} df_{ri'}(t)\right) \quad ; \quad \forall(i,t)) \qquad (33)
\end{aligned}
$$

$$df_{ri}(t) = \alpha_{ri}\cdot x_{ri} + \beta_{ri}\cdot z_{ri}(t) \quad ; \quad \forall(r,i,t) \qquad (34)$$

$$\sum_{\forall(j,r,i)|m_{jri}=k} x_i(t)\cdot\eta_{jril}(t)\cdot\lambda_{jii} \le C_k\cdot N_{kl}(t) \quad ; \quad \forall(k,l,t) \qquad (35)$$

$$
\begin{aligned}
&\sum_{i=1}^{I}\sum_{r=1}^{R_i}\sum_{j=1}^{J_{ri}} x_{ri}(t)\cdot\eta_{jril}(t)\cdot\lambda_{jri} \ge \\
&\frac{q}{L}\sum_{l=1}^{L}\sum_{i=1}^{I}\sum_{r=1}^{R_i}\sum_{j=1}^{J_{ri}} x_{ri}(t)\cdot\eta_{jril}(t)\cdot\lambda_{jri} \quad ; \quad \forall(l,t) \qquad (36)
\end{aligned}
$$

3.5. The Fitness Function

The purpose of the fitness function is to measure the fitness of the candidate solutions in the population with respect to the objective and constraint functions of the model. For a given solution, its fitness is given by Eq. (37) as the sum of the objective function of the model (Eq. (1)) and the penalty terms of constraint violations. The value of the model objective function is the sum of the objective function of the LP subproblem, machine procurement cost, and system reconfiguration cost. The penalty terms are to enforce the cell size and machine collocation constraints. The factors f_{cs} and f_{mc} are used for scaling these penalty terms. Finally, for a minimization problem, the raw fitness score F needs to be transformed so that the minimum raw fitness will correspond to the maximum transformed fitness. This is achieved by using the

equation in Eq. (38) where $\widetilde{F}$ is the transformed fitness function.

$$\begin{aligned} F &= \text{Model Objective Function} \\ &+ f_{cs} \cdot \sum_{t=1}^{T} \sum_{l=1}^{L} \max \left\{ 0, \quad \sum_{k=1}^{K} N_{kl}(t) - LB_l \ , \quad \sum_{k=1}^{K} N_{kl}(t) - UB_l \right\} \\ &+ f_{mc} \cdot \sum_{t=1}^{T} \sum_{l=1}^{L} \left\{ \sum_{\forall (k^c, k^d) \in \Omega} |r_{k^c l}(t) - r_{k^d l}(t)| \right\} \end{aligned} \tag{37}$$

$$\widetilde{F} = \begin{cases} 1 & ; \ if \ F_{max} = F_{min} \\ \frac{F_{max} - F}{F_{max} - F_{min}} & ; \ if \ \frac{F_{max} - F}{F_{max} - F_{min}} > 0.1 \\ 0.1 & ; \ otherwise. \end{cases} \tag{38}$$

3.6. Genetic Operators

Genetic operators evolve the population in creating promising candidate solutions to replace the less promising ones. These operators are generally classified as selection, crossover, and mutation operators. In the proposed genetic algorithm, the selection operator is implemented by simulating a biased roulette wheel (Goldberg 1989). Hong *et al.* (2000) suggested that each problem, even each stage of the genetic process in a single problem, may require appropriately defined multiple genetic operators for best results. In this section we present several crossover and mutation operators that are specific to solving the proposed model.

Crossover Operators: The crossover operators produce children by exchanging information contained in the parents. In this section we present several crossover operators tailored to the structure of the solution representation shown in Figure 2. These are:

- Single point crossover,
- Period swamp crossover,
- All-cell swamp crossover,
- All-part swamp crossover,
- Single-cell swamp crossover,
- Single-part swamp crossover and
- Route swamp crossover

Single-point crossover operator is a standard crossover operator used in most genetic algorithms. It randomly generates a single crossover point along the length of the chromosome and swamps the right-hand-side segments of the parents. The period-swap crossover operator randomly selects a period in the planning horizon and exchanges the segments of the parent chromosomes corresponding to that period. All-cell swap crossover operator randomly selects a cells sub-segment and exchanges this sub-segment between the parents. All-part swap crossover operator works similarly to exchange the parts sub-segment. Single-cell swap crossover operator randomly selects a single cell along the length of the chromosome and exchanges machine assignment information of this cell between parents. Single-part swap crossover operator works similarly to exchange the information about the various production routes of this part between parents. Route swamp crossover randomly selects a production route of a part along the length of the chromosome and exchanges the operations-cell assignment of this route between parents.

Mutation Operators: The mutation operators act on a single chromosome to alter the information contained in the genes. These operators are usually applied under certain probabilities much less than the crossover probabilities. In this section we present five mutation operators used in the genetic algorithm. These are:

- Machine mutator,
- Setup mutator,
- Part level cell mutator,
- Route level cell mutator,
- Operation level cell mutator,

Machine mutator applied along the entire length of a chromosome to step up or down the number of the machine of each type installed in each cell during each period. The setup mutator is applied to each of the $z_{ri}(t)$ in the chromosome to switch its value between 0 and 1. The part-level cell mutator randomly alters the c_{jri}'s of all the operations in all the routes of a part to other identical values in $\{1, 2, \cdots L\}$. The route-level cell mutator randomly alters the c_{jri}'s of all the operations of a given route of a given part to other identical values in $\{1, 2, \cdots L\}$. The part- and route-level mutation operators are applied during the first phase of the genetic search where the quest is to find the best configuration with independent cells. Operation-level cell mutator alters the value of each of c_{jri}'s of the operations of the parts in various route. However, this operator is applied for each operation independently and may result in different values of c_{jri}'s of the operations of a part. Hence, it may result in inter-cell movements. For this reason, this operator is applied in the second phase of the genetic search where the attempt is

to optimize the cost of inter-cell movement along other cost terms of the model. Moreover, this operator is applied at a lower mutation rate than part-level cell mutator to avoid unnecessary perturbations.

3.7. Two Search Phases

Generating independent cells is simpler than generating cells which optimize inter-cell movements and other cost terms of the objective function. Moreover, the best configuration with independent cells could be a neighborhood solution to the best configuration with optimal inter-cell movements as generating relatively independent cells is one of the design objectives. With this consideration, the search process was divided into two phases. In the first phase, the algorithm attempts to quickly find the best configuration with independent cells. To perform this task, the algorithm applies all the genetic operators except the single-point crossover and the operation-level cell mutator as these operators may result in solutions with inter-cell movement. In the second phase, the algorithm finds the configuration with the best inter-cell movements based on the solution found in the first phase. To perform this task, the algorithm applies all the genetic operators except the part-level and route-level cell mutation operators as these operators only results in independent cells. The steps of LPEGA are represented in the flow chart given in Figure 3 using the notations given below. The steps were coded in C++ and run on a Pentium-4 (2.93 GH, 512 MB Ram) PC computer. Simplex subroutines within ILOG CPLEX package (ILOG Inc. 2003) was used to solve the embedded linear programming model.

PS	Population size
p	Index for individual in a given population
g	Generation counter,
g_{max}	Maximum number of generations,
$Phase$	An indicator number which equals to 1 for the first phase or 2 for the second phase,
g_{phase}	Generation at which the value of $Phase$ should be set equal to 2 if it were not previously set to this value by other conditions,
b	Number of successive generations counted without any improvement of the best individual so far found,
b_{max1}	Maximum value of b at which point the second phase is to be entered if $Phase$ was equal to 1,
b_{max2}	Maximum value of b in the second phase at which point the search will terminate.

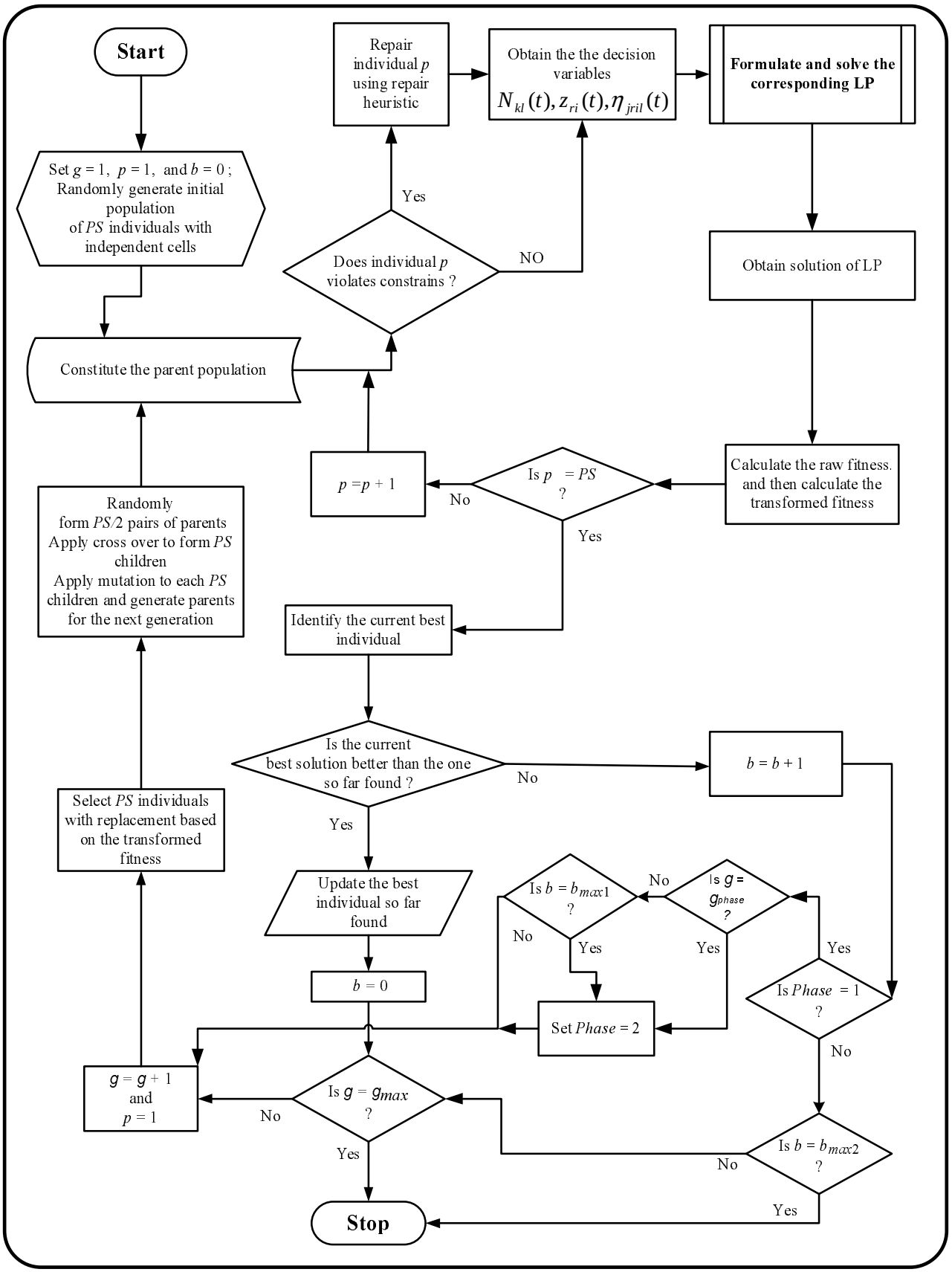


Figure 3. The steps of LPEGA

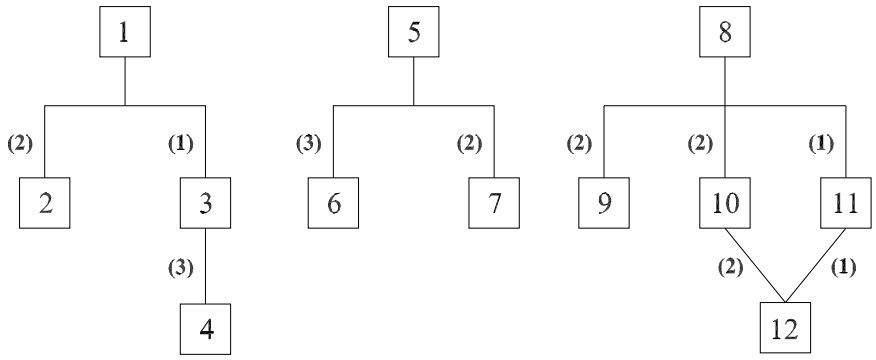


Figure 4. Bill-of-Materials for 12 part types

4. Numerical Examples

4.1. Model Analysis

In this section we present a numerical example showing some of the basic features of the proposed model and illustrating the need for an integrated approach in manufacturing system analysis. The example problem consists of processing 12 part types using 6 types of machines in 3 cells during 4 planning periods. The bill-of-materials describing assembly relationship of the parts is given in Figure 4. In this figure, the numbers inside the rectangles are part indices and those in the braces are the quantities of the lower level items required by one unit of their immediate predecessor items. In Tables 1 and 2 are data for the part types. In Table 1 are unit inventory holding, defective item replacement, inter-cell movement, and subcontracting costs. This table also shows the independent demand of the parts in the four planning periods. Table 2 shows data pertaining to the various production routes of the parts. These are the data of machine type and processing time required by each operation in each route, setup cost of each route and the constants α and β for calculating the defect allowances. Table 3 contains the data for the different machine types. The data includes machine procurement cost, unit capacity per period, installing and uninstalling costs, operation costs, and the number of machines of each type available at the beginning of the planning horizon.

With the data given in Figure 4 and Tables 1-3, the proposed model was solved using the general branch and cut algorithm in ILOG CPLEX where the solution generated by the proposed genetic algorithm was used as a starting incumbent solution. In solving the model, three different cases were considered. These three cases were differentiated by considering (1) JIT philosophy, (2) neither JIT nor disruptive philosophy and (3) disruptive philosophy, respectively. Decisions regarding production route selection, production lot sizing, defect allowance, inventory level, and the sequence by which the parts visit the cells are given in Tables 4 and 5 for

Table 1. Data for the parts

Part	Costs related to: Inventory holding	Replacing Defective	Inter-cell Movement	Sub-contracting	Independent Demand in Period t: 1	2	3	4
1	0.35	5	0.12	16	1500	1500	1400	200
2	0.35	10	0.13	24	1100	1400	1200	1300
3	0.35	10	0.12	24	1230	0	1350	1400
4	0.25	5	0.12	16	1200	1250	1400	1350
5	0.35	10	0.13	24	0	1800	1600	2400
6	0.50	10	0.12	24	1100	1500	1200	1600
7	0.35	10	0.13	24	0	1400	1600	300
8	0.35	5	0.12	24	0	1300	1600	2400
9	0.50	10	0.13	24	1100	1200	4000	1200
10	0.50	5	0.12	28	0	1200	1500	200
11	0.35	10	0.12	28	1200	1500	200	1520
12	0.35	10	0.12	28	2100	5000	1400	1300

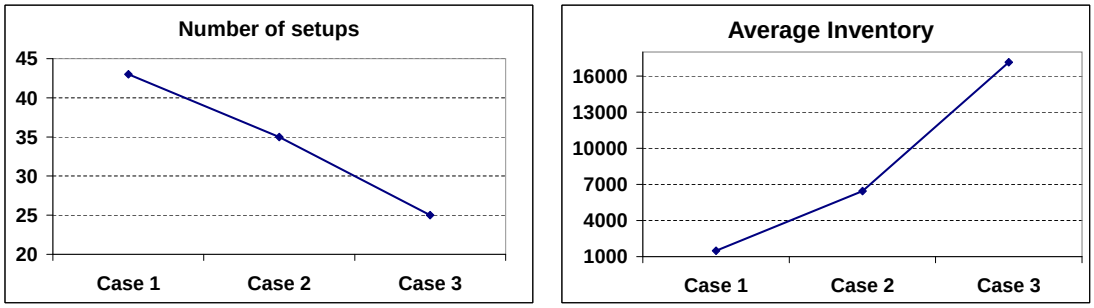
Table 2. Data for the parts (Continued)

Part	Route	Operation Data (m_{jri}, λ_{jri})	S_i	Disruptive-philosophy: α	β	JIT-philosophy: α	β
1	1	(3, 3)-(6, 4)	1920	0.02	180	0.08	-120
2	1	(1, 4)-(3, 5)-(6, 5)	1900	0.03	210	0.12	-140
	2	(1, 4)-(4, 5)-(3, 5)	1920	0.04	210	0.16	-140
3	1	(2, 2)-(5, 3)-(4, 3)	1400	0.03	150	0.12	-100
4	1	(3, 3)-(5, 4)-(2, 3)	1920	0.02	150	0.08	-100
5	1	(2, 4)-(6, 4)	1930	0.01	180	0.04	-120
6	1	(2, 3)-(3, 3)-(6, 2)	1770	0.02	180	0.08	-120
	2	(2, 4)-(6, 4)	1700	0.02	210	0.08	-140
	3	(2, 2)-(1, 3)-(4, 3)	1800	0.03	210	0.12	-140
7	1	(6, 4)-(2, 5)	1800	0.02	210	0.08	-140
	2	(2, 3)-(5, 3)-(3, 3)	1820	0.02	240	0.08	-160
8	1	(1, 4)-(4, 4)-(2, 4)	1890	0.02	180	0.08	-120
9	1	(6, 3)-(3, 3)	1900	0.03	240	0.12	-160
10	1	(1, 4)-(4, 3)-(3, 4)	1930	0.03	180	0.12	-120
	2	(3, 5)-(1, 4)-(4, 2)	1780	0.03	240	0.12	-160
11	1	(1, 3)-(4, 4)-(3, 3)	1920	0.02	240	0.08	-160
	2	(5, 3)-(4, 6)	1940	0.03	240	0.12	-160
	3	(1, 2)-(4, 3)-(5, 4)	1990	0.02	210	0.08	-140
12	1	(2, 2)-(6, 3)	1890	0.02	240	0.08	-160

Table 3. Data for the machines

Machine Type	Procurement Cost	Capacity (hrs)	Installation Cost	Uninstalling cost	Operation Cors/hrs	No of machines available at $t = 0$
1	1100	430	60	60	8	3
2	1100	400	80	80	9	3
3	1000	420	80	80	7	3
4	1200	430	90	90	7	3
5	1100	430	60	60	8	3
6	1100	410	70	70	7	3

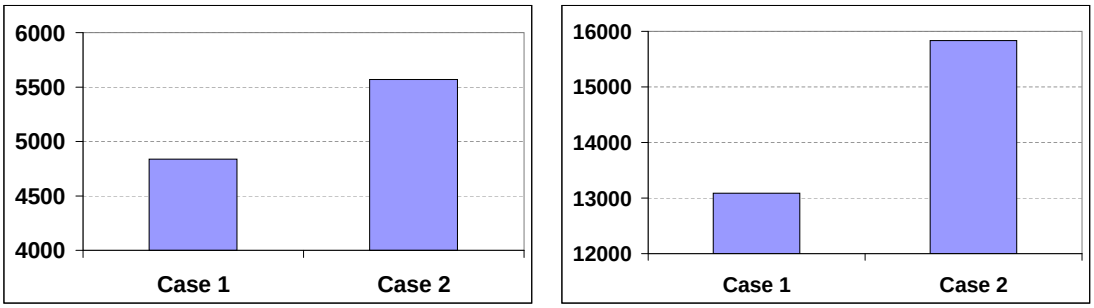
the the first two cases. Certain aspects of the solutions of the three cases are summarized in Figures 5, 6 and 7. From Table 4, one can see that the total number of setups $\left(\sum_{i=1}^{I}\sum_{r=1}^{R_i} z_{ri}(t)\right)$ is equal to 10 in the first period and equal to 11 in each of the remaining three periods. This makes the total number of setups under JIT philosophy being equal to 43. In Figure 5a, it can be seen that the total number of setups decreases as we shift from the first to the second and then to the third case. This results in an increase in the average inventory level given by $(\sum_{t=1}^{T-1}\sum_{i=1}^{I} I_i(t))/(T-1)$ as shown in Figure 5b. A decrease in the number of setups from 43 in the first case to 35 in the second case is also reflected in Figure 6(a) by an increase in the average run length from about 4,840 to 5,570. With the JIT approach, this increment of run length causes an increment of the total number of defective items from about 13,080 to 15,830 as shown in Figure 6(b). In Figure 5(a), it can be seen that there are less number of setups when the disruptive philosophy is in place. This reduced number of setups causes the average run length to increase from about 5,570 to 8,180 and the average number of defects to fall from 12,180 to 9,700 as shown in Figure 7(a) and 7(b), respectively. Here, it is important to note that 12,180 is the total number of defective items in the second case that would have been calculated assuming disruptive philosophy. The last columns corresponding to each period in Tables 4 and 5 show the sequence by which the parts visit the manufacturing cells along the selected routes. In these columns, it can be seen that the parts visit only one cell in most of the selected routes. This reduces the inter-cell movement for the better performance of the system. The cell formation and reconfiguration decisions for these three different cases are presented in Table 6. The required system reconfiguration to adapt to changes in product mix variations can be seen in this table. For example, under the JIT philosophy, cell-1 has 2 units of type 2 machine and 4 units of type 4 machine in the first period. In the second period, each of the quantities of these machine types was increased by one unit in this cell. In this table, it is also possible to see that the variations in the production planning decision among the three cases resulted in the variations in the cell formation and configuration decision. As a simple comparison, the total number of machines used in the system is equal to 25 under JIT philosophy and 30 under



(a) (b)

Figure 5. Number of setups and average inventory level when JIT, disruptive or neither philosophy is considered

disruptive philosophy. The reason for requiring less number of machines under JIT philosophy is the use of large number of setups. The use of large number of setups reduces the production volume in each setup and distribute production more equally in each planning period, avoiding peak production volumes that may require more machines. The number of machines used in case 2 (when neither philosophies are considered) is 23, less than the number of machines used in any other cases. The reason for this is that, when neither philosophies are considered, all the parts produced have no defects, making the total production volume less. However, with such number of machines it would be impossible to satisfy demand for the parts if the production process is imperfect and produce defective items. The numerical example demonstrated that the consideration of the impact of production run length on product quality may greatly affect the production planning decisions. The change in production planning decisions in turn may affect the cell formation and reconfiguration decisions and vice vera. Such inter-dependency on the various design and operational issues signifies the need for an integrated approach in manufacturing system analysis.



(a) (b)

Figure 6. (a) Average run length and (b) total number of defective as per the JIT philosophy when this philosophy is applied (Case 1) and not applied (Case 2)

Table 4. Solution under JIT philosophy

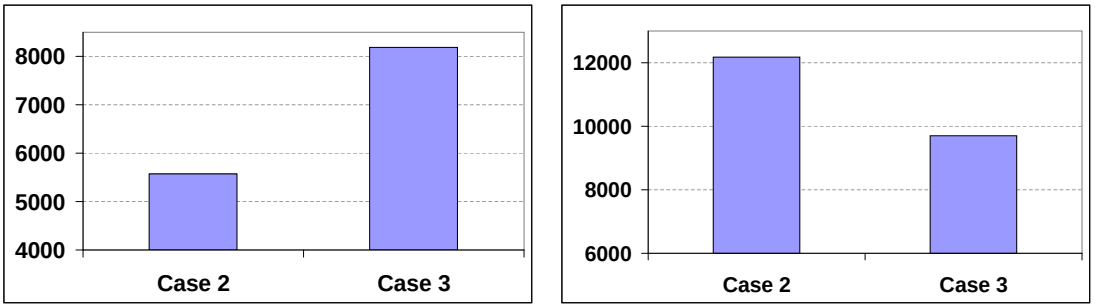
		Period 1					Period 2					Period 3					Period 4				
i	r	$z_{ri}(1)$	$x_{ri}(1)$	$df_{ri}(1)$	$I_i(1)$	Seq.*	$z_{ri}(2)$	$x_{ri}(2)$	$df_{ri}(2)$	$I_i(1)$	Seq.	$z_{ri}(3)$	$x_{ri}(3)$	$df_{ri}(3)$	$I_i(1)$	Seq.	$z_{ri}(4)$	$x_{ri}(4)$	$df_{ri}(4)$	$I_i(1)$	Seq.
1	1	1	1500	0	0	2–1	1	3239	139	1600	3–1	0	0	0	200		0	0	0	0	
2	1	1	4500	400	0	2–2–1	1	8477	877	0	3–3–1	1	1205	5	0	2–2–1	1	1318	18	0	2–1–1
	2	0	0	0			0	0	0			0	0	0			0	0	0		
3	1	1	2989	259	0	3–3–3	1	3409	309	0	3–3–3	1	1420	70	0	2–2–2	1	1477	77	0	3–3–3
4	1	1	10098	708	0	3–3–3	1	11359	809	0	2–2–2	1	5815	365	0	3–3–3	1	5924	374	0	3–3–3
5	1	0	0	0	0		1	1800	0	0	1–1	1	1600	0	0	1–1	1	2400	0	0	1–1
6	1	0	0	0	650		0	0	0	0		0	0	0	0		0	0	0	0	
	2	1	1750	0		1–1	1	6641	391		1–1	1	6370	370		1–1	1	9413	613		1–1
	3	0	0	0			0	0	0			0	0	0			0	0	0		
7	1	0	0	0	0		1	1750	0	0	1–1	0	0	0	0		0	0	0	0	
	2	0	0	0			1	3359	109		3–3–3	1	5043	243		3–3–3	1	5370	270		3–3–3
8	1	1	1300	0	1300	2–2–3	0	0	0	0		1	1826	26	200	2–2–2	1	2261	61	0	2–2–3
9	1	1	4023	323	0	1–2	1	1333	0	133	1–2	1	8303	836	0	1–1	1	6182	582	0	1–1
10	1	0	0	0	1200		0	0	0	0		0	0	0	0		0	0	0	0	
	2	1	4136	336		2–2–2	0	0	0			1	5614	514		2–2–2	1	5045	445		2–2–2
11	1	1	2725	58	167	2–2–2	0	0	0	0		1	2000	0	0	2–2–2	1	3870	150	0	2–2–2
	2	0	0	0			1	1333	0		2–2	0	0	0			0	0	0		
	3	0	0	0			0	0	0			0	0	0			0	0	0		
12	1	1	13268	901	0	1–1	1	6710	377	0	1–1	1	14609	1009	0	1–1	1	15283	1063	0	1–1
Total		10	46289	2985	3317	NA	11	49410	3011	1733	NA	11	53805	3438	400	NA	11	58543	3653	0	NA

* The sequence by which cells are visited

Table 5. Solution when neither JIT nor Disruptive philosophy is applied*

		Period 1					Period 2					Period 3					Period 4				
i	r	$z_{ri}(1)$	$x_{ri}(1)$	$df_{ri}(1)$	$I_i(1)$	Seq.*	$z_{ri}(2)$	$x_{ri}(2)$	$df_{ri}(2)$	$I_i(1)$	Seq.	$z_{ri}(3)$	$x_{ri}(3)$	$df_{ri}(3)$	$I_i(1)$	Seq.	$z_{ri}(4)$	$x_{ri}(4)$	$df_{ri}(4)$	$I_i(1)$	Seq.
1	1	1	1500	0	0	3–3	1	3100	139	1600	2–3	0	0	0	200		0	0	0	0	
2	1	1	4100	400	0	2–3–3	1	10100	1218	2500	1–1–3	0	0	0	1300		0	0	0	0	
	2	0	0	0			0	0	0			0	0	0			0	0	0		
3	1	1	2730	259	0	1–1–1	1	3133	314	33	2–2–2	1	2717	257	1400	2–2–2	0	0	0	0	
4	1	1	9390	708	0	1–1–1	1	10650	817	0	2–2–2	1	10900	839	1350	2–2–2	0	0	0	0	
5	1	1	1800	0	1800	1–3	0	0	0	0		1	1600	0	0	3–3	1	2400	0	0	3–3
6	1	0	0	0	1500		0	0	0	0		0	0	0	0		0	0	0	0	
	2	0	0	0			0	0	0			0	0	0			0	0	0		
	3	1	8000	932		2–2–2	0	0	0			1	6000	659		2–2–2	1	8800	1041		2–2–2
7	1	0	0	0	1400		0	0	0	0		0	0	0	0		0	0	0	0	
	2	1	5000	261		1–1–1	0	0	0			1	4800	243		2–2–2	1	5100	270		2–2–2
8	1	1	1300	0	1300	1–1–1	0	0	0	0		1	1983	42	383	1–1–3	1	2017	45	0	1–1–2
9	1	1	4900	486	1200	3–3	0	0	0	0		1	7966	904	0	3–1	1	5234	532	0	3–1
10	1	0	0	0	1200		0	0	0	0		1	5466	609	0	1–1–1	0	0	0	0	
	2	1	3800	336		1–1–1	0	0	0			0	0	0			1	4234	396		1–1–1
11	1	0	0	0	0		0	0	0	2183		0	0	0	0		0	0	0	0	
	2	1	2500	159		1–2	1	3683	320		2–2	0	0	0			1	3537	301		2–2
	3	0	0	0			0	0	0			0	0	0			0	0	0		
12	1	1	12200	887	0	2–3	1	8683	581	0	3–3	1	12333	899	0	3–3	1	13304	983	0	3–3
Total		12	57220	4428	8400	NA	6	39349	3390	6316	NA	9	53765	4453	4633	NA	8	44626	3566	0	NA

* The number of defective items were calculated assuming JIT philosophy based on a lot size required to produce $x_{ri}(t)$ good items where $x_{ri}(t)$ is the lot size determined when the philosophy is not applied



(a) (b)

Figure 7. (a) Average run length and (b) total number of defective as per the disruptive philosophy when this philosophy is not applied (Case 2) and applied (Case 3)

4.2. Computation Performance

In addition to the example problem discussed above, several other example problems were developed to evaluate the computational efficiency of the developed genetic algorithm. These examples were solved using the developed heuristic. The heuristic solutions were compared with those generated by ILOG CPLEX and lower bounds determined by optimally solving the model after relaxing certain constraints. If the workload balancing constraint (Eq. 8), the cell size constraint (Eq. 10) and the machine separation constraint (Eq. 13) are relaxed, an optimal solution of the relaxed problem would have all the machines placed in one cell. Thus, relaxing these three sets of constraints is equivalent to relaxing only Eq. (10) and Eq. (13) and setting the number of cells to be generated equal to unity (i.e., $L = 1$). With such relaxation and simplification, the proposed model can be rapidly solved using ILOG CPLEX for the example problems considered. Moreover, the optimal values of the objective function of the relaxed problems were observed to be well above the lower bounds determined by ILOG CPLEX after several hours of computation in solving the original model. Here, we present the computational results from two of the example problems: Problem 1 and Problem 2. The data for Problem 1 is presented in the previous section. With this data, after linearizing, the proposed model has a total of 3,012 variable; 1,348 of them are binary and 216 are general integer variables. The corresponding number of constraints is 5,960. Problem 2 consists of processing 25 part types using 12 machine types in 3 cells during 4 planning periods. For this problem, the linearized model has a total of 6,356 variables; 2,868 of them are binary and 432 are general integer variables. The corresponding number of constraints is 12,700.

In Table 7 is the convergence history of ILOG CPLEX and LPEGA in solving Problems 1 and 2. As can be seen from Table 7, the lower bound and the best objective function value for Problem 1 were 4.22568 and 4.66757, respectively, found by CPLEX after 46 hours of computation. At this point of the computation, the optimality gap was $(4.66757 - 4.22568)/4.66757 \times$

Table 6. Machine cell formation and reconfiguration decision under three different cases

		Case 1						Case 2						Case 3					
		No. of machine of type						No. of machine of type						Machine Type					
Cell	Period	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	1	0	2	0	0	0	4	1	3	3	1	3	0	0	1	2	1	1	2
	2	0	3	0	0	0	5	2	2	3	1	0	0	3	2	0	3	1	0
	3	0	3	1	0	0	5	2	0	3	1	0	0	3	3	2	3	3	0
	4	0	4	1	0	0	5	1	0	3	1	0	0	3	3	2	3	3	0
2	1	2	0	3	2	1	0	2	2	0	2	0	0	3	0	0	2	2	0
	2	1	2	2	2	2	0	1	2	2	2	3	0	0	4	0	0	2	3
	3	2	1	2	2	1	0	1	3	2	2	3	0	0	2	2	0	2	3
	4	2	0	2	2	1	0	2	3	2	2	3	0	0	2	2	0	2	3
3	1	1	2	2	1	2	0	0	0	2	0	0	4	0	4	3	0	3	3
	2	2	1	3	1	1	0	0	1	0	0	0	4	0	0	5	0	3	3
	3	1	2	2	1	2	0	0	2	0	0	0	4	0	1	2	0	1	3
	4	1	2	2	1	2	0	0	2	0	0	0	4	0	1	2	0	1	3
Total*		25						23						30					

* Total number of machines used in the system in the entire planning horizon= $\sum_{l=1}^{L}\sum_{k=1}^{K} N_{k,l,4}$

$100 = 9.47\%$. From this table, it can also be seen that starting form the first 45 seconds of computation time, LPEGA found solutions better than those generated using CPLEX in 46 hrs. The optimality gap of the final solution found using LPEGA with reference to the CPLEX lower bound was 5.69%. An improved lower bound for Problem 1 was also determined by solving the problem to optimality after relaxing the constraints in Eq. (10) and Eq. (13) and setting the number of cells to 1. The improved lower bound was 4.33638 and the optimality gap of the final solution found using LPEGA with reference to this improved lower bound was 3.22%. This suggested that the optimality gap of the LPEGA solution with reference to an optimal solution of the original problem is less than 3.22%. Encouraging results were also observed in solving Problem 2. Starting form the first 20 seconds of computation time, LPEGA found solutions better than those generated using CPLEX in 100 hrs. The lower bound and the best objective function value found using CPLEX after 100 hrs of computation were 7.64544 and 9.60757, respectively. The optimality gaps of the final solutions determined using CPLEX and LPEGA with reference to the CPLEX lower bound were 20.4% and 6.48%, respectively. The improved lower bound for Problem 2 was 7.96076. This implies that the optimality gap of the LPEGA solution with reference to an optimal solution of the original problem was less than 2.62%.

The computational experiment on Problem 1 was repeated for 14 parameter settings given in Table 8. The convergence graphs of LPEGA for these test runs are given in Figure 8. From

this figure it can be seen that LPEGA was able to converge to an acceptable solution for several parameter settings. In all of the test runs, solutions found using LPEGA in the first few hungered of generations were better than those found using CPLEX after more than 46 hours. We were also able to find several other parameter settings for which the genetic algorithm performs very well. This is an indication of the relative robustness of the proposed method. The encouraging results were obtained mainly due to (1) the solution approach of embedding a LP process into a meta heuristic, (2) systematically dividing the search into two phases, (3) constraint handling techniques and (4) the model specific genetic operators. For example, dividing the search into the 2 phases greatly improves the convergence behavior of LPEGA as shown in Figures 9.

Table 7. Comparison of LPEGA with CPLEX using two example problems

Problem 1				Problem 2			
	CPLEX	Objective			CPLEX	Objective	
Time	Lower Bound	CPLEX	LPEGA	Time	Lower Bound	CPLEX	LPEGA
00:00:02	4.22277	11.0157	6.89016	00:00:10	7.64214	28.6191	9.94213
00:00:04	4.22277	11.0157	5.40412	00:00:20	7.64214	28.6191	9.02501
00:00:09	4.22277	11.0157	4.90499	00:00:26	7.64214	28.0692	8.85474
00:00:24	4.22277	9.80000	4.74073	00:03:51	7.64214	28.0692	8.45511
00:00:45	4.22277	7.78200	4.63842	00:06:46	7.64214	15.6975	8.39693
00:01:54	4.22355	6.15200	4.58880	00:09:28	7.64215	12.5122	8.34907
00:05:19	4.22356	5.42330	4.53236	00:17:18	7.64215	12.0664	8.34561
00:07:37	4.22356	5.42330	4.49681	00:30:08	7.64215	10.8868	8.24639
00:10:24	4.22356	5.22184	4.49561	01:54:04	7.64215	10.4873	8.22151
00:31:18	4.22360	4.95993	4.48685	02:26:09	7.64217	10.1762	8.21806
00:31:25	4.22360	4.95993	4.48085	04:56:30	7.64236	10.1762	8.20888
01:12:13	4.22371	4.95733	4.48085	05:24:34	7.64244	10.1762	8.20469
01:37:13	4.22387	4.94089	*	06:56:21	7.64260	10.1762	8.18498
02:05:45	4.22397	4.94089	*	07:46:22	7.64264	10.1762	8.17949
05:05:41	4.22443	4.88142	*	08:02:33	7.64265	10.1762	8.17490
06:32:32	4.22454	4.79367	*	08:23:35	7.64267	10.1762	8.17490
09:39:14	4.22470	4.77058	*	17:35:51	7.64337	10.1533	*
14:18:17	4.22494	4.76690	*	20:32:36	7.64349	10.1497	*
14:44:19	4.22496	4.76394	*	32:16:58	7.64399	9.82868	*
14:58:10	4.22498	4.71146	*	36:22:09	7.64417	9.81213	*
15:04:39	4.22499	4.70108	*	43:34:55	7.64443	9.75404	*
17:22:05	4.22513	4.68635	*	50:54:01	7.64467	9.74824	*
18:13:55	4.22516	4.68337	*	62:49:00	7.64498	9.71716	*
32:58:37	4.22555	4.66757	*	83:08:22	7.64529	9.60757	*
46:16:26	4.22568	4.66757	*	100:00:00	7.64544	9.60757	*

* A termination criterion was met.
Note: Values are in 100,000.

5. Discussion and Conclusion

As discussed in the first section of this paper, an integrated approach should be pursued in manufacturing system analysis, since different aspects of a system are interrelated in many ways. In addition, a comprehensive model consisting of different aspects of the system can help

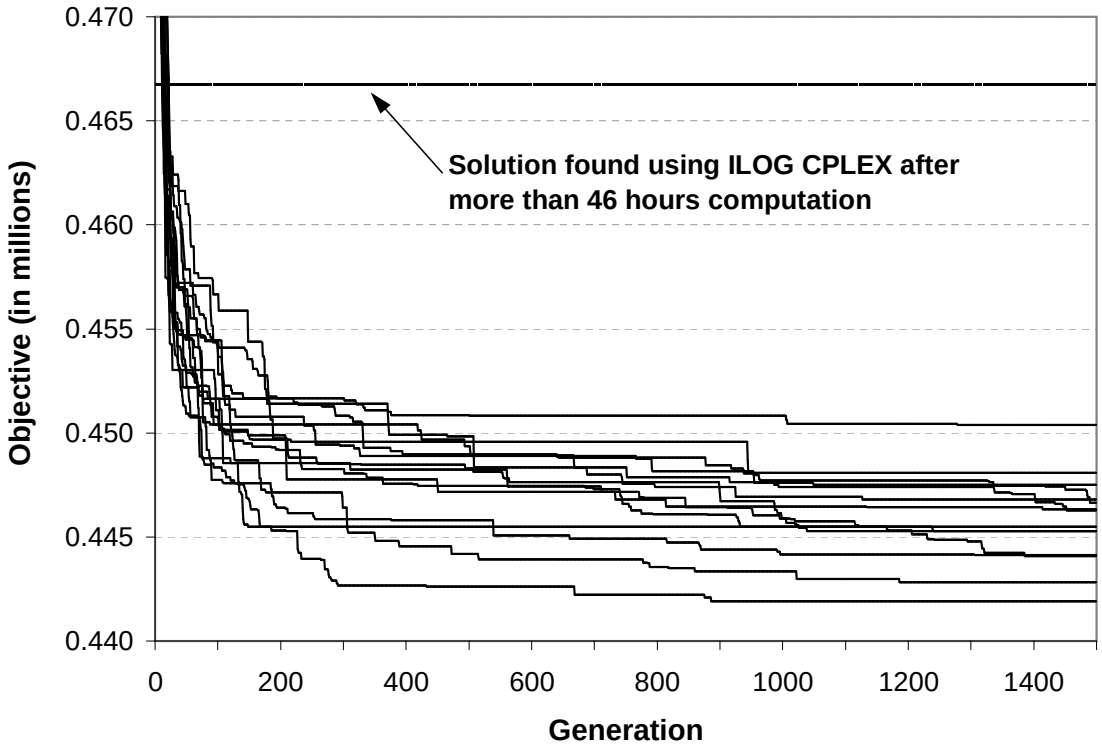


Figure 8. Convergence of LPEGA for the 14 test runs where the computation time is less than an hour for most of the tests

one to understand the problem better. In this paper, we developed a comprehensive mathematical model for integrated and dynamic manufacturing cell formation, considering a multi-item and multi-level lot sizing aspects and the impact of lot size on product quality. The model attempts to minimize production and quality related costs and incorporates a number of manufacturing attributes and practical constraints. These include dynamic system configuration, alternative routings, sequence of operations, machine capacity constraint, workload balancing, cell size limit and machine proximity requirements. The proposed model is NP-hard and may not be solved to optimality or near optimality using of-the-shelf optimization packages. To this end, we developed a heuristic method based on genetic algorithm to solve the proposed model. During the course of the search, the genetic algorithm interactively uses the simplex algorithm to solve a linear programming subproblem corresponding to each integer solution visited in the search process. Numerical examples were presented to demonstrate the need for an integrated approach in manufacturing system analysis and to illustrate the computational efficiency of the developed heuristic. Computational performance of the proposed heuristic search method is very encouraging based the results of the testing problems. Our future research in this area includes developing integrated methods in designing manufacturing systems and supply chain networks considering product quality, product recovery and disposal policies.

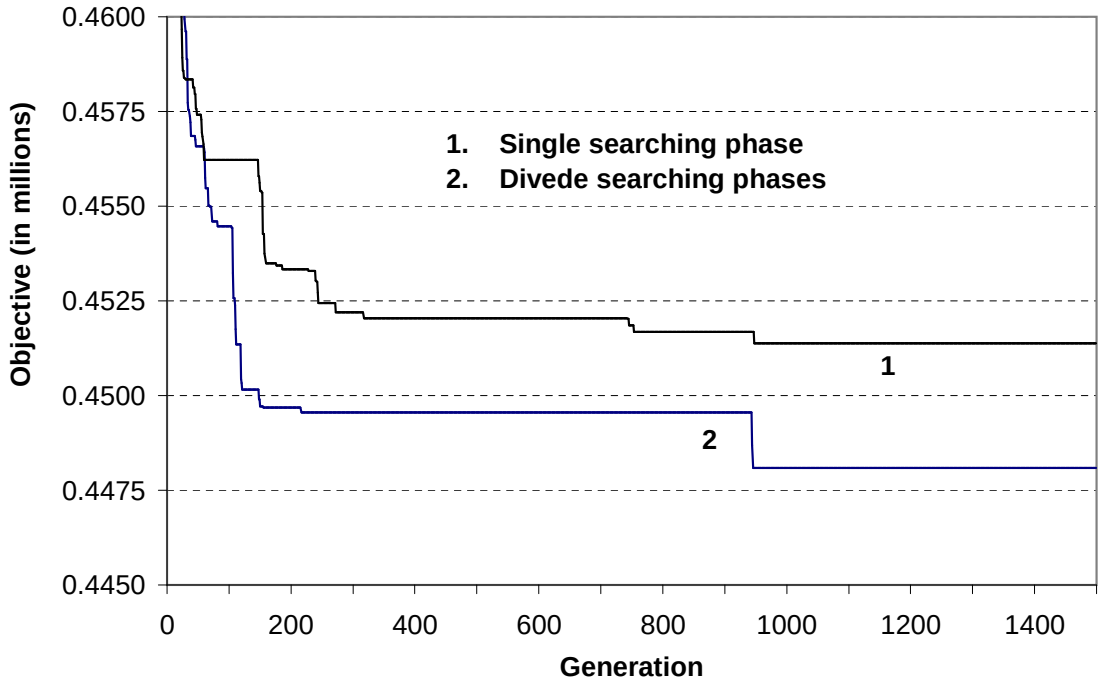


Figure 9. The impact of dividing the search into two phases on the convergence of LPEGA

Acknowledgements: 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.

References

Arvindh, B. and Irani, S., 1994. Cell formation: the need for an integrated solution of the subproblems. *International Journal of Production Research*, **32**, 1197–1218.

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

Bard, J. and Golany, B., 1991. Determining the number of kanbans in a multiproduct multistage production system. *International Journal of Production Research*, **29**, 881–895.

Berretta, R. and Rodrigues, L., 2004. A memetic algorithm for a multistage capacitated lot-sizing problem. *International Journal of Production Economics*, **87**, 67–81.

Bitran, G. and Chang, L., 1987. A mathematical programming approach to deterministic kanban system. *Management Science*, **33**, 427–441.

Burbridge, J., 1989. Production Flow Analysis for Planning Group Technology. Oxford Science Publication, Oxford,

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

Chen, M., 2001. A model for integrated production planning in cellular manufacturing systems. *Integrated Manufacturing Systems*, **12**, 275–284.

Chen, S. J. and Cheng, C. S., 1995. A neural network-based cell formation algorithm in cellular manufacturing. *International Journal of Production Research*, **32**, 293–318.

Clark, A., 2003. Optimization approximations for capacity constrained material requirements planning. *International Journal of Production Economics*, **84**, 115–131.

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

Garvin, D. A., 1988. Managing quality: The strategic and competitive edge. Free Press, New York,

Goldberg, D., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York,

Grubbström, R. and Huynh, T., 2006. Multi-level, multi-stage capacity-constrained production-inventory systems in discrete time with non-zero lead times using mrp theory. *International Journal of Production Economics*, **101**, 53–62.

Herer, Y. and Shalom, L., 2000. The kanban assignment problem a non-integral approach. *European Journal of Operational Research*, **120**, 260–276.

Hong, Z., Wang, H., and Chen, W., 2000. Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of Heuristics*, **6**, 439–455.

Huge, E. C. and Anderson, A. D., 1988. The Spirit of Manufacturing Excellence: An Executives Guide to the New Mind Set. Dow Jones-Irwin, Homewood, ILL,

Hyer, N. and Wemmerlöv, U., 2002. REORGANIZING THE FACTORY Computing through cellular manufacturing. Productivity Press, Portland, Oregon, USA, pp. 311–368.

ILOG Inc., 2003. CPLEX 9.0 Users Manual. 1080 Linda Vista Ave. Mountain View, CA 94043, (http://www.ilog.com).

Jaber, M. and Bonney, M., 2003. Lot sizing with learning and forgetting in set-ups and in product quality. *International Journal of Production Economics*, **83**, 95–111.

Jamal, A., 1993. Neural network and cellular manufacturing. *Industrial Management & Data Systmes*, **39**, 21–25.

Kim, C. and Hong, H., 1999. An optimal production run length in deteriorating production processes. *International Journal of Production Economics*, **58**, 183–189.

Li, C. and Cheng, T., 1994. An economic production quantity model with learning and forgetting considerations. *Production and Operations Management*, **3**, 118–132.

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.

Porteus, E. L., 1986. Optimal lot sizing, process quality improvement and setup cost reduction. *Operations Research*, **34**, 137–144.

Price, W., Gravel, M., and Nsakanda, A., 1994. A review of optimization models of kanaban-based production systems. *Journal of Operational Research*, **75**, 1–12.

Price, W., Gravel, M., Nsakanda, A., and Cantin, F., 1995. Modeling th eperformance of a kanban assembly shop. *International Journal of Production Research*, **33**, 1171–1177.

Riggs, J., 1981. Production Systems: Planning, analysis and control. Wiley, New York,

Saidi-Mehrabad, N. and Safaei, N., 2006. A new model of dynamic cell formation by a neural approach. *The International Journal of Advanced Manufacturing Technology*, **Online First (DOI: 10.1007/s00170-006-0518-2)**.

Schonberger, R. J. and Knod, E. M., 1994. Operations Management: Continuous Improvement, 5th Edition. Irwin, Burr Ridge, IL,

Singh, N., 1996. Systems Approach to Computer-Integrated Design and Manufacturing. Wiley, New York,

Tavakkoli-Moghaddam, R., Aryanezhad, M., 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.

Teghem, J., Pirlot, M., and Antoniadis, C., 1995. Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem. *Journal of Computational and Applied Mathematics*, **64**, 91–102.

Urban, T. L., 1998. Analysis of production systems when run length influences product quality. *International Journal of Production Research*, **36**, 3085–3094.

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

Table 8. Values of the parameters of GA used for 14 test runs on problem 1

Parameter Name	Test Run 1	2	3	4	5	6	7	8	9	10	11	12	13	14
Population Size	100	80	120	140	70	90	130	70	80	60	120	86	160	130
Crossover probability for														
Single point crossover	0.8	0.7	0.7	0.9	0.6	0.9	0.65	0.8	0.99	1.0	0.95	0.95	0.95	0.85
Period swamp crossover	0.9	0.8	0.9	0.9	0.7	0.9	0.75	0.8	0.99	1.0	0.95	0.99	0.9	0.95
All-cell swamp crossover	0.8	0.8	0.9	0.9	0.7	0.9	0.75	0.75	0.99	1.0	0.95	0.99	0.9	0.95
All-part swamp crossover	0.8	0.8	0.8	0.8	0.5	0.9	0.5	0.5	0.99	1.0	0.95	0.95	0.95	0.95
Single-cell swamp crossover	0.7	0.7	0.7	0.8	0.5	0.8	0.4	0.4	0.99	1.0	0.8	0.8	0.8	0.8
Single-part swamp crossover	0.7	0.7	0.7	0.7	0.7	0.8	0.4	0.4	0.99	1.0	0.8	0.8	0.8	0.8
Route swamp crossover	0.4	0.3	0.4	0.4	0.4	0.5	0.3	0.3	0.99	1.0	0.5	0.5	0.3	0.4
Mutation Probability for														
Machine mutator	0.005	0.003	0.01	0.03	0.02	0.005	0.005	0.005	0.01	0.008	0.005	0.005	0.01	0.03
Setup mutator	0.005	0.01	0.02	0.03	0.02	0.02	0.008	0.02	0.01	0.008	0.02	0.02	0.02	0.02
Part level cell mutator	0.02	0.03	0.03	0.03	0.04	0.04	0.008	0.02	0.01	0.005	0.04	0.04	0.04	0.04
Route level cell mutator	0.02	0.02	0.02	0.02	0.04	0.04	0.008	0.02	0.02	0.02	0.04	0.04	0.03	0.06
Operation level cell mutator	0.008	0.01	0.01	0.02	0.02	0.03	0.006	0.006	0.006	0.006	0.03	0.03	0.03	0.006