A mathematical model and a parallel multiple search path simulated annealing for an integrated distributed layout and machine cell formation

Published in 2017 in the Journal of Manufacturing Systems, Vol. 43, 195 -212

Please cite this article as:

Defersha, F. M. and Hodiya, A. (2017). A mathematical model and a parallel multiple search path simulated annealing for an integrated distributed layout and machine cell formation. Journal of Manufacturing Systems, Vol. 43, 195-212.

A Mathematical Model and a Parallel Multiple Search Path Simulated Annealing for an Integrated Distributed Layout Design and Machine Cell Formation

Fantahun M. Defersha[a,*], Abenet Hodiya[a,b,]

[a] *School of Engineering, University of Guelph, 50 Stone Road East, Guelph, Ontario, Canada, N1G 2W1*
[b] *Current address: PowerCor Manufacturing (Linamar), 545 Elmira Rd N, Guelph, Ontario, Canada N1K 1C2*

Abstract

Facility layout problem is a well-researched problem of finding configurations of departments and machines on a plant floor with the objective of improving material handling efficiency. With this objective, different techniques of configuring facilities have been documented in literature. Among them are cellular and distributed layouts. Cellular layouts are applicable in scenarios where demand and product mix are relatively stable and rational part families/machine cells can be identified. With this assumption, the literature provides many techniques for their design. Distributed layout, on the other hand, are recommended in volatile environments where product demand and mix are changing very rapidly. However, we argue that a real-life scenario may lay within the spectrum of these two extremes. In this paper, we attempt to bridge this gap by developing a mathematical model that integrates distributed layout design and machine cell formation with an objective to minimize a weighted sum of material handling and inter cellular movement costs. Through distributing the machines over the shop floor, the model attempts to minimize material handling cost. By identifying possible machine cells and part families, it attempts to minimize inter cellular movements. At the same time, the model ensures that machines that belong to the same cell are laid out on contiguous physical locations so that the advantages of cellular manufacturing systems can be fully exploited. Operations sequence, alternative routing, workload balancing among cells and other pragmatic issues are also incorporated in the model. We developed a parallel multiple search path simulated annealing to solve the proposed model efficiently. Several numerical examples are presented to illustrated the model and the computational performance of the developed algorithm.

Keywords: Mathematical Model; Distributed Layout; Cell Formation; Multiple Search Path Simulated Annealing; High Performance Parallel Computing.

1. Introduction

Over the past 50 years, group technology and its implementation via cellular manufacturing system (CMS) can be regarded as one of the most significant advances in the quest for faster, better, cheaper production and delivery of manufactured goods (Askin, 2013). It is a production philosophy aimed at increasing productivity by utilizing the similarities of products in their design and manufacturing attributes. CMS in particular involves (i) grouping parts having similar processing

*Corresponding author
Email address: fdefersh@uoguelph.ca (Fantahun M. Defersha)

requirements into part families, and (ii) organizing dissimilar machines along other supporting resources and operators into relatively autonomous cells such that each cell produces a part family with at most efficiency. Surveys on CMS user industries were conducted by Askin and Estrada (1999), Hyer and Wemmerlöv (1989), Wemmerlöv and Johnson (1997) and Wemmerlöve and Hyer (1989) and all found astounding results. In some of these surveys, respondents reported on average reductions of cycle time by 61%, setup time by 53%, distance/move time by 61%, response time to customer by 50% and work-in-process inventory by 48%. Improvements in product quality and job satisfaction, on average, by 31% and 27%, respectively, were also reported along with many other benefits. To this end, since the early pioneer articles by McAuley (1972), Rajagopalan and Batra (1975) and King (1980), CMSs have attracted more than 40 years intensive research in various aspects of their design and operations.

The central theme in the vast majority of those researches has been part family and machine cells formation. The most influential articles in this area that have received more than 220 citations each (more than 10 citations per year since their publication as per Google Scholar) include Ballakur and Steudel (1987), Chandrasekharan and Rajagopalan (1986), Choobineh (1988), Gonçalves and Resende (2004), Vakharia and Wemmerlöve (1990) and Wemmerlöv and Hyer (1986). Multi-period cell formation with dynamic reconfiguration have also been proposed in literature. Prominent article in this area with higher rate citations per year are Defersha and Chen (2006), Kioob *et al.* (2009) and Safaei *et al.* (2008). The determination of the physical locations of the machines and the cells has also been a subject of research since the 1990's. To the best of our knowledge, the very first mathematical model for an integrated facility layout and cell formation was proposed in Alfa *et al.* (1992). The model attempts to determine the locations of the machines from a fixed reference point and at the same time identify the cells to which the machines belong where the areas of the cells are given a priory. Integrated cell formation and a linear or a u-shaped inter-cellular layout (without detail layout of machines) was reported in Arvindh and Irani (1994). A mathematical model and simulated annealing for cell formation and layout design was proposed in Wang *et al.* (1998) where the main structure of the cellular system is given. Krishnan *et al.* (2012) developed hierarchial cell formation and facility layout procedure. At the top level of the hierarchy, machine cells and part families are identified. Based on the formed machine cells, a genetic algorithm is used to layout the machines based on a simple S-shaped space filling curve (the concept of space filling curvets in facility layout was initially introduced in Bozer *et al.* (1994)). There are also other numerous studies for facility layout in cellular manufacturing. The most recent once include Kia *et al.* (2015), Forghani *et al.* (2015) and Mohammadi and Forghani (2016). Kia *et al.* (2015) developed an integrated cell formation and layout where the cells can have unequal areas. The cells are restricted to be rectangular in shape. Forghani *et al.* (2015) combined the quadratic assignment problem (QAP) with two-dimensional facility layout problem and then formulate an integrated cell formation and layout problem and solve the problem using genetic algorithm. It was assumed machines in each cell are arranged in a single row. This work was later expanded to an S-shaped layout in Mohammadi and Forghani (2016).

Though cellular manufacturing systems are widely accepted as being superior to traditional process layout, there are several studies that challenge those claims (see for example Suresh and Meredith (1994), Flynn and Jacobs (1987) and Morris and Tersine (1990)). Primary concerns regarding the suitability of CM is its inflexible to changes in demand and workload patterns (Kannan and

Ghosh, 1995). It is assumed that part spectrum and demand are almost stable for a considerable long planning horizon (2-5 years) (Baykasoglu, 2003). However, when product demand and mix are volatile, meaningful part families and machine cells cannot be identified. In such scenarios, other layout methodologies, that do not require machine cells be identified and dedicated to distinct part families, such as fractal and holography/distributed layouts were proposed with the objective of minimizing part travel distance/time. The notion of fractal factory originated from Warnecke (1993) in which the mathematical concept of fractal geometry, used to describe objects that replicate their whole structure, is applied in managing organizations. The theory suggested that organizations be structured using fractal units that are self-similar, self-organizing, self-optimizing entities. This theory was first applied to facility layout by the name *fractal layout* in Venkatadri *et al.* (1997) and Montreuil *et al.* (1999) where fractal cells (having roughly the same composition of machines, factories within factory) are to be dispersed in the shop floor. The main objective is to reduce material movements by forming small multifunction fractal cells capable of processing most of the demanded products. Other studies on fractal layout can also be found in Aririguzo *et al.* (2013) and Saad and M. (2004). Holography layout was introduced in (Montreuil *et al.*, 1993; Montreuil and Venkatadri, 1991) in which machines are to be randomly dispersed through the shop floor. The assumption is when product mix changes and new parts are introduced, efficient routes can easily be identified that minimizes travel distances. The idea of this type of layout get a sizable attention in literature by a different name, distributed layout (see for instance Urban *et al.* (2000); Benjaafar and Sheikhzadeh (2000); Baykasoglu (2003); Lahmer and Benjaafar (2005); Baykasoğlu and Göçken (2010); Hamedi *et al.* (2012); Nageshwaraniyer *et al.* (2013); Shafigh *et al.* (2017)). Urban *et al.* (2000) proposed a model in which material flow requirements dictate the placement of machines without following a functional or a cellular arrangement. Benjaafar and Sheikhzadeh (2000) showed that creating replicates of the same department and distributing them throughout the plant floor greatly reduce material handling cost in a stochastic product demand scenario. Lahmer and Benjaafar (2005) presented a procedure for the design of distributed layouts in settings with multiple periods where product demand and product mix vary from period to period. Baykasoğlu and Göçken (2010) conducted a simulation study on distributed and factional layouts and the authors demonstrated that distributed layout can greatly reduce material handling time. Semi-distrusted layout deign procedure using genetic algorithm was presented in Hamedi *et al.* (2012). Nageshwaraniyer *et al.* (2013) developed a metaheuristic by incorporating the features of symbiotic and clonal algorithms to solve a model for distributed layout. Shafigh *et al.* (2017) develop a linear programming embedded simulated annealing to solve model that combines distributed layout, dynamic reconfiguration and production planning.

The assertion from the above literature review is that cellular manufacturing systems are applicable when demand and product mix are relatively stable, whereas fractal or distributed layouts are for a very volatile (chaotic as per Montreuil *et al.* (1993)) environment. However, we argue that a real life scenario may always lay within the spectrum of these two extremes. In this work, we attempt to bridge this gap by developing a mathematical model that integrates distributed layout design and machine cell formation. In developing the model, we consider two primary objectives, one from each category. The main objective in distributed layout is the minimization of travel distance of parts by distributing resources to increase their accessibility from different regions of the layout. Whereas, in cellular layout, the main objective is the minimization of inter-cellular movement by

enabling all parts in a family be processed within a single dedicated cell as much as possible. The proposed model is aimed at minimizing a weighted sum of these two objectives. Through distributing the machines over the shop floor, the model attempts to minimize material handling cost. By identifying possible machine cells and part families, it attempts to minimize inter cellular movements. At the same time, the model ensures that machines that belong to the same cell are laid out on contiguous physical locations so that the advantages of cellula manufacturing systems can be fully exploited. Moreover, through identifying the machine cells over the distributed layout, detail inter-cellular and intra-cellular layout are obtained. Operations sequence, alternative routing, workload balancing among cells and other pragmatic issues are also incorporated. The proposed model can also be used for virtual cell formation and reconfiguration (without physical machine relocation) over a given distributed layout with a unique advantage of forming each virtual cell with a group of machines on contiguous physical location. Experimental results show that solving the proposed model using off-the-shelf optimization packages is difficult even for small size problems. To solve the model efficiently, we develop a multiple search path simulated annealing. We further enhance the algorithm using high performance parallel computation. The remainder of the paper is organized as follows. In Section 2, we present the proposed mathematical model. The solution procedure is in Section 3. Numerical examples are in Section 4, illustrating the feature of the model and the computational performance of the proposed algorithm. Finally, in Section 5 are discussion, conclusion and feature research.

2. Mathematical Model

In this section, we develop a new mathematical model for an integrated distributed layout and cellular manufacturing systems design. In doing so, we draw concepts from (i) irregularly shaped departments facility layout, (ii) distributed layout and (iii) machine cell formation. As it is the case in many studies (see for example Baykasoğlu and Göçken (2010); Hamedi *et al.* (2012); Lahmer and Benjaafar (2005); Rosenblatt and Golany (1992)), we assume a rectangular factory floor divided into grids in which machines are separated by one unit distance. Given this assumption, the problem description, notations and mathematical formulation are detailed in the following subsections.

2.1. Problem description:

Consider a manufacturing facility processing P products using M machines installed on L locations (where $L = M$). Currently, it is assume that identical or similar machines are located close to each other forming functional departments whereas each individual machine in the systems is given a unique identifier (index) as $m = 1$, $m = 2$, $\cdots$, $m = M$. A product p can be processed along R_p alternative routes where an alternative route is defined as a sequence of machines required to process a product from raw material to a finished good. For example, alternative routes for a typical product can be given as Route-1: $m = 2 \rightarrow 4 \rightarrow 5 \rightarrow 3$, Route-2: $m = 2 \rightarrow 4 \rightarrow 7$, and Route-3: $m = 2 \rightarrow 4 \rightarrow 8$. This may be the case where the last two operations on machines 5 and 3 in Route-1 can be combined and performed on machine 7 if Route-2 is chosen, or on machine 8 if Route-3 is chosen. Machines 7 and 8, in Route-2 and -3, respectively, may be two identical or similar machines. The production volume of a product in the planing horizon can be split among its alternative routes. Given L locations as shown in Figure 1-(a), the problem is to disaggregate the functional departments and distribute the machines (as shown in Figure 1-(b)) and identify machine cell (as shown in

Figure 1-(c)) in order to minimize the cost of material handling and the total number of inter-cellular movements by all the parts. A particular location can be a member of a cell if it is adjacent (share boundaries) with one or more locations in that cell. A cell should not have a disjoint sets of locations (i.e. a single closed-loop boundary can be identified that contains all the locations of this cell and none from other cells). This is to ensure that machines of a particular cell are laid out in contiguous physical locations. Each machine has a capacity expressed in hours during the planning horizon and therefore cannot be assigned a workload more than its capacity. The number of locations (machines) that can be added to a particular cell has both lower and upper limits and the workload among the cells need to be balanced. The notations used and the proposed mathematical model are presented in the following subsections.

Figure 1: A typical machine distribution and a cell formation process based on location adjacency

2.2. Notations

Input Data and Indexes

M Number of machines where machines are indexed by $m = 1, 2, ..., M$.

P Number of products where products are indexed by $p = 1, 2, ..., P$.

R_p Number of alternative processing routes for product p where alternative processing routes are indexed by $r = 1, 2, ..., R_p$.

$J_{p,r}$ Number of operations of a product p along processing routes r where operations are indexed by $j = 1, 2, ..., J_{p,r}$.

$M_{j,p,r}$ The index of the machine type used to process operation j of product p along processing route r.

$T_{p,r,j}$ Processing time in minutes of operation j of product p along processing route r (on machine $M_{j,p,r}$).

L Number of locations where locations are indexed by $l = 1, 2, ..., L$.

C Number of cells where cells are indexed by $c = 1, 2, ..., C$.

LB Minimum number of locations (machines) in a cell c.

UB Maximum number of locations (machines) in a cell $LB = I$.

I Same us UB where the order in which the locations are added to a cell is indexed by $i = 1, 2, ..., I = UB$. (Note: Locations are added to a cell in certain order for the sake of mathematical formulation. For example, one may say location l is the i^{th} location to be added to cell c. However, the final solution is not impacted by the order in which locations are added to a cell.)

Υ Intercellular workload balancing factor.

$N_{l,l'}$ A binary data which equals to 1 if location l and l' are adjacent, 0 otherwise;

$E_{l,l'}$ Material handling distance between location l and l'.

F_p Material handling cost per unit distance for a single product p.

V_p Intercellular movement cost per a single product p .

D_p Demand for product p during the planning horizon.

K_m Capacity of machine type m in hours during the planning horizon.

Θ_1 Objective function weight factor for the total material transportation cost.

Θ_2 Objective function weight factor for the total intercellular movement cost.

M Large positive number.

Binary variables:

$a_{m,l}$ A binary variable which equals to 1 if machine m is in location l, 0 otherwise;

$x_{l,c}$ A binary variable which equals to 1 if location l is in cell c, 0 otherwise;

$\eta_{j,p,r,c}$ A binary variable which equals to 1 if operation j of product p along process plan r is processed in cell c, 0 otherwise;

$q_{l,i,c}$ A binary variable which equals to 1 if location l is the i^{th} location to be added to cell c, 0 otherwise; (also see the definition of the notation I)

$y_{m,c}$ A binary variable which equals to 1 if machine m is in cell c, 0 otherwise;

Continuous variables:

$\alpha_{p,r}$ The production sublot size of product p along process plan r.

$\hat{d}_{p,r,j}$ The distance between the locations where consecutive operations of product p along process plan r are processed and multiplied by the sublot size $\alpha_{p,r}$.

$\Omega_{j,p,r,c}$ The operation of the production sublot size of product p according to process plan r in a cell.

2.3. Model Formulation

Based on the problem description and notations, the mixed integer non-linear mathematical model for integrated facility layout and cell formation is presented below.

Minimize:

$$z = \Theta_1 z_1 + \Theta_2 z_2 \tag{1}$$

Where:

$$z_1 = \sum_{p=1}^{P} \sum_{j=1}^{J_{p,r}-1} \sum_{r=1}^{R_p} F_p \times \hat{d}_{p,r,j}$$

$$z_2 = (\frac{1}{2}) \sum_{c=1}^{C} \sum_{p=1}^{P} \sum_{j=1}^{J_{p,r}-1} \sum_{r=1}^{R_p} (V_{p\times} \alpha_{p,r} \mid (\eta_{j+1,p,r,c} - \eta_{j,p,r,c}) \mid)$$

Subject to:

$$x_{l,c} \leq \sum_{\{l' | N_{l,l'}=1\}} x_{l',c} \; ; \quad \forall (l,c) \tag{2}$$

$$q_{l,i,c} \leq \sum_{i'=1}^{i-1} \sum_{\{l' | N_{l,l'}=1\}} q_{l',i',c} \; ; \quad \forall (l,i,c) \mid i > 1 \tag{3}$$

$$\sum_{c=1}^{C} \sum_{i=1}^{I} q_{l,i,c} = 1 \; ; \quad \forall (l) \tag{4}$$

$$\sum_{i=1}^{I} q_{l,i,c} = x_{l,c} \; ; \quad \forall (l,c) \tag{5}$$

$$\sum_{l=1}^{L} q_{l,i,c} \leq 1 \; ; \quad \forall (i,c) \tag{6}$$

$$LB \leq \sum_{l=1}^{L} x_{l,c} \leq UB \; ; \quad \forall (c) \tag{7}$$

$$\sum_{c=1}^{C} x_{l,c} = 1 \; ; \quad \forall (l) \tag{8}$$

$$a_{m,l} + x_{l,c} \leq y_{m,c} + 1 \; ; \quad \forall (l,m,c) \tag{9}$$

$$\sum_{l=1}^{L} a_{m,l} = 1 \; ; \quad \forall(m) \tag{10}$$

$$\sum_{m=1}^{M} a_{m,l} = 1 \; ; \quad \forall(l) \tag{11}$$

$$\sum_{c=1}^{C} y_{m,c} = 1 \; ; \quad \forall(m) \tag{12}$$

$$\eta_{j,p,r,c} = y_{m,c} \; ; \quad \forall(j,p,r,c) \mid (m = M_{j,p,r}) \tag{13}$$

$$\sum_{\forall(j,p,r)|M_{j,p,r}=m} \alpha_{p,r} \cdot T_{p,r,j} \leq K_m \; ; \quad \forall(m) \tag{14}$$

$$\sum_{p=1}^{P}\sum_{r=1}^{R}\sum_{j=1}^{J_{p,r}} \Omega_{j,p,r,c} \geq \frac{\Upsilon}{C} \sum_{p=1}^{P}\sum_{r=1}^{R}\sum_{j=1}^{J_{p,r}}\sum_{c'=1}^{C} \Omega_{j,p,r,c} \; ; \quad \forall(c) \tag{15}$$

$$\Omega_{j,p,r,c} \geq \alpha_{p,r} \cdot T_{p,r,j} - M(1 - y_{m,c}) \; ; \quad \forall(j,p,r,c) \mid (m = M_{j,p,r}) \tag{16}$$

$$\Omega_{j,p,r,c} \leq \alpha_{p,r} \cdot T_{p,r,j} + M(1 - y_{m,c}) \; ; \quad \forall(j,p,r,c) \mid (m = M_{j,p,r}) \tag{17}$$

$$\Omega_{j,p,r,c} \leq M \cdot y_{m,c} \; ; \quad \forall(j,p,r,c) \mid (m = M_{j,p,r}) \tag{18}$$

$$\hat{d}_{p,r,j} \geq E_{l,l'} \times \alpha_{p,r} + M(a_{m,l} + a_{m',l'}) - 2M \; ;$$
$$\forall(j,p,r,l,l') \mid (j < J_{p,r}, \; m = M_{j,p,r} \; \& \; m' = M_{j+1,p,r}) \tag{19}$$

$$\hat{d}_{p,r,j} \leq E_{l,l'} \times \alpha_{p,r} - M(a_{m,l} + a_{m',l'}) + 2M \; ;$$
$$\forall(j,p,r,l,l') \mid (j < J_{p,r}, \; m = M_{j,p,r} \; \& \; m' = M_{j+1,p,r}) \tag{20}$$

$$\sum_{r=1}^{R_p} \alpha_{p,r} = D_p \; ; \quad \forall(p) \tag{21}$$

$$a_{m,l}, \; x_{l,c}, \; y_{m,c}, \; \eta_{j,p,r,c}, \; \& \; q_{l,i,c} \quad \textit{are binary.} \tag{22}$$

The objective function in Eq. (1) is a weighted sum of the number of the material handling cost (z_1) between all pairs of locations and intercellular movements cost (z_2). The first term represents the sum of the costs for the distances travelled from machine to machine by all the parts while being processed from raw pieces to finished goods. As it can be seen in Benjaafar and Sheikhzadeh (2000), this cost term can be minimized even without cell formation by simply distributing the machines over the shop floor. Thus, this term is well aligned with the objective of distrusted layout design. The second term of the objective function measures how well the system is disaggregated into relatively independent cells which conforms with the main objective of cellular manufacturing system design. Hence, in proposing this model, we are attempting to unify the concepts of distributed layouts and cellular manufacturing systems with the objective of handling manufacturing scenarios that may not be well addressed by either layout concepts independently.

The constraint in Eq. (2) states that a particular location l can be in cell c if it is adjacent to one or more locations that belong to the same cell c. However, this constraint alone will not prevent a cell from having two or more non-adjacent sets of locations as long as each set has more than one location. In order to model the constraints that will prevent a cell from being disjoint, we

assume locations are added to a cell in a certain order as defined by the variable $q_{l,i,c}$ and a location can be added to a cell if it is adjacent to one or more locations that are already added to the cell. This is enforced by the constraint in Eq. (3) and is illustrated in Figure 1(d)-(i). For example, in Figure 1(d), location L18 is assumed to be the first location being added to a cell. The potential locations that can be the second addition to the cell are L13, L17, L19 and L23 as indicated by the cross-hatch. Let Location L17 is the 2nd location to be added to the cell as shown in Figure 1(e) where the potential location for the 3rd addition are cross-hatched. This process continue in a similar fashion and terminates when the number of locations added to a cell are within the lower and upper limits. Eq. (4) states that a particular location l will be added exactly to one cell c in one addition step i. The logical relationship between $q_{l,i,c}$ and $x_{l,c}$ is enforced by the constraint in Eq. (5). In a particular addition cycle of a location to a cell, at most one location can be added to the cell as stated by Eq. (6). The constraint in Eq. (7) is to enforce the lower and upper limits on the size of a cell. A location l can be added to exactly one cell and this is enforced by Eq. (8). The constraint in Eq. (9) enforces a logical relationship among the binary variable $a_{m,l}$, $x_{l,c}$ and $y_{m,c}$. A machine can occupy exactly one location, a location can be assigned to exactly one machine, and a machine can belong to exactly one cell as enforced by Eqs. (10), (11) and (12), respectively. The constraint in Eq. (13) defines the logical relationship between the binary variables $\eta_{j,p,r,c}$ and $y_{m,c}$. Machine capacity constraint is enforced by Eq. (14). The constraint in Eq. (15) is to enforce workload balancing among the cells where the factor $\Upsilon \in (0, 1)$ is used to determine the degree of the workload balance. This constraint is similar to the workload balancing constraint appeared for the first time in Defersha and Chen (2006). If the number of cells is C, the minimum allowable workload of a cell is $\frac{\Upsilon}{C} \times 100\%$ of the total workload of the systems in terms of processing time. If Υ is chosen close to 1.0, the allowable workload of each cell will be close to the average workload given by $\frac{100}{C}\%$ of the total workload of the systems. The constraints in Eqs. (16), (17) and (18) are to calculate the workload in cell c because of the j^{th} operation along the r^{th} processing route of product p. The total distance traveled by product p along its route r for the processing of its consecutive operation j and $j + 1$ is calculated by Eqs. (19) and (20). The constraint in Eq. (21) states that the sum of the sublots of product p along all of its processing routes should be equal to its total production demand for the planning period. The integrality constraints are in Eq. (22).

2.4. Linearizing the Model

The proposed mathematical model is non-linear because of the absolute value in the objective function. This term can be linearized in two steps. First, the absolute value $\mid \eta_{j+1,p,r,c} - \eta_{j,p,r,c} \mid$ is substituted by a binary variable $z_{j,p,r,c}$ with the additional constraints in Eqs. 23-25. Second, the resulting quadratic term $(\alpha_{p,r} \cdot z_{j,p,r,c})$ is replaced by a continuous variable $w_{j,p,r,c}$ with another set of additional constraints given in Eqs. 26-28. Here it is important to note that absolute value $\mid \eta_{j+1,p,r,c} - \eta_{j,p,r,c} \mid$ can be equal to 1 for at most one value of c. Hence, the subscript c can be (and should be) dropped from $z_{j,p,r,c}$ and $w_{j,p,r,c}$ and these variables have to be replaced by $z_{j,p,r}$ and $w_{j,p,r}$, respectively, in Eqs. 23-28.

$$\eta_{j+1,p,r,c} - \eta_{j,p,r,c} \leq z_{j,p,r,c} \; ; \quad \forall (j, p, r, c) \,|\, (j < J_{p,r}) \tag{23}$$

$$-\eta_{j+1,p,r,c} + \eta_{j,p,r,c} \leq z_{j,p,r,c} \; ; \quad \forall (j, p, r, c) \,|\, (j < J_{p,r}) \tag{24}$$

$$z_{j,p,r,c} \quad \in \quad \{1, 0\} \; ; \quad \forall (j, p, r, c) \,|\, (j < J_{p,r}) \tag{25}$$

$$w_{j,p,r,c} \geq \alpha_{p,r} + M \cdot z_{j,p,r,c} - M \; ; \quad \forall (j,p,r,c) \,|\, (j < J_{p,r}) \tag{26}$$

$$w_{j,p,r,c} \leq \alpha_{p,r} \; ; \quad \forall (j,p,r,c) \,|\, (j < J_{p,r}) \tag{27}$$

$$w_{j,p,r,c} \leq M \cdot z_{j,p,r,c} \; ; \quad \forall (j,p,r,c) \,|\, (j < J_{p,r}) \tag{28}$$

3. Solution Procedure

Despite the importance and magnitude of the effort that has been put into computational science, in many ways the construction of new algorithms remains more of an art than a science (Knoll *et al.*, 2005). Preexisting theories give little or no guidance for the choice of solution representation and the design of search operators for new problems (Moraglio, 2007). In light of these and other similar assertions from literature and based on our experience, we argue that a new problem usually requires a new design of solution representation, initialization technique, implementation strategies, and search operators. Many of these important components have been detailed in the following subsections in relation to developing a simulated annealing for the proposed mathematical model.

3.1. Solution Representation

Solution representation is the first and the most important step in applying a metaheuristic algorithm. It must be designed in such a way that all feasible solutions are accessible to the search process and model constraints are encoded in it. The solution representation designed to solve the proposed model is depicted in Figure 2. The left-hand-side segment (LHS-Segment) encodes the cell formation through assigning a cell index $c_l \in \{1, 2, \cdots, C\}$ to each location l. This encodes the variable $x_{c,l}$ and the constraint in Eq. (8). The constraints in Eqs. (2)-(7) are being taken care by the initialization process and the search operators as explained in the subsections 3.2 and 3.4. The middle-segment (MDL-Segment) encodes the distributed layout design aspect of the mathematical model by assigning a location index $l_m \in \{1, 2, \cdots, L\}$ to each machine m. In this segment, $\{l_1, l_2, l_3, \cdots, l_{M=L}\}$ is a permutation of the indices of $M = L$ locations. Hence, an index of a particular location appears exactly once to grantee the constraints in Eqs. (10) and (11). The RHS-Segment encodes the sizes of the sublots $\alpha_{p,r}$. The element $\theta_{p,r}$ takes a binary value to indicate whether route r of product p is used or not. In this segment, the summation $\sum_{r=1}^{R_p} \theta_{p,r}$ for each p should be kept to be greater or equal to 1 to ensure that at least one route is opened. The size of a sublot of product p along one of its route r is calculated using Eq. (29). This equation along with the requirement on the RHS-segment that $\sum_{r=1}^{R_p} \theta_{p,r} \geq 1$ grantees the constraint in Eq. (21).

$$\alpha_{p,r} = \frac{\theta_{p,r} \times \beta_r}{\sum_{r'=1}^{R_p} (\theta_{p,r'} \times \beta_{p,r'})} \times D_p \tag{29}$$

3.2. Initialization

Initializing the LHS-Segment: Randomly assigning a cell index $c_l \in \{1, 2, \cdots, C\}$ to each location l in the LHS-Segment of an initial solution will not provide a layout with cells that can be demarcated from one another as shown in Figure 3-(f). In order to ensure that an initial solution has cells that can be demarcated from one antother, we develop a simple initialization procedure. The pseudocode of this procedure is provided in Pseudocode 1 and its implementation is exemplified in Figure 3 where a total of 56 locations are to be demarcated into five cells. First, we arbitrarily

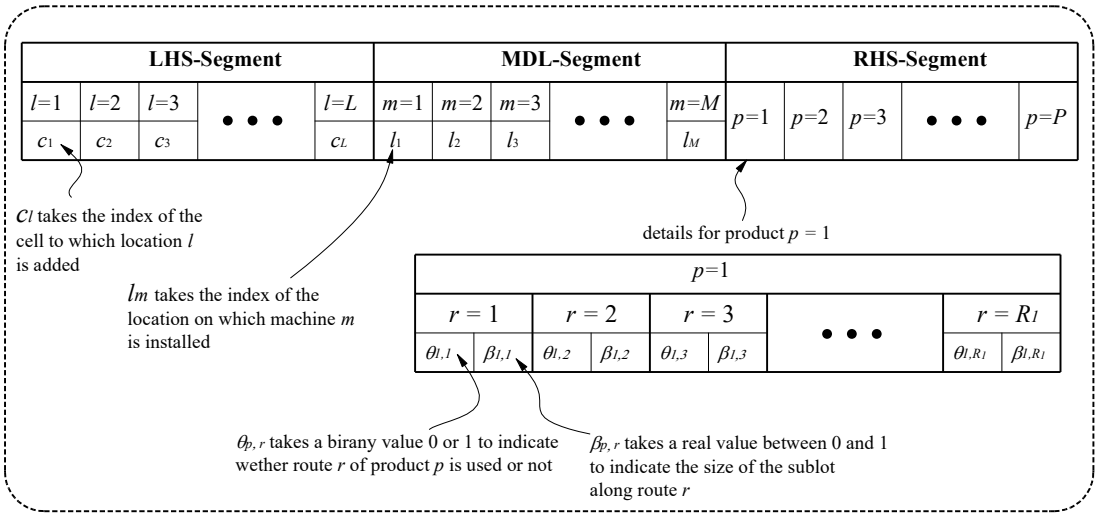


Figure 2: Solution representation for an integrated cell formation and distributed layout

assign five locations as the starting locations for the five cells as shown in Figure 3-(a). This is in line-(3) of Pseudocode 1 after setting a variable $Feasible = False$. In the first execution of the "for-loop" (i.e. Iteration 1) from line-(5), the first location that can be added to a cell is location 3 as it is adjacent to location 11 that has already been added to cell 5 in a previous iteration (in this case iteration-start). Location 4 cannot be added to a cell as it is not adjacent to any location that has already been added to a cell in a previous iteration. This process continues up to $l = 56$ and the result from this iteration is shown in Figure 3-(b). Since all the locations are not yet added to the cells, the "while-loop" in line-4 calls for a second execution of the inner "for-loop" (i.e Iteration 2). Now let us examine this iteration when $l = 35$ (see Figure 3-(c)), in which case the location is adjacent to two locations that were added to two different cells in the previous iteration (location 35 is adjacent to location 34 in cell 4 and to location 36 in cell 2). At this stage of the current iteration, locations 17 and 26 have been already added to cell 4 making the total number of locations added to this cell to be 6, and location 28 has been already added to cell 2 making the number of locations added to cell 2 to be 6 as well. Since the number of location so far added to these cells are equal, location 35 can be added to either cell 4 or cell 2. This tie is broken arbitral and the location is added to cell 2. When this iteration continues and reaches $l = 42$, more locations have already been added to cell 2 than to cell 4. Hence, location 42 is added to cell 4. This process continues to provide the complete result of Iteration-2 as shown in Figure 3-(d). During the third iteration (see Figure 3-(e)), locations 1 and 5 can only be added to cell 5, locations 6 and 8 to cell 1. Location 29 can be added either to cell 1 or 2. However, cell 2 has the smaller number of locations added so far than cell 1, hence location 29 has to go to cell 2. By the same analysis, location 50 is added to cell 4. At Iteration-4, all the locations are added to the cells as shown in Figure 3-(f) at which point the inner "while-loop" exits, sending the control to line-16 (see Pseudocode 1). The number of locations added to each cell are compared to the set lower and upper bounds (LB and UB). If one or more cells do not meet this condition, the control will go back to line-3 and the whole process repeats. If all the cells generated respect the bound limits, a solution can be initialized by coping the the final location-cell assignments to the LHS-Segment as shown in Figure 3-(g).

Initializing the MDL- and RHS-Segments: Unlike that of the LHS-Segment, the initial-

Pseudocode 1: LHS-Segment Initialization

```
1  set Feasible = False
2  while Feasible=False do
3      start Arbitrarily identify C locations as a starting locations for the formation of the C
         cells.
4      while there are locations that are not yet assigned to a cell do
5          for l = 1 to L do
6              if location l not yet assigned to a cell then
7                  Identify all the cells that location l can be added
8                  /* Before the current executions of this "for loop", if a cell has
                   one or more locations already added to it that are adjacent to
                   location l, then this cell is a potential cell to which location l
                   can be added.  Locations added to a cell in the current execution
                   of this "for-loop" are not used for adjacency test to add a new
                   location to this cell, but are added to the locations count which
                   can be used for tie braking in the following "if-statement".  */
9                  if there are one or more cells to which location l can be add then
10                     Assign location l to the cell that has the smallest number of locations
                         assigned to far; break ties arbitrarily.
11                     Increase the number of locations assigned to this cell by one.
12                 end
13             end
14         end
15     end
16     if  the number of locations added to cell each is within the lower and upper limits then
17         set Feasible = True
18     end
19 end
```

izations of the MDL- and RHS-Segments are quite intuitive. The MDL-Segment can be initialized by randomly permutating the indices of $M = L$ locations and copy the resulting permutation, $[l_1, l_2, l_3, \cdots, l_M | l_m \neq l_{m'} \forall (m \neq m')$ & $l_m \in \{1, 2, \cdots, L\}]$, to this segment. RHS-Segment is initialized in such a way that the θ and the β corresponding to each route r of a product p are assigned (i) a binary value 0 or 1 and (ii) a real value between 0 and 1, respectively. In this initialization process, corresponding to each product p, one has to make sure that at least one of its θ' is set to 1 to ensure the opening of at least one route for this product. The overall initialization procedure takes only few minutes to generate several thousands of starting solutions for large size problems. Hence, it incurs no computational burden on the algorithm.

3.3. Evaluation

The purpose of evaluation in a metaheuristic is to measure the relative goodness of candidate solutions with respect to the objective function and the constraints of the model to be solved. The solution representation along with the initialization procedure discussed in the previous subsections and the search operators (see subsection 3.4) satisfy many of the model constraints. The only constraints that may be violated by a randomly generated solution are the machine capacity and the workload balancing constraints in Eqs. (14) and (15), respectively. Hence, a measure of goodness

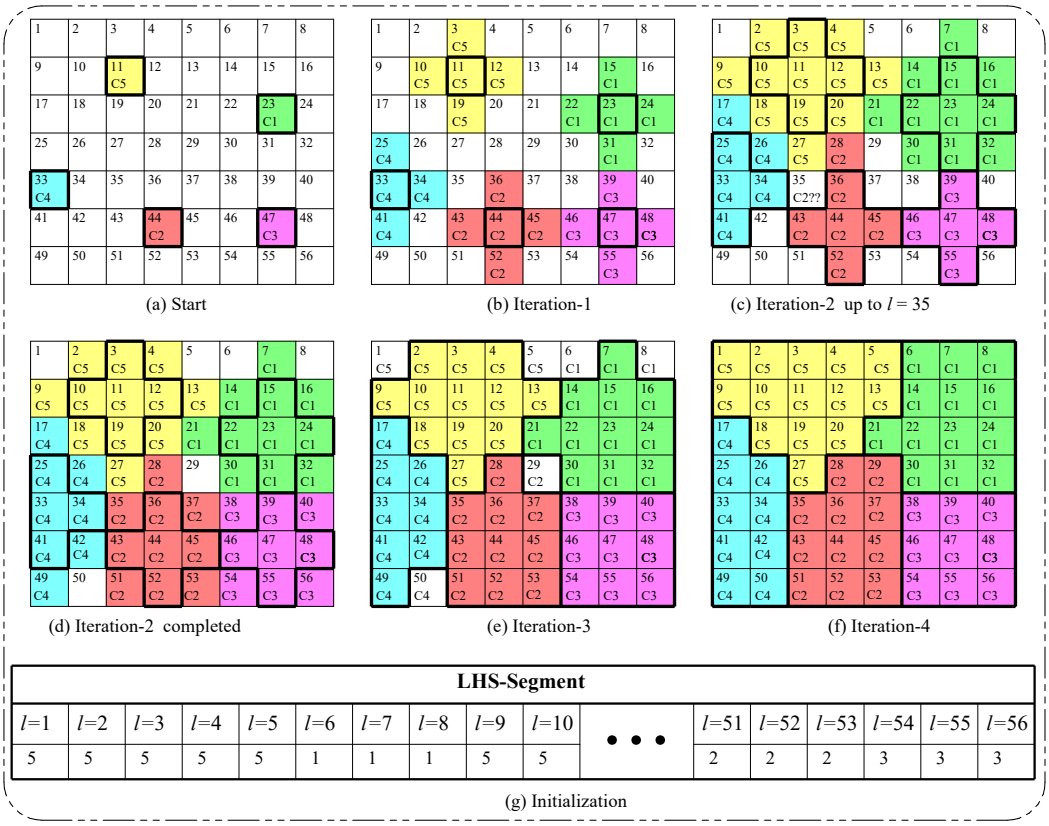


Figure 3: Initialization of the LHS-Segment where an iteration is a single execution of the "for-loop" of Pseudocode 1

of a solution should involve penalty terms proportional to any violations of these constraints. This measure of goodness is give in Eq. (30). In this equation, the first two terms are the two terms of the objective function of the mathematical model. The last two terms are penalties for violations of machine capacity and workload balancing constraints, respectively.

$$E = \Theta_1 z_1 + \Theta_2 z_2 + \Theta_3 z_3 + \Theta_4 z_4 \tag{30}$$

Where:

$$z_1 = \sum_{p=1}^{P} \sum_{j=1}^{J_{p,r}-1} \sum_{r=1}^{R_p} F_p \times \hat{d}_{p,r,j}$$

$$z_2 = (\frac{1}{2}) \sum_{c=1}^{C} \sum_{p=1}^{P} \sum_{j=1}^{J_{p,r}-1} \sum_{r=1}^{R_p} (V_{p\times} \alpha_{p,r} \mid (\eta_{j+1,p,r,c} - \eta_{j,p,r,c}) \mid)$$

$$z_3 = \sum_{m}^{M} \max \left(0, \quad \left[\sum_{\forall (j,p,r) | m_{j,p,r}=m} \alpha_{p,r} \cdot T_{p,r,j} \right] - K_m \right)$$

$$z_4 = \sum_{c}^{C} \max \left(0, \quad \frac{\Upsilon}{C} \sum_{p=1}^{P} \sum_{r=1}^{R} \sum_{j=1}^{J_{p,r}} \sum_{c'=1}^{C} \Omega_{j,p,r,c} - \sum_{p=1}^{P} \sum_{r=1}^{R} \sum_{j=1}^{J_{p,r}} \Omega_{j,p,r,c} \right)$$

In order to evaluate the measure of goodness given in Eq. (30), it is not necessary to explicitly decode all the variable $\eta_{j,p,r,c}$, $\hat{d}_{p,r,j}$, and $\Omega_{j,p,r,c}$ from a solution under consideration and apply

the equations. Instead, one can take advantage of the structure of the solution representation and compute the measure without explicitly decoding these variables as shown in Pseudocodes 2 and 3. In these pseudocode, it is assumed that a solution $\mathbf{X}$ is a data structure that can be dot operated to access its member data. For example, the index of the cell c_l to which location l is assigned and the index of the location l_m on which machine m is installed can be accessed from this solution using a dot operator as **X.LHS-Segment.Location**$[l]$.c_l and **X.MDL-Segment.Machine**$[m]$.l_m, repsectively (see Figure 2 for the solution strucutre). Similarly, **X.RHS-Segment.Product**$[p]$**.Route**$[r]$ can be dot operated in order to access the values of θ_r and β_r. With this assumption, the pseudocodes for evaluating Eq. (30) are partly explained as follows.

Pseudocode 2 is for calculating the costs of material handling z_1 and the inter cellular movement z_2. In lines 6 and 7 of this pseudocode, two locations (l and l') of two machines ($M_{p,r,j}$ and $M_{p,r,j+1}$) required to process two consecutive operation (j and $j+1$) along an opened route r of product p are obtained from a solution. Then, in line 8, the material haling cost between these two locations is recursively added to z_1. In lines 9 and 10, the cell indices (c and c') are obtained in which these two consecutive operations are performed. If these two indices are not equal, then intercellular movement cost is recursively added to z_2 in line 12. The machine capacity and workload balancing constraints violations are computed in Pseudocode 3. In line 10, the load to process operation j of product p along an opened route r is recursively added to the total system load. This operation is to be performed on machine $M_{p,r,j}$ and the load on this machine is incremented in line 10. The location l on which this machine is installed is obtained in line 12 and, in line 13, the cell index c to which this location is assigned is obtained. Then, the load on this cell is incremented. Once the loads on the system, on each machine and on each cell are calculated, the violation of machine capacity and workload balancing constraints are recursively computed in lines 22 and 28, respectively.

Pseudocode 2: Calculating z_1 and z_2 of a solution $\mathbf{X}$

Input: Using Eq. (29), calculate $\alpha_{p,r}$ for each $p \in \{1, 2, \cdots, P\}$ and $r \in \{1, 2, \cdots, R_p\}$

```
1  for p = 1 to P do
2      for r = 1 to R_p do
3          if X.RHS-Segment.Product[p].Route[r].θ_r = 1 then
4              for j = 1 to J_{p,r} − 1 do
5                  l = X.MDL-Segment.Machine[M_{p,r,j}].l_m
6                  l' = X.MDL-Segment.Machine[M_{p,r,j+1}].l_m
7                  z_1 = z_1 + (F_p × E_{l,l'} × α_{p_r})
8                  c = X.LHS-Segment.Location[l].c_l
9                  c' = X.LHS-Segment.Location[l'].c_l
10                 if c ≠ c' then
11                     z_2 = z_2 + (V_p × α_{p,r})
12                 end
13             end
14         end
15     end
16 end
17                 /* For many of the notation, see Section 2.2 and Section 3.1.  */
```

Pseudocode 3: Calculating z_3 and z_4 of a solution **X**

Input: Using Eq. (29), calculate $\alpha_{p,r}$ for each $p \in \{1, 2, \cdots, P\}$ and $r \in \{1, 2, \cdots, R_p\}$

float LoadOnMachine[M] /* Load on machine $m = 1, 2, \cdots, M$ */
float LoadOnCell[C] /* Load on cell $c = 1, 2, \cdots, C$ */
float TotalSystemLoad; /* Total workload on the system */
/* Calculate the load on each machine, the load on each cell, and the total workload on the system */
for $p = 1$ **to** P **do**
  for $r = 1$ **to** R_p **do**
    if X.RHS-Segment.Product[p]**.Route**[r].$\theta_r = 1$ **then**
      for $j = 1$ **to** $J_{p,r}$ **do**
        TotalSystemLoad = TotalSystemLoad + $T_{p,r,j} \times \alpha_{p,r}$
        LoadOnMachine[$M_{p,r,j}$] = LoadOnMachine[$M_{p,r,j}$] + $T_{p,r,j} \times \alpha_{p,r}$
        l = **X.MDL-Segment.Machine**[$M_{p,r,j}$]**.**l_m
        c = **X.LHS-Segment.Location**[l]**.**c_l
        LoadOnCell[c] = LoadOnCell[c] + $T_{p,r,j} \times \alpha_{p,r}$
      end
    end
  end
end
/* Calculate machine capacity constraint violation */
for $m = 1$ **to** M **do**
  if LoadOnMachine[m] > K_m **then**
    $z_3 = z_3$+LoadOnMachine[m] - K_m
  end
end
/* Calculate workload balancing constraint violation */
for $c = 1$ **to** C **do**
  if LoadOnCell[c] < $\frac{\Upsilon}{C}$×TotalSystemLoad **then**
    $z_4 = z_4 + \left(\frac{\Upsilon}{C} \times \text{TotalSystemLoad} - \text{LoadOnCell}[c]\right)$
  end
end
/* For many of the notation, see Section 2.2 and Section 3.1. */

3.4. Search Operators

A typical solution encodes the decisions regarding the boundaries of the cells, the distribution of machines on shop floor, and the sizes of the sublots of the products. Thus, in searching the solution space, operators are needed that can alter the traits of a solution that determined these interrelated decisions while respecting model constraints. These operators are described in the following subsections.

3.4.1. Cell Boundary Perturbation Operator

Cell boundary perturbation operator (CBPO) modifies the boundaries of the cells by changing the cell assignments of locations. CBPO, while achieving boundary perturbation, is also required to ensure that all the locations of a particular cell are always within a single closed-loop boundary and respect the lower and upper bounds on the sizes of the cell. In applying this operator, the information in the LHS-Segment of the solution is first mapped into a rectangular greed of the specific

layout problem being solved. Once the perturbation operator is applied, the resulting cellular layout is copied back to the LHS-Segment. In order to describe this operator, additional notations and definitions are given here under where the examples are based on Figure 4 in which there are five cells with their sizes being limited between $LB = 4$ and $UB = 6$.

Additional Notations and Definitions (Refer to Figure 4-(a) for the examples):

NALOC(l) Number of Adjacent Locations to location l that are Outside the Cell to which location l belongs. If NALOC(l) > 0, location l in a given cell shares boundary with other location(s) in another cell(s). E.g., NALOC(18) = 0; NALOC(17) = 2.

NALWC(l) Number of Adjacent Locations to location l that are Within the same Cell to which location l also belongs. E.g. NALWC(18) = 4; NALWC(19) = 1.

NL(c) Number of locations in cell c. E.g. NL(1) = 5, NL(3) = 6.

Donor A donor cell is a cell that release a location for an adjacent cell when CBPO is applied.

Recipient A recipient cell is a cell that receive an additional location from an adjacent cell when CBRO is applied.

Donatable A location is said to be debatable if it can be donated to an adjacent cell without causing the donor cell be fragmented. A cell is fragmented if a single closed-loop boundary cannot be identified that contains all member locations of this cell and none from other cells.

Bond(l, c) The set of edges that a given location l in cell c is adjacent with other locations that belong to the same cell c where an edge is identified as North(N), East(E), West(W) or South(S) edge. E.g. Bond(1, 1) = {E}; Bond(3, 1) = {S, W}; Bond(18, 3) = {N, E, S, W}; Bond(1, 2) is undefined since location 1 is not in cell 2.

NE(l) The location situated in the NE corner of location l. E.g. NE(6) = 2; NE(13) = 9; NE(3) is undefined.

NW(l) The location situated in the NW corner of location l. E.g. NW(7) = 1; NW(24) = 18; NW(16) is undefined.

SE(l) The location situated in the SE corner of location l. E.g. SE(13) = 19; SE(14) =20; SE(20) is undefined.

SW(l) The location situated in the SW corner of location l. E.g. SW(13) = 17; SW(14) = 18; SW(16) is undefined.

Given the above definitions, Table 1 provides the different sets of conditions for a location to be donatable. For location l in cell c to be donatable, condition Set-0 and one of the conditions set from the remaining 9 sets must be satisfied. Condition Set-0 states that for a location to be donatable, (a) it must lie along a cell boundary and be adjacent to another cell (mathematically NALOC(l) > 0), and (b) the size of the donor cell should be higher than the lower bound (i.e. NL(c) $> LB$). Give the conditions in Set-0, if one of the conditions set from Set-1 to Set-9 is also satisfied, the donor

cell will not be fragmented if the location is donated to a recipient cell. For a cell to be recipient, it should have lesser number of locations than the upper bound ($\text{NL}(c') < UP$) and be adjacent to a donatable location from the donor cell. Figure 4 illustrates examples of allowable and not allowable boundary perturbations. For example, the move from (a) to (b) is allowed since location L6 from the donor cell C5 satisfies conditions Set-0 and Set-1 and the recipient cell C1 satisfies the conditions $\text{NL}(1) < UP$. The move from (c) to (d) is not allowed because the size of recipience cell C1 is already equal to upper bound and adding more locations to this cell will violate upper bound constraint. The move from (e) to (f) is not allowed because location L16 salsifies only Set-0 (and none from Set-1 to Set-9). Moving this location to cell C3 will cause cell C2 be fragmented. The CBPO operator always perform only allowable boundary changes. Now let as consider the procedure how this operator was applied for the move from (a) to (b). First a list of donatable locations are generated as shown in the first column of Table 2. The corresponding conditions that these locations stratify to be donatable are indicated in the second column of this table. Once, the list of donatable locations are generated, potential recipient cells are determined for each donatable location as shown in the third column. The last step is then to arbitrarily select one of the donatable locations that have recipient cell and assign this location to its recipient cell (if there are more than once recipient cells, select one arbitrarily). In this first application, location 6 was chosen arbitrarily from the many donatable locations and it was assigned to cell 1. This decision is indicated in fourth column of Table 2 and the resulting configuration is depicted in Figure 4-(b)). A second application of CBPO (the layout in Figure 4-(b) as a starting configuration) is illustrated in the last four columns of Table 2 and the resulting configuration is shown in Figure 4-(c). In the search process, this operator is applied on a solution with a probability δ_1.

Table 1: Conditions for a given location l in a given cell c to be donatable to other adjacent cell.

Conditions Set	Descriptions of Conditions	Conditions Set	Descriptions of Conditions
0	(a) NALOC(l) > 0 (b) NL(c) > LB	5	(a) NALWC(l) = 2 (b) Bond(l, c) = {S, E} (c) SE(l) is also in cell c
1	(a) NALWC(l) = 1	6	(a) NALWC(l) = 3 (b) Bond(l, c) = {N, E, W} (c) NE(l) and NW(l) are also in cell c
2	(a) NALWC(l) = 2 (b) Bond(l, c) = {N, W} (c) NW(l) is also in cell c	7	(a) NALWC(l) = 3 (b) Bond(l, c) = {N, E, S} (c) NE(l) and SE(l) are also in cell c
3	(a) NALWC(l) = 2 (b) Bond(l, c) = {N, E} (c) NE(l) is also in cell c	8	(a) NALWC(l) = 3 (b) Bond(l, c) = {N, W, S} (c) NW(l) and SW(l) are also in cell c
4	(a) NALWC(l) = 2 (b) Bond(l, c) = {S, W} (c) SW(l) is also in cell c	9	(a) NALWC(l) = 3 (b) Bond(l, c) = {S, E, W} (c) SE(l) and SW(l) are also in cell c

NOTE: A donatable location must satisfy conditions Set-0 and one set from the remaining 9 sets.

3.4.2. Other Search Operators

The other operators need to be developed are for perturbing a solution to alter location assignments of the machines and sizes of sublot. These operators are named as Machine-Location, Alternative-Route, and Sublot-Size Perturbation Operators (MLPO, ARPO and SSPO, respectivley). MLPO operates in the MDL-Segment of a solution by arbitrarily selecting two machines and

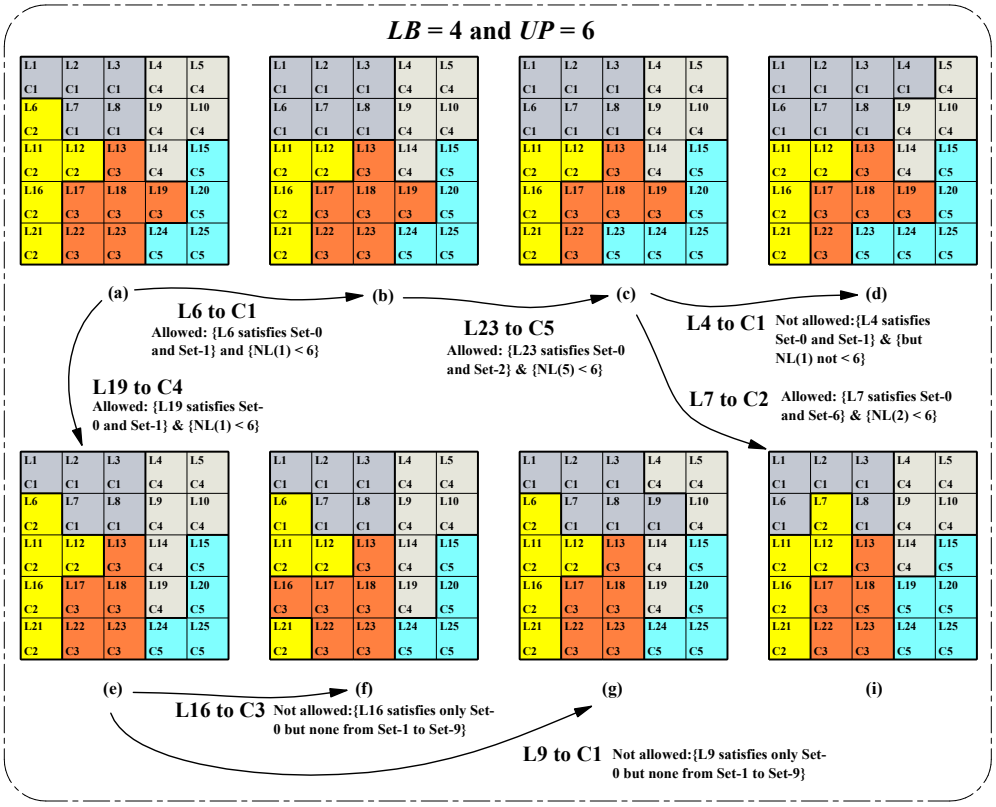


Figure 4: Examples of allowed and not allowed cell boundary changes: In each allowed move, (i) the location from the donor cell satisfies conditions Set 0 and one set of conditions from the remaining 9 sets of Table 1, and (ii) the size of the recipient cell is lower than the upper limit

swapping their location assignments with a small probability δ_2. ARPO applied with a small probability δ_3 on each $\theta_{p,r}$ in the RHS-Segment of a solution to flip its value from 0 to 1 or vice versa. In applying this operator, it is also necessary to keep $\sum_r^{R_p;} \theta_{p,r} \geq 1$ so that each product p has at least one of its route opened for its processing. SSPO applied with a small probability δ_4 on each $\beta_{p,r}$ to arbitrarily step-up or down its value with a step amount φ using equation $\beta_{p,r} = \min\{1, \quad \beta_{p,r} + \varphi\}$ or $\beta_{p,r} = \max\{0, \quad \beta_{p,r} - \varphi\}$, respectively. Each time this operator is applied, the step amount φ is determined with the equation $\varphi = \varphi_{max} \times \text{Rand}()$ where $\varphi_{max} \in (0, 1)$ is algorithm parameter and Rand() is a function generating a random number in (0, 1).

3.5. Simulated Annealing

The design of the components of the solution procedure presented in the previous subsections are unique to solving the newly proposed mathematical model. However, the fundamental concept of simulated annealing (SA) is the same across domains. The name and inspiration came from a technique called annealing in metallurgy, used to reduce the hardness of a metal by heating and gradual cooling. This gradual cooling is interpreted as a gradual decrease in the probability of accepting worse solutions as the SA explores a search space in which a particular solution X is analogous to a state a physical system and a function $E(X)$ to the internal energy of that system at that state. The goal of the search is to bring the system from an arbitrary initial state X_0 of high internal energy E to a state where this energy is at its minimum possible. In doing so, the algorithm visits a sequence of solutions X_0, X_1, $\cdots$, X_n, X_{n+1}, $\cdots$, X_N where this sequence of visitation is

Table 2: Procedure of the applications of CBPO for the moves from (a)-(b) and (b)-(c) in Figure 4

First application: Figure-4(a) to Figure-4(b)				Second application: Figure-4(b) to Figure-4(c)			
Donatable location	Condition sets satisfied Set-0 & Set-	Potential recipient cell(s)	Arbitrarily chosen move	Donatable location	Condition sets satisfied Set-0 & Set-	Potential recipient cell(s)	Arbitrarily chosen move
1	1	2		3	4	4	
3	4	4		4	5	None*	
4	5	1		6	3	2	
6	1	1	L6→C1	7	6	2	
7	3	2		8	2	4	
8	2	4		10	2	5	
10	2	5		13	1	4	
12	1	1		14	1	5	
13	1	1, 2, 4**		17	5	2	
14	1	5		19	1	4, 5	
17	5	2		22	3	2	
19	1	4, 5		23	2	5	L23→C5
21	1	None*					
22	3	2					
23	2	5					

***The size of the only adjacent recipient cell is already at the upper bound.**
****If location 13 were to be donated, one of the three recipient cells will be selected arbitrarily.**

guided by Eq. (31). In this equation, X'_n is a neighborhood solution generated by slightly perturbing X_n. The parameter T_n is the *temperature* at the n^{th} iteration. The corresponding sequence of T_n is generated in such a way that $T_{n+1} \leq T_n$ with its value approaching to zero as n increases. This sequence of temperature is called cooling schedule and the common approach is to keep the temperature at the same level for Q number of iteration and then reduce its value with an equation $T_n = T_q = \lambda T_{q-1}$. The subscript q is incremented by one every Q iterations. The coefficient λ, called the cooling exponent, has a value close to but less than one.

$$X_{n+1} = \begin{cases} X'_n & \text{if } E(X'_n) \leq E(X_n) \\ X'_n & \text{with a probability of } \left(\exp\left[\frac{E(X_n)-E(X'_n)}{T_n}\right]\right) \\ X_n & \text{otherwise} \end{cases} \tag{31}$$

In the basic SA discussed above, a single search path (X_0, X_1, $\cdots$, X_n, X_{n+1}, $\cdots$, X_N) is followed which is also a common implementation approach in literature. However, from the point of view of performance, following a single search path may not be necessary or advisable (Lee and Lee, 1996). In this paper, we adopt an implementation of SA similar to those reported in Defersha and Chen (2008) and Defersha (2015) which involves multiple search paths and parallel computation as illustrated in Figure 5. In this figure, $X_{p,s,n}$ is the solution at the n^{th} iteration along the s^{th} search path in the p^{th} process (a process is an instance of SA with S search paths that is being executed by one computing unit, core, or cpu). For better result, the search paths within a process may communicate every Z_1 iteration to start the search from the best solution so far known within that process at the current temperature level. Moreover, the processes may communicate every $Z_2 \gg Z_1$ iteration to restart their search paths from the best solution so far known across all the processes.

SA was chosen over genetic algorithm (GA) because it was not possible to design a meaningful crossover operator that cuts the LHS-segment into sub-segments and exchange between parent

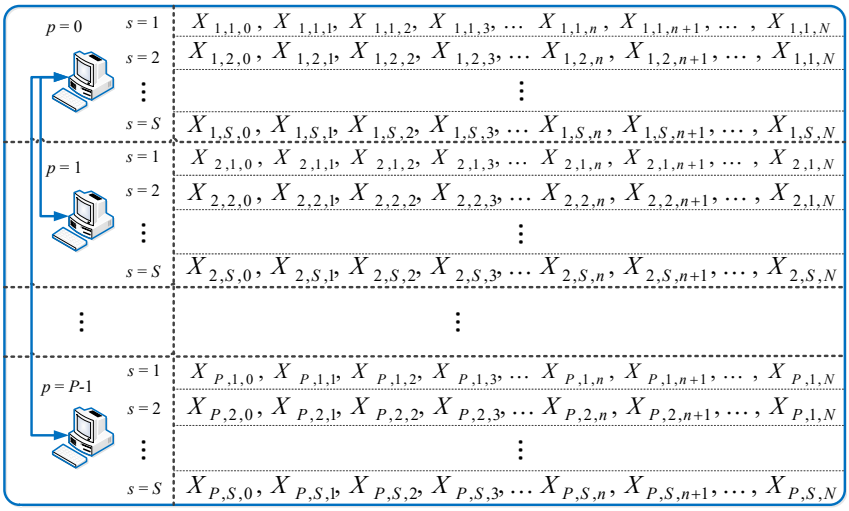


Figure 5: Schematics of a multiple search path parallel simulated annealing (the search paths are allowed to communicate within and across processors periodically).

chromosomes, since such exchange will generate solutions with disjoint cells (violating model constraints). Nevertheless, we tried to retain the global search capability of the population based GA by following multiple search path in which at each iteration many solution points are considered. It is also important to note that SA is better known for its local search capability. As such, the proposed multiple search path SA also retains this local search capability as each search path represents a simple SA. Overall, the proposed parallel multiple search path SA attempts to benefit from both global search (like GA) and local improvement strategies (like SA).

3.6. Computer Implementation

All the components of the algorithm (solution representation, initialization, search operators, evaluation) and its parallel implementation were coded in C++ programming language in which MPI message-passing library was used for communication. The code was tested in a parallel computation platforms of Calcul Québec (http://www.calculquebec.ca/en/). The particular computing environments where we test our code consists of several thousands of computing cores (20176 cores in a cluster named guillimin and 30984 cores in another cluster named mp2). The test problems were run using up to 384 cores. Each core excuses its own process[1] where a process is a single instance of SA with S search paths and uses separately seeded pseudo-random number generator to enable exploring different parts of the search space. The process with rank 0 (see footnote [2]), in addition to excusing its own S search paths, is designated to periodically gather the best solution know in each process and determine the overall best and broadcast this solution to all the other processes. The steps of the parallel multiple search path SA and the notations used to describe them are presented.

[1] In computing, a process is an instance of a computer program that is being executed. It contains the program code and its current activity.

[2] In parallel computation using MPI (massage passing interface), processes that are concurrently running for a give computational job are ranked from 0 to $P-1$ where P is the total number processes.

Notations:

p Process Rank, $p = 0,\ 2,\ ...,\ P-1$ where P is the number of concurrently running processes (a process is an instance of SA with S search paths that is being executed by one computing unit, core, or cpu)

s Index of search paths, $s = 1,\ 2,\ ...,\ S$ where S is the number of search paths followed by each process.

n Iteration counter, $n = 1,\ 2,\ ...,\ N$ where N is the maximum number of iterations in each search path.

$X_{p,s,n}$ The solution at the n^{th} iteration along the s^{th} search path in the p^{th} process.

λ Cooling schedule coefficient.

q Index for the temperature levels in the cooling schedule.

T_q Temperature at the q^{th} level, $T_q = \lambda \times T_{q-1} = \lambda^q \times T_0$.

Q Number of iterations to be performed in each search path at each temperature level.

Z_1 Number of iterations performed in each search path in each processor before a processor restarts all of its search paths (at the current level of temperature) from the best solution it has found so far.

F Global Communication frequency factor where $Z_2 = F \times Z_1$ is the number of iterations to be performed by each search path before communication is effected among the processors.

BS_p Best solution so far found in the p^{th} processor.

BS Best solution so far found by all the processors

Rand() Random number generator. Each processor uses a different seed for the random number generator.

Algorithm Steps:

Step 0. Initialization

Set p = My_Process_Rank

Initialize counter: $n = 0$ and $q = 0$. Initialize the Best-Individual BI_p with a null value. If $p = 0$, set the Best-Individual BI so far found with a null value too.

Generate initial solution points $X_{p,1,0},\ X_{p,2,0},\ \cdots\ X_{p,S,0}$ by applying the initialization technique presented in Section 3.2.

For $s = 1$ to S : Determine $E(X_{p,s,0},)$ using the method presented in Section 3.3.

Step 1. For $s = 1$ to S

Move: Using the perturbation operators presented in Section 3.4, perturb $X_{p,s,n}$ to get $X'_{p,s,n}$.

Evaluate: Determine $E(X'_{p,s,n})$.

Decide: **If** $E(X'_{p,s,n}) \leq E(X_{p,s,n})$, **then** $X_{n+1,sp} = X'_{p,s,n}$
Else If $\exp\left\{\left[E(X'_{p,s,n}) - E(X_{p,s,n})\right]/T_q\right\} >$ Rand(), **then** $X_{n+1,sp} = X'_{p,s,n}$
Else $X_{p,s,n+1} = X_{p,s,n}$

Update: **If** $E(X_{p,s,n+1}) < E(BI_p)$, **then** $BI_p = X_{p,s,n+1}$

Step 2. Set $n = n + 1$.

If n mod $Q = 0$, then set $q = q + 1$, and $T_q = \lambda \times T_{q-1}$

If $(n < N + 1)$ AND $(n$ mod $Z_2 \neq 0)$ go to Step 1.

If $(n < N + 1)$ AND $(n$ mod $Z_2 = 0)$ go to Step 3.

If $n = N + 1$, STOP.

Step 3. If $p \neq 1$, send BI_p to the process whose Process_Rank = 0.

If $p = 0$, receive the best solutions found by each process and determine BI and send this solution to all the other processors.

If $p \neq 0$, receive BI from the process whose Process_Rank = 0.

Set $X_{p,s,n} = BI$ for all s and go to Step 1.

4. Numerical Examples

In this section we consider several numerical examples to illustrate the model and the performance of the proposed algorithm. For this purpose, we generate several problem instances with varying sizes. The general features of these problems are in Table 3. The objective function weight factors, Θ_1 and Θ_2, were set in such a way that the two terms of the objective function have comparable values in the final solutions so that both terms are optimized. The number of cells and their size limits (LB and UB) are to be set at the discretion of the designer based on several design factors. Thus the values indicated in the table are arbitrary values.

4.1. Model Illustration

In this example we attempted to illustrate the applicability of the proposed model in designing both distributed layouts and cellular manufacturing systems. For this illustration we choose Problem-1 (see Table 3) having 30 part types and 25 machines. The assumed initial departmentalized functional layout is given in Figure 6-(a). The data D_p, V_p and F_p are in Table 4. We further subdivide this problem into two problems (Problem-1a and Problem-1b). In Problem-1a, processing routing were generated with a controlled randomness in such a way that rational part families based on similarities in manufacturing requirements can be identified. The part routings for this problem are given in columns 3 to 7 of Table 5. This problem may represent a scenario in which cellular manufacturing systems are appropriate. Whereas, the processing routings for Problem-1b, given in Table 6, were generated purely randomly and part families may not exist and cellular manufacturing

Table 3: General features of the problems considered

Attribute	Problem No.			
	1	2	3	4
Number of Parts P	30	60	320	600
Number of machines M	25	40	120	192
Layout shape	5×5	8×5	12×10	16×12
Number of cells C	4	6	10	12
Cell size LB	5	6	10	12
Cell size UB	7	8	14	20
Max number of Routes	2	2	3	3
Max number of operations	5	8	12	60
Min number of operations	3	4	6	8
Workload balancing factor Υ	0.85	0.9	0.9	0.9
Objective Term factors (Θ_1, Θ_2)	(0.1, 2)	(0.1, 2)	(0.1, 1000)	(0.1, 1000)

Note: Max number of Routes = $\max_{\forall p}\{R_p\}$; Max (min) number of operations = $\max_{\forall p,r}\{J_{p,r}\}$ ($\min_{\forall p,r}\{J_{p,r}\}$)

Table 4: Part data for Problems 1a and 1b

p	D_p	V_p	F_p	p	D_p	V_p	F_p	p	D_p	V_p	F_p
1	1300	1.2	2	11	1300	1.9	1	21	1200	1.6	2
2	1100	1.5	2	12	1100	1.6	1	22	1300	1.7	2
3	1100	1.5	3	13	1000	1.8	3	23	1400	1.1	3
4	1000	1.6	1	14	1200	1.7	1	24	1100	1.8	2
5	1100	1.6	2	15	1200	1.2	2	25	1100	1.4	1
6	1000	2.0	1	16	1200	1.5	1	26	1300	1.9	2
7	1000	1.6	2	17	1500	1.8	1	27	1100	1.6	2
8	1400	1.9	1	18	1100	2.0	2	28	1400	1.2	2
9	1100	1.6	3	19	1200	1.7	1	29	1300	1.9	1
10	1500	1.2	2	20	1100	2.0	1	30	1100	1.1	3

systems may not be appropriate. The inter-departmental movements of the parts in these two problems, based on the layout in Figure 6-(a), were indicated in columns 8 of Tables 5 and 6, respectively, and are comparable.

With the objective of reducing material handling and inter-cellular movement, the two problems were solved using the proposed algorithm. The proposed cellular/distributed layouts are given in Figures 6-(b) and 6-(c) and the corresponding intercellular movement are indicated in the last columns of Tables 5 and 6. In Figure 7-(a), it can be seen that there are substantial material handling cost reductions in both problems by distributing the machines over the shop floor. As such, the advantages of reduced material handling through distributed layout can be achieved in both problems. However, the intercellular movement in Problem-1b is substantially higher than that in Problem-1a as shown in Figure 7-(b). Thus, added advantages of cellular manufacturing (such as group setup, ease of scheduling, team sprit, operators accountability and satisfaction) may happen only in Problem-1a since most parts in this problem are processed in one cell only. Here it is important to note that, in literature, it has been asserted that cellular manufacturing systems are applicable in scenarios where demand is more stable and logical part families can be identified. With this assumption, a tremendous amount of models and algorithms have been developed for cellular manufacturing systems design. Distributed layout, on the other hand, are recommended in volatile environments where product demand and mix are changing very rapidly and many research articles are published in this area as well. However, we argue that a real life scenarios may lie at any point within the spectrum of these two extreme scenarios. Thus, the work in this paper bridges the research gap by integrating

distributed layout and cellular manufacturing in a single model/algorithm that can be applied in all possible scenarios along this spectrum. The model provides detail layout of the system at a machine level which we call it a distributed layout. Moreover, by assigning location to cells, it simultaneously determines both the inter and intra cellular layout and ensures machines of a particular cell lie in contiguous physical locations so that the advantages of cellula manufacturing systems can be fully exploited.

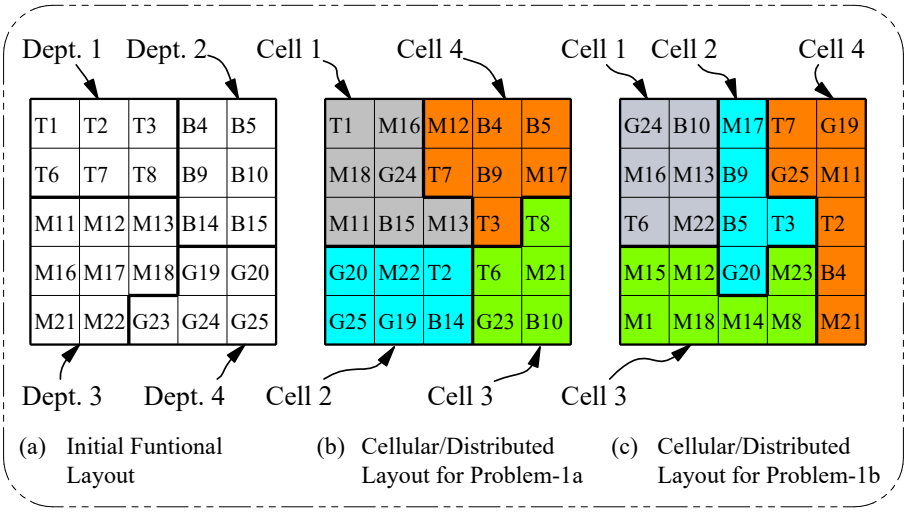
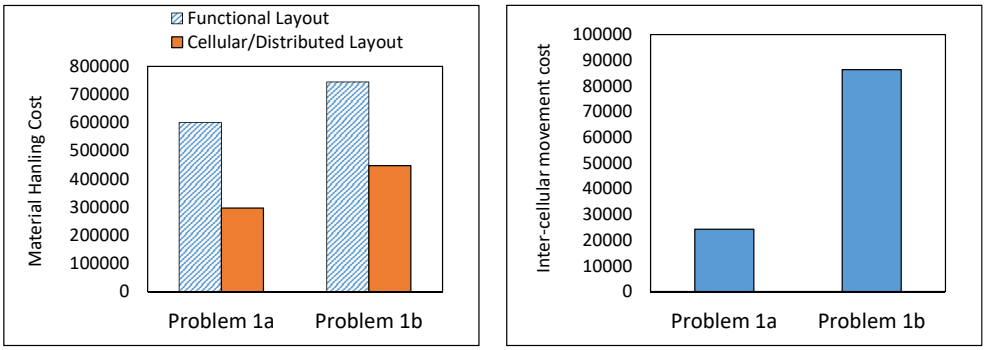


Figure 6: Initial functional layout and proposed cellular/distributed layouts for problems 1a and 1b).



(a) (b)

Figure 7: Cost savings in problems 1a and 1b).

4.2. Algorithm Performance

4.2.1. SA vs CPLEX in Solving a Small Problem

As the proposed model and the algorithm are new, we are not able to compare computational performances against published results. Instead, we attempted to show the relative performance of the algorithm against the state-of-the-art general purpose optimization package, IBM ILOG CPLEX (version 12.6). Figure 8-(a) shows a 25-hour convergence history of CPLEX in solving Problem-1 with the linearized objective function terms of the model. The first feasible solution with an objective function value of 263,330 was found after about 50 minutes of computation on a desktop PC having Intel Xeon CPU (3.2 Hz, 16.0 GB Ram). This solution kept on gradually improving, for instance to 63,350 at 7:28:25. The least value of the objective function found using CPLEX was 60,008 at

Table 5: The routing information for Problem 1a and partial solution

		Required machine and processing time ($M_{j,p,r}$, $T_{p,r,j}$) for operation j					Sequence of Dept. visitation in layout Fig. 6-a	Sequence of cell visitation in layout Fig. 6-b
p	r	1	2	3	4	5		
1	1	(12,9)	(16,8)	(18,8)	(15,7)		D3-D1	C4-C1-C4
2	1	(2,3)	(14,4)	(22,4)	(19,4)	(22,3)	D1-D2-D3-D4-D1	C2
3	1	(3,13)	(7,12)	(17,13)			D1-D3	C4
4	1	(8,9)	(21,8)	(23,7)	(6,8)		D1-D3-D4-D1	C3
5	1	(20,13)	(24,11)	(12,12)			D4-D3	C2-C1-C4
6	1	(11,3)	(16,4)	(24,5)	(20,5)	(15,3)	D3-D4-D2	C1-C2
7	1	(9,11)	(3,12)	(9,12)			*	C4
	2	(7,9)	(5,8)	(17,8)	(9,8)		D1-D2-D3-D2	*
8	1	(17,4)	(5,3)	(4,4)	(9,5)	(17,4)	D3-D2-D3	C4
9	1	(2,8)	(19,7)	(25,8)	(19,7)		D1-D4	C2
10	1	(22,9)	(19,9)	(14,7)	(19,7)		D3-D4-D2-D4	C2
11	1	(7,13)	(4,12)	(5,13)			D1-D2	C4
12	1	(24,3)	(18,3)	(24,4)	(16,3)	(1,4)	D4-D3-D4-D3-D1	C1
13	1	(19,4)	(14,5)	(22,4)	(19,5)	(14,4)	D4-D2-D3-D4-D2	C2
14	1	(13,9)	(24,9)	(15,8)	(13,7)		D3-D4-D2-D3	C1
15	1	(11,7)	(20,8)	(18,8)	(16,8)		D3-D4-D3	C1-C2-C1
16	1	(10,11)	(6,12)	(21,12)			D2-D1-D3	C3
17	1	(7,8)	(9,7)	(17,9)	(5,8)		D1-D2-D3-D2	C4
18	1	(13,8)	(15,7)	(11,9)	(18,8)		D3-D2-D3	C1
19	1	(19,7)	(14,9)	(22,7)	(25,7)		D4-D2-D3-D4	C2
20	1	(21,12)	(10,13)	(23,11)			D3-D2-D4	C3
21	1	(4,7)	(3,9)	(4,7)	(17,8)		D2-D1-D2-D3	C4
22	1	(11,11)	(20,13)	(15,12)			D3-D4-D2	C1-C2-C1
23	1	(23,3)	(10,5)	(8,4)	(6,3)	(21,4)	D4-D2-D1-D3	C3
24	1	(12,11)	(16,12)	(15,12)			D3-D2	C4-C1
25	1	(21,12)	(6,12)	(21,11)			D3-D1-D3	C3
26	1	(21,11)	(6,12)	(2,12)			D3-D1	C3-C1
27	1	(12,7)	(13,8)	(15,9)	(1,7)		D3-D2-D1	C4-C1
28	1	(8,8)	(21,7)	(8,7)	(6,8)		D1-D3-D1	C3
	2	(23,3)	(8,4)	(23,4)	(6,4)	(8,3)	*	*
29	1	(21,4)	(10,4)	(8,4)	(21,4)	(10,4)	D3-D2-D1-D3-D2	C3
30	1	(21,9)	(6,8)	(8,9)	(10,8)		D3-D1-D2	C3

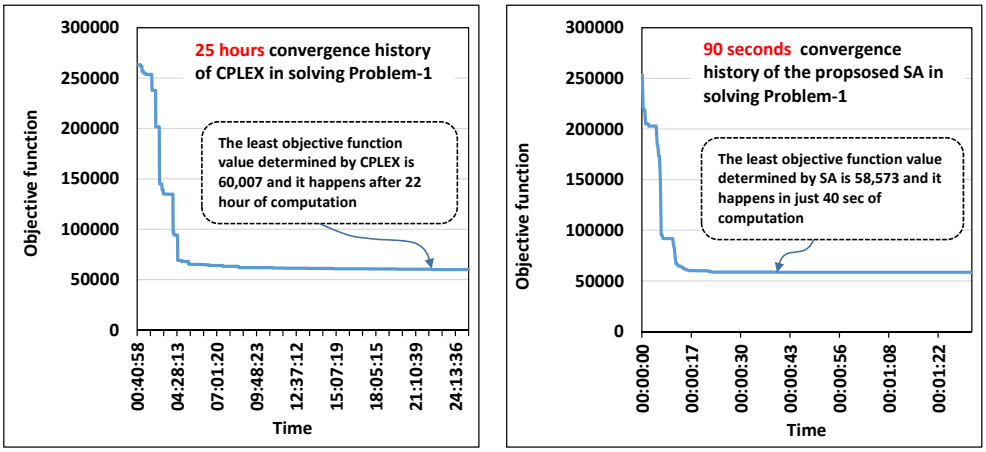
22:23:04. After about 25 hours, CPLEX stopped computing as the size of the node file generated exceeds the available memory of 750 GB in a working directory that we were able to allocate for this computation. This clearly demonstrates the difficulty of solving the proposed model, even for a small size problem, using the state-of-the-art general purpose optimization package. Whereas, as shown in Figure 8-(b), the proposed algorithm converged very quickly within 90 seconds (40,000 iterations using 1200 search paths) and found a solution better than the one found using CPLEX after 22 hours of computation. This illustrates the potential of the algorithm in solving large size problems. Figure 9 shows the computational time to perform 10,000 iteration using 1500 search path as the size of Problem-1 is increased in terms of the number machines and the number of parts. As it can be seen from this figure, the computational time increases close to linearly as the function of the problem size. This contrasts with the exact algorithm in CPLEX (branch-and-cut) of which computational time may increase exponentially as the problem size increases. This further illustrate the suitability of the algorithm in solving large size problems.

Table 6: The routing information for Problem 1b and partial solution

		Required machine and processing time ($M_{j,p,r}$, $T_{p,r,j}$) for operation j					Sequence of Dept. visitation in layout Fig. 6-a	Sequence of cell visitation in layout Fig. 6-c
p	r	1	2	3	4	5		
1	1	(6,9)	(15,8)	(20,8)	(12,7)		D1-D2-D4-D3	C1-C3-C2-C3
2	1	(2,3)	(4,4)	(9,3)	(5,5)	(16,4)	D1-D2-D3	C4-C2-C1
4	1	(8,4)	(21,3)	(3,3)	(11,3)	(4,5)	D4-D3-D1	C4
5	1	(6,3)	(22,5)	(24,3)	(7,4)	(20,4)	D1-D3-D2-D1-D4	
6	1	(17,12)	(25,13)	(21,13)			D1-D3-D2-D1-D4	C1-C3-C4-C2
7	1	(5,11)	(20,11)	(2,12)			D2-D1	C2-C1-C4
	2	(16,4)	(17,5)	(13,4)	(9,5)	(13,4)	*	*
8	1	(5,9)	(3,8)	(4,8)	(19,9)		D2-D1-D2-D4	C2-C4
9	1	(6,4)	(1,4)	(14,3)	(15,4)	(12,4)	D1-D2-D3	C1-C3
10	1	(13,12)	(22,13)	(20,12)			D3-D4	C1-C2
11	1	(3,13)	(6,12)	(17,11)			D3-D1-D3	C1-C2
12	1	(5,8)	(20,8)	(3,7)	(5,9)		D2-D4-D1-D2	C2
13	1	(2,12)	(4,11)	(21,12)			D1-D2-D3	C4
14	1	(20,4)	(14,5)	(8,4)	(18,4)	(1,4)	D4-D2-D1-D3-D1	C2-C3
15	1	(14,4)	(9,3)	(17,4)	(12,3)	(5,5)	D2-D3	C3-C2-C3-C2
16	1	(15,8)	(12,8)	(6,8)	(16,7)		D2-D3-D1-D3	C3-C1
17	1	(23,9)	(2,8)	(24,9)	(10,9)		D4-D1-D4-D2	C3-C4-C1
18	1	(18,8)	(12,8)	(6,8)	(16,7)		D3-D1-D3	C3-C1
19	1	(1,8)	(8,8)	(13,8)	(23,8)		D1-D3-D4	C3-C1-C3
20	1	(17,9)	(11,8)	(7,8)	(19,9)		D3-D1-D4	C2-C4
21	1	(14,5)	(12,4)	(4,4)	(2,5)	(3,3)	D2-D3-D2-D1	C3-C4-C2
22	1	(14,7)	(3,7)	(22,9)	(12,8)		D2-D1-D3	C3-C2-C1-C3
23	1	(23,3)	(12,5)	(5,4)	(9,4)	(2,5)	D4-D2-D1	C3-C2-C4
24	1	(5,8)	(22,8)	(13,8)	(10,7)		D2-D3-D2	C2-C1
25	1	(23,3)	(12,4)	(14,4)	(1,5)	(7,5)	D4-D3-D2-D1	C3-C4
26	1	(18,4)	(2,4)	(7,3)	(10,4)	(14,5)	D3-D1-D2	C3-C4-C1-C3
27	1	(10,12)	(6,12)	(20,11)			D2-D1-D4	C1-C2
28	1	(9,11)	(25,12)	(3,11)			D2-D4-D1	C2-C4-C2
29	1	(22,3)	(2,3)	(11,4)	(25,4)	(13,3)	D3-D1-D3-D4-D3	C1-C4-C1
30	1	(14,8)	(18,8)	(12,8)	(10,7)		D2-D3-D2	C3-C1

4.2.2. Empirical Studies

Empirical studies on the computational behaviour of both sequential and parallel multiple search path SAs for cell formation (without layout consideration) have been conducted in Defersha and Chen (2008). Defersha (2015), later demonstrated the suitability of similar implementation of SA for flowshop scheduling. As the implementation strategies followed in this paper are similar to those found in Defersha and Chen (2008); Defersha (2015), we are not presenting detail empirical studies of the computational behavior of the proposed algorithm. However, a limited empirical study is essential since the algorithm developed is specific to the proposed model with new solution representation, new initialization techniques and new move operators. This empirical study is to demonstrate the algorithm performance improvements achieved through (1) following multiple search paths, (2) interaction of search paths and (3) high performance parallel computation. For the demonstrations (1) and (2), the problem with 60 part types and 40 machines (Problem 2) is considered. This problem was first solved using a single search path SAs in which the algorithm runs for 150,000,000 iterations (for about 2 hours using 2.1 GHz cpu). The algorithm was executed for eight complete runs by changing the seed of the random number generator (of the programming language used, c++) and then the average value of the objective function of the final solutions was computed. We repeat this experiment for a multiple search path SA by increasing the number of the search paths at a time



(a) CPLEX (b) SA

Figure 8: Convergence histories of CPLEX and the proposed SA in solving problem-1

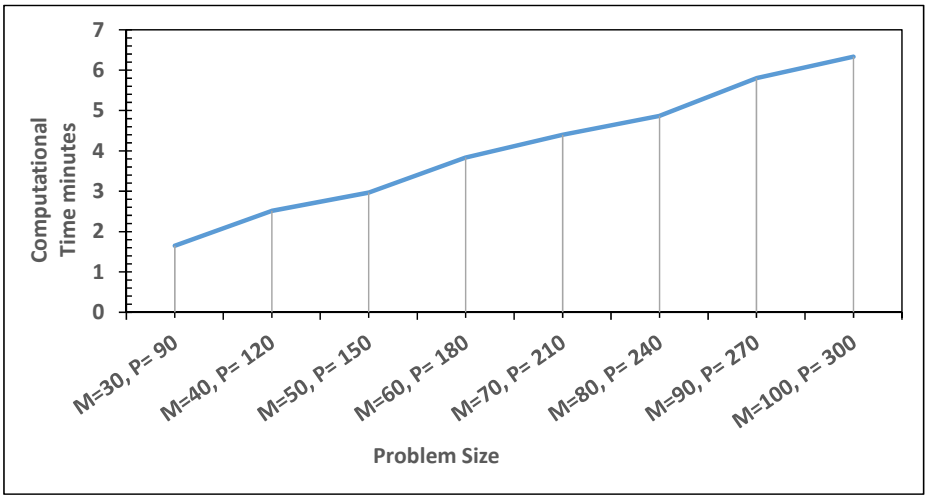


Figure 9: Time required to perform 10,000 iteration using 1500 search paths as a function of problem size (M = number of machines, P = Number of Parts).

from 2 to 5, to 10 e.t.c. As we increase the number of search paths in each test, each search path is shortened to keep the computational time and the total number of iteration remain the same as that of the single search path SA. The averages values of the objective function are plotted in Figure 10. From this figure it can be seen that as the number of search paths is increased from 1 to 20, the quality of the final solution generally keeps on improving. This clearly demonstrates the benefit of excusing multiple short SA runs instead of a single long run.

However, as the number of search paths is further increased for the same total number of iterations, the qualities of the final solutions start to deteriorate since each search path cannot get enough run length to converge within a given time limit. A technique that will enable a large number of search paths to run to convergence within the given time can be parallel computing. In this case, the search paths are to be divided in to smaller batches and allocated to concurrently available processors. For better performance, the search paths within and across processes can interact periodically. In those interactions, the best solutions found from all the search paths are collected and the overall best (winner) solution is distributed to all the search paths and each search path continues its search from this solution. Figure 11 shows the convergence history in eight runs without search

path interaction and in another eight runs with search path interaction of the 15-search path SA in solving Problem 2. The result clearly demonstrates a substantial performance improvement both in convergence rate and final solution quality as a result of search paths interaction. Moreover, from this convergence graph, the final solution qualities form the eight runs of the multiple search path SA with interaction are more or less the same regardless of the starting sets of solution.

In order to illustrate the algorithm performance improvement that can be achieved using parallel computation, we consider Problems 3 and 4 (see Table 3) as these problems are very large in size and may represent real life scenarios with greater computational challenge. Figure 12 shows the convergence history of the process with rank 0 in solving Problem 3 as the number of processes is increased from 1, 8, 24, 48, to 96. In these computations, the number of search paths in each process was kept at 500 and each search path runs for 50,000 iterations. The search paths were allowed to interact every 1000 iterations within a process and every 25,000 iterations across the processes. The computational time remains at about 2 hours and it is not impacted as the number of processes is increased as each process is excused in its own assigned CPU in the parallel computing environment. The abrupt change in convergence history in the process with rank 0 (or in any given process) happens when all the processes communicate to determine the winer solution and continue their respective iterations from this solution. As it can be seen from this convergence history, there is a 32% improvement in the final solution quality by using parallel computation with search path interactions. The level of improvement is even bigger in Problem 4, which is at 58% as shown in Figure 13 as the number of process is increased from 1, 8, $\cdots$, to 384 for a fixed computation time (which was 6 hours and 12 minutes in this problem). This clearly demonstrate that, though SA is generally able to find good solutions for small size problems, the search process is likely to become trapped in a local optima when solving large size problems. In this case, as illustrated by these examples, the best alternative can be the use of high performance parallel computation.

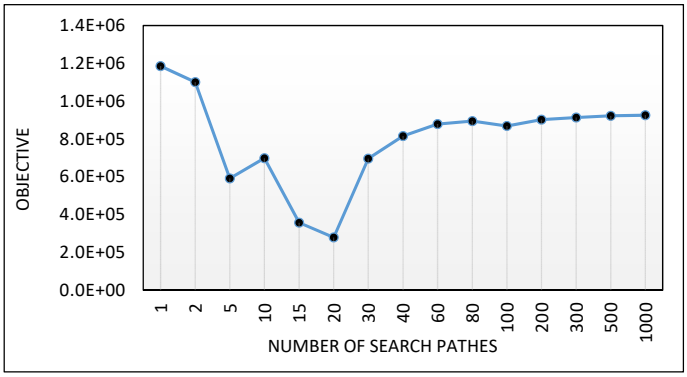


Figure 10: The effect of increasing the number of search paths on the final solution quality for the same total number of iteration

5. Discussion and Conclusions

CMS involves the grouping of parts having similar processing requirements into part families, and organizing machines along other supporting recourses into cells such that each cell produces a part family with at most efficiency. Several industrial surveys in literature found that system performance improvements achieved through CMS implementations are astounding, and the area draws enormous

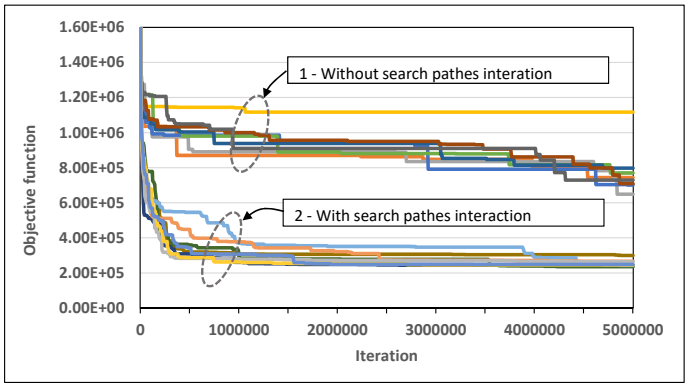


Figure 11: The effect of search paths interaction on the convergence of 15-search-path SA in solving Problem 2 during the first 500000 iterations (One hour computational time)

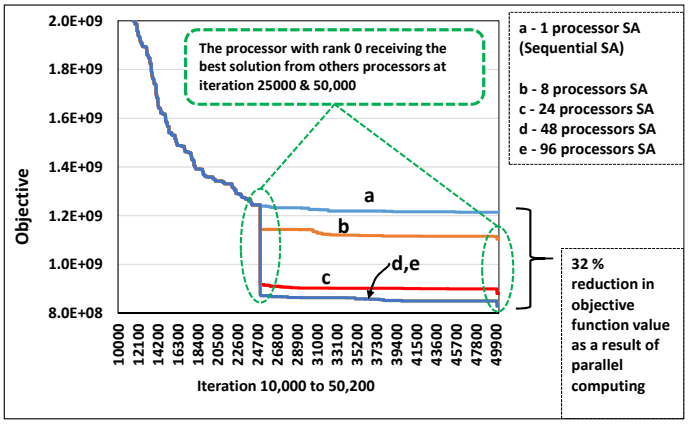


Figure 12: The convergence history of the process with rank 0 in the parallel SA in solving Problem 3. The abrupt change in convergence in a given process happens when all the processes communicate to determine the winer solution and continue their respective iteration from this solution.

research. However, there are some studies that challenge the claims, the primary concerns being CMS are not applicable in today's volatile environments where demand and product mix change rapidly. To address this issue, distributed layout has been emerging as an alternative to cellular layouts and it is gaining traction in literature. However, we argue that changes in product demand and mix in a real life system may always lay within the spectrum of being stable and volatile. This paper bridges this gap by providing a mathematical model and a solution procedure that integrate distributed layout design and machine cell formation. The mathematical model:

1. Minimizes material handling cost through distributing the machines over the shop floor both in stable or volitive (or any given scenario) as it was demonstrated in the numerical example.

2. Minimizes intercellular movement by identifying cells and part families whenever possible.

3. Ensures that machines that belong to the same cell are laid out on contiguous physical locations so that the advantages of cellula manufacturing systems can be fully exploited.

4. Provides detail layout at a machine level and determine both inter-cellular and intra-cellular configurations.

5. Incorporates many pragmatic issues such as operations sequence, alternative routing, lot splitting, workload balancing among cells and constraints on machine capacity.

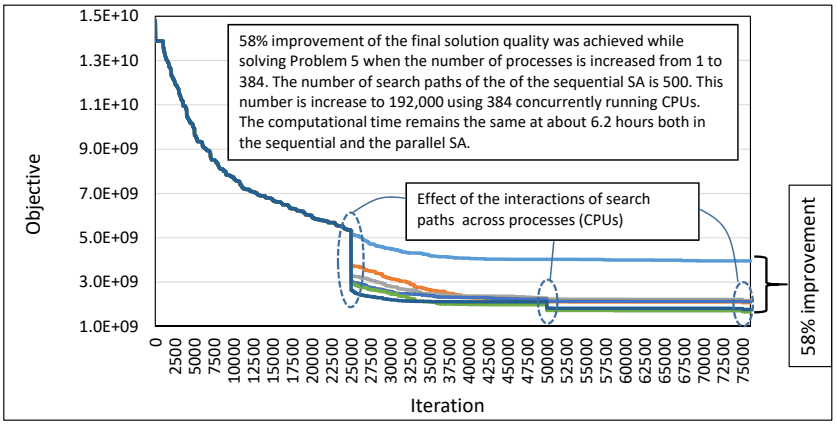


Figure 13: The convergence history of the process with rank 0 in the parallel SA in solving Problem 5 as the number of concurrently running processes is increased from 1, to 8, 24, 48, 96, 192, and to 384.

6. If a distribute layout is given, enables the formation and reconfiguration of virtual cells over this layout with a unique advantage of forming each virtual cell with a group of machines on contiguous physical location. From a solution procedure point of view, this is possible by populating the MDL-Segment of the solution representation using the information obtained from the given distributed layout and search for the cell boundaries while keeping the machine configuration unchaged.

Numerical example showed that solving the proposed model using off-the-shelf optimization package is difficult even for small size problems. To this end, we develop an efficient SA based algorithm. The SA follows multiple search paths with interactions. From the results of the test problems, it is evident that instead of excusing a single long SA run, it is much preferable to execute multiple short runs. The interaction of the search paths also resulted in substantial improvement of the convergence speed of the algorithm. The parallel implementation of the algorithm demonstrated substantial improvement in final solution quality when solving large problems. These clearly demonstrate that, though SA is generally able to find good solutions for small size problems, the search process is likely to become trapped in a local optima when solving large size problems. In this case, as illustrated by these examples, the best alternative can be the use of high performance parallel computation. Our future work in this area include extensive simulation based investigation of manufacturing systems based on non-conventional layouts and developing algorithms and methods for production scheduling of these non-conventional manufacturing systems where there is a limited research.

Acknowledgements:

This research is supported by the Discovery Grant from the National Science and Engineering Research Counsel of Canada, NSERC. We would like to thank Compute Canada (https://www.computecanada.ca/) and its Consortiums specially Calcul Québec (http://www.calculquebec.ca/en/) for providing access to high performance parallel computing infrastructure.

References

Alfa, A. S., Chen, M., and Heragu, S. S., 1992. Integrating the grouping and layout problem in cellular manufacturing. *Comuters and Industrial Engineering*, **23 (1-4)**, 55–58.

Aririguzo, J. C., Saad, S. M., and Nkwogo, U. O., 2013. A genetic algorithm approach to designing and modelling of a multi-functional fractal manufacturing layout. Proceedings of the 11th International Conference on Manufacturing Research (ICMR2013). Cranfield, UK, pp. 399–404.

Arvindh, B. and Irani, S. A., 1994. Cell formation: the need for an integrated solution of the subproblems. *International Journal of Operations Research*, **32 (5)**, 1197–1218.

Askin, R. G., 2013. Contributions to the design and analysis of cellular manufacturing systems. *International Journal of Production Research*, **51**, 6778–6784.

Askin, R. G. and Estrada, S., 1999. An investigation of cellular manufacturing practices. Wiley, New York, pp. 25–34.

Ballakur, A. and Steudel, H. J., 1987. A within-cell utilization based heuristic for designing cellular manufacturing systems. *International Journal of Production Research*, **25**, 639–665.

Baykasoglu, A., 2003. Capability-based distributed layout approach for virtual manufacturing cells. *International Journal of Production Research*, **41**, 2597–2618.

Baykasoğlu, A. and Göçken, M., 2010. Capability-based distributed layout and its simulation based analyses. *Journal of Intelligent Manufacturing*, **21 (4)**, 471–485.

Benjaafar, S. and Sheikhzadeh, S., 2000. Design of flexible plant layouts. *IIE Transactions*, **32**, 309–322.

Bozer, Y. Z., Meller, R. D., and Erlebacher, S. J., 1994. An improvement-type layout algorithm for single and multiple-floor facilities. *Management Science*, **40**, 918–932.

Chandrasekharan, M. P. and Rajagopalan, R., 1986. An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *International Journal of Production Research*, **24**, 451–464.

Choobineh, F., 1988. A framework for the design of cellular manufacturing systems. *International Journal of Production Research*, **26**, 1161–1172.

Defersha, F. M., 2015. A simulated annealing with multiple-search paths and parallel computation for a comprehensive flowshop scheduling problem. *International Transactions in Operations Research*, **22**, 669691.

Defersha, F. M. and Chen, M., 2006. 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., 2008. A parallel multiple markov chain simulated annealing for multi-period manufacturing cell formation problems. *International Journal of Advanced Manufacturing Technology*, **37**, 140156.

Flynn, B. B. and Jacobs, R. R., 1987. An experimental comparison of cellular (group technology) layout with process layout. *Decision Sciences*, **18**, 562–581.

Forghani, K., Mohammadi, M., and Ghezavati, V., 2015. Integrated cell formation and layout problem considering multi-row machine arrangement and continuous cell layout with aisle distance. *International Journal of Advanced Manufacturing Technology*, **78**, 687–705.

Gonçalves, J. F. and Resende, M. G. C., 2004. An evolutionary algorithm for manufacturing cell formation. *Computers & Industrial Engineering*, **47**, 247–273.

Hamedi, M., Ismailand, N. B., Esmaeilian, G. R., and Ariffin, M., 2012. Developing a method to generate semi-distributed layouts by genetic algorithm. *International Journal of Production Research*, **50 (4)**, 953–975.

Hyer, N. L. and Wemmerlöv, U., 1989. Groupt technology in the us manufacturing industry: A survey of current practice. *International Journal of Production Research*, **27**, 1287–1304.

Kannan, V. R. and Ghosh, S., 1995. Using dyanamic cellular manufacturing to simply scheduling in cell based production system. *Omega, The International Journal of Management Science*, **23**, 443–452.

Kia, R., Shirazi, H., Javadian, N., and Tavakkoli-Moghaddam, R., 2015. Designing group layout of unequal-area facilities in a dynamic cellular manufacturing system with variability in number and shape of cells. *International Journal of Production Research*, **53**, 3390–3418.

King, J., 1980. Machine-component grouping in pfa: an approach using a rank-order clustering algorithm. *International Journal of Production Research*, **18**, 213–232.

Kioob, S. A., Bulgak, A. A., and Bektas, T., 2009. Integrated cellular manufacturing systems design with production planning and dynamic system reconfiguration. *European Journal of Operational Research*, **192**, 414–428.

Knoll, D., Morel, J., Margolin, L., and Shashkov, M., 2005. Physically motivated discretization methods. *Los Alamos Sience*, **26**, 188–212.

Krishnan, K. K., Mirzaei, S., Venkatasamy, V., and Madhusudanan, P. V., 2012. A comprehensive approach to facility layout design and cell formation. *Intranational Journal of Advanced Manufacturing Technology*, **59**, 737–573.

Lahmer, M. and Benjaafar, S., 2005. Design of distributed layouts. *IIE Transactions*, **37**, 303–318.

Lee, S.-Y. and Lee, K. G., 1996. Synchronous and asynchronous parallel simulated annealing with multiple markov chains. , **7**, 903–1007.

McAuley, J., 1972. Machine grouping for efficient production. *Production Engineering*, **51**, 53–57.

Mohammadi, M. and Forghani, K., 2016. Designing cellular manufacturing systems considering s-shaped layout. *Computers & Industrial Engineering*, **98**, 221–236.

Montreuil, B., LeFrancois, P., Marcotte, S., and Venkatadri, U., 1993. Holographic layout of manufacturing systems operating in chaotic environments. Tech. rep., Technical Report.

Montreuil, B. and Venkatadri, U., 1991. Strategic interpolative design of dynamic manufacturing systems layout. *Management Science*, **37**, 682–694.

Montreuil, B., Venkatadri, U., and Rardin, R. L., 1999. Fractal layout organization for job shop environments. *International Journal of Production Research*, **37**, 501521.

Moraglio, A., 11 2007. Towards a gemoetric unification of evolutionary algorithms. Ph.D. thesis, Department of Computer Science, University of Esssex.

Morris, J. S. and Tersine, R. J., 1990. A simulation analysis of factors influencing the attractiveness of group technology cellular layouts. *Management Science*, **36**, 1567–1578.

Nageshwaraniyer, S., Khilwani, N., Tiwari, M., Shankar, R., and Ben-Arieh, D., 2013. Solving the design of distributed layout problem using forecast windows: A hybrid algorithm approach. *Robotics and Computer-Integrated Manufacturing*, **29 (1)**, 128–138.

Rajagopalan, R. and Batra, J., 1975. Design of cellular production systems–a graph theoretic approach. *International Journal of Production Research*, **13**, 56–68.

Rosenblatt, M. J. and Golany, B., 1992. A distance assignment approach to the facility layout problem. *European Journal of Operational Research*, **57**, 253–270.

Saad, S. M. and M., L. A., 2004. Layout design in fractal organization. *International Journal of Production Research*, **42**, 3529–3550.

Safaei, N., Saidi-Mehrabad, M., and Jaba-Ameli, M. S., 2008. A hybrid simulated annealing for solving an extended model of dynamic cellular manufacturing system. *European Journal of Operational Research*, **185**, 563–592.

Shafigh, F., Defersha, F. M., and Moussa, S. E., 2017. A linear programming embedded simulated annealing in the design of distributed layout with production planning and systems reconfiguration. *International Journal of Advanced Manufacturing Technology*, **88**, 11191140.

Suresh, N. C. and Meredith, J. R., 1994. Coping with the loss of pooling synergy in cellular manufacturing systems. *Management Science*, **40**, 466–483.

Urban, T., Chiang, W. C., and Russel, R. A., 2000. The integrated machine allocation and layout problem. *International Journal of Production Research*, **38 (13)**, 2911–2930.

Vakharia, A. J. and Wemmerlöve, U., 1990. Designing a cellular manufacturing system: a materials flow approach based on operation sequences. *IIE Transactions*, **22**, 84–97.

Venkatadri, U., Rardin, R. L., and Montreuil, B., 1997. A design methodology for fractal layout organization. *IIE Transactions*, **29**, 911924.

Wang, T.-Y., Lin, H.-C., and Wu, K.-B., 1998. An improved simulated annealing for facility layout problems in cellular manufacturing systems. *Computers and Industrial Engineering*, **34**, 309–319.

Warnecke, H. J., 1993. The Fractal Company A Revolution in Corporate Culture. Springer, Berlin,

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

Wemmerlöve, U. and Hyer, N. M. L., 1989. Cellular manufacturing in the US industry: A survey of users. *International Journal of Production Research*, **27**, 1511–1530.