A new fuzzy genetic algorithm for the dynamic bi-objective cell formation problem considering passive and active strategies

Menouar Boulif a,*, Karim Atif b

a Département d’Informatique, Faculté des Sciences, Université M’hamed Bouguerra à Boumerdès, Avenue de l’indépendance, 35000 Boumerdès, Algeria
b Département d’Informatique, Faculté Génie Electrique et Informatique, Université des Sciences et de la Technologie Houari Boumediene, BP 32, El-Alia, 16111 Babazouar, Alger, Algeria

Received 22 April 2005; received in revised form 25 March 2007; accepted 28 March 2007
Available online 2 April 2007

Abstract

This paper deals with manufacturing cell formation considering the dynamic behavior of the production system. First, we discuss the importance of taking into account the dynamic aspect of the problem that has been poorly studied in the related literature. We argue that by considering a multi-periodic planning horizon, we can tackle the problem according to two strategies: passive and active. The first strategy consists of maintaining the same composition of machines during the overall planning horizon, while the second allows performing a different composition for each period. We present a graph theory model for the problem. For the passive cell formation problem (PCFP), we prove that it amounts to a cell formation problem in a static system. In order to solve the active cell formation problem (ACFP), we propose a Shortest Path heuristic (SP) and a Genetic Algorithm (GA) based method. When the decision maker wants to choose the most adequate strategy for its environment, we state that the decision problem between active and passive strategies can be solved by solving ACFP. However, the complexity of ACFP justifies the need to control the presented solving approaches. In this situation, we propose a new fuzzy logic enhancement to the GA. The results, using this enhancement, are better than those obtained using the SP method or the GA alone.

Keywords: Manufacturing cell formation; Dynamic production system; Active and passive strategies; Graph partitioning; Linguistic fuzzy modeling; Genetic algorithm

1. Introduction

Cellular Manufacturing Systems (CMS), an implementation of the Group Technology (GT) concept, consist of dividing the manufacturing system into cells so that similar parts are all produced in the same cell. Such systems are specifically designed for job shops whose production is average in terms of variety and volume.

* Corresponding author.
E-mail address: boumen@umbb.dz (M. Boulif).
The success of CMS is rooted in their proven ability to reduce set-up times, in-process inventories, lot sizes and production equipment while improving productivity and facilitating the mastering of the production system.

There are three important steps in CMS design: (1) cell formation, (2) machine layout and (3) cell layout. This paper deals with machine clustering which is one of the main tasks of the first step.

Several works have been proposed for manufacturing cell formation (MCF) problem. These works can be broadly divided into methods based on the part-machine incidence matrix, on similarity coefficients and on heuristics. The first group of this non-exclusive classification generally proceed by swapping rows and/or columns of the part-machine incidence matrix (PMIM) to yield a diagonal block structure from which part families and machine cells are obtained [17,18]. In the second one, a measure of similarity between machines (or parts) is used to determine the clusters [24,9,15]. Finally, the third group has been of a great interest for researchers since the NP-hardness of the problem has been demonstrated. Recent researches in this group concern using meta-heuristics [10,22,11], or combining them for building more efficient methods [16,4].

In order to fill in the gap between theoretical models and real-life circumstances, some researches tried to consider some real-life aspects in their models, like routing flexibility [21] and tooling requirement [30]. However, to make the problem less difficult, the majority of these works suppose that part types and their demands are constant during the overall planning horizon. This strong and constraining assumption weakens all approaches trying to solve the related problems. Indeed, the evolution of the production environment, characterized by an important demand disturbance and a merciless market, leads to an adaptation deficiency: any modification in the market tendencies causing a change in part types or demands, would lead to a possible decrease or even a collapse of the manufacturing system performance. Therefore, to face these risks would require taking into account the dynamic aspect of the production environment.

Being informed by the dynamic requirements around the beginning of the nineties [14], the dynamic aspect of the MCF problem has been tackled by Wicks [34] who proposed a multi-period formulation to the part family and machine cell formation problem. He used three minimizing design objectives, namely, the material handling cost, the investment in additional machines and the system reconfiguration costs over the planning horizon. He presented a Genetic Algorithm (GA) based method for clustering machines into cells. The GA embodies a problem specific heuristic for assigning parts to the cells. The work of Wicks has been published later on [35]. Chen [5] developed a dynamic programming based approach, using a mixed integer programming model, to minimize intercellular material and machine costs as well as reconfiguration cost. First, he solves a reconfiguration cost relaxed version of his model to get the best cell configuration for each period. Then, he uses a dynamic programming procedure to find a solution for its original problem, by considering in each period all the previous cell configurations. It can be stated that a best solution for the original problem can be composed of cell configurations that are not the best in any period of the planning horizon. Taboun et al. [31] proposed a mathematical programming model for the multi-periodic machine cell and part family formation problem. However, they restricted the change only to part families. They introduced six criteria in their objective function, namely, cell configuration, machine capital investment, machine procurement and salvage, idle time, intercellular movement and part subcontracting costs; the first two costs being considered for the initial period only. Due to the big size of the proposed mathematical programming model, a two-stage heuristic is used to solve it. In the first stage, part families are formed and possible machine contents of cells are determined using a maximum similarity-minimum machines heuristic, and in the second stage, a part family formation relaxed version of the original model is solved in order to allocate the remaining machines to the cells. As mentioned earlier, the authors sought to find a cell composition that will be maintained during the planning horizon. It can be stated that this so called passive strategy is not convenient for all situations because it can happen that a modification in the product types must be followed by a modification in the cell composition in order to maintain competitiveness. Mungwatanna [25] developed a mathematical model for the machine cell formation problem considering multi-routing parts; then, he used a simulated annealing heuristic to solve it. In all the works that consider reconfiguration cost [5,25,34,35] the authors suppose the number of cells is known a priori, and it is also supposed to be static during the overall planning horizon. Mungwatanna [25] argues this restrictive supposition will allow reducing the complexity of the model and consequently, the size of the search space. However, we can say that a dynamic number of cells can be an important optimizing parameter. Furthermore, as the following example shows, it can be a necessity. Indeed, consider the situation where all the cells of the plant are saturated. Additional machines will require elevating the number of cells, since they cannot be placed
in the existing cells. We can say that this restriction gave a considerable help to simplify the reconfiguration cost definition. Nevertheless, this definition must be reformulated when we consider a varying number of cells. Further discussion on the reconfiguration cost definition is given in Section 2.2.4.

In addition, all the mentioned works do not consider some important realistic constraints such as cohabitation and non-cohabitation. Cohabitation constraints allow taking into account those machines whose dangerous or particular energy sources require that they be placed in close proximity so as to facilitate their supervision and satisfy their energy needs, and non-cohabitation constraints allow considering that, for example, high precision machines must be placed far from those that generate a high level of vibration.

Our paper aims to contribute to these research efforts by proposing an approach that tries to approximate real life circumstances in MCF problem solving. We think that this aim can be achieved by taking into account the dynamic behavior of the input data (e.g., product types and part demands variation); by considering realistic constraints (e.g., machine cohabitation and non-cohabitation) and by avoiding restrictive assumptions like static number of cells. We also tried to give a useful help in decision making by considering the decision problem between passive and active strategies. All these issues are addressed in a new approach using a combination of the Genetic Algorithm (GA) and the Fuzzy logic (FL).

The remaining of this article is organized as follows: in Section 2, we state the problem and show that it can be tackled according to two strategies: active and passive; then, we present for the first strategy a graph theory based formulation. Under the passive strategy, we show that the problem amounts to a static cell formation problem. Section 3 is devoted to the problem solving under the passive strategy. In the following Section, we present two solving methods for the active strategy: the first is based on the shortest path algorithm, while the second is based on the genetic algorithm. In Section 5, we discuss the need to control the search done by any heuristic method when the decision maker seeks the most adequate strategy for the manufacturing system environment. Consequently, we propose to provide to the GA an enhancement based on the FL. In Section 6, we illustrate the application of the presented methods on a hypothetical example that shows the worthwhile role of the FL enhancement. Finally, Section 7 presents our conclusions as well as our recommendations for further research.

2. Problem modeling

2.1. Problem statement

MCF problem consists of grouping (or clustering) machines into cells and part types into families. Each part family is dedicated to a machine cell such that parts of the same family are essentially processed in their associated cell. When the two clustering tasks (part and machine clustering) are not done simultaneously, it is possible to deduce the part clustering from the machine clustering, and vice versa.

In the Static Manufacturing Cell Formation problem, the data inputs (i.e., part types, part demands) are supposed constant during the overall planning horizon but for the Dynamic Manufacturing Cell Formation (DMCF) problem, these inputs can vary significantly.

As a first approach to the DMCF problem, we suppose that these data inputs vary according to a known number of periods in the planning horizon. These data, supposed constant in each period, define what we call a scenario. Given these considerations, the DMCF problem can be stated according to two strategies, passive and active, as follows:

Consider a dynamic manufacturing system verifying the following hypotheses:

1. The planning horizon is composed of a given number of periods, each with a known duration.
2. In each period, the manufacturing system is submitted to a unique scenario of part demands.
3. Each part type has a unique operation sequence and each operation is done in a unique machine.
4. For each period, the set of available machines meets the production requirements.

On one hand, for the Active Cell Formation Problem (ACFP), we seek a sequence of cell compositions, one composition for each period, minimizing the cost of production. On the other hand, for the Passive Cell Formation Problem, we seek one cell composition able to tolerate, for the best, the scenario changes.
All the cell compositions must respect the following practical constraints:

1. an upper bound for cell size;
2. upper and lower bounds for the number of cells;
3. cohabitation constraints requiring some machines to be placed in the same cell; and
4. non-cohabitation constraints requiring the separation of some machine couples which are then placed in different cells.

In order to define the production cost to be minimized, several criteria have been used in the literature. One of the most considered is the number of intercellular part moves. Because exclusive use of the binary incidence matrix inputs (PMIM) limits the number of decisive and meaningful parameters taken into consideration, recent research tends to combine PMIM information with other inputs [15,22]. One of these, the production volume transported, is commonly chosen since any realistic approach to the cell formation problem must take part volume data into account [32].

However, if the minimization of the intercellular part moves enables the optimization of the cell composition, tackling the MCF problem taking into account the dynamic aspect, requires the optimization of the reconfiguration cost too. Indeed, when different compositions are set in two consecutive periods, a cost is generated due to both cell and machine movements that can induce a considerable unproductive time.

The following section presents a bi-criterion formulation for the stated problem, taking into account the highlighted considerations of the objective function.

2.2. A graph theory formulation for ACFP

The static MCF problem has a classical graph theory based formulation [26,4] that we shall extend to ACFP by defining the input data, the constraints and the bi-criterion objective function as follows:

2.2.1. Input data

(1) Let us consider \( M = \{ M_1, M_2, \ldots, M_m \} \) the set of the \( m \) available machines and \( P = \{ P_1, P_2, \ldots, P_p \} \) the set of the \( p \) part types. Parts in \( P \) are manufactured on machines in \( M \) over a planning horizon of \( H \) periods, each with a duration \( d_h \) expressed in time units.

Remark 1. The set \( P \) gathers all the \( H \) different part sets to be manufactured in the \( H \) periods and, similarly, \( M \) gathers all the \( H \) machine sets available in each period.

(2) For each part type \( P_k (k = 1, 2, \ldots, p) \), we suppose given:
   (i) \( R_k \): a single sequence (or routing) of machines to be visited by the part, \( R_k = (M_{k,1}, M_{k,2}, \ldots, M_{k,s_k}) \), where: \( (M_{k,j}) \in M (j = 1, 2, \ldots, s_k) \) and \( s_k \) is the number of machines in \( P_k \)'s routing.
   (ii) \( r_{kh} \): the mean production volume\(^1\) of part type \( P_k \) per time unit during the period \( h (h = 1, \ldots, H) \).

(3) For each \( (P_k, M_i, M_j) \in P \times M \times M \), we denote as \( v_{k,i,j} \) the number of times in which \( M_i \) follows \( M_j \) or inversely \((i,j = 1, \ldots, m)\) in the routing of \( P_k \).

(4) For each couple \((M_i, M_j) \in M \times M \), we denote the \((M_i, M_j)\) inter-machine traffic \((i,j = 1, \ldots, m)\) during the period \( h (h = 1, \ldots, H) \) by:

\[
t_{i,j,h} = \sum_{k=1}^{p} r_{kh} \times v_{k,i,j}
\]

\(^{1}\) Since the part demand fluctuates, the periods are determined by identifying the time intervals in which this demand varies slightly. The maintained part demand value in each identified period is set equal to the mean (or the maximum) of the actual demands in that period.
(5) For each period \( h, h = 1, \ldots, H \), we define a network \( N_h = (M, E, W_h) \) where \( M \) is the set of available machines during the planning horizon, \( E = \{ e_{ij} | (i, j) \in M \times M \text{ and } \exists k = 1, \ldots, H: t_{ijk} \neq 0 \} \) is the set of edges that link every machine couple connected by a part movement in any period, and \( W_h \) is an edge weighting function that defines the amount of part traffic between machines during the period \( h \) (i.e., \( W_h(e_{ij}) = t_{ijk} \)).

(6) For every period \( h \), we define:

(i) a partition \( C_h \) of \( M \) in \( J_h \) cells, \( J_h \in \{1, \ldots, m\} \), such that \( C_h = \{C_{h1}, C_{h2}, \ldots, C_{hJ_h}\} \) and \( \forall h = 1, \ldots, H: \)

\[
\bigcup_{i=1}^{J_h} C_{hi} = M \text{ and } C_{hi} \cap C_{hj} = \emptyset \quad \forall i, j \in \{1, \ldots, J_h\}, \ i \neq j;
\]

(ii) the subset \( E(C_h) \) of intercellular edges related to the partition \( C_h \):

\[
E(C_h) = \{e_{kl} \in E | (M_k, M_l) \in C_{hi} \times C_{hj}; \ i, j = 1, 2, \ldots, J_h; \ i \neq j \text{ and } k, l = 1, 2, \ldots, m\}
\]

(iii) the normalized intercellular traffic \( T(C_h) \) of the partition \( C_h \):

\[
T(C_h) = d_h \cdot \sum_{e_{kl} \in E(C_h)} W_h(e_{kl}) / \left( \sum_{h} d_h \cdot \sum_{e_{kl} \in E} W_h(e_{kl}) \right)
\]

(7) for every consecutive composition couple \( (C_h, C_{h+1}), h = 1, \ldots, H - 1 \), we define the cell reconfiguration (recomposition) normalized cost, \( R(C_h, C_{h+1}) \), that must evaluate the cost of the changes to be applied to \( C_h \) to get the next composition, \( C_{h+1} \). The formal definition of this cost will be the subject of a discussion in Section 2.2.4.

(8) for every \( H \) consecutive compositions \( S = (C_1, \ldots, C_H) \) that cover the entire planning horizon we define:

(i) the entire normalized intercellular traffic \( TE(S) \):

\[
TE(S) = \sum_{h=1}^{H} T(C_h)
\]

(ii) the entire normalized reconfiguration cost \( RE(S) \):

\[
RE(S) = \sum_{h=1}^{H-1} R(C_h, C_{h+1}) / (H - 1)
\]

2.2.2. Constraints

Let \( S = (C_1, \ldots, C_H) \) be a sequence of \( H \) compositions that respect the constraints quoted in the problem statement (see Section 2.1). This can be formally rephrased by:

(1) Given the maximum number of machines allowed in cells during the period \( h, N_{max_h} \), we must have

\[
\forall C_{hi} \in C_h \ (i = 1, 2, \ldots, J_h), \ \text{Card}(C_{hi}) \leq N_{max_h},
\]

where \( \text{Card}(C_{hi}) \) is the size of the cell \( C_{hi} \) (number of \( C_{hi} \) assigned machines).

(2) Given an upper \( (J_{max_h}) \) and a lower \( (J_{min_h}) \) bounds for the number of cells allowed in period \( h, J_h \), we must verify:

\[
J_{min_h} \leq J_h \leq J_{max_h}
\]

(3) Given a set of cohabitant machines couples that must be in the same cell during the period \( h, SC_h \), we have to respect:

\[
\forall (M_k, M_l) \in SC_h, \exists C_{hi} \in C_h : M_k, M_l \in C_{hi} \quad \text{where } k, l \in \{1, \ldots, m\} \text{ and } i \in \{1, \ldots, J_h\}
\]

(4) Given a set of non-cohabitant machines couples that must be in different cells during the period \( h, SN_h \), we have to respect:
\[(M_k, M_l) \in SN_h, \exists C_{hi}, C_{hj} \subset C_h : M_k \in C_{hi} \quad \text{and} \quad M_l \in C_{hj} \quad \text{where} \quad k, l \in \{1, \ldots, m\} \]

and

\[i, j \in \{1, \ldots, J_h\}, i \neq j\]

**Remark 2.** For the convenience of our solving approach, we expand \(E\) to \((E \cup_h (SC_h \cup SN_h))\). By this fact, we add fictitious edges between machines that must be cohabitant or non-cohabitant given that there is no part movements between them in any period.

### 2.2.3. Multi-objective function

Let \(CS\) be the set of composition sequences that respect the previous constraints. The problem is to find a composition sequence \(S^* \in CS\) such that:

\[
Z(S^*) = \text{Min}_{S \in CS}(w_1 TE(S) + w_2 RE(S))
\]

(5)

Where \(w_1\) and \(w_2\) are two non-negative rational numbers defined by the decision maker and whose sum equals one.

That is, we seek a composition sequence that respects all constraints and have the minimum weighted sum of both the entire intracellular traffic and reconfiguration cost. Note that the fractional weighting parameters permit to give to each of the two objectives the relative importance it deserves.

**Remark 3.** As mentioned earlier (in Section 2.1), our model assumes the set of existing machines satisfies the production requirements. This can be met by achieving the following tasks. First, we determine, for each period, the number of machines that can undertake the production of all the planned products. This number can be obtained using the following rule (see [6] for a similar rule):

\[
nr_{ih} = \left\lfloor \frac{\sum_{k=1}^{p} \sum_{l=1}^{q} ot_{khl} \cdot r_{kh}}{at_i} \right\rfloor,
\]

where \(nr_{ih}\) is the number of type \(i\) machines required in period \(h\), \([\ast]\) is the smallest integer greater than or equal to \(*\), \(ot_{khl}\) is the time needed to process the \(l\)th operation of one unit of part \(P_k\) on machine type \(i\) and \(at_i\) is the processing time capacity of machine type \(i\). After acquiring the needed machines, we give to each machine a different index even for those who are identical. Later on, every part volume is split in several ratios so as to develop all the different sequences of machines to be visited by the same units of part flows. This splitting is realized by taking care of not to exceed the processing capacity of each machine (see [21] for a heuristic method to carry out this splitting). We reindex the parts afterwards so as to associate to each different sequence of machines a different part index. The volume of each part so created is set to the volume ration of the associated machine sequence. These procedures give the inputs to be handled by our model.

An example will make this reindexation scheme more clear. Suppose we have a part \(P_1\) to be manufactured according to a demand of seven units a day. The routing is composed of two operations, say Op_1 and Op_2, that require respectively two and one hour. In addition, we suppose the first operation can be achieved only on a machine \(M_1\) whilst the second can be done on another machine, say \(M_2\). Both machines have a capacity of ten hours. If the second machine can cope with all the required units, the first can only deal with five. To overcome this global capacity shortage requires acquiring a new machine that can do the first operation Op_1 for the reminder units. The new acquired machine is represented using a new index, say \(M_3\), even if it was exactly the same as \(M_1\). Then, we consider that part \(P_1\), which is processed according to two routings \(M_1 M_2\) and \(M_1 M_2\), is represented by two parts: \(P_1\) whose routing is \(M_1 M_2\) and production volume is five per day, and \(P_2\) whose routing is \(M_3 M_2\) and production volume is two units per day.

### 2.2.4. The definition of the reconfiguration cost

When we take into account the dynamic aspect of the MCF problem, a transition cost must be considered between every couple of consecutive periods due to the reconfiguration of the manufacturing system at the start of the succeeding period. This so called reconfiguration cost is a decisive parameter in determining which strategy to use. Indeed, the more important the cost is, the more we resort to a static sequence (passive strategy).
This cost measures the order of magnitude of the changes that must be applied to the structure of the system and their effects, concerning in major part machine movements (changes), and idle time (effects), respectively. Therefore, it is obvious that an exact evaluation of this cost must use information about the physical location of machines. However, tackling the problem with such information requires dealing with the layout problem too (integrated approach), yielding a big size model, not easy to solve for large real-size instances.

In order to have a good compromise between neglecting this cost and having a prohibitive exact evaluation, we can resort to an estimation exclusively based on the information of the resource decomposition into cells, independently of their positions (machines and cells) on the shop floor.

Previous researches [5,25,34,35] generally define this cost by considering a static number of cells that allows having a similar cell indexation in all the periods of the planning horizon. Then, for every two consecutive compositions, the moved machines are detected by considering the changes that occurred in the cells having the same indexes in the two periods. However, this method presents a significant shortcoming, due to the redundancy of the cell indexation. Indeed, if the indexation is not appropriate, a great reconfiguration cost can be obtained even if the cell compositions are exactly the same in all the periods of the planning horizon! Furthermore, another drawback comes from the fact that this method is not suitable for a varying number of cells.

In order to make up for these limitations, we propose to evaluate this cost by codifying first the compositions in a suitable manner, then by using a function that measures the dissimilarity between the composition codes. For instance, we can consider for every composition $C_h$, $h = 1, \ldots, H$, the $m \times m$ matrix $B_h$ defined as follows:

$$B_{hij} = \begin{cases} 1 & \text{if } M_i \text{ and } M_j \text{ are in the same cell in the period } h \\ 0 & \text{otherwise} \end{cases}$$

This codification is suitable because it describes faithfully the machine clusters of the composition without adding cell indexation information, the cause of the misleading indexation redundancy. Then, using a distance function that computes the dissimilarity between the codes of every consecutive compositions, $C_h$ and $C_{h+1}$, yields the reconfiguration cost definition defined by formula (6) that uses a normalized Hamming distance:

$$R(C_h, C_{h+1}) = 2 \cdot \left( \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} |B_{hij} - B_{h+1,ij}| \right) / ((m - 1) \cdot m)$$

In order to give to some machines a relatively greater importance in the reconfiguration cost definition, we can use a codification that considers mainly machine couples that are linked by a part traffic. Indeed, it is the moving of those machines that will influence the production system performance. Such a codification will be presented in Section 4.2.1.

2.3. A formulation for the passive cell formation problem PCFP

PCFP occurs when the decision maker imposes a passive strategy. That is, the same composition must be maintained during the overall planning horizon. This attitude is due generally to the fact that the cost of inter- and/or intracellular rearrangement is so important, the decision maker doesn’t want take the risk of maladjustment by using an active strategy.

Let us consider the ACFP formulation (see Section 2.2). PCFP formulation can be directly deduced from ACFP by imposing to the sequence $S$ to be made of a unique composition. That is, $S = (C_1, \ldots, C_H)$ with $C_1 = \cdots = C_H = C$. We shall denote $CS'$ the set of such sequences. This restriction has a direct effect on the bi-objective function $Z(S')$ whose definition becomes: $Z(S') = \min_{S \in CS'} TE(S)$, since the so defined reconfiguration cost is null and thus the weighting procedure becomes superfluous.

Taking inspiration from Batta’s work [2], who has shown that the dynamic passive layout problem amounts to a static (mono-periodic) layout problem, we prove in what follows that a similar result can be found for our passive cell formation problem model, PCFP.

Proposition. The passive cell formation problem, PCFP, amounts to a static (mono-periodic) cell formation problem.
Proof. The objective function of PCFP can be developed as follows:

\[ Z(S^*) = \min_{S \in CS^*} TE(S) \]
\[ = \min_{S \in CS^*} \sum_h T(C_h) \]
\[ = \min_{S \in CS^*} \sum_h \left( d_h \cdot \sum_{e_{kl} \in E(C_h)} W_h(e_{kl}) \right) \]
\[ = \min_{S \in CS^*} \sum_h \left( d_h \cdot \sum_{e_{kl} \in E} W_h(e_{kl}) \right) \cdot W_h(e_{kl}) \]

However, since the same partition is maintained during the \( H \) periods (i.e., \( \forall h : E(C_h) = E(C) \)), we can replace \( C_h \) by \( C \) and invert the two summation signs, the set of inter cell edges \( E(C_h) \) being not depending on \( h \). These operations yield:

\[ = \min_{C \in CP} \sum_{e_{kl} \in E(C)} \sum_h \left( d_h \cdot \sum_{e_{kl} \in E} W_h(e_{kl}) \right) \cdot W_h(e_{kl}) \]

where \( CP \) is the set of compositions that verify the constraints related to each period (see Section 2.2.2).

By this formulation, we define a graph partitioning problem in the network \( N = (M, E, W) \) where the edge weight function \( W \) is defined as follows:

\[ \forall e_{kl} \in E : W(e_{kl}) = \sum_h \left( d_h \cdot \sum_{e_{kl} \in E} W_h(e_{kl}) \right) \cdot W_h(e_{kl}) \quad (7) \]

2.4. The problem of decision between passive and active strategies

If we seek for ACFP the best composition sequence, for the Passive Cell Formation Problem (PCFP) we seek one cell composition that can tolerate, for the best, the system changes. Dealing with the decision problem between passive and active strategies can give a worthwhile help to the decision maker aiming to determine the most adequate one between the two strategies, when the reconfiguration cost is relatively great. However, if it can be stated that a passive strategy will be more adequate for an environment with an important reconfiguration cost, for what threshold this can be guaranteed?

It is important to state here that PCFP is a particular case of ACFP. Indeed, for ACFP, a feasible solution can be composed by a sequence of identical cell compositions. That is, the solution space of PCFP is included in ACFP space and hence, theoretically, the decision problem between the two strategies does not arise, since solving ACFP allows solving the decision problem too.

3. Solving methods for the PCFP

The proposition presented in Section 2.3 has a straightforward consequence: it is possible to solve PCFP by applying any solving method for the static manufacturing cell problem on an instance of a flow graph deduced from the \( H \) flow graphs of the planning horizon. For example, Fig. 1 depicts four flow graphs associated to a manufacturing system whose planning horizon is split on four scenarios with duration of 4, 2, 4, 2 months respectively (see Fig. 1a for the associated flow graphs). When the decision maker imposes a passive strategy, the problem amounts to solve a static manufacturing cell problem instance, whose flow graph is deduced from the previous by considering for each edge, connecting machine \( M_i \) to \( M_j \), the weight defined by Eq. (7). That is, equal to the sum of the normalized duration weighted flows linking these machines in the four graphs (see Fig. 1b).

Since a large panoply of solving methods for the static cell formation problem exists in the literature (see, for example, [3] for a comparison study between genetic algorithms with different coding, neural network, taboo search, hybrid method applications), solving PCFP will not be tackled in our paper.
4. Solving methods for the ACFP

In this section, we present two solving methods: the first takes inspiration from Rosenblat works achieved in the dynamic intracellular facility layout problem [28], the second is a generalization of a GA that proves satisfactory results for the static cell formation problem [4].

4.1. The shortest path method

An instance of the problem can be represented by an $H$-partite graph where vertices of every subset $h$ are the set of all feasible compositions $C_{ih}$ ($C_{ih}$ is the $i$th possibly implemented composition in the $h$th period, $i \in \{1, \ldots, T_h\}$ and $T_h$ is the number of compositions considered in period $h$). The edge set is defined for all vertex couples of the form $(C_{ih}, C_{jh+1})$ to which we associate a value equal to the weighted sum of the transition cost between the two compositions, $R(C_{ih}, C_{jh+1})$ (see Section 2.2.4), and their intercellular flow mean, i.e., $w_2 \cdot R(C_{ih}, C_{jh+1})/(H - 1) + w_1 \cdot (T(C_{ih}) + T(C_{jh+1}))/2$; except for the edges incident to the first and the $H$th subset vertices defined by $w_2 \cdot R(C_{i1}, C_{j2})/(H - 1) + w_1 \cdot T(C_{i1}) + w_1 \cdot T(C_{j2})/2$ and $w_2 \cdot R(C_{iH-1}, C_{jH})/(H - 1) + w_1 \cdot T(C_{iH-1})/2 + w_1 \cdot T(C_{jH})$ respectively (see Fig. 2).

This edge weight definition involves that getting an optimal solution for ACFP amounts to determine a shortest path in the graph so defined. However, for large real life problems, the number of different compositions to be considered in each period is so important\(^2\) that it makes it very expensive to find an optimal solution. Thus, for each period $h$, we consider a moderate number of compositions $T_h$. These compositions can be defined randomly or by a heuristic. Note that Chen’s approach [5] is a shortest path method based on dynamic

\(^2\) Although there is a possibility to reduce this number and still maintain optimality (see [29,13]), it would be very prohibitive since it requires solving optimally $H$ instances of an NP-complete problem.
programming. He considers, for each period, $H$ compositions obtained by solving the static problem instance engendered by each scenario.

In order to get a good solution for the $ACFP$, $T_h$ becomes an important parameter, but due to the NP-completeness of the static problem [4], a good value for $T_h$ cannot be established practically without experimentation.

For large size instances, this limited number of compositions to be considered in each period becomes a serious handicap, leading to a poorly explored solution space. Thus, it will be more interesting to investigate on meta-heuristic methods that have proven abilities in exploring huge and complex spaces, and this is what we shall address in the next sections using a genetic algorithm approach.

4.2. The genetic algorithm method

Following his publication, "Adaptation in Natural and Artificial Systems", Holland [12] is considered to be the founder of the Genetic Algorithms method. These algorithms are based on an analogy to the phenomenon of natural selection in biology. First, a chromosome structure is defined to represent the solutions of the problem. Using this structure, a population is generated, either randomly or by using a given heuristic. Then, members of this initial population are selected based on an evaluation function called fitness that associates a value to each member according to its objective function. The higher a member’s fitness value, the more likely it is to be selected. Therefore, the less fit individuals are replaced by those who perform better. Genetic operators are then applied to the selected members to produce a new generation. This process is repeated until a certain number of iterations, $i_{max}$, is reached.

Using the Genetic Algorithms method to solve the stated problem requires specifying:

1. the structure of the genetic code representing solutions.
2. a random or heuristic method to generate an initial population.
3. an adaptation function to evaluate the fitness of each member of the population.
4. the genetic operators to generate sons by reproduction.
5. some control parameter values (population size, iteration number, probabilities of genetic operators, etc.).

In what follows, we shall address these issues briefly, and for further explanation for the binary GA, its qualities compared with previous proposed GAs, and its application to the static MCF problem, see [4].

4.2.1. Binary code

A sequence of compositions can be represented by a vector of dimension $|E| \times H$, $E$ being the edge set and $H$ the number of periods. The vector is therefore composed of $H$ code portions of length $|E|$, in which each
allele is associated to an edge of the flow graph. Each allele of a code portion $h$, $h = 1, \ldots, H$, of the vector indicates if the corresponding edge is intracellular or not during the $h$th period. An example of such a codification is given in Fig. 3.

In this example, two cell compositions are considered. The first-period composition has three cells while that of the second has only two. Since the number of edges is 9, the length of the genetic chains is $9 \times 2 = 18$. The first code portion is related to the first period and defines three intracellular edges, i.e., zero-alleles’ edges $e_1, e_6, e_8,$ and six intercellular edges, i.e., one-alleles’ edges $e_2, e_3, e_4, e_5, e_7, e_9$; while the second period portion defines five intracellular edges, i.e., $e_3, e_4, e_5, e_6, e_8$, and four intercellular edges, i.e., $e_1, e_2, e_7, e_9$. The cell composition to be implemented in each period is defined by the connected components of the flow graph created by the intracellular edges.

**Remark 4.** The GA binary codification can be used in order to compute the reconfiguration cost. Indeed, as mentioned earlier (see Section 2.2.4), instead of considering all machine couples of the matrix $B$ in the codification of a given composition, we can only consider those that are linked by a positive traffic in at least one period. Doing so enables us to give more importance to these machines in the definition of the reconfiguration cost, since their gathering or separation has more influence in the objective function.

### 4.2.2. Initial population

The initial population is generated randomly, but the probability $P_1$ of generating one-alleles (intercellular edges) is set greater than that of generating zero-alleles (intracellular edges) in order to prevent single cell compositions.

### 4.2.3. Fitness and selection

To allow the GA to maintain unfeasible population solutions and to get advantages of the good information they can hide, the fitness is calculated in such a manner that enables those solutions contributing to the exploration scheme.

First, the minimization problem is transformed into a maximization problem via the formula, $Z'(S) = B - Z(S)$, where $Z(S)$ is the value of the objective function for a given solution $S$, and $B^3$ is an upper bound of $Z(S)$. Thus, maximizing $Z'$ is equivalent to minimizing $Z$. Second, the obtained value is translated

---

3 Since the objective function is normalized and its value is between 0 and 1, we can set $B$ equal to 1.
using the formula, \( Y(S) = Z'(S) + (u - v(S)) \times B \), where \( u \) is the number of constraints, and \( v(S) \) is the number of \( S \) unverified constraints. This translation makes it possible to sort the solutions in \( u + 1 \) consecutive intervals of length \( B \), with the first being for those solutions which do not check all \( u \) constraints, the next being for those which do not check \( u - 1 \) constraints, and so on, until the feasible solutions are put in the \((u + 1)\)th and last interval. The \( Y(S) \) value obtained can then be fine-tuned using a function that allows the feasible domain to be widened (see Fig. 4).

The “Roulette wheel” random procedure was used to select an individual [8]. On this wheel, each individual in the population has a slot proportional to its fitness, computed using the formula, \( Pr(S_k) = F(S_k)/\sum_{i=1}^{s} F(S_i) \), where \( s \) is the GA population size, \( S_k \) is the \( k \)th solution of the GA population \( (k \in \{1,2,\ldots,s\}) \), and \( F(S_k) \) is the fitness (adaptation) of the solution \( S_k \), derived from the fine-tuning procedure. A number between 0 and 1 is selected randomly \( s \times P_2 \) times, where \( P_2 \) is a rational value between 0 and 1. Each time, the individual related to the section containing the generated number is selected. The less fit individuals give up therefore their places (in the population) to those that are more fit, and the \( s \times (1 - P_2) \) remaining individuals are replaced by new randomly-generated ones.

4.2.4. Genetic operators

(a) Crossover

We use a multi-cutting point crossover that derives two offspring from two parents according to the following algorithm:

\textit{Step 1:}
Choose randomly:
1. two individuals (the parents);
2. a number \( nc \) of cutting points \( (1 \leq nc \leq H) \);
3. \( nc \) cutting points \( c_1, \ldots, c_{nc} \), at most one in each period.

\textit{Step 2:}
The first offspring is constructed by the concatenation of the \( c_1 \) first alleles of the first parent with the alleles of rank \( c_1 + 1 \) till \( c_2 \) of the second, then the alleles of rank \( c_2 + 1 \) till \( c_3 \) of the second, and so on. The second offspring is constructed by the concatenation of the remaining parents' code portions. This procedure is repeated with another pair until \( s \times P_3 \) individuals have been replaced, where \( P_3 \) is a value between 0 and 1.

(b) Mutation

The number of individuals that undergo this operator is determined by the value of the product \( s \times P_4 \), where \( P_4 \) is a value between 0 and 1. Two composition code portions associated to two different periods of each chosen individual are exchanged.
4.2.5. Control parameters

The control parameters $s$, $i_{\text{max}}$, $P_1, \ldots, P_4$ referred to throughout the previous sections depend on the problem instance. Consequently, regardless of the attention paid to their settings, for each instance, empirical experimentation is generally necessary to choose the parameter values making the GA to perform at its best.

5. Fuzzy enhancement

As mentioned earlier, PCFP is a particular case of ACFP, and hence solving ACFP optimally would solve the problem of decision between passive and active strategies too. However, the hope to solve ACFP optimally is a too optimistic objective because of the complexity of the problem. Indeed, it can be stated that the ACFP is NP-complete because it has an NP-complete special case (when $H$ equals 1, see [4,7]). Furthermore, the larger the size of the problem is, the bigger the amount of active solutions is, compared with the passive ones. Therefore, the search for a good solution, in case it would be a passive or pseudo-passive (we mean by pseudo-passive a solution that most of the transitions are between identical compositions), becomes like searching a needle in a hay. This thought leads us to attempt controlling the GA implicitly. However, in order to maintain the GA principle of random exploration, this search controlling must be soft enough. Hence, we propose to use a fuzzy logic guidance to push the GA to step up the exploration of the most promising solution areas.

In the general wide domain of optimization, helping the GA with the fuzzy logic is an idea that has been materialized for a long time. The purpose of using fuzzy logic was to speed up the GA convergence and/or to obtain better quality solutions [27]. The general principle of these works was to dynamically adapt the values of the GA operator parameters using fuzzy inference rules as to reach those values able to enhance the GA performance. Indeed, we can find adaptation of population size [19,20], crossover rate [20,33] and mutation rate [20,33,1]. These researches allow building GA systems that outperform significantly conventional GAs. However, these works, albeit interesting, do not consider the causes that “propel” the GA in its search for good solutions but rather their effects. Indeed, the fuzzy models use as input either the convergence speed, the population diversity or its average fitness. We think, it will be more interesting to deal with cause inputs first, admitting that nothing forbids to complement them by effect inputs. In this perspective, discussion about the cause inputs detection and how to exploit them will be addressed in the next section.

5.1. Detecting the influencing information

The fact that a transition is passive or active (that is, maintaining the same cell composition or using a different one for the succeeding period) depends mainly on the data defining the two part demand scenarios related to the two periods associated with this transition. Each scenario engenders part flows between the machines defining the flow graph. When we look at these flow graphs, if the concentration of the flows is “almost” the same in the two graphs, the cell compositions would be probably similar. Thus, if we represent each graph by a geographical map or an image indicating relief, we can predict a passive transition for two “close” images.

In addition, there is another parameter not less influencing, namely the reconfiguration cost. Indeed, if this cost is “very prohibitive”, then resorting to a passive transition is more likely, even if the flow graph similarity is not very great. Therefore, using these two influencing information in the search guidance can be worthwhile for the control of any heuristic solving approach.

5.1.1. Extracting information from the planning horizon scenarios

Each Scenario engenders a flow graph whose information is gathered in the flow matrix, which is an $m \times m$ matrix defining for every couple of machines the amount of part flow linking them. However, since the flow matrix is symmetric, it can be compacted by representing only those components that are above its main diagonal. This yields a compressed matrix whose dimension is $(m/2) \times (m - 1)$ or $m \times ((m - 1)/2)$, depending on the parity of the number of machines $m$. 
Moreover, a suitable information extraction from the flow graph must reflect the flow zone concentration and not the flow amount. Indeed, if we reconstitute a graph by adding to every edge weight the same constant, an optimal solution for the first would be necessarily optimal for the other. Therefore, it would be judicious to normalize the information of the compacted matrix by subtracting from all the weights their minimum, then dividing all the resulting values by their maximum. In what follows, we shall designate this resulting matrix by (reduced) flow matrix. If we associate then to every component value of the flow matrix a suitable grey scale, it can be confounded to a relief map that we call flow image. Fig. 5 depicts the steps that yield the flow images from the flow graphs.

For every consecutive flow images, we define the amount of closeness (or similarity) by calculating the distance function between their associated flow matrices. If the distance separating the flow matrices is small, the flow images are considered “near”, otherwise they are considered “far”, and in order to exploit this information using fuzzy logic, we define the linguistic variable Distance, whose values are in \{Near, Far\}.
5.1.2. Extraction of the reconfiguration cost importance information

The second influencing information, that is the relative importance of the reconfiguration cost, can be extracted from any of the weights associated to the two criteria defined by the decision maker (see 2.2.3). We choose to extract it from the weight $w_2$ associated to the reconfiguration cost criterion. If this cost is close to one it means that it is very important in comparison to the intercellular traffic cost; on the other hand, if it is close to zero, it is considered of a weak importance. This knowledge will be exploited by defining the linguistic variable $Reconfiguration$, whose values are in the set $\{Small, Big\}$ to designate a “small” and a “big” reconfiguration cost respectively.

5.2. The inference rules

In order to exploit the knowledge stored in the fuzzy inputs, we use if-then type inference rules that will give an appropriate output value for each transition, depending on the input values of the influencing information. These rules are defined as follows:

- IF “Distance is Near” AND “Reconfiguration is Small” THEN “Similarity is Medium” OR,
- IF “Distance is Near” AND “Reconfiguration is Big” THEN “Similarity is Great” OR,
- IF “Distance is Far” AND “Reconfiguration is Small” THEN “Similarity is Weak” OR,
- IF “Distance is Far” AND “Reconfiguration is Big” THEN “Similarity is Medium”.

The decision to take is defined by the linguistic variable $Similarity$ whose terms are in $\{Weak, Medium, Great\}$, to designate “weak”, “medium” and “great” $Similarity$ respectively. This variable describes the amount of similarity that might tie the two consecutive compositions of the considered transition.

In order to have the outputs related to the variable $Similarity$, given the input values related to the variables $Distance$ and $Reconfiguration$, we use the Mamdani’s Max–Min method [23].

5.3. The fuzzy enhanced GA

Incorporating the fuzzy reasoning to our solving approach to direct the GA search to the promising areas, consists of redefining the following points of the GA described in Section 4.2.

5.3.1. The generator of individuals

The generator of individuals procedure is called to get a new individual either for generating all individuals of the initial population (see 4.2.2), or in the replacing step that enables replacing non-selected individuals by new ones (see 4.2.3).

Each new individual is generated sequentially starting from a code portion related to a randomly chosen period. The probability of choosing a period depends both on its duration and on the relative amount of part volume produced in it. Therefore, it can be defined by the normalized entire flow given by the formula

$$d_h \cdot \sum_{e_{ik} \in E} W_h(e_{ik}) / \sum_{h} d_h \cdot \sum_{e_{ik} \in E} W_h(e_{ik}).$$

Doing so allows periods with relatively long duration and considerable amount of part production volume to stand a good chance of being chosen first, since over the composition of the selected period, the rest of the compositions will be built. In fact, the composition $C_h$ related to this first chosen period, we call base composition, is generated randomly. Then, applying a perturbation on the $C_h$ genetic code yields the composition $C_{h+1}$. This perturbation is proportional to the inference rule result associated with the incident transition $(h, h + 1)$. Afterward, we get $C_{h+2}$ from $C_{h+1}$, using transition $(h + 1, h + 2)$ inference result, and so on till having the composition $C_H$. The same procedure is repeated in the opposite direction to get the compositions $C_{h-1}, C_{h-2}$, and so on till having $C_1$.

5.3.2. The crossover

Using the crossover to direct the search to passive or active solution area can be achieved by controlling the number of cutting points and their locations. Indeed, a simple crossing over, with only one cutting point, encourages the maintenance of passive or pseudo passive solutions, because a considerable number of parents’ transitions are retained. On the other hand, increasing the number of cutting points encourages the active
solutions. The reason being that the probability of having offspring with quite different compositions becomes higher.

Thus, we can enhance the previous GA crossover algorithm (see 4.2.4) by modifying the steps 1.2 and 1.3 defining the number of cutting points and their associated locations in the genetic code according to the following:

Let \( r_1, r_2, \ldots, r_{H-1} \) be the results of inference rules related to the transitions \((1,2), (2,3), \ldots, (H-1,H)\) respectively (see Section 5.2).

For each code portion associated to a period \( h, h = 1, \ldots, H \), is assigned a cutting point with a probability \( p_h \) defined as follows:

\[
p_h = \begin{cases} 
1 - r_1 & \text{if } h = 1 \\
1 - r_{H-1} & \text{if } h = H \\
1 - \left(\frac{r_h + r_h}{2}\right) & \text{otherwise}
\end{cases}
\]

Assuming that the alleles are indexed from 1 to \( H \times |E| \), the location of each probably assigned cutting point is, for each period \( h, h = 1, \ldots, H \), a uniformly chosen element of the set of indexes \( \{ (h - 1) \times |E| + 1, \ldots, h \times |E| \} \).

Doing so enables avoiding cutting points in composition code portions that have a great similarity with the neighbors, and encourages putting them in those portions that have a weak similarity with them.

5.3.3. The mutation

The mutation can benefit from the inference rule results \( r_i, i = 1, \ldots, H - 1 \), by swapping the consecutive quite similar compositions. This can be achieved by covering the genetic code of the individual, chosen to undergo a mutation, transition by transition, and swapping progressively the consecutive codes, related to the two compositions incident to the covered transition, with a probability equal to the corresponding inference rule result.

6. An illustrative example

In this section, an example is presented in order to illustrate the benefits of the fuzzy enhancement. A comparative evaluation with earlier methods was impossible for several reasons: These methods use either different objective functions or require additional non-practically justified input data (like predetermined number of cells), or do not take into account some important practical constraints like cohabitation and non-cohabita-

Table 1
The planning horizon scenarios

<table>
<thead>
<tr>
<th>Parts</th>
<th>Machines</th>
<th>Volumes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M1</td>
<td>M2</td>
</tr>
<tr>
<td>P_1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>P_2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>P_3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>P_4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_6</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_8</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_9</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_10</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_11</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_12</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_13</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_14</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_15</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P_16</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Period durations: 2 4 2 4
tion. However, in order to show the fuzzy enhanced GA efficiency, we compare it with the shortest path method and the GA without the fuzzy enhancement (see Section 4). The three applications were coded using a gcc C++ compiler and were processed on a Cyrix MII 300 microcomputer with a clock speed of 233 MHz and 32 Mo of RAM. In the following paragraphs, the three methods are referred to as SP for the shortest path method, GA for the Genetic Algorithm without fuzzy enhancement and FGA for the Fuzzy Genetic Algorithm.

6.1. Example inputs

A 384 size example is presented below. The size is assumed to be equal to the product $p \times m \times H$ (number of parts $\times$ number of machines $\times$ number of periods).

Table 1 shows the four scenarios of the planning horizon. The scenario of a period $h$ is the subset of parts that have a positive part production volume in that period. For example, the first scenario consists of manufacturing the sub-set of parts $\{P1,P2,P3,P4\}$ according to the corresponding operation sequences, with the volumes 1,3,5,2 respectively.

The maximum number of machines in each cell, $N_{max_h}$, is set equal to 3 for all periods and for technical reasons, machines M1 and M3 must be placed in different cells.

In addition, we take into consideration two opposite situations: The first is when the decision maker feels that the reconfiguration cost has a weak importance in comparison to the traffic cost, thus he gives to them the weights 0.3 and 0.7 respectively. In the second, we consider the inverse case (that is, 0.7 for the reconfiguration cost weight and 0.3 for that of the traffic cost).

6.2. GA knowledge embodying

As described in Section 5.1, two input parameters influence on the sought best solution: the flow matrices and the importance of the reconfiguration cost. The information of the first is extracted by the linguistic variable Distance whose term membership functions are given in Fig. 6a.

Table 2 shows the Distance term membership factors for the illustrative example.

For the second influencing input, we use the linguistic variable Reconfiguration whose term membership functions are depicted on Fig. 6b. Table 3 shows the associated membership factors for the illustrative example.

These two linguistic variables are bound by the rules (see Section 5.2) that yield, by applying Mamdani’s Max–Min method, then the mean of maximums procedure [35,36], the crisp values $r_1, r_2, r_3$ given in Table 4.

![Fig. 6. Input and output linguistic variable membership functions.](image)
These values are used by the FGA to predict the form of the sought solution, as described in Section 5.3, considering that each \( r_i \), \( i \in \{1,2,3\} \), describes the expected similarity between the two consecutive compositions associated to the periods \( i \) and \( i + 1 \).

6.3. Result discussion

The three methods were run several times in order to determine those parameter values that would render the results stable. A result is assumed to be stable if the best related solution is always obtained at least twice if the program is run five times. The best solution for each method was then reported with its own computational running time.

Table 2

<table>
<thead>
<tr>
<th>Distance term membership factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
</tr>
<tr>
<td>(1,2)</td>
</tr>
<tr>
<td>(2,3)</td>
</tr>
<tr>
<td>(3,4)</td>
</tr>
</tbody>
</table>

Table 3

<table>
<thead>
<tr>
<th>Reconfiguration term membership factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconfiguration cost weight</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 4

<table>
<thead>
<tr>
<th>Similarity defuzzification results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconfiguration cost weight</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0.7</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th>Performance for a weak importance reconfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>Traffic</td>
</tr>
<tr>
<td>Reconfiguration</td>
</tr>
<tr>
<td>Aggregation</td>
</tr>
<tr>
<td>CPU time (s)</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>Best cell compositions for a weak importance reconfiguration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>SP</td>
</tr>
<tr>
<td>GA</td>
</tr>
<tr>
<td>FGA</td>
</tr>
</tbody>
</table>
For the first situation ($w_1 = 0.7$ and $w_2 = 0.3$), the results obtained by applying the three methods to the above example are reported in Table 5.

The two GA based methods reached a same solution and it was better than that of the SP method (See Table 6 for the details concerning the solutions found by the three methods). However, the FGA could reach stability with less computational cost.

Indeed, Table 7 shows that the FGA requires a smaller population size and fewer generations to reach the good solution. Such economy stems from the Fuzzy enhancement ability to direct earlier the search to the promising areas.

For the second situation ($w_1 = 0.3$ and $w_2 = 0.7$), the results obtained are reported in Table 8. The GA reached a better solution in comparison with the SP method.

The cell compositions of the solutions found by the three methods are depicted in Table 9.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Methods</th>
<th>SP</th>
<th>GA</th>
<th>FGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size ($s$)</td>
<td>–</td>
<td>200</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>Number of generations ($i_{max}$)</td>
<td>–</td>
<td>2000</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>One-allele generation probability ($P_1$)</td>
<td>–</td>
<td>0.80</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td>Selection rate ($P_2$)</td>
<td>–</td>
<td>0.70</td>
<td>0.70</td>
<td></td>
</tr>
<tr>
<td>Crossover rate ($P_3$)</td>
<td>–</td>
<td>0.90</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>Mutation rate ($P_4$)</td>
<td>–</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>Number of solutions per period</td>
<td>40</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

For the first situation ($w_1 = 0.7$ and $w_2 = 0.3$), the results obtained by applying the three methods to the above example are reported in Table 5.

The two GA based methods reached a same solution and it was better than that of the SP method (See Table 6 for the details concerning the solutions found by the three methods). However, the FGA could reach stability with less computational cost.

Indeed, Table 7 shows that the FGA requires a smaller population size and fewer generations to reach the good solution. Such economy stems from the Fuzzy enhancement ability to direct earlier the search to the promising areas.

For the second situation ($w_1 = 0.3$ and $w_2 = 0.7$), the results obtained are reported in Table 8. The GA reached a better solution in comparison with the SP method.

The cell compositions of the solutions found by the three methods are depicted in Table 9.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Cell compositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>{1}</td>
</tr>
<tr>
<td>GA</td>
<td>{1,2,4}</td>
</tr>
<tr>
<td>FGA</td>
<td>{1,5,6}</td>
</tr>
</tbody>
</table>

Table 8
Performance for a great importance reconfiguration

<table>
<thead>
<tr>
<th>Methods</th>
<th>SP</th>
<th>GA</th>
<th>FGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic</td>
<td>0.6627</td>
<td>0.5361</td>
<td>0.4940</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Aggregation</td>
<td>0.1988</td>
<td>0.1608</td>
<td>0.1482</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>11.47</td>
<td>81.27</td>
<td>2.01</td>
</tr>
</tbody>
</table>

Table 9
Best cell compositions for a great importance reconfiguration

<table>
<thead>
<tr>
<th>Methods</th>
<th>Period 1</th>
<th>Period 2</th>
<th>Period 3</th>
<th>Period 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>{1}</td>
<td>{2}</td>
<td>{3,4,5}</td>
<td>{6}</td>
</tr>
<tr>
<td>GA</td>
<td>{1,2,4}</td>
<td>{3,5,6}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FGA</td>
<td>{1,5,6}</td>
<td>{2,3,4}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10
Parameter values for a great importance reconfiguration

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Methods</th>
<th>SP</th>
<th>GA</th>
<th>FGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size ($s$)</td>
<td>–</td>
<td>200</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Number of generations ($i_{max}$)</td>
<td>–</td>
<td>2500</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>One-allele generation probability ($P_1$)</td>
<td>–</td>
<td>0.80</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td>Selection rate ($P_2$)</td>
<td>–</td>
<td>0.70</td>
<td>0.70</td>
<td></td>
</tr>
<tr>
<td>Crossover rate ($P_3$)</td>
<td>–</td>
<td>0.90</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>Mutation rate ($P_4$)</td>
<td>–</td>
<td>0.02</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>Number of solutions per period</td>
<td>40</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>
However, the GA was outperformed by the FGA. Indeed, the GA was unable to overcome falling into a local optimum in spite of having elevated the mutation rate and decreased the selection rate in order to enhance the diversification of the population genetic code (see Table 10 for the parameter values used).

The same conclusions can be drawn from examples of medium size (see Appendix A). Indeed, if we compare the general performance of the SP method with the GA, we can deduce that this latter performs better and when the problem size increases, the SP shortcoming become more critical. In addition, if we compare the GA performance with that of the FGA, we can see that the fuzzy enhancement gives a significant help to the GA. Actually, when the system allows active reconfiguration (that is, the scenarios are quite different and the reconfiguration cost is not very important), the fuzzy embodying knowledge allows getting stability faster.

On the other hand, when the system doesn’t tolerate a big reconfiguration cost, the fuzzy enhancement enables the improvement of the GA abilities to search for good solutions in the dynamic system complex space, by intensifying the prospecting of promising areas.

In short, we can say that overall, the FGA performance proves the fuzzy enhancement was worthwhile; either the reconfiguration cost is relatively more important or not. The examples used in this comparison are relatively average in size, and the fuzzy enhancement was able to enhance the solution given by the GA alone. Based on its performance here, one can predict that the fuzzy enhanced GA will continue to produce high quality solutions for large instances.

7. Conclusion

Cell formation is one of the main problems to be solved in the design of a cellular manufacturing system. In this paper, an approach is proposed for solving the problem under its dynamic and deterministic aspects. We first highlighted the necessity of taking the dynamic aspect into account. Then, we proposed a graph partitioning formulation of the problem under the highlighted aspect that considers real life circumstances. Two strategies are considered corresponding to two possible orders of the decision maker. For the first, the passive strategy, we show that it amounts to a cell formation problem in a static system. For the second, the active strategy (ACFP), we propose a Shortest Path heuristic (SP) and a Genetic Algorithm (GA) based method.

Table A.1
Second example planning horizon scenarios

<table>
<thead>
<tr>
<th>Parts</th>
<th>Machines</th>
<th>Volumes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M1</td>
<td>M2</td>
</tr>
<tr>
<td>P1</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>P3</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>1</td>
<td>2,4</td>
</tr>
<tr>
<td>P5</td>
<td>2</td>
<td>.</td>
</tr>
<tr>
<td>P6</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P7</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>P8</td>
<td>7</td>
<td>.</td>
</tr>
<tr>
<td>P9</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P10</td>
<td>7</td>
<td>.</td>
</tr>
<tr>
<td>P11</td>
<td>3</td>
<td>.</td>
</tr>
<tr>
<td>P12</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>P13</td>
<td>3</td>
<td>.</td>
</tr>
<tr>
<td>P14</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P15</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P16</td>
<td>.</td>
<td>2</td>
</tr>
<tr>
<td>P17</td>
<td>1,4</td>
<td>.</td>
</tr>
<tr>
<td>P18</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>P19</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>P20</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Period durations 1 2 4 3 3
to solve the problem. When the decision maker wants to choose the most adequate strategy for its environment, a problem of decision between the two strategies occurs. We show that this decision problem can be solved by solving ACFP. However, the NP-completeness complexity of ACFP justifies the need to control the presented solving methods when large real life instances are considered. Thus, a new Fuzzy GA enhancement is proposed for embodying the information of the system inputs in the GA search engine, thrusting it to focus its prospecting in promising areas. The results obtained show that the GA outperforms the SP method and furthermore, that the Fuzzy enhanced GA outperforms the GA alone.

Table A.2
Second example performance for a weak importance reconfiguration (w2 = 0.3)

<table>
<thead>
<tr>
<th>Methods</th>
<th>SP</th>
<th>GA</th>
<th>FGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic</td>
<td>0.6563</td>
<td>0.5641</td>
<td>0.4786</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>0</td>
<td>0.0545</td>
<td>0</td>
</tr>
<tr>
<td>Aggregation</td>
<td>0.4594</td>
<td>0.4113</td>
<td>0.3350</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>28.47</td>
<td>152.82</td>
<td>53.17</td>
</tr>
</tbody>
</table>

Table A.3
Second example best cell compositions for a weak importance reconfiguration (w2 = 0.3)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Period 1</th>
<th>Period 2</th>
<th>Period 3</th>
<th>Period 4</th>
<th>Period 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>{1,3,4,7,13}</td>
<td>{1,3,4,7,13}</td>
<td>{1,3,4,7,13}</td>
<td>{1,3,4,7,13}</td>
<td>{1,3,4,7,13}</td>
</tr>
<tr>
<td></td>
<td>{2,5,9,11,14}</td>
<td>{2,5,9,11,14}</td>
<td>{6,8,10,12,15}</td>
<td>{6,8,10,12,15}</td>
<td>{6,8,10,12,15}</td>
</tr>
<tr>
<td>GA</td>
<td>{1,5,7,14,15}</td>
<td>{1,5,7,14,15}</td>
<td>{1,4,5,7,11}</td>
<td>{1,4,5,7,11}</td>
<td>{1,4,5,7,11}</td>
</tr>
<tr>
<td></td>
<td>{2,3,8,10,13}</td>
<td>{2,3,8,10,13}</td>
<td>{6,9,12,14,15}</td>
<td>{6,9,12,14,15}</td>
<td>{6,9,12,14,15}</td>
</tr>
<tr>
<td>FGA</td>
<td>{4,6,9,12}</td>
<td>{4,6,9,12}</td>
<td>{1,7,9,10,13}</td>
<td>{1,7,9,10,13}</td>
<td>{1,7,9,10,13}</td>
</tr>
<tr>
<td></td>
<td>{2,3,8,12,15}</td>
<td>{2,3,8,12,15}</td>
<td>{4,5,6,11,14}</td>
<td>{4,5,6,11,14}</td>
<td>{4,5,6,11,14}</td>
</tr>
<tr>
<td></td>
<td>{4,5,6,11,14}</td>
<td>{4,5,6,11,14}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.4
Second example performance for a great importance reconfiguration (w2 = 0.7)

<table>
<thead>
<tr>
<th>Methods</th>
<th>SP</th>
<th>GA</th>
<th>FGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic</td>
<td>0.7549</td>
<td>0.5641</td>
<td>0.5066</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Aggregation</td>
<td>0.2265</td>
<td>0.1692</td>
<td>0.1520</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>28.82</td>
<td>170.46</td>
<td>78.06</td>
</tr>
</tbody>
</table>

Table A.5
Second example best cell compositions for a great importance reconfiguration (w2 = 0.7)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Period 1</th>
<th>Period 2</th>
<th>Period 3</th>
<th>Period 4</th>
<th>Period 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>{1,4,8,10,14}</td>
<td>{2,5,11}</td>
<td>{1,4,8,10,14}</td>
<td>{2,5,11}</td>
<td>{1,4,8,10,14}</td>
</tr>
<tr>
<td></td>
<td>{3,7,12,13,15}</td>
<td>{6,9}</td>
<td>{3,7,12,13,15}</td>
<td>{6,9}</td>
<td>{3,7,12,13,15}</td>
</tr>
<tr>
<td>GA</td>
<td>{1,5,11,13,14}</td>
<td>{2,7,8}</td>
<td>{1,5,11,13,14}</td>
<td>{2,7,8}</td>
<td>{1,5,11,13,14}</td>
</tr>
<tr>
<td></td>
<td>{10,12}</td>
<td>{3,4,6,9,15}</td>
<td>{10,12}</td>
<td>{3,4,6,9,15}</td>
<td>{10,12}</td>
</tr>
<tr>
<td>FGA</td>
<td>{1,7,9,10,13}</td>
<td>{2,3,5}</td>
<td>{1,7,9,10,13}</td>
<td>{2,3,5}</td>
<td>{1,7,9,10,13}</td>
</tr>
<tr>
<td></td>
<td>{11,14}</td>
<td>{4,6,8,12,15}</td>
<td>{11,14}</td>
<td>{4,6,8,12,15}</td>
<td>{11,14}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parts</td>
<td>M1</td>
<td>M2</td>
<td>M3</td>
<td>M4</td>
<td>M5</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>P1</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P2</td>
<td>6</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P4</td>
<td>1</td>
<td>5</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P5</td>
<td>1</td>
<td>5</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P6</td>
<td>1</td>
<td>5</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P7</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P8</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P9</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P10</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P11</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P12</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P13</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P14</td>
<td>4</td>
<td>2,5</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>P15</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P16</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P17</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P18</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P19</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P20</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P21</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P22</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P23</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>P24</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

Period durations 2 4 3 2 3 2
Table A.7
Third example performance for a weak importance reconfiguration (w2 = 0.3)

<table>
<thead>
<tr>
<th>Methods</th>
<th>SP</th>
<th>GA</th>
<th>FGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic</td>
<td>0.6878</td>
<td>0.5854</td>
<td>0.5187</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>0.0719</td>
<td>0.0854</td>
<td>0.1236</td>
</tr>
<tr>
<td>Aggregation</td>
<td>0.5030</td>
<td>0.4354</td>
<td>0.4002</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>116.47</td>
<td>1123.76</td>
<td>817.64</td>
</tr>
</tbody>
</table>

Table A.8
Third example best cell compositions for a weak importance reconfiguration (w2 = 0.3)

<table>
<thead>
<tr>
<th>Period</th>
<th>Cell compositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1,5,12,14,16} [2,6,8,13,18]</td>
</tr>
<tr>
<td></td>
<td>{3,4,11,20} [7,9,10,15,19]</td>
</tr>
<tr>
<td></td>
<td>{5,11,13,16} [6,14]</td>
</tr>
<tr>
<td>2</td>
<td>{1,5,12,14,16} [2,6,8,13,18]</td>
</tr>
<tr>
<td></td>
<td>{1,10,15,17,19} [2,4,8,20]</td>
</tr>
<tr>
<td></td>
<td>{3,4,11,20} [7,9,10,15,19]</td>
</tr>
<tr>
<td>3</td>
<td>{1,5,12,14,16} [2,6,8,13,18]</td>
</tr>
<tr>
<td></td>
<td>{1,9,12,16} [2,4,8]</td>
</tr>
<tr>
<td></td>
<td>{3,5,6,19}</td>
</tr>
<tr>
<td>4</td>
<td>{1,5,12,14,16} [2,6,8,13,18]</td>
</tr>
<tr>
<td></td>
<td>{1,9,12,16} [2,4,8]</td>
</tr>
<tr>
<td></td>
<td>{3,5,6,19}</td>
</tr>
<tr>
<td>5</td>
<td>{1,5,12,14,16} [2,6,8,13,18]</td>
</tr>
<tr>
<td></td>
<td>{7,10,15} [11,18,20]</td>
</tr>
<tr>
<td></td>
<td>{14}</td>
</tr>
<tr>
<td>6</td>
<td>{1,2,7,9,10} [3,5,6,12,14]</td>
</tr>
<tr>
<td></td>
<td>{4,11}</td>
</tr>
<tr>
<td></td>
<td>{8,13,15,18,20}</td>
</tr>
</tbody>
</table>

Table A.9
Third example performance for a great importance reconfiguration (w2 = 0.7)

<table>
<thead>
<tr>
<th>Methods</th>
<th>SP</th>
<th>GA</th>
<th>FGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic</td>
<td>0.7154</td>
<td>0.7463</td>
<td>0.6325</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>0.0472</td>
<td>0.0067</td>
<td>0</td>
</tr>
<tr>
<td>Aggregation</td>
<td>0.2477</td>
<td>0.2286</td>
<td>0.1898</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>60.76</td>
<td>595.27</td>
<td>219.38</td>
</tr>
</tbody>
</table>

Table A.10
Third example best cell compositions for a great importance reconfiguration (w2 = 0.7)

<table>
<thead>
<tr>
<th>Period</th>
<th>Cell compositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1,3,5,9,12} [2,6,13,14,16]</td>
</tr>
<tr>
<td></td>
<td>{4,7,15,17,19} [8,10,11,18,20]</td>
</tr>
<tr>
<td>2</td>
<td>{1,3,5,9,12} [2,6,13,14,16]</td>
</tr>
<tr>
<td></td>
<td>{4,7,15,17,19} [8,10,11,18,20]</td>
</tr>
<tr>
<td>3</td>
<td>{1,3,5,9,12} [2,6,13,14,16]</td>
</tr>
<tr>
<td></td>
<td>{4,7,15,17,19} [8,10,11,18,20]</td>
</tr>
<tr>
<td>4</td>
<td>{1,3,5,9,12} [2,6,13,14,16]</td>
</tr>
<tr>
<td></td>
<td>{4,7,15,17,19} [8,10,11,18,20]</td>
</tr>
<tr>
<td>5</td>
<td>{1,3,5,9,12} [2,6,13,14,16]</td>
</tr>
<tr>
<td></td>
<td>{4,7,15,17,19} [8,10,11,18,20]</td>
</tr>
<tr>
<td>6</td>
<td>{1,3,5,9,12} [2,6,13,14,16]</td>
</tr>
<tr>
<td></td>
<td>{4,7,15,17,19} [8,10,11,18,20]</td>
</tr>
</tbody>
</table>
We intend to continue our research in the following directions. First, we hope to develop new methods for embodying data information in the GA search process in order to take into consideration further criteria such that minimizing part external operation time, inter- and intracellular load unbalance, ... Second, we would like to investigate the possibility of extending the proposition of Section 2.4 to other criteria by defining suitable criterion modeling. Third, we are interested in the idea of tackling the dynamic MCF problem from a stochastic point of view by considering, deterministic and non-deterministic inputs. Finally, we want to tackle the problem by integrating the routing flexibility aspect.

Appendix A. Additional computational data and the associated results

A.1. Second example

Size: 1500 (20 parts, 15 machines and 5 periods)
Constraints: Maximum number of machines in each cell = 5, non-cohabitation of M8 with M11 (Tables A.1–A.5).

A.2. Third example

Size: 2880 (24 parts, 20 machines and 6 periods)
Constraints: Maximum number of machines in each cell = 5, non-cohabitation of M4 with M16 (Tables A.6–A.10).

References
