CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Hybrid of Metaheuristic Methods for Solving the Cell Formation Problem

Luong Thuan Thanh
Jacques A. Ferland
Nguyen Dinh Thuc
Van Hien Nguyen

March 2012

CIRRELT-2012-14

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

UNIVERSITÉ LAVAL · UQÀM · HEC MONTRÉAL · ÉCOLE POLYTECHNIQUE MONTRÉAL · Université de Montréal

Hybrid of Metaheuristic Methods for Solving the Cell Formation Problem

Luong Thuan Thanh[1], Jacques A. Ferland[1,2,*], Nguyen Dinh Thuc[3], Van Hien Nguyen[1,4]

[1] Institute for Computational Science and Technology, Information Technology Park (VNU-IT.park), Quarter 6, Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and Department of Computer Science and Operations Research, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

[3] Faculty of Information Technology - HCMUS – VNU, 227 Nguyen Van Cu, Dist. 5, Hochiminh city, Vietnam University of Science, Vietnam National University at Ho Chi Minh City, Vietnam

[4] Department of Mathematics, University of Namur (FUNDP), rue de Bruxelles 61, B-5000 Namur, Belgium

Abstract. In this paper we solve the cell formation problem with three different hybrids of metaheuristic methods. The first method is an implementation of the simulated annealing method (SA) using different neighborhoods of the current solution. The solution generated at each iteration is obtained by using a diversification of the current solution combined with an intensification to improve this solution. Different diversification and intensification strategies are combined to generate different neighborhoods. The second method is an adaptive simulated annealing method (ASA) where the neighborhood used at each iteration is selected randomly among the four neighborhood identified above. The procedure is adaptive in the sense that the probabilities are updated during the process according to the success of using the different neighborhoods. A third set of methods is derived by modifying the hybrid method (HM) combining a local search algorithm (LSA) with a genetic algorithm (GA) introduced in (Elbenani et al., 2011). All the variants perform well to deal with the 35 benchmark cell formation problems commonly used in the literature, but the dominating one is a modified HM followed by a SA method afterward. It allows improving the best-known solution of 2 of the 35 benchmarked problems used in the literature, reaching the best-known solution of 32 others, and missing the best-known solution of the other one by a factor of 0.016%.

Keywords. Metaheuristics, evolutionary computation, fractional programming, simulated annealing, genetic algorithm, combinatorial optimization.

Acknowledgements. This research was supported by the Institute for Computational Science and Technology at Ho Chi Minh City (ICST HCMC), Vietnam.

---

* Corresponding author: JacquesA.Ferland@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1 INTRODUCTION

The *Group Technology* is an approach often used in manufacturing and engineering management taking advantage of similarities in production design and processes. In this context, the *Cellular Manufacturing* refers to maximize the overall efficiency of a production system by grouping together *machines* providing service to similar *parts* into a subsystem (denoted *cell*). The corresponding problem is formulated as a *(Machine-Part) Cell Formation Problem*. As a consequence, the interactions of the machines and the parts within a cell are maximized, and those between machines and parts of other cells are reduced as much as possible.

The cell formation problem is a NP hard optimization problem (Dimopoulos and Zalzala, 2000). For this reason, several heuristic methods have been developed over the last forty years to generate good solutions in reasonable computational time. To learn more about the different methods, we refer the reader to the survey papers proposed in (Goncalves and Resende, 2004), and in (Papaioannou and Wilson, 2010) where the authors survey the different techniques classified as follows:

- Cluster analysis: techniques for recognizing structure in a data set
- Graph partitioning approaches where a graph or a network representation is used to formulate the cell formation problem
- Mathematical programming methods: the cell formation problem is formulated like a non linear or linear integer programming problem
- Heuristic, metaheuristic and hybrid metaheuristic: The most popular methods are: simulated annealing, tabu search, genetic algorithms, colony optimization, particle swarm optimization, neural networks and fuzzy theory.

In (Ghosh *et al.,* 2010), the authors introduce a survey of various genetic algorithms used to solve the cell formation problem. The success of genetic algorithms in solving this problem induced researchers to consider different variants and hybrids in order to generate very robust techniques.

In this paper, we introduce solution methods hybridizing different approaches. The first method is an implementation of the *simulated annealing* (*SA*) (Kirkpatrick *et al.*, 1983, Cerny,1994) using different neighborhoods of the current solution. The solution selected in the neighborhood at each iteration is obtained by applying a diversification strategy to the current solution and by using an intensification strategy to improve it. The first intensification strategy is an *approximation method* used in a local search method (*LSA*) introduced in (Elbenani *et al.*, 2011), and the second one relies on the Dinkelbach method (Dinkelbach 1967). They are combined with two different diversification strategies to generate four different neighborhoods leading to four variants of the simulated annealing method. The second method is specified by referring to the Adaptive Large Neighborhood Search (*ALNS*) introduced in (Pisinger and Ropke 2007). It is an *adaptive simulated*

annealing method (*ASA*) where the neighborhood used at each iteration is selected randomly among the four neighborhood identified above. The procedure is adaptive in the sense that the probabilities are updated during the process according to the success of using the different neighborhoods. A third set of methods are derived by modifying the hybrid method (*HM*) combining a local search algorithm (*LSA*) with a genetic algorithm (*GA*) introduced in (Elbenani *et al.*, 2011). The first modification is to replace the approximation method used in *LSA* by an *Exact procedure* based on the Dinkelbach method to solve fractional programming problem. The second modification is to apply a *SA* method afterward on the solution generated with the *HM*.

Numerical results are obtained comparing numerically the efficiency of the variants with respect to the best-known solutions of 35 benchmark problems commonly used by authors to evaluate their methods. All the variants perform well to deal with the cell formation problem, but the dominating one is the modified *HM* using the *Exact procedure* followed by a *SA* method afterward. It allows improving the best-known solution of 2 of the 35 benchmarked problems, reaching the best-known solution of 32 others, and missing the best-known solution of the other one by a factor of 0.016%.

The cell formation problem is summarized in Section 2. Section 3 is devoted to the simulated annealing procedure. We introduce the different diversification and intensification strategies to develop the different neighborhoods. The *ASA* and the modified *HM* methods are summarized in Sections 4 and 5, respectively. The Section 6 includes the numerical results.

2 PROBLEM FORMULATION

To formulate the cell formation problem, consider the following two sets

$I =$ set of m machines: $i = 1, \ldots, m$

$J =$ set of n parts: $j = 1, \ldots, n.$

The production incidence matrix $A = \left[a_{ij} \right]$ indicates the interactions between the machines and the parts:

$$a_{ij} = \begin{cases} 1 \text{ if machine } i \text{ process part } j \\ 0 \text{ otherwise.} \end{cases}$$

Furthermore, a part j may be processed by several machines. A production cell k $\left(k = 1, \ldots, K\right)$ includes a subset (group) of machines $C_k \subset I$ and a subset (family) of parts $F_k \subset J$. The problem is to determine a solution including K production cells $\left(C, F\right) = \left\{ \left(C_1, F_1\right), \ldots, \left(C_K, F_K\right) \right\}$ as *autonomous* as possible. Note that the K production cells induce partitions of the machines set and of the parts set:

$$C_1 \bigcup \ldots \bigcup C_K = I \quad \text{and} \quad F_1 \bigcup \ldots \bigcup F_K = J$$

and for all pairs of different cell indices k_1 and $k_2 \in \left\{1, \ldots, K\right\}$

$$C_{k_1} \bigcap C_{k_2} = \phi \quad \text{and} \quad F_{k_1} \bigcap F_{k_2} = \phi.$$

To illustrate the production cells concept, consider a machine-part incidence matrix in Table

1. Table 2 illustrates a partition into 3 different cells illustrated in the gray zones. The solution includes the 3 machine groups {(1,4,6), (3,5), (2)} and the 3 part families {(2,4,6,8), (1,7), (3,5)}.

Parts		1	2	3	4	5	6	7	8
Machines	1	0	1	0	1	1	1	0	1
	2	1	0	1	0	1	0	0	0
	3	1	0	1	0	0	0	1	0
	4	0	1	0	1	0	1	0	1
	5	1	0	0	0	0	0	1	1
	6	1	1	0	0	0	1	1	1

Table 1. Incidence matrix

Parts		2	4	6	8	1	7	3	5
Machines	1	1	1	1	1	0	0	0	1
	4	1	1	1	1	0	0	0	0
	6	1	0	1	1	1	1	0	0
	3	0	0	0	0	1	1	1	0
	5	0	0	0	1	1	1	0	0
	2	0	0	0	0	1	0	1	1

Table 2. Matrix solution

The *exceptional elements* (1,5), (6,1), (6,7), (3,3), (5,8) and (2,1) correspond to entries having a value 1 that lay outside of the gray diagonal blocks.

Sarker and Khan (2001) carry out a comparative study of different *autonomy* measures for the solution of a cell formation problem. In this paper we consider the grouping efficacy *Eff* (Kumar and Chandrasekharan 1990) that is mostly used:

$$Eff = \frac{a - a_1^{Out}}{a + a_0^{In}} = \frac{a_1^{In}}{a + a_0^{In}} \qquad (1)$$

where $a = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}$ denotes the total number of entries equal to 1 in the matrix *A*, a_1^{Out} denotes the number of *exceptional* elements, and a_1^{In} and a_0^{In} are the numbers of one and of zero entries in the gray diagonal blocks, respectively. The objective function of the problem is maximizing *Eff* .

To formulate the mathematical formulation of the problem, we introduce the following binary variables:

for each pair $i = 1,\ldots,m;\ k = 1,\ldots,K$

$$x_{ik} = \begin{cases} 1 & \text{if machine } i \text{ belongs to cell } k \\ 0 & \text{otherwise} \end{cases}$$

for each pair $j = 1,\ldots,n;\ k = 1,\ldots,K$

$$y_{jk} = \begin{cases} 1 & \text{if part } j \text{ belongs to cell } k \\ 0 & \text{otherwise.} \end{cases}$$

To evaluate the objective function *Eff*, it is easy to verify that

$$a_1^{out} = a - \sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} x_{ik} y_{jk}$$

$$a_0^{In} = \sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n} \left(1 - a_{ij}\right) x_{ik} y_{jk}.$$

In this paper we are considering the following model $M(x,y)$ of the cell partitioning problem:

$$M(x,y) \qquad \text{Max } Eff = \frac{\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} x_{ik} y_{jk}}{a + \sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n} \left(1 - a_{ij}\right) x_{ik} y_{jk}}$$

$$\text{Subject to } \sum_{k=1}^{K} x_{ik} = 1 \qquad i = 1,\ldots,m \tag{2}$$

$$\sum_{k=1}^{K} y_{jk} = 1 \qquad j = 1,\ldots,n \tag{3}$$

$$\sum_{i=1}^{m} x_{ik} \geq 1 \qquad k = 1,\ldots,K \tag{4}$$

$$\sum_{j=1}^{n} y_{jk} \geq 1 \qquad k = 1,\ldots,K \tag{5}$$

$$x_{ik} = 0 \text{ or } 1 \qquad i = 1,\ldots,m; k = 1,\ldots,K \tag{6}$$

$$y_{jk} = 0 \text{ or } 1 \qquad j = 1,\ldots,n; k = 1,\ldots,K \tag{7}$$

The constraints (2) and (3) ensure that each machine and each part is assigned to exactly one cell, respectively. The constraints (4) and (5) ensure that each cell includes at least one machine and one part (no empty cell allowed). Finally, the variables are binary in (6) and (7). In our numerical experimentation we fix the number K of cells for each problem to its value in the best-known solution reported in the literature, and constraints (4) and (5) eliminate any empty cell

3 SIMULATED ANNEALING

To deal with the cell formation problem, we use a straightforward implementation of the simulated annealing method presented in (Ferland and Costa, 2001.

***Procedure Simulated Annealing* (*N*)**
Initialization:
Let (C^0, F^0) an initial solution; TP^0 the initial temperature
Let iter := 0; $TP := TP^0$; $fcount := 0$
Let $(C, F) := (C^*, F^*) := (C^0, F^0)$; stop := false
While not stop
$changes := 0$; $trials := 0$
While $trials < SF$ **and** $changes < coff$
Generate a solution $(C', F') \in N(C, F)$
$\Delta := Eff(C', F') - Eff(C, F)$
If $\Delta > 0$
then $(C, F) := (C', F')$ **and** changes := changes + 1
else generate a random number $r \in (0,1)$
If $r < e^{\Delta/TP}$ **then** $(C, F) := (C', F')$ **and** changes := changes + 1
If $Eff(C', F') > Eff(C^*, F^*)$ **then** $(C^*, F^*) := (C', F')$ **and** $fcount := 0$
$trials := trials + 1$
$TP := \alpha TP$
Iter := iter + 1
If *changes/trials* < mpc **then** *fcount* := *fcount* + 1
If *iter* ≥ itermax **or** *fcount* = *flimit* **then** stop := true
(C^*, F^*) is the best solution generated □

In this implementation of the simulated annealing, we complete several iterations with the same temperature *TP*. This temperature is modified when the number of trial solutions (*trials*) or when the number of times that the current solution is changed (*changes*) reaches threshold values *Sf* or *coff*, respectively. The parameter α is used to modify the temperature. Two stopping criteria are used. The first is fixed in terms of the number of different temperature values used (itermax). To apply the second criterion, we keep track of the number of consecutive temperature values (*fcount*) where the number of *changes* over the number of *trials* is smaller than a threshold value mpc. When *fcount* reaches the value *flimit*, the procedure stops.
To complete the presentation of the procedure, we indicate how the initial solution

$\left(C^0, F^0\right)$ is generated, and we describe the different neighborhoods *N* that we are using.

3.1 Initial Solution

To generate the initial solution, we use a procedure quite similar to the one proposed in (Rojas *et al.*, 2004) that is introduced in (Elbenani *et al.*, 2011). First we determine *K* machine groups $C_1^0,\ldots,C_K^0$. Then the *K* part families $F_1^0,\ldots,F_K^0$ are specified on the basis of the *K* machines groups known.

Denote

$$a_{i\bullet} = \sum_{j=1}^{n} a_{ij} \text{ and } a_{\bullet j} = \sum_{i=1}^{m} a_{ij}$$

the number of parts processed by machine *i* and the number of machines processing *j*, respectively. To initiate the machine groups formation, select the *K* machines having the largest values $a_{i\bullet}$, and assign them to the different groups $C_k^0, k=1,\ldots K$. Then each of the other machines left is assigned to the group C_k^0 including machines processing mostly the same parts. More specifically, denote *INA* the set of machine left. The assignments are completed as follows:

I. For all machines $i \in INA$, determine the group

$$\bar{k}(i) = \underset{k=1,\ldots,K}{\text{Min}} \left\{ \frac{1}{\left|C_k^0\right|} \sum_{j=1}^{n} \sum_{i_k \in C_k^0} \left| a_{ij} - a_{i_k j} \right| \right\} \qquad gr_i = \underset{k=1,\ldots,K}{\text{ArgMin}} \left\{ \frac{1}{\left|C_k^0\right|} \sum_{j=1}^{n} \sum_{i_k \in C_k^0} \left| a_{ij} - a_{i_k j} \right| \right\}.$$

II. Determine the machine $\bar{i} \in INA$

$$\bar{i} = \underset{i \in INA}{\text{ArgMin}} \left\{ \bar{k}(i) \right\}$$

and assign $\bar{i}$ to group $C_{gr_{\bar{i}}}^0$; i.e., $C_{gr_{\bar{i}}}^0 = C_{gr_{\bar{i}}}^0 \bigcup \{\bar{i}\}$.

III. Eliminate $\bar{i}$ from *INA*, and repeat I) until *INA* becomes empty.

On the basis of the *K* machine groups $C_1^0,\ldots,C_K^0$, determine the *K* part families $F_1^0,\ldots,F_K^0$. For each part *j*, denote

$\tilde{a}_{1j}^{In}(k) = \sum_{i \in C_k^0} a_{ij}$ the number of machines in group k that are processing part j

$\tilde{a}_{0j}^{In}(k) = \left|C_k^0\right| - \tilde{a}_{1j}^{In}(k)$ the number of machines in group k that are not processing part j

$\dfrac{\tilde{a}_{1j}^{In}(k)}{a_{\bullet j} + \tilde{a}_{0j}^{In}(k)}$ an approximation of the impact on the grouping efficacy *Eff* of assigning part j to family k.

Then each part *j* is assigned to the family $F_{\tilde{k}(j)}^0$ where $\tilde{k}(j) = \underset{k=1,\ldots,K}{\text{ArgMax}} \left\{ \dfrac{\tilde{a}_{1j}^{In}(k)}{a_{\bullet j} + \tilde{a}_{0j}^{In}(k)} \right\}$ in

order to generate a good initial solution $\left(C^0, F^0\right)$ having the grouping efficacy

$$Eff = \frac{\sum_{j=1}^{n} \tilde{a}_{1j}^{In}\left(\tilde{k}(j)\right)}{a + \sum_{j=1}^{n} \tilde{a}_{0j}^{In}\left(\tilde{k}(j)\right)}.$$

Note that if some family F_k^0 is empty, then we apply the *repair process* to reassign one part to it inducing the smallest decrease of the grouping efficiency.

3.2 Neighborhoods

Different neighborhoods are used to obtain different variants of the simulated annealing method. Each neighborhood is obtained by using a diversification strategy to destroy and recover a new solution, and an intensification strategy to improve the new solution. This solution generated is denoted

$$\left(C', F'\right) \in N\left(C, F\right).$$

3.2.1 Diversification of the solution $\left(C, F\right)$

The procedure is applied on the current solution $\left(C, F\right)$ in order to modify (destroy) the assignment of some elements (machines and/or parts) to be reassigned to other cells selected randomly in order to recover a new solution $\left(C'', F''\right)$. We consider two different ways to destroy the current solution $\left(C, F\right)$:

- ***D1***: Modify the assignment of $\lceil \%n \rceil$ parts and of $\lceil \%m \rceil$ machines (the destroy percentage % being a parameter of the method).
- ***D2***: Select randomly between two strategies: modify either $\lceil \%n \rceil$ parts or modify $\lceil \%m \rceil$ machines.

3.2.2 Intensification of the solution $\left(C'', F''\right)$

To intensify the search around the solution $\left(C'', F''\right)$, we modify successively the machine groups on the basis of the part families and the part families on the basis of the machine groups until no modification is possible. The solution $\left(C', F'\right) \in N\left(C, F\right)$ is the best solution generated during the process. In this paper we consider two different ways for modifying the part families (machine groups) on the basis of the machine groups (part families).

***I1*: *Approximation method*:**

The first procedure to modify the part families on the basis of the machine groups is introduced in (Elbenani *et al.*, 2011), and it is similar to the process for fixing the part families on the basis of the machine groups introduced in the preceding Section 3.1 (where we generate the initial solution). A procedure can be obtained *mutatis mutandis* to modify

the machine groups on the basis of the part families.
Note that whenever the machines groups (or the part families) include an empty one, then we apply a *repair process* to reassign one machine to it inducing the smallest decrease of the grouping efficacy.

I2: ***Exact procedure*****:**
The exact procedure to modify the part families on the basis of the machine groups relies on the Dinkelbach approach (Dinkelbach 1967) to solve fractional programming problems. Indeed, since the group efficacy

$$Eff = \frac{a_1^{In}}{a + a_0^{In}} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} x_{ik} y_{jk}}{a + \sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(1 - a_{ij}\right) x_{ik} y_{jk}}$$

is fractional, it seems appropriate to use the Dinkelbach algorithm to solve the problem of modifying part families on the basis of the machine groups. Indeed, once the machine groups are fixed to $\overline{C}$ (i.e., $x = \overline{x}$), the problem $M\left(x, y\right)$ reduces to

$$M\left(\overline{x}, y\right) \qquad \text{Max } Eff = \frac{a_1^{In}\left(\overline{x}, y\right)}{a + a_0^{In}(\overline{x}, y)} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} \overline{x}_{ik} y_{jk}}{a + \sum_{k=1}^{K}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(1 - a_{ij}\right) \overline{x}_{ik} y_{jk}}$$

$$\text{Subject to } \sum_{k=1}^{K} y_{jk} = 1 \qquad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} y_{jk} \geq 1 \qquad k = 1, \ldots, K$$

$$y_{jk} = 0 \text{ or } 1 \qquad j = 1, \ldots, n; k = 1, \ldots, K.$$

The Dinkelbach procedure to deal with $M\left(\overline{x}, y\right)$ requires solving a sequence of problems where the objective function becomes linear by combining the numerator and the denominator of *Eff*:

$$M\left(\lambda, \overline{x}, y\right) \qquad \text{Max } E(\lambda) = a_1^{In}\left(\overline{x}, y\right) - \lambda\left(a + a_0^{In}(\overline{x}, y)\right) =$$

$$- \lambda a + \sum_{j=1}^{n}\left[\sum_{k=1}^{K}\sum_{i \in \overline{C}_k} a_{ij} - \lambda \sum_{k=1}^{K}\sum_{i \in \overline{C}_k}(1 - a_{ij})\right] y_{jk}$$

$$\text{Subject to } \sum_{k=1}^{K} y_{jk} = 1 \qquad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} y_{jk} \geq 1 \qquad k = 1, \ldots, K$$

$$y_{jk} = 0 \text{ or } 1 \qquad j = 1, \ldots, n; k = 1, \ldots, K$$

for different values of λ. As we shall see, this problem is trivial to solve.
First, introduce the Dinkelbach procedure solving $M(\lambda, \bar{x}, y)$:

Dinkelbach procedure

- *Initialization.*
  - Start with the solution $(\bar{C}, F^0)$
  - Take $\lambda_0 := Eff(\bar{C}, F^0) = \dfrac{a_1^{In}(\bar{x}, y^0)}{a + a_0^{In}(\bar{x}, y^0)}$, and $\zeta := 1$.
- *Step* ζ.
  - Solve the problem $M(\lambda_{\zeta-1}, \bar{x}, y)$

    Let $(\bar{C}, F^\zeta)$ be an optimal solution of this problem.

    Let $E(\lambda_\zeta)$ be the optimal of this problem.
  - *Stopping rule.* If $E(\lambda_\zeta) = 0$, then STOP: $(\bar{C}, F^\zeta)$ is an optimal solution and $Eff(\bar{C}, F^\zeta)$ is an optimal value of $M(\bar{x}, y)$).
  - Otherwise, let $\lambda_\zeta := Eff(\bar{C}, F^\zeta)$. Let $\zeta := \zeta + 1$, and go back to *Step* ζ. □

In the Dinkelbach procedure, the initial solution $(\bar{C}, F^0)$ is the current solution on hand. A sequence of different part families are generated, and this sequence converges to an optimal *K* part families on the basis of the *K* machine groups $\bar{C}$. The algorithm converges since the sequence $\{\lambda_\zeta\}$ is strictly increasing (Crouzeix et al 2008).

Now consider the problem $M(\lambda, \bar{x}, y)$. Since the objective function

$$E(\lambda) = -\lambda a + \sum_{j=1}^{n} \left[\sum_{k=1}^{K} \sum_{i \in \bar{C}_k} a_{ij} - \lambda \sum_{k=1}^{K} \sum_{i \in \bar{C}_k} (1 - a_{ij}) \right] y_{jk}$$

is separable in *j*, the optimal assignment of part *j* is determined by the index $\bar{k} \in \{1, \ldots, K\}$ where

$$\left[\sum_{i \in \bar{C}_{\bar{k}}} a_{ij} - \lambda \sum_{i \in \bar{C}_{\bar{k}}} (1 - a_{ij}) \right] = \operatorname{Max}_{k=1,\ldots,K} \left\{ \left[\sum_{i \in \bar{C}_k} a_{ij} - \lambda \sum_{i \in \bar{C}_k} (1 - a_{ij}) \right] \right\}$$

and assigning $j \in F_{\bar{k}}$. Note that referring to the formulation of the cell formation problem, it follows that

$$\left[\sum_{i\in\bar{C}_k} a_{ij} - \lambda\sum_{i\in\bar{C}_k}(1-a_{ij})\right] = \left[\begin{pmatrix}\text{the number of 1 in the column } j \text{ that}\\ \text{belong to the rows included in the set } \bar{C}_k\end{pmatrix} - \lambda\begin{pmatrix}\text{the number of 0 in the column } j \text{ that}\\ \text{belong to the rows included in the set } \bar{C}_k\end{pmatrix}\right]$$

A similar procedure can be obtained *mutatis mutandis* to modify the machine groups on the basis of the part families.

Note that whenever the machines groups (or the part families) include an empty one, then we apply a *repair process* to reassign one machine (one part) to it inducing the smallest decrease of the grouping efficiency. This exact procedure is also used by the authors in (Khoa et all 2011) to develop a multi starts procedure to solve the Cell Formation Problem.

3.2.3 Four different neighborhoods

In this paper we compare numerically four different variants specified using the following neighborhoods:

N^1: generated with the diversification ***D1*** and the intensification ***I1***

N^2: generated with the diversification ***D1*** and the intensification ***I2***

N^3: generated with the diversification ***D2*** and the intensification ***I1***

N^4: generated with the diversification ***D2*** and the intensification ***I2***.

4 ADAPTIVE SIMULATED ANNEALING

Referring to the Adaptive Large Neighborhood Search (ALNS) proposed by the authors in (Pisinger and Ropke 2007), we develop a new variant of the Simulated Annealing where the neighborhood used at each iteration is selected randomly in a set of neighborhoods available. In our implementation we use the set of neighborhoods $\{N^1, N^2, N^3, N^4\}$ specified in Section 3.2.3. A probability P_i is associated with each $N^i, i = 1,\ldots,4,$ and the neighborhood is selected according to these probabilities.

The same values of the probabilities should be used for a fixed number *coiteration* of iterations of the Simulated Annealing procedure before being updated according to the performance of the neighborhoods N^i during the procedure. In order to do this, associate a score parameter π_i with each neighborhood N^i. The scores should be proportional to the efficiency of the neighborhoods, and hence larger scores induce that the neighborhoods should be chosen with larger probabilities.

To update the scores after completing *coiteration* iterations, we specify a scalar co_i indicating the number of times that N^i is selected and a value δ_i measuring the efficiency of N^i. These values are updated each time neighborhood N^i is selected as follows:

$$co_i := co_i + 1$$
$$\delta_i := \delta_i + \sigma$$

where

$$\sigma = \begin{cases} \sigma_1 & \text{if } \Delta > 0 \text{ and the best solution is improved} \\ \sigma_2 & \text{if } \Delta > 0 \text{ and the best solution is not improved} \\ \sigma_3 & \text{if the current solution is modified according to the probability} \\ \sigma_4 & \text{if the current solution does not change,} \end{cases}$$

and $\sigma_1 > \sigma_2 > \sigma_3 > 0 > \sigma_4$. Then after completing *coiteration* iterations, the scores $\pi_i, i = 1,\ldots,4,$ are updated as follows:

$$\pi_i := \text{Max}\left\{\varepsilon, (1-\rho)\pi_i + \rho\left(\frac{\delta_i}{co_i}\right)\right\}$$

$$co_i := \delta_i := 0,$$

and the probabilities $P_i, i = 1,\ldots,4,$ become

$$P_i := \frac{\pi_i}{\sum_{l=1}^{4} \pi_l}.$$

The values of ε prevents π_i to become negative, and the value of π_i is modified more extensively when the value of $\rho \in [0,1]$ is larger. Moreover, it follows that the probability P_i should increase when the neighborhood N^i is successful to increase the value of *Eff*.

5 MODIFYING THE *HYBRID METHOD* (*HM*) IN (Elbenani et al 2011)

The *Hybrid Mehod* (*HM*) introduced in (Elbenani et al 2011) generates very good solutions for the 35 benchmark problems. This hybrid method integrates a *Local Seach Algorithm* (*LSA*) within a steady state *Genetic Algorithm* (GA).

The *LSA* includes two different procedures, one to diversify and the other to intensify the search. They are applied successively for a fixed number of iterations. To diversify more extensively the feasible domain, a destroying procedure is used to select either a subset of machines or a subset of parts for which the assignment is modified. Then a recovering procedure allows generating a new solution by reassigning a new group to each machine or a new family to each part of the subset in order to reduce the grouping efficacy as little as possible. Note the difference with the destroying procedure used to specify the neighborhoods in Section 3.2.1 where the elements are reassigned randomly to a new group or a new family. The intensification strategy is described in Section 3.2.2 where the *Approximation method* (*I1)* is used to modify the part families (machine groups) on the basis of the machine groups (part families).

The purpose of the *Hybrid Method HM* is to allow improving even more the quality of the solutions using a steady state *GA* to diversify even more the procedure. Each solution is encoded as a vector of $(n+m)$ elements including the family of each of the *n* parts and the group of each of the *m* machines. To generate the initial population *S*, we first introduce the solution generated in Section 3.1. Then each of the other solution in *S* is obtained according to the following procedure. First we decide to generate either the machine groups or the part

families, each alternative having a probability of 0.5. If the first alternative is selected, then each machine *i* is assigned randomly to a group *k*. We also prevent that each group is not empty by applying a repair process to move a machine from the group including the most to the empty group. Then the part families are determined on the basis of these machine groups as described in Section 3.1. The *LSA* is applied to improve the solution which is included in the population *S*. The procedure to complete the second alternative is similar. The role of machines and parts are exchanged.

At each iteration (generation) of the *GA,* two solutions are selected according to a tournament strategy based on their fitness measured in term of their *Eff*. A *uniform crossover* is applied to generate two offspring solutions. If required, the repair process is applied to insure that no group or no family is empty. A *mutation operator* is specified by selecting randomly one machine and one part that are reassigned to a new group and a new family selected randomly. The *mutation* is performed according to a probability *pm*. Finally, the *LSA* method is applied to improve each offspring solution before updating the population of solutions.

The *GA* stops whenever the best solution is not improved for a fixed number *nga* of consecutive iterations.

In this paper, we introduce two different modifications of *HM*, and we compare them numerically. The first modification (*HM_E*) is obtained by modifying the *LSA* to replace the *Approximation method* by the *Exact procedure* to modify the part families (machine groups) on the basis of the machine groups (part families). The purpose of the second modification is to verify if the solutions obtained with the methods *HM* and *HM_E* can be improved by applying a *SA* method afterward to their solutions. The corresponding methods are denoted *HM_SA* and *HM_E_SA*.

6 NUMERICAL RESULTS

To complete the numerical experimentation, we consider the 35 benchmark problems that are commonly used by authors to evaluate the efficiency of their methods. The first 5 columns of Table 3 indicate the problem number, the reference where it is specified (Problem source), its size (values of *m*, *n*, and *K*), and the value of its best-known solution (*BKS*). Moreover the values of the best-known solutions are identified by refereeing to the following references (Goncalves and Resende, 2004, James *et al.*, 2007, Luo and Tang, 2009, Mahdavi *et al.*, 2009, Tunnukij and Hicks, 2009, Elbenani *et al.*, 2011, and Ying *et al.*, 2011). Furthermore, the authors in (Elbenani and Ferland, 2012) are using an exact method based on the Dinkelbach approach to show that the best-known value is in fact equal to the optimal value for the following problems: P1 to P17, P19 to P24, P28, P30, P34, and P35. This is indicated in Table 3 by marking these optimal values with the index *.

The numerical tests are completed on a PC equipped with an INTEL Core 2 Duo processor running at 2.2 GHZ, and having a 2 GB of central memory on a Linux system. To complete the comparisons of the variants, we always use the average *Eff* (Aver. *Eff*) and the average solution time (Aver. Time) over 10 runs for solving each problem.

The first part of this section is dedicated to compare the different variants of the *SA* method. Then we verify if the adaptive impact allows obtaining better results using the *ASA* methods. The different variants of the *HM* method are compared numerically to verify if the *Exact procedure* allows also improvements and to see the benefit of applying a *SA* method

afterward. Finally, we conclude this section by comparing numerically the best variants of these three methods.

6.1 The *SA* method

The purpose of this analysis is threefold. First we compare the average group efficacy (*A.Eff*) of the four variants of *SA*. As a consequence we should identify the best diversification (***D1*** or ***D2***) and the best intensification (***I1*** or ***I2***) strategies. In the second part, we compare the impact of the percentage % of modified elements in the diversification strategies. Three different values are considered: 20%, 30%, and 50%. Finally, we verify how the quality of the solutions can be improved when the solution time allowed increases.

Comparing the neighborhoods

To compare the neighborhoods, we implement the *SA* method with the following values for the parameters:

$$TP^0 = 100K \qquad \text{mpc} = 0.5 \qquad \alpha = 0.2$$
$$\text{itermax} = 10K \qquad Sf = 2K \qquad coff = 2K$$
$$flimit = 5K.$$

The last four columns of Table 3 include the *A.Eff* for the *SA* using the four different neighborhoods $N^i, i = 1,\ldots,4.$ For each problem, the best value of *A.Eff* is marked in bold. To reduce the length of the paper, we report only the table where the percentage is fixed at 30%, but the tables for the other two values of % are quite similar. Additional results comparing the average *A.Eff* (Aver. *Eff*) and the average solution time (Aver. Time) for the 35 problems, and the number of problems where the *BKS* is reached or exceeded for the four neighborhoods are included in Table 4. The numerical results in Tables 3 and 4 indicate that the variants using neighborhoods N^2 and N^4 allows generating better results than using N^1 and N^3. Furthermore, the overall average (last row of the Table 3) for the variant with N^2 exceeds slightly that of *BKS,* and that of the variant with N^4 is slightly smaller than that of *BKS.* Hence these variants seem very efficient to solve the cell formation problem.

This analysis above allows concluding that the intensification strategy ***I2*** seems more efficient than ***I1***. Furthermore, since the variant N^2 is slightly more efficient than N^4, it follows that the diversification ***D1*** seems to be slightly more efficient than ***D2*** when combined with the intensification ***I2***.

Table 3: Compare *A. Eff* of the four neighborhoods when %=30%

P	Problem source	*m*	*n*	*K*	*BKS*	N^1	N^2	N^3	N^4
P1	King and Nakornchai (1982)	5	7	2	**82.35***	**82.35**	**82.35**	**82.35**	**82.35**
P2	Waghodekar and Sahu (1984)	5	7	2	**69.57***	69.25	**69.57**	69.41	**69.57**
P3	Seifoddini (1989)	5	18	2	**79.59***	**79.59**	**79.59**	**79.59**	**79.59**
P4	Kusiak and Cho (1992)	6	8	2	**76.92***	**76.92**	**76.92**	**76.92**	**76.92**
P5	Kusiak and Chow (1987)	7	11	5	**60.87***	**60.87**	**60.87**	**60.87**	**60.87**

P6	Boctor (1991)	7	11	4	**70.83***	**70.83**	**70.83**	**70.83**	**70.83**
P7	Seifoddini and Wolfe (1986)	8	12	4	**69.44***	**69.44**	**69.44**	68.84	**69.44**
P8	Chandrasekharan and Rajagopalon (1986a)	8	20	3	**85.25***	**85.25**	**85.25**	**85.25**	**85.25**
P9	Chandrasekharan and Rajagopalon (1986b)	8	20	2	**58.72***	58.62	58.56	58.4	58.5
P10	Mosier and Taube (1985a)	10	10	5	**75***	**75**	**75**	**75**	**75**
P11	Chan and Milner (1982)	10	15	3	**92***	**92**	**92**	**92**	**92**
P12	Askin and Subramanian (1987)	14	24	7	**72.06***	71.64	**72.06**	71.54	**72.06**
P13	Stanfel (1985)	14	24	7	**71.83***	**71.83**	**71.83**	**71.83**	**71.83**
P14	McCormick (1972)	16	24	8	**53.26***	52.96	**53.26**	52.83	**53.26**
P15	Srinivasan et al. (1990)	16	30	6	**69.53***	67.83	**69.53**	68.02	69.11
P16	King (1980)	16	43	8	**57.53***	57.41	**57.53**	57.38	**57.53**
P17	Carrie (1973)	18	24	9	**57.73***	**57.73**	**57.73**	**57.73**	**57.73**
P18	Mosier and Taube (1985b)	20	20	5	42.96	**43.01**	**43.12**	42.83	**43.06**
P19	Kumar et al. (1986)	20	23	7	**50.81***	**50.81**	**50.81**	50.68	**50.81**
P20	Carrie (1973)	20	35	5	**77.91***	76.33	**77.91**	76.33	**77.91**
P21	Boe and Cheng (1991)	20	35	5	**57.98***	56.93	**57.98**	56.86	**57.98**
P22	Chandrasekharan and Rajagopalon (1989)	24	40	7	**100***	**100**	**100**	**100**	**100**
P23	Chandrasekharan and Rajagopalon (1989)	24	40	7	**85.11***	**85.11**	**85.11**	**85.11**	**85.11**
P24	Chandrasekharan and Rajagopalon (1989)	24	40	7	**73.51***	**73.51**	**73.51**	**73.51**	**73.51**
P25	Chandrasekharan and Rajagopalon (1989)	24	40	11	**53.29**	**53.29**	**53.29**	**53.29**	**53.29**
P26	Chandrasekharan and Rajagopalon (1989)	24	40	12	**48.95**	**48.95**	**48.95**	48.85	**48.95**
P27	Chandrasekharan and Rajagopalon (1989)	24	40	12	**46.58**	46.57	**46.58**	46.52	46.55
P28	McCormick (1972)	27	27	5	**54.82***	**54.82**	**54.82**	54.78	**54.82**
P29	Carrie (1973)	28	46	10	**47.08**	46.39	**47.08**	46.23	**47.08**
P30	Kumar and Vannelli (1987)	30	41	14	**63.31***	62.99	**63.31**	62.9	**63.31**
P31	Stanfel (1985)	30	50	13	**60.12**	**60.12**	**60.12**	60.09	**60.12**
P32	Stanfel (1985)	30	50	14	**50.83**	50.8	**50.83**	50.74	**50.83**
P33	King and Nakornchai (1982)	36	90	17	46.67	**46.71**	**47.18**	**46.7**	**47.17**
P34	McCormick (1972)	37	53	3	**60.64***	58.31	60.63	58.26	60.63
P35	Chandrasekharan and Rajagopalon (1987)	40	100	10	**84.03***	**84.03**	**84.03**	**84.03**	**84.03**
Aver *Eff*					65.92	65.66	**65.93**	65.61	65.91

Table 4: Comparing the four neighborhoods for %=30%

	Aver. *Eff*	Aver. Time (sec)	*BKS* reached	*BKS* exceeded
N^1	65.66	10.01	20	2
N^2	65.93	15.49	31	2
N^3	65.61	7.22	15	1
N^4	65.91	9.91	29	2

Impact of the value of the destroying percentage %

Now consider the results summarized in Tables 5 and 6 to analyze the efficiency of the variant using N^2 with different percentages %. For each problem, the best-solution is marked in bold, and the smallest solution time (*A.* Time) average solution time over the 10 runs, is marked in italic bold. The results in Table 6 indicate that the percentage 30% allows reaching or exceeding the best-known solution more often, but the percentage 20% allows a smaller average solution time. Thus if the user put more emphasis on the quality of the solution, then the percentage 30% is more appropriate, but if the solution time must be reduced, then the percentage of 20% is more convenient.

Table 5: Compare *A. Eff* of N^2 when %=20% , 30% and 50%

P	*BKS*	$N^2(20\%)$		$N^2(30\%)$		$N^2(50\%)$	
		A. Eff	*A.* Time (sec.)	*A. Eff*	*A.*Time (sec.)	*A. Eff*	*A.*Time (sec.)
P1	**82.35**	**82.35**	***0.018***	**82.35**	0.027	**82.35**	0.033
P2	**69.57**	**69.57**	***0.02***	**69.57**	0.028	**69.57**	0.03
P3	**79.59**	**79.59**	***0.037***	**79.59**	0.048	**79.59**	0.052
P4	**76.92**	**76.92**	***0.026***	**76.92**	0.028	**76.92**	0.04
P5	**60.87**	**60.87**	***0.374***	**60.87**	0.426	**60.87**	0.465
P6	**70.83**	**70.83**	***0.227***	**70.83**	0.245	**70.83**	0.302
P7	**69.44**	**69.44**	***0.251***	**69.44**	0.28	**69.44**	0.338
P8	**85.25**	**85.25**	***0.146***	**85.25**	0.166	**85.25**	0.2
P9	**58.72**	58.53	***0.045***	58.56	0.051	**58.72**	0.058
P10	**75**	**75**	***0.421***	**75**	0.493	**75**	0.624
P11	**92**	**92**	***0.14***	**92**	0.154	**92**	0.192
P12	**72.06**	**72.06**	***2.252***	**72.06**	2.735	**72.06**	3.34
P13	**71.83**	**71.83**	***2.206***	**71.83**	2.755	**71.83**	3.355
P14	**53.26**	**53.26**	***4.83***	**53.26**	5.22	**53.26**	6.124

P15	**69.53**	**69.53**	***1.621***	**69.53**	1.904	**69.53**	2.435
P16	**57.53**	**57.53**	***6.932***	**57.53**	7.759	**57.53**	8.85
P17	**57.73**	**57.73**	***6.288***	**57.73**	7.34	**57.73**	8.427
P18	42.96	**43.04**	***1.398***	**43.12**	1.702	**43.1**	2.204
P19	**50.81**	**50.81**	***3.336***	**50.81**	3.8	**50.81**	4.761
P20	**77.91**	**77.91**	***1.254***	**77.91**	1.484	**77.91**	1.889
P21	**57.98**	**57.98**	***1.483***	**57.98**	1.764	**57.98**	2.216
P22	**100**	**100**	***4.284***	**100**	4.362	**100**	5.296
P23	**85.11**	**85.11**	***4.423***	**85.11**	4.865	**85.11**	7.36
P24	**73.51**	**73.51**	***4.637***	**73.51**	5.502	**73.51**	8.611
P25	**53.29**	**53.29**	***15.459***	**53.29**	19.5	**53.29**	25.57
P26	**48.95**	**48.95**	***21.828***	**48.95**	29.264	48.88	41.684
P27	**46.58**	**46.58**	***21.194***	**46.58**	27.573	46.51	43.48
P28	**54.82**	**54.82**	***1.306***	**54.82**	1.631	**54.82**	1.98
P29	**47.08**	47.07	***21.323***	**47.08**	22.886	**47.08**	32.465
P30	**63.31**	63.29	***47.698***	**63.31**	58.074	**63.31**	65.175
P31	**60.12**	**60.12**	***32.113***	**60.12**	39.162	**60.12**	51.847
P32	**50.83**	**50.83**	***47.931***	**50.83**	55.442	**50.83**	77.801
P33	46.67	**47.17**	***161.88***	**47.18**	204.38	**47.18**	229.04
P34	**60.64**	60.63	***1.008***	60.63	1.021	60.63	1.055
P35	**84.03**	**84.03**	***29.171***	**84.03**	30.058	**84.03**	44.132
Aver.	65.93	**65.93**	12.787	**65.93**	15.489	**65.93**	19.469

Table 6: Table Table 6: Comparing the % for N^2

%	Aver. *Eff*	Aver. Time (sec.)	*BKS* reached	*BKS* exceeded
20	65.93	12.787	29	2
30	65.93	15.489	31	2
50	65.93	19.469	30	2

Impact of increasing solution time

To complete this analysis, we consider only the better variants N^2 and N^4. Furthermore, we implement the *SA* method using smaller values for the two parameters $Sf = K$ and $coff = \lceil 0.5K \rceil$, and the destroy percentage is fixed at 30%:

$TP^0 = 100K$ $\quad$ mpc $= 0.5$ $\quad$ $\alpha = 0.2$ $\quad$ $destroy\ \ percentage = 30\%$

$Sf = K$ $\quad$ $coff = \lceil 0.5K \rceil$

the purpose being to reduce the solution time for the tests.

Since the numerical results indicate that the variants N^2 and N^4 reached the optimal value for several problems, we only verify the impact of increasing the solution time for the 13 problems where the optimal value is unknown (P9, P15, P16, P18, P25 to P27, P29 to P34) and where one of the methods does not reach the optimal solution.

To modify the solution time, we increase the values of the stopping criteria itermax and *flimit*:

$$\text{cSA 0: itermax} = 10K \quad \text{and} \quad flimit = 5K$$
$$\text{cSA 1: itermax} = 20K \quad \text{and} \quad flimit = 15K$$
$$\text{cSA 2: itermax} = 45K \quad \text{and} \quad flimit = 40K.$$

The first column of Table 7 includes the problems considered, and columns 2 to 4 and 5 to 7 are associated with the variants N^2 and N^4, respectively. Each entry in Table 7 includes the values *A*. *Eff* and *A*. Time in the first and the second row, respectively. Furthermore, for each problem and for each variant, the best *A*. *Eff* is marked in bold when it is reached for the smallest *A*. Time. Denote this best value of *A. Eff* by *B. Eff*, and the smallest *A*. Time by *B.* Time .

Table 7: Increasing solution time for N^2 and N^4 when % = 30%

P	N^2			N^4		
	cSA 0	cSA 1	cSA 2	cSA 0	cSA 1	cSA 2
P9	58.56	58.62	**58.68**	58.46	58.47	**58.53**
	0.021	0.044	0.114	0.009	0.027	0.063
P15	69.32	**69.53**	69.53	68.27	68.69	**69.32**
	0.891	2.328	6.091	0.733	1.453	3.45
P16	**57.53**	57.53	57.53	57.49	57.51	**57.53**
	4.103	9.969	25.835	2.696	5.642	13.228
P18	43.06	43.09	**43.13**	43.06	43.04	**43.10**
	0.81	2.125	5.188	0.522	1.228	2.949
P25	**53.29**	53.29	53.29	**53.29**	53.29	53.29
	9.5	26.583	74.111	5.378	15.092	40.755
P26	**48.95**	48.95	48.95	48.92	48.92	**48.95**
	15.808	37.656	101.088	9.752	21.426	56.407
P27	46.56	46.57	**46.58**	46.50	**46.58**	46.55
	14.621	37.596	103.407	8.826	22.368	55.708
P29	**47.08**	47.08	47.08	47.06	**47.08**	47.08
	13.051	30.501	86.155	11.051	26.037	68.5

P30	63.27	**63.31**	63.31	63.27	**63.31**	63.31
	27.465	73.476	217.905	17.629	42.194	118.13
P31	**60.12**	60.12	60.12	**60.12**	60.12	60.12
	19.889	55.452	164.04	11.66	30.849	84.967
P32	**50.83**	50.83	50.83	**50.83**	50.83	50.83
	27.099	77.366	225.759	17.132	42.349	118.892
P33	47.16	47.17	**47.19**	47.13	47.16	**47.18**
	90.153	241.075	604.44	72.655	172.285	452.21
P34	**60.63**	60.63	60.63	**60.63**	60.63	60.63
	0.519	1.077	2.739	0.405	0.897	2.184
	54.34	54.36	54.37	54.23	54.28	54.34
	17.225	45.788	124.375	12.188	29.373	78.265

Increasing solution time seems to have a larger impact for the variant N^4. Indeed, referring to the last row of Table 7, it follows that multiplying the solution time by a factor of 2.66 and 2.41 to move from c*SA* 0 to c*SA* 1 induces an increase of the Aver. *Eff* by factors of 1.0004 and 1.0009 for N^2 and N^4, respectively. Similarly, to move from c*SA* 1 to c*SA* 2 by multiplying the solution time by a factor of 2.72 and 2.66 induces also a larger increase of Aver. *Eff* for N^4 than for N^2 (factors of 1.0002 and 1.0011 for N^2 and N^4, respectively). This observation also follows from the fact that, for each problem, the case number (c*SA* 0, c*SA* 1, or c*SA* 2) where *B*. *Eff* is reached is in general smaller or equal in N^2 than in N^4.

Comparing N^2 ***and*** N^4

Now considering only the problems used in Table 7, evaluate the average values (Aver. *B*. *Eff* and Aver. *B*. Time) similar to those in the last row of Table 7, but where, for each problem, we use the *B*. *Eff* and the *B*. Time reaching it. These elements are included in the second and third columns of Table 8. Furthermore, determine the number of problems where each variant reaches a better *B*. *Eff* than the other. They are included in the 2×2 matrix in columns 4 and 5 of Table 8 (i.e., N^2 $\left(N^4\right)$ reaches a better *B*. *Eff* than N^4 $\left(N^2\right)$ in 4 (0) problems).

Table 8: Comparing N^2 and N^4

Variant	Aver. *B*. *Eff*	Aver. *B*. Time (sec.)	N^2	N^4	*BKS*
N^2	54.37	67.609	–	4	2
N^4	54.34	50.268	0	–	2
BKS	–	–	2	3	–

The results in Table 8 lead to the conclusion of selecting the variant N^2 to get better solutions and N^4 to reduce the solution time.

6.2 The *ASA* method

First note that the parameters to implement the *SA* method are the same as those used above when analyzing the impact of increasing the solution time for *SA*:

$$TP^0 = 100K \qquad \text{mpc} = 0.5 \qquad \alpha = 0.2$$
$$Sf = K \qquad coff = \lceil 0.5K \rceil$$

The additional parameters required in *ASA* are specified as follows:

$coiteration$ (frequency to modify the scores) $= 2K$

the values to update the scores: $\sigma_1 = 7 \qquad \sigma_2 = 3$

$\sigma_3 = 2 \qquad \sigma_4 = -1$

ρ (parameter to modify the scores) = 0.7

To implement the *ASA* method, we use the 4 neighborhoods N^i. Different variants are obtained with different values for the original scores $\pi_i^0, i = 1,\ldots,4$, and for different values for the destroy percentage % of the neighborhoods. Preliminary tests using more than 20 variants indicate that the most promising variants are those where the initial scores of π_2^0 and π_4^0 for N^2 and N^4 are larger than π_1^0 and π_3^0 for N^1 and N^3. For this reason, we complete the numerical comparison with the following variants:

$$ASA^1 : \pi_1^0 = 10,\ \pi_2^0 = 100,\ \pi_3^0 = 10,\ \pi_4^0 = 100;\ \%^i = 30\%,\ i = 1,\ldots,4$$
$$ASA^2 : \pi_1^0 = 25,\ \pi_2^0 = 100,\ \pi_3^0 = 25,\ \pi_4^0 = 75;\ \%^i = 30\%,\ i = 1,\ldots,4.$$

To compare the two variants ASA^1 and ASA^2, and to analyze the impact of increasing the solution time, we only consider the 14 problems where the optimal solution is unknown (P9, P15, P16, P18, P25 to P34) or where the optimal solution is not reached. The results are summarized in Table 9 having the same format as Table7.

Table 9: Increasing solution time for ASA^1 and ASA^2 when % = 30%

P	ASA^1			ASA^2		
	c*SA* 0	c*SA* 1	c*SA* 2	c*SA* 0	c*SA* 1	c*SA* 2
P9	58.5	58.56	**58.65**	58.53	58.56	**58.62**
	0.019	0.034	0.079	0.017	0.036	0.075
P15	68.97	68.65	**69.37**	68.87	68.47	**69.37**
	0.76	1.526	3.143	0.632	1.403	3.395
P16	57.38	57.49	**57.53**	57.4	**57.52**	57.51
	1.911	5.715	12.281	2.059	5.205	12.248

P18	42.96	43	**43.08**	42.96	43.06	**43.07**
	0.609	1.298	3.02	0.531	1.381	2.921
P25	**53.29**	53.29	53.29	**53.29**	53.29	53.29
	5.347	14.39	34.736	5.535	14.809	37.016
P26	48.78	48.82	**48.95**	48.71	48.92	**48.95**
	8.481	20.303	47.069	7.717	19.418	51.294
P27	46.36	**46.57**	46.57	46.53	46.57	**46.58**
	8.719	20.502	48.748	8.281	19.488	48.16
P28	54.77	**54.82**	54.82	54.81	**54.82**	54.82
	0.604	1.371	3.26	0.639	1.393	3.048
P29	47.01	**47.08**	47.08	**47.08**	47.08	47.08
	9.688	22.047	53.966	9.483	23.145	53.541
P30	63.27	63.27	**63.31**	**63.31**	63.31	63.31
	18.583	42.778	116.322	18.571	45.845	112.592
P31	**60.12**	60.12	60.12	**60.12**	60.12	60.12
	10.779	31.394	81.746	10.303	28.589	76.467
P32	**50.83**	50.83	50.83	50.8	**50.83**	50.83
	14.011	40.208	109.265	16.971	40.238	108.891
P33	47.16	47.16	**47.18**	47.15	47.17	**47.19**
	58.589	181.303	422.702	65.454	176.526	425.27
P34	60.57	**60.63**	60.63	**60.63**	60.62	60.63
	0.457	0.904	2.115	0.46	0.917	2.063
	54.28	54.31	54.39	54.3	54.31	54.38
	9.897	27.412	67.032	10.475	27.028	66.927

The last row of Table 9 indicates that we can get better Aver. *Eff* with ASA^1 than ASA^2 using similar solution time. Furthermore, increasing solution time seems to have a larger impact for ASA^1. Indeed, moving from c*SA* 0 to c*SA* 1 by multiplying the solution time by factors 2.76 and 2.51 for ASA^1 and ASA^2 induces a larger increase of Aver. *Eff* for ASA^1 than for ASA^2 (factors of 1.0006 and 1.0002 for ASA^1 and ASA^2, respectively). Similarly, when moving from c*SA* 1 to c*SA* 2 by multiplying the solution time by a factor 2.45 for ASA^1 and ASA^2 induces a larger increase of Aver. *Eff* for ASA^1 than for ASA^2 (factors of 1.0015 and 1.0013 for ASA^1 and ASA^2, respectively).

Comparing ASA^1 ***and*** ASA^2

Determine the elements of Table 10 as those in Table 8.

Table 10: Comparing ASA^1 and ASA^2

Variant	Aver. *B. Eff*	Aver. *B.* Time (sec.)	ASA^1	ASA^2	*BKS*
ASA^1	54.39	48.541	–	3	2
ASA^2	54.38	44.45	2	–	2
BKS	–	–	4	4	–

The results in Table 10 lead to the conclusion of selecting the variant ASA^1 to get better solutions and ASA^2 to reduce the solution time.

6.3 The modified *HM* methods

In this section we are using N^2 as the *SA* applied afterward on the results of *HM* and *HM_E*. The parameter values to implement N^2 are the following:

$$TP^0 = 100K \qquad \text{mpc} = 0.5 \qquad \alpha = 0.2 \qquad destroy\ \ percentage = 20\%$$
$$\text{itermax} = 45K \qquad Sf = K \qquad coff = \lceil 0.5K \rceil$$
$$flimit = 40K.$$

The parameter values for the *HM* method are those specified in (Elbenani et al., 2011). Furthermore, to evaluate the impact of increasing the solution time, we modify the number of generations (iterations) in *HM*:

$$\text{c}HM\ 0 = 5m$$
$$\text{c}HM\ 1 = 10m$$
$$\text{c}HM\ 2 = 20m.$$

Here also, we only verify the impact of increasing the solution time only for the 20 problems where the optimal solution is unknown (P18, P25 to P27, P29, P31 to P33) or where the optimal solution is not reached. The numerical results summarized in Tables 11 and 12 indicate that increasing the number of generations is more significant for *HM* , but not for the other three variants.

Table 11: Increasing solution time for *HM* and *HM_E*

P	*HM*			*HM_E*		
	cHM 0	cHM 1	cHM 2	cHM 0	cHM 1	cHM 2
P2	69.41	**69.57**	69.57	**69.57**	69.57	69.57
	0.006	0.016	0.028	0.022	0.042	0.084
P7	**68.84**	68.84	68.84	**69.44**	69.44	69.44
	0.099	0.183	0.357	0.254	0.462	0.881
P13	**71.83**	71.83	71.83	71.59	**71.83**	71.83
	1.173	2.316	4.586	3.547	6.937	12.852
P14	52.9	**53.16**	53.16	**53.26**	53.26	53.26
	2.455	5.592	9.074	5.261	10.312	20.24
P15	**69.53**	69.53	69.53	**67.42**	67.42	67.42
	1.684	3.142	6.055	3.744	7.389	14.626
P16	57.33	57.34	**57.36**	**57.53**	57.53	57.53
	6.027	10.418	20.63	12.094	22.552	43.909
P17	57.28	57.28	**57.37**	**57.73**	57.73	57.73
	4.612	7.702	14.525	9.445	18.853	38.21
P18	43.07	43.09	**43.09**	43.14	**43.15**	43.15
	1.797	3.189	5.561	3.43	6.558	12.228
P19	50.68	50.72	**50.81**	**50.81**	50.81	50.81
	3.202	5.94	13.027	6.113	12.102	24.114
P20	76.76	77.06	**77.42**	**77.58**	77.58	77.58
	2.031	4.734	10.753	4.851	9.605	19.076

P21	57.17	57.25	**57.32**	**57.07**	57.07	57.07
	2.84	5.009	9.251	5.363	10.679	21.547
P25	**53.29**	53.29	53.29	**53.29**	53.29	53.29
	14.739	29.038	57.301	37.485	73.22	144.079
P26	**48.95**	48.95	48.95	**48.95**	48.95	48.95
	19.443	36.898	72.585	49.315	91.669	174.905
P27	**46.58**	46.58	46.58	**46.58**	46.58	46.58
	19.482	36.19	69.489	48.524	90.732	175.611
P28	54.77	54.78	**54.79**	**54.82**	54.82	54.82
	2.094	4.366	8.778	6.895	13.556	26.907
P29	46.85	46.86	**46.9**	**47.08**	47.08	47.08
	23.727	43.249	94.496	47.605	90.735	176.955
P30	62.99	**63.1**	63.1	**63.31**	63.31	63.31
	55.89	102.323	162.1	89.325	166.114	320.859
P31	**60.12**	60.12	60.12	**60.12**	60.12	60.12
	38.584	75.711	148.866	88.49	175.606	347.844
P32	**50.83**	50.83	50.83	**50.83**	50.83	50.83
	66.315	111.24	199.169	122.935	243.632	485.504
P33	46.99	47.03	**47.06**	47.18	**47.19**	47.19
	320.063	555.388	963.506	361.8	608.213	1021.15
P34	60.38	**60.39**	60.39	**60.63**	60.63	60.63
	4.892	9.514	17.581	6.168	11.854	22.946
	57.45	57.50	57.54	57.52	57.53	57.53
	28.15	50.103	89.891	43.46	79.563	147.835

Table 12: Increasing solution time for HM_N^2 and $HM\_E\_N^2$

P	HM_N^2			$HM\_E\_N^2$		
	c*HM* 0	c*HM* 1	c*HM* 2	c*HM* 0	c*HM* 1	c*HM* 2
P2	**69.57**	69.57	69.57	**69.57**	69.57	69.57
	0.015	0.022	0.035	0.028	0.05	0.09
P7	**69.44**	69.44	69.44	**69.44**	69.44	69.44
	0.222	0.304	0.477	0.382	0.59	1.004
P13	**71.83**	71.83	71.83	**71.83**	71.83	71.83
	2.662	3.807	6.073	5.021	8.396	14.274
P14	**53.26**	53.26	53.26	**53.26**	53.26	53.26
	4.656	7.797	11.273	7.485	12.551	22.423
P15	**69.53**	69.53	69.53	**69.53**	69.53	69.53
	2.754	4.206	7.128	4.632	8.272	15.499
P16	57.49	**57.51**	57.51	**57.53**	57.53	57.53
	11.07	15.27	25.61	16.979	27.612	48.773
P17	**57.73**	57.73	57.73	**57.73**	57.73	57.73
	9.085	12.154	18.991	13.903	23.343	42.548
P18	43.12	**43.14**	43.14	43.14	**43.15**	43.15
	2.734	4.112	6.489	4.334	7.488	13.13
P19	**50.81**	50.81	50.81	**50.81**	50.81	50.81
	5.644	8.366	15.453	8.564	14.593	26.505
P20	**77.91**	77.91	77.91	**77.91**	77.91	77.91
	2.732	5.442	11.505	5.542	10.305	19.752
P21	**57.98**	57.98	57.98	**57.98**	57.98	57.98
	3.826	5.988	10.229	6.234	11.546	22.416
P25	**53.29**	53.29	53.29	**53.29**	53.29	53.29
	30.416	44.723	73.054	52.927	89.03	159.598

P26	**48.95**	48.95	48.95	**48.95**	48.95	48.95
	40.603	58.062	93.754	70.438	113.151	196.055
P27	**46.58**	46.58	46.58	**46.58**	46.58	46.58
	41.085	57.789	91.087	70.126	112.609	197.198
P28	**54.82**	54.82	54.82	**54.82**	54.82	54.82
	3.167	5.441	9.836	7.97	14.647	27.953
P29	**47.08**	47.01	47.04	**47.08**	47.08	47.08
	43.855	63.163	114.475	67.381	110.681	196.499
P30	**63.31**	63.29	63.29	**63.31**	63.31	63.31
	104.436	149.963	209.489	137.444	214.687	368.535
P31	**60.12**	60.12	60.12	**60.12**	60.12	60.12
	76.482	113.589	187.219	127.111	214.142	387.09
P32	**50.83**	50.83	50.83	**50.83**	50.83	50.83
	120.043	164.933	252.806	176.217	297.387	539.877
P33	**47.19**	47.19	47.19	47.18	**47.19**	47.19
	529.706	762.139	1171.75	558.764	809.098	1217.85
P34	**60.63**	60.63	60.63	**60.63**	60.63	60.63
	5.52	10.141	18.202	6.946	12.616	23.628
	57.69	57.69	57.69	57.69	57.69	57.69
	49.558	71.305	111.187	64.211	100.609	168.605

To ease the comparison of the variants, we generate the elements of Table 13 as we did in Table 8. These results allow verifying the positive impact of replacing the *Approximation method* modifying the part families (machine groups) on the basis of the machine groups (part families) by the *Exact procedure*. Indeed, the variant *HM_E* reaches a better *B. Eff* than the variant *HM* for 11 (out of 21) problems while the reverse is true for only 2 problems. Similarly, *HM_E_* N^2 get a better *B. Eff* than *HM_* N^2 for 2 problems.

Table 13: Comparing *HM, HM_E, HM_* N^2, and *HM_E_* N^2

Variants	Aver. *B. Eff*	Aver. *B.* Time (sec.)	*HM*	*HM_E*	*HM_* N^2	*HM_E_* N^2	*BKS*
HM	57.54	67.595	–	2	0	0	2
HM_E	57.53	55.505	11	–	2	0	2
*HM_* N^2	57.69	49.823	12	3	–	0	2
*HM_E_* N^2	57.69	76.282	12	3	2	–	2
BKS	–	–	10	4	2	1	–

The improvement of applying N^2 afterward is also indicated in Table 13. *HM_* N^2 and *HM_E_* N^2 reach better *B. Eff* than variant *HM* and *HM_E* for 12 and 3 problems, respectively. Globally, the results in Table 13 allow to conclude that the variant *HM_E_* N^2 dominate the other three as far as the *B. Eff* is concerned, even though its

*B.*Time is larger.

6.4 Comparing the best variants of *SA, ASA, and HM* modified

The numerical results in this experimentation indicates that the three variants N^2, ASA^1, and $HM\_E\_N^2$ all reach the best-known solution for 29 problems. For the other 6 problems left, the results are summarized in Table 14. For each problem, the *B. Eff* for the variants having a smaller or a larger value than *BKS* is indicated in italic or in bold, respectively. For P18 and P33, the *B.Eff* obtained with the three variants exceeds the *BKS*.

Table 14: Grouping efficacy of N^2, ASA^1 ,and $HM\_E\_N^2$

P	*BKS*	N^2	ASA^1	$HM\_E\_N^2$
P9	58.72*	*58.68*	*58.65*	58.72
P15	69.53*	69.53	*69.37*	69.53
P18	*42.96*	**43.13**	**43.08**	**43.15**
P27	46.58	46.58	*46.57*	46.58
P33	*46.67*	**47.19**	**47.18**	**47.19**
P34	60.64*	*60.63*	*60.63*	*60.63*

The elements of Table 15 are obtained as those in Table 8. We observe that $HM\_E\_N^2$ reaches a better *B. Eff* than N^2 and ASA^1 for 2 and 5 of the 6 problems. Similarly, N^2 reaches a better *B. Eff* than ASA^1 for 5 problems. Furthermore, note that the performance of the variants to reach a better *B. Eff* seems to increase with their *B.* Time. Finally, $HM\_E\_N^2$, N^2, and ASA^1 fail to reach to best-known solution for 1, 2 and 4 problems, respectively.

Table 15: Comparing, N^2, ASA^1 ,and $HM\_E\_N^2$

Variants	*B. Eff*	*B*. Time (sec.)	N^2	ASA^1	$HM\_E\_N^2$	*BKS*
N^2	65.93	25.841	–	5	0	2
ASA^1	65.93	19.965	0	–	0	2
$HM\_E\_N^2$	65.94	52.87	2	5	–	2
BKS	65.92	–	2	4	1	–

In summary, all the variants analyzed in this paper are quite efficient to deal with the cell formation problem, but the three variants compared in Tables 14 and 15 are the most efficient of their categories. Finally, the variant $HM\_E\_N^2$ seems to include all the best features and dominates over all.

7 CONCLUSION

In this paper we introduce three different methods to deal with the cell formation problem: a Simulated Annealing method (*SA*), an Adaptive Simulated Annealing method (*ASA*), and a Hybrid Method (*HM*) combining a Local Search Algorithm (*LSA*) and a Genetic Algorithm (*GA*). Four variants of *SA* are considered using different neighborhoods. Each neighborhood is obtained by combining one of the two diversification strategies to destroy and recover a new solution, and one of the two intensification strategies to improve the solution. The two diversification strategies are as follows:

- ***D1***: Modify the assignment of $\lceil \%n \rceil$ parts and of $\lceil \%m \rceil$ machines
- ***D2***: Select randomly between two strategies: modify either $\lceil \%n \rceil$ parts or modify $\lceil \%m \rceil$ machines

where the parameter % takes the values 20%, 30%, or 50%. The intensification strategies are using one of the following method to modify the part families (machine groups) on the basis of the machine groups (part families):

- ***I1***: *Approximation method* introduced in (Elbenani *et al.*, 2011)
- ***I2***: *Exact procedure* based on the Dinkelbach method.

The *ASA* method is a modification of the *SA* where the neighborhood used at each iteration is selected randomly among the four neighborhoods mentioned above. The method is adaptive in the sense that the probability of selecting a neighborhood is modified during the procedure according to the results obtained when using the neighborhood. Finally, we consider the *HM* introduced in (Elbenani et al., 2011) modified as follows. The first modification is to replace the *Approximation method* used in the *LSA* to modify the part families (machine groups) on the basis of the machine groups (part families) by the *Exact procedure*. The second modification is to apply a *SA* method afterward.

A numerical experimentation is completed using the 35 benchmarked problems commonly used in the literature. The results indicate that better results are obtained using the *Exact procedure* rather than the *Approximation method*. Increasing solution time seems to be more beneficial for the *SA* and the *ASA* variants than for the modified *HM* variants. All the variants are quite efficient to deal with the cell formation problems, but if we compare the best performing variants of the three methods, the modified *HM* using the *Exact procedure* followed by an *SA* afterward seems to dominate.

REFERENCES

Askin, R.G. and Subramanian, S.P., 1987. A cost-based heuristic for group technology configuration. *International Journal of Production Research*, 25, 101–113.

Boctor, F.F., 1991. A linear formulation of the machine-part cell formation problem. *International Journal of Production Research*, 29, 343–356.

Boe, W.J. and Cheng, C.H., 1991. A close neighbour algorithm for designing cellular manufacturing systems. *International Journal of Production Research*, 29, 2097–2116.

Carrie, A.S., 1973. Numerical taxonomy applied to group technology and plant layout. *International Journal of Production Research*, 11, 399–416.

Cerny, V., 1985. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Application*, 45, 41 – 51.

Chan, H.M. and Milner, D.A., 1982. Direct clustering algorithm for group formation in cellular manufacture. *Journal of Manufacturing Systems*, 1, 65-75.

Chandrasekharan, M.P. and Rajagopalan, R., 1987. ZODIAC: an algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research*, 25, 835–850.

Chandrasekharan, M.P. and Rajagopalan, R., 1989. GROUPABILITY: an analysis of the properties of binary data matrices for group technology. *International Journal of Production Research*, 27, 1035–1052.

Chandrasekharan, M.P. and Rajagopalan, R., 1986a. MODROC: an extension of rank order clustering for group technology. *International Journal of Production Research*, 24, 1221–1233.

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

Crouzeix, J.P, Ferland, J.A. and Nguyen, V.H., 2008. Revisiting Dinkelbach-type algorithms for generalized fractional programs. *Operational Research Society of India*, 45, 96-110.

Dimopoulos, C. and Zalzala, A.M.S., 2000. Recent developments in evolutionary computations for manufacturing optimization: problems, solutions, and comparisons. *IEEE Transactions on Evolutionary Computations*, 4, 93–113.

Dinkelbach, W., 1967. On nonlinear fractional programming. *Managenment Science*, 13,492 – 498.

Elbenani, B., Ferland, J.A., Bellemare J., 2011. Genetic algorithm and large neighborhood search to solve the cell formation problem. *Expert Systems with Applications* (to appear).

Elbenani, B. and Ferland, J.A., 2012. Cell formation problem solved exactly with the Dinkelbach algorithm. *Publication CIRRELT*-2012-07, University of Montreal, Canada.

Farahani, M.H. and Hosseini, L., 2011. An ant colony optimization approach for the machine-part cell formation problem. *International Journal of Computational IntelligenceSystems*, 4, 486–496.

Ferland, J.A. and Costa D., 2001. Heuristic search methods for combinatorial programming problems. *Publication # 1193, Department of computer science and operation research*, University of Montreal, Canada.

Ghosh, T., Dan, P.K., Sengupta, S., Chattopadhyay, M., 2010. Genetic rule based techniques in cellular manufacturing (1992-2010): a systematic survey. *International Journal of Engineering Science and Technology*, 2, 198–215.

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

James, T.L., Brown, E.C., Keeling, K.B., 2007. A hybrid Grouping Genetic Algorithm for the cell formation problem. *Computers & Operations Research*, 34, 2059–2079.

Khoa, T., Ferland, J.A., Tien, D., 2011. A randomized local search algorithm for the machine-part cell formation problem. In *Advanced Intelligent Computing Technology and Applications-ICIC2011* (to appear).

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

Kirkpatrick, S., Gelatt, C.D. Jr, Vecchi, M.P., 1983. Optimization by simulated annealing. *Science*, 220, 671 – 680.

King, J.R. and Nakornchai, V., 1982. Machine-component group formation in group technology: review and extension. *International Journal of Production Research*, 20, 117–133.

Kumar, C. and Chandrasekharan, M., 1990. Grouping efficiency: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Research*, 28, 233–243.

Kumar, K.R. and Vannelli, A., 1987. Strategic subcontracting for efficient disaggregated manufacturing. *International Journal of Production Research*, 25, 1715–1728.

Kumar, K.R., Kusiak, A., Vannelli, A., 1986. Grouping of parts and components in flexible manufacturing systems. *European Journal of Operational Research*, 24, 387–397.

Kusiak, A. and Cho, M., 1992. Similarity coefficient algorithm for solving the group technology problem. *International Journal of Production Research*, 30, 2633–2646.

Kusiak, A. and Chow, W.S., 1987. Efficient solving of the group technology problem, *Journal of Manufacturing Systems*, 6, 117–124.

Luo, L., Tang, L., 2009. A hybrid approach of ordinal optimization and iterated local search for manufacturing cell formation. *International Journal of Advanced Manufacturing Technology*, 40, 362– 372.

Mahdavi, I., Paydar, M.M., Solimanpur, M., Heidarzade, A., 2009. Genetic algorithm approach for solving a cell formation problem in cellular manufacturing. *Expert Systems with Applications*, 36, 6598–6604.

McCormick, W.T., Schweitzer, P.J., White, T.W., 1972. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20, 993–1009.

Mosier, C.T. and Taube, L., 1985a. The facets of group technology and their impact on implementation. *OMEGA*, 13, 381–391.

Mosier, C. and Taube, L., 1985b. Weighted similarity measure heuristics for the group technology machine clustering problem, *OMEGA*, 13, 577–83.

Papaioannou, G. and Wilson, J.N., 2010. The evolution of cell formation problem methodologies based on recent studies (1997-2008): Review and directions for future research, *European Journal of Operational Research*, 206, 509–521.

Pisinger, D. and Ropke, S., 2007. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34, 2403 – 2435.

Rojas, W., Solar, M., Chacon, M, Ferland, J.A., 2004. An efficient genetic algorithm to solve the manufacturing cell formation problem. In *Adaptive Computing in Design and Manufacture VI,* I.C. Parmee (ed), Springer-Verlag, 173–184 .

Sarker, B. and Khan, M., 2001. A comparison of existing grouping efficiency measures and a new grouping efficiency measure. *IIE Transactions*, 33, 11–27.

Seifoddini, H., 1989. A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. *International Journal of Production Research*, 27, 1161–1165.

Seifoddini, H. and Wolfe, P.M., 1986. Application of the similarity coefficient method in group technology, *IIE Transactions*, 18, 271–277.

Srinivasan, G., Narendran, T.T., Mahadevan, B., 1990. An assignment model for the part-families problem in group technology, *International Journal of Production Research*,

28, 145–152.

Stanfel, L.E., 1985. Machine clustering for economic production. *Engineering Costs and Production Economics*, 9, 73–81.

Thanh, L.T., Ferland, J.A., Thuc, N.D. Nguyen, V.H., 2011.Simulated annealing method with different neighborhoods for solving the cell formation problem. *Proceedings FEC, ECTA 2011, Paris, France, 24 -26 Octobre, 2011 (Ed. A. Rosa, J. Kacprzyk, J. Filipe, A.D. Correia)*, 525 – 533.

Tuyen, D., Tien, D., and Ferland, J.A., 2011 "A Metaheuristic Approach for Cell Formation Problem", SoICT 2011, Hanoi, Vietnam, 13, 14 octobre 2011.

Tunnukij, T. and Hicks, C., 2009. An Enhanced Genetic Algorithm for solving the cell formation problem. *International Journal of Production Research*, 47, 1989–2007.

Waghodekar, P.H. and Sahu, S., 1984. Machine-component cell formation in group technology MACE. *International Journal of Production Research*, 22, 937–948.

Ying K.C., Lin S.W., Lu C.C. 2011. Cell formation using a simulated annealing algorithm with variable neighborhood. *European Journal of Industrial Engineering*, 5, 22–42.