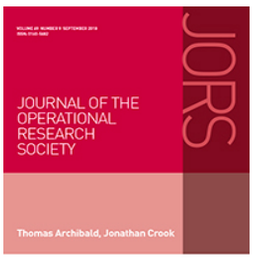

Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: https://www.tandfonline.com/loi/tjor20

GRASP with path relinking for the manufacturing cell formation problem considering part processing sequence

Juan A. Díaz & Dolores E. Luna

To cite this article: Juan A. Díaz & Dolores E. Luna (2018) GRASP with path relinking for the manufacturing cell formation problem considering part processing sequence, Journal of the Operational Research Society, 69:9, 1493-1511, DOI: 10.1080/01605682.2017.1404183

To link to this article: https://doi.org/10.1080/01605682.2017.1404183

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY, 2018
VOL. 69, NO. 9, 1493–1511
https://doi.org/10.1080/01605682.2017.1404183

GRASP with path relinking for the manufacturing cell formation problem considering part processing sequence

Juan A. Díaz[a] and Dolores E. Luna[b]

[a]Dpt. Actuaría Física y Matemáticas, Universidad de las Américas, Puebla, México; [b]Dpt. Ingeniería Industrial y Mecánica, Universidad de las Américas, Puebla, México

ABSTRACT

In this study, a hybrid heuristic is proposed for the Manufacturing Cell Formation Problem considering part processing sequence. The hybrid heuristic is based on a GRASP heuristic and then it uses a path relinking strategy on a set of elite solutions obtained by GRASP. The proposed GRASP heuristic constructs feasible solutions using a two-stage randomised greedy procedure, that are improved with a local search procedure. This procedure explores two neighbourhoods: machine reassignments and machine interchanges. The path relinking is used as an intensification strategy to improve the GRASP solutions. The performance of the proposed heuristic is evaluated with data-sets from the literature and a set of larger size instances randomly generated. The obtained results show that the GRASP heuristic provides optimal or near optimal solutions with very small computational burden. These results are further improved with a Path Relinking strategy. The most remarkable features of the proposed heuristic are its simplicity and its ease of implementation.

ARTICLE HISTORY
Received 21 October 2015
Accepted 6 November 2017

KEYWORDS
GRASP; path relinking; manufacturing cell formation

1. Introduction

One of the first steps in the implementation of modern manufacturing practices such as *Just in Time* and *Lean Manufacturing* is to modify plant layouts to reduce unnecessary movements, transportation, inventory levels of work in progress and waiting times. All these factors directly impact customer service and productivity. It is becoming increasingly popular to use cellular manufacturing to achieve these objectives. Because cellular manufacturing groups a set of machines and dedicates them to the production of a set of parts with similar manufacturing characteristics, a significant reduction in the time spent in setting up machines is expected. Reduction in set-up times will allow reducing batch sizes which will result in a reduction of work in progress. Since the machines belonging to a manufacturing cell will be physically grouped, material handling will also be reduced. The combination of reduced set-up times and reduced material handling should result in a flow time reduction which will allow improving customer service.

Implementing manufacturing cell systems requires several steps. Some of these steps are:

(1) Cell formation. The objective of this step is to identify sets of similar parts to form part families and to identify machine sets to form manufacturing cells, in order to reduce material handling among cells.
(2) Cell layout. This step deals with the physical layout of machines within cells to further improve material handling.
(3) Scheduling parts within the cell. This step is a production planning problem (Muruganandam, Prabhaharan, Asokan, & Baskaran, 2005).

The Machine Cell Formation Problem (MCFP) considers the following situation. Given a set of machines, a set of parts, and information regarding which parts need to be processed in each machine, it is required to partition the set of machines into machine-cells and the set of parts into part-families in order to reduce part movement among cells. Therefore, in the ideal case, each machine-cell can be viewed as an isolated job-shop system. However, in practice it could be economically prohibitive, since it would require to replicate machines that process parts assigned to different part-families.

The MCFP has been extensively studied in the literature. Papaioannou and Wilson (2010) present a literature review of this problem covering the years 1997–2008. They are focused on presenting both formulations and solution approaches for different versions of the problem. In particular, there is a vast literature for the case where the part-machine incidence matrix is binary and, a one in position k, j indicates that part k requires machine j for its process. Using this representation, to identify part families and manufacturing cells, the methods proposed in the literature seek to obtain a

CONTACT Dolores E. Luna dolorese.luna@udlap.mx

block diagonal matrix with the fewest possible number of ones outside the diagonal blocks. These ones represent intercellular movements, and therefore, measure the amount of material handling that must be done among cells. Many objective functions using different measures of efficiency and efficacy have been proposed over the years to measure solution quality. An efficacy measure is often used. This efficacy measure penalises both, intercellular movements, by counting the number of machines that parts must visit out of the machine-cell where they are assigned, and lack of part-families' homogeneity, by counting the number of times that parts do not need to be processed by all machines of the machine-cell where they are assigned. An excellent review of solution methods for this version of the problem is found in Gonçalves and Resende (2004). Wu, Chang, and Yeh (2009) propose a hybrid heuristic that combines a Boltzmann function and a mutation operator, Wu, Chung, and Chang (2010) propose a water flow like algorithm and Díaz, Luna, and Luna (2012) propose a GRASP algorithm. More recently, Noktehdan, Seyedhosseini, and Saidi-Mehrabad (2015) use a heuristic method denominated "league championship algorithm", that is used for grouping problems, and apply it to the MCFP. Martins, Pinheiro, Protti, and Ochi (2015) propose a hybrid metaheuristic approach that combines iterated local search and variable neighbourhood search, and Karoum and Elbenani (2016) describe a hybrid clonal heuristic.

The main disadvantage of representing the relationship between parts and machines with a binary matrix is that, unfortunately, it does not correctly represent the flow of parts among manufacturing cells. For example, suppose that a part k requires for its process a machine j that is not in the cell to which it is assigned. Depending on the order of processing, if machine j is required at the beginning or end of the process sequence, only one movement between cells would be made. In contrast, if machine j is required in any intermediate process operation, two movements between cells are required. The version that we study in this work, considers the order of processing to better estimate the material handling among manufacturing cells. This version of the problem has been less studied. In this case, the machine-part matrix is integer, and it represents the sequential order in which machines are used to produce the different parts. In this sense, more emphasis is made on the number of times a part is moved from one cell to another. Therefore, this version of the problem better reflects the amount of material handling required.

The Machine Cell Formation Problem (MCFP) considering part processing sequence considers the following situation. Given a set of machines $M = \{1, \ldots, n\}$, and a set of parts, $P = \{1, \ldots, m\}$, an instance of the MCFP is usually represented by means of a part-machines matrix $(a_{kj})_{k \in P, j \in M}$, where $a_{kj} = s$ means that the s^{th} operation of part k is performed on machine j. Null values on the part-machines matrix mean that machine j is not required to process part k (see Figure 1). The objective is to find a partition of the machines set into machine-cells and a partition of the parts set into part-families, in order to minimise intercellular movements. Once a partition of the machines into machine-cells is at hand, part-families can be easily identified by assigning each part to the machine-cell, where more consecutive machines are needed to process that part (ties are arbitrarily broken). Sofianopoulou (1997) presents an integer LP model to assign parts to machine-cells that can be solved by any commercial software, since the constraints coefficient matrix of the model is totally unimodular. Usually, a parameter is given to specify the maximum number of machines per machine-cell. In Figure 2, a feasible solution for the problem instance given in Figure 1 is shown, where the maximum number of machines per machine-cell is equal to 6. As can be observed in Figure 2, the diagonal blocks of the diagonal block matrix define machine-cells and part-families. For example, let's observe part 19 which requires 8 operations and it is assigned to machine cell containing machines 3, 4, 8, 11, 12, and 14. Operations 1 through 6 are performed on that machine-cell. Then, to perform the 7^{th} operation required by part 19, it has to be transported to the machine-cell which contains machines 1, 6, 10, and 15 and, finally, the 8^{th} operation of part 19 is performed on the machine-cell containing machines 2, 5, 7, 9, and 13. As can be observed in Figure 2 the objective function value of the solution is 17. This means that it is required to move 17 parts from one machine-cell to another.

Although several exact procedures have been proposed for MCFP considering part processing sequence (see, for example, Faigle, Schrader, & Suletzki, 1987; Sorensen, 1995; Spiliopoulos & Sofianopoulou, 2008) it is highly likely that there exists even moderate or medium size problem instances where exact solution cannot be obtained with reasonable computational effort, since the problem is known to be NP-hard (see, for example, Sorensen, 1995). Also, several heuristic procedures have been proposed for the MCFP considering part processing sequence. A simulated annealing algorithm was proposed in Sofianopoulou (1997). In this approach, neighbourhood solutions are generated by reassigning a randomly chosen machine i, from the machine-cell where it is assigned, to another machine-cell where a randomly selected machine j is assigned. Initial solutions are obtained by generating one machine-cell at a time. The machines are assigned to the cell that is being generated by randomly selecting a machine from the set of machines that have not been yet assigned, until the maximum number of machines allowed per cell is reached. A feasible initial solution is obtained when all machines have been

Machines

Parts	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		2			3		1		4				5	6	
2	1					3				4					2
3				2		4						1		3	
4		2			3		1		4				5		
5		2			3				4	1					
6	1	4				3				5					2
7		2			3		1	7	4				5		6
8	1			5		2				4					3
9			6	3				2			5	1		4	
10		5		2				4			6	1		3	
11				2				5			4	1		3	
12			6	2				4			5	1		3	
13	1					3									2
14	3	2					1		4				5		
15	3					2				4					1
16	1					3				4					2
17		2			3		1						4		
18	1					4				3					2
19		8	6	2				4			5	1		3	7
20				1				4			5	2		3	
21			1	2				4			5			3	
22		2			3		1		4						
23				2				4	5		6	1		3	
24			6	2	7			4			5	1		3	
25			5	2							4	1		3	

Figure 1. Incidence matrix.

Machines

Parts	3	4	8	11	12	14	2	5	7	9	13	1	6	10	15
3		2			1	3							4		
9	6	3	2	5	1	4									
10		2	4	6	1	3	5								
11		2	5	4	1	3									
12	6	2	4	5	1	3									
19	6	2	4	5	1	3	8								7
20		1	4	5	2	3									
21	1	2	4	5		3									
23		2	4	6	1	3				5					
24	6	2	4	5	1	3		7							
25	5	2		4	1	3									
1						6	2	3	1	4	5				
4							2	3	1	4	5				
5							2	3		4				1	
7			7				2	3	1	4	5				6
14							2		1	4	5	3			
17							2	3	1		4				
22							2	3	1	4					
2												1	3	4	2
6							4					1	3	5	2
8		5										1	2	4	3
13												1	3		2
15												3	2	4	1
16												1	3	4	2
18												1	4	3	2

Figure 2. Machine-cells and part-families.

assigned to one machine-cell. Spiliopoulos and Sofianopoulou (2003) propose a tabu search algorithm. In this algorithm they explore two different neighbourhoods: (1) reassignment of one machine from one cell to another and (2) interchange of two machines assigned to different cells. They allow reassignment moves that violate the maximum cell size allowed as a diversification strategy, since reassignment moves change solution structure (number of machines in each cell). The proposed tabu search algorithm uses short and long term memory to obtain a proper balance between diversification and intensification of the search procedure. An Ant Colony System is proposed in Spiliopoulos and Sofianopoulou (2008), where solutions are built by artificial ants based on a priori available information and the history of the search procedure. Also, the procedure uses tight bounds to guide and accelerate the search, making the procedure more robust, since less parameters need to be tuned. Saeedi, Solimanpur, Mahdavi, and Javadian (2010) propose three different heuristic procedures for the MCFP considering part processing sequence: a genetic algorithm, an ant colony system and a simulated annealing algorithm. They consider a different formulation where the objective function penalises both intercellular movements and voids. Díaz, Luna, and Zetina (2013) compare three different algorithms for this problem, a GRASP algorithm, a reactive GRASP algorithm and a hybrid algorithm that uses a tabu search instead of the local search used in the reactive GRASP algorithm. The results for the three algorithms are good, but the hybrid algorithm obtains the best results of all. More recently, (Raja & Anbumalar, 2016) study a version of the MCFP related to the one studied in this paper. They combine the cell formation problem with sequence data and the cell layout problem. They test their methodology with two problems from the literature.

In this work, we propose a hybrid heuristic for the Manufacturing Cell Formation Problem considering part processing sequence. The proposed algorithm uses a GRASP methodology combined with a Path Relinking strategy. GRASP was first introduced in Feo and Resende (1995). It is an iterative procedure. At each iteration a *Construction phase* is used to obtain an initial solution that is then improved with a local search procedure that usually explores one or more neighbourhoods. Path Relinking is an intensification strategy that explores trajectories connecting elite solutions obtained by heuristic methods such as GRASP, tabu search and scatter search (Glover, 1996). The performance of the proposed hybrid heuristic is evaluated using data-sets from the literature and a set of larger size instances ran-

domly generated. The results show that the algorithm obtains optimal or near optimal solutions with very small computational burden.

This paper is structured as follows. In Section 2, the problem is defined and formulated. In Section 3, the constructive and improvement phase of the proposed GRASP heuristic, as well as the Path Relinking strategy are described in detail. Section 4 presents the results obtained with the proposed heuristic for different datasets from the literature and a set of larger size instances randomly generated and finally, in Section 5 we draw some conclusions.

2. Machine cell formation problem considering part processing sequence

The machine cell formation problem is a particular case of the Graph Partitioning Problem that is an NP-hard problem (see for example Sorensen, 1995).

As in Sofianopoulou (1997), given a set of machines $M = \{1, \ldots, n\}$, a set of parts, $P = \{1, \ldots, m\}$, and a part-machines matrix $(a_{kj})_{k\in P, j\in M}$, let define $w_{ij} \forall i, j \in M, i < j$, as the number of parts that are sequentially processed in machines i and j, that is,

$$w_{ij} = \sum_{k\in P} f_{ijk} \tag{1}$$

where

$$f_{ijk} = \begin{cases} 1 \text{ if } a_{ki} \times a_{kj} \neq 0 \text{ and } |a_{ki} - a_{kj}| = 1 \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

is a parameter that is set to one when machines i and j process sequentially part k. Figure 3 illustrates these calculations using the instance given in Figure 1. For example, $w_{6,10} = 3$ because parts 2, 16, and 18 use machines 6 and 10 sequentially.

We define the following decision variables,

$$x_{it} = \begin{cases} 1 \text{ if machine } i \text{ is allocated to cell } t \\ 0 \text{ otherwise} \end{cases}, \quad i \in M, t = 1, \ldots, T \tag{3}$$

and

$$z_{ijt} = \begin{cases} 1 \text{ if machines } i \text{ and } j \text{ are allocated to cell } t \\ 0 \text{ otherwise} \end{cases}, \quad i, j \in M : i < j, t = 1, \ldots, T \tag{4}$$

where $T = \lceil \frac{n}{S} \rceil$ is the number of machine-cells to determine and S is the maximum cell size allowed.

Machines

Machines	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		1	0	0	0	2	0	0	1	1	0	0	0	0	5
2			0	0	6	1	6	1	0	2	1	0	0	0	1
3				1	1	0	0	0	0	0	5	0	0	0	1
4					0	0	0	1	0	1	0	9	0	10	0
5						0	0	0	5	0	0	0	1	0	0
6							0	0	0	3	0	0	0	1	6
7								0	0	0	0	0	0	0	0
8									1	0	6	1	0	7	1
9										0	1	0	4	0	0
10											0	0	0	0	2
11												0	0	3	0
12													0	1	0
13														1	1
14															0
15															

Figure 3. Flow between machines matrix.

The problem can be formulated in the following way

$$\text{maximise } flow = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \sum_{t=1}^{T} z_{ijt} \tag{5}$$

$$\text{subject to } \sum_{t=1}^{T} x_{it} = 1 \quad i \in M \tag{6}$$

$$\sum_{i\in M} x_{it} \leq S \quad t = 1, \ldots, T \tag{7}$$

$$z_{ijt} \leq x_{it} \quad i, j \in M : i < j, t = 1, \ldots, T \tag{8}$$

$$z_{ijt} \leq x_{jt} \quad i, j \in M : i < j, t = 1, \ldots, T \tag{9}$$

$$z_{ijt} \geq x_{it} + x_{jt} - 1 \quad i, j \in M : i < j, \; t = 1, \ldots, T \tag{10}$$

$$x_{it} \in \{0, 1\} \quad i \in M, t = 1, \ldots, T \tag{11}$$

$$z_{ijt} \in \{0, 1\} \quad i, j \in M : i < j, \; t = 1, \ldots, T \tag{12}$$

The objective function (5) is to maximise intracell part flows. This is equivalent to minimise intercell part flows since it can be determined with the following expression: $\sum_{i,j\in M:i<j} w_{ij} - flow$. Constrains (6) ensure that each machine is allocated to exactly one machine-cell. Constrains (7) guarantee that at most S machines are allocated to each machine-cell. Constraints (8), (9) and (10) are used to linearise binary products $z_{ijt} = x_{it}x_{jt}$ for $i, j \in M : i < j, t = 1, \ldots, T$.

As can be observed in the above formulation of the problem, several equivalent symmetric solutions can be obtained by re-indexing machine-cells. This may cause branch and bound methods to be inefficient, as mentioned in Sherali and Smith (2001), since the inherent symmetry in the model degrades the performance of enumeration algorithms. The problem can

be reformulated to reduce to some extent the degradation caused by the symmetry of the above formulation (see formulation below). This reduction is achieved by fixing the allocation of an arbitrary machine, for example machine 1, to some machine-cell, for example, machine-cell 1, (see constraint (14)) and by ensuring that the number of machines in a machine-cell does not increase with the machine-cell index (see constraints (17)).

$$\text{maximise } u = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \sum_{t=1}^{T} z_{ijt} \tag{13}$$

$$\text{subject to } x_{11} = 1 \tag{14}$$

$$\sum_{t=1}^{T} x_{it} = 1 \quad i \in M \tag{15}$$

$$\sum_{i \in M} x_{it} \leq S \quad t = 1, \ldots, T \tag{16}$$

$$\sum_{i \in M} \left(x_{it} - x_{i,t+1}\right) \geq 0 \quad t = 2, \ldots, n \tag{17}$$

$$z_{ijt} \leq x_{it} \quad i, j \in M : i < j, t = 1, \ldots, T \tag{18}$$

$$z_{ijt} \leq x_{jt} \quad i, j \in M : i < j, t = 1, \ldots, T \tag{19}$$

$$z_{ijt} \geq x_{it} + x_{jt} - 1 \quad i, j \in M : i < j, \; t = 1, \ldots, T \tag{20}$$

$$x_{it} \in \{0, 1\} \quad i \in M, t = 1, \ldots, T \tag{21}$$

$$z_{ijt} \in \{0, 1\} \quad i, j \in M : i < j, \; t = 1, \ldots, T \tag{22}$$

3. Hybrid algorithm

In this section, we describe the proposed hybrid heuristic for the MCFP considering part processing sequence. The heuristic is formed by three elements, a two stage randomised adaptive greedy procedure that generates initial solutions, a local search procedure for improving the initial solutions and a path relinking procedure to further improve the solutions. In Section 3.1, we describe the constructive phase of the GRASP algorithm, in Section 3.2 we describe the local search procedure used to improve the solutions generated by the constructive procedure and, in Section 3.3 we describe the path relinking strategy and the hybrid algorithm.

3.1. Greedy procedure

Solutions of the MCFP considering part processing sequence are represented by machine set partitions. Let $G = (M, E)$ be a complete graph where M is the machines set and E is an edges set. Associated to each $e = \{i, j\} \in E$ there is a weight w_{ij} that, as explained above, represents the number of parts that are successively processed by machines $i, j \in M$. Given a set $C = \{C_1, C_2, \ldots, C_T\}$ of T machine subsets such that $\bigcup_{i=1}^{T} C_i = M$ and $C_i \cap C_j = \emptyset, \forall i, j \in \{1, \ldots, T\}, i \neq j$, let $\sigma : M \rightarrow \{1, 2, \ldots, T\}$, where $\sigma(i) = j$ if machine i is assigned to cell C_j. Therefore, the objective function value for a given solution C is obtained by the sum of edge weights whose ends belong to different machine subsets,

$$z(C) = \sum_{e=\{i,j\} \in E : \sigma(i) \neq \sigma(j)} w_{ij}.$$

The constructive phase of the algorithm obtains initial feasible solutions using a two-stage procedure, that are then improved using a local search phase which explores machine reassignment and machine interchange neighbourhoods. In the first stage of the constructive phase, an edge is selected from a restricted candidate list RCL^1 of edges and both ends of that edge are added to initialise a new machine set, while in the second stage additional machines are added to this machine set using another restricted candidate list RCL^2 of unassigned machines. We now describe both stages of the greedy procedure in detail.

The greedy procedure is a two stage iterative approach. At iteration t, a new machine cell, C_t is formed. The purpose of the first stage of the greedy procedure is to create and initialise a new machine-cell. Therefore, it is desirable to initialise the cell at iteration t with two machines that process similar parts in successive steps of their manufacturing process. For this reason, within the first step of the constructive phase, we randomly select one edge from the RCL^1. For each candidate edge, we define a greedy value that measures the benefit of including both ends of the edge in the same subset of nodes. Since each node represents a machine, machine pairs with higher values of w_{ij} would tend to be grouped together. Let $w_{\max} = \max_{\{i,j\} \in E(M \setminus A)} \{w_{ij}\}$, $w_{\min} = \min_{\{i,j\} \in E(M \setminus A)} \{w_{ij}\}$, where $E(M \setminus A) = \{\{i, j\} \in E : i, j \in M \setminus A\}$ and A is the set of nodes that have already been assigned to a machine subset. We select a pair of machines to initialise a new subset from RCL^1, that contains the candidate edges set, that is, those edges whose $w_{ij} \geq T_1$, where $T_1 = w_{\max} - \alpha(w_{\max} - w_{\min})$ is a threshold value and $\alpha \in [0, 1]$ is a parameter that controls the degree of greediness or randomness of the greedy procedure.

Once a new machine-cell is created and initialised, during the second stage of the constructive procedure more machines are added to that machine-cell. During this stage, it is desirable to add machines that also process similar parts, in successive steps of their manufacturing process, to those that have already been added to that machine-cell. For this purpose, additional unassigned machines are selected from a restricted candidate list RCL^2 and added to C_t.

Different greedy criteria can be used for membership of a node $i \in M \setminus A$ to RCL^2. One of them could be

the sum of the w_{ij} values of the edges whose end points are in the cut-set $\delta(i, C_t)$, where $\delta(U, V) = \{\{i, j\} \in E : i \in U, j \in V\}$. That is, the *attractiveness measure* $a_{i,C_t} = \sum_{\{i,j\}\in\delta(i,C_t)} w_{ij}$ of node i to the partition set C_t. Let $a_{\max} = \max_{i\in M\setminus A}\{a_{i,C_t}\}$, $a_{\min} = \min_{i\in M\setminus A}\{a_{i,C_t}\}$. We randomly select a node to be included in the node partition C_t from RCL^2, that contains the candidate nodes set, that is, those nodes whose $a_{ij} \geq T_2$, where $T_2 = a_{\max} - \alpha(a_{\max} - a_{\min})$ is a threshold value and $\alpha \in [0, 1]$ is a parameter that controls the degree of greediness or randomness of the greedy procedure. This procedure is repeated until $|C_t| = S$ or $M \setminus A$ is empty. Then, the first stage of the greedy procedure is reinitiated if $M \setminus A$ is not empty. Otherwise, the greedy procedure is stopped, since an initial feasible solution is at hand.

The randomised greedy procedure of the proposed GRASP algorithm is depicted in Algorithm 3.1.

Algorithm 3.1 Randomised greedy procedure of hybrid Algorithm

$A \leftarrow \emptyset$
$t \leftarrow 0$
repeat
  $w_{\max} \leftarrow \max_{\{i,j\}\in E(M\setminus A)} \{w_{ij}$
  $w_{\min} \leftarrow \min_{\{i,j\}\in E(M\setminus A)} \{w_{ij}\}$
  $T_1 \leftarrow w_{\max} - \alpha(w_{\max} - w_{\min})$
  $RCL^1 \leftarrow \{\{i,j\} \in E(M \setminus A) : w_{ij} \geq T_1\}$
  Randomly select an edge $e = \{i,j\}$ from RCL^1
  $t \leftarrow t + 1$
  $C_t \leftarrow \{i,j\}$
  $A \leftarrow A \cup \{i,j\}$
  $Stop \leftarrow$ **false**
  repeat
    if $|C_t| = S$ or $A = M$ **then**
      $Stop \leftarrow$ **true**
    else
      $a_{\max} \leftarrow \max_{i\in M\setminus A} \; a_{i,C_t}$
      $a_{\min} \leftarrow \min_{i\in M\setminus A} \{a_{i,C_t}\}$
      $T_2 \leftarrow a_{\max} - \alpha(a_{\max} - a_{\min})$
      $RCL^2 \leftarrow \{i \in M \setminus A : a_{i,C_t} \geq T_2\}$
      Randomly select a node v from RCL^2
      $C_t \leftarrow C_t \cup \{v\}$
      $A \leftarrow A \cup \{v\}$
    end if
  until $Stop$
until $A = M$

3.2. Local search procedure

Once an initial solution is at hand using the constructive phase procedure of the proposed GRASP algorithm, it is improved by means of a local search procedure that explores two different neighbourhoods: (1) interchange of two machines assigned to different cells, and (2) reassignment of one machine from one cell to another. Let $C = \{C_1, C_2, \ldots, C_T\}$ be an initial solution for the MCFP given by a partition of the machines set M into T subsets. Therefore, the machine interchange neighbourhood is given by all solution that can be reached from the current one by selecting two machines assigned to two different subsets and swapping them,

$$N_1(C) = \Big\{ C' = \{C'_1, \ldots, C'_i, \ldots, C'_j, \ldots, C'_T\} : \exists i, j \in \{1, \ldots, T\}, i \neq j, s \in C_i, t \in C_j, C'_i = C_i \setminus \{s\} \cup \{t\}, C'_j = C_j \setminus \{t\} \cup \{s\}, C'_t = C_t, \forall t \neq i, j \Big\},$$

and the reassignment neighbourhood is given by all solutions that can be obtained from the current solution, by reassigning one machine from one subset to another,

$$N_2(C) = \Big\{ C' = \{C'_1, \ldots, C'_i, \ldots, C'_j, \ldots, C'_T\} : \exists i, j \in \{1, \ldots, T\}, i \neq j \text{ and } |C_j| < S, s \in C_i, C'_i = C_i \setminus \{s\}, C'_j = C_j \cup \{s\}, C'_t = C_t, \forall t \neq i, j \Big\}.$$

In both neighbourhoods *best improvement* strategy is used. The algorithm performs local search in neighbourhood $N_1(C)$ and, once a local optimum with respect to $N_1(C)$ is reached, local search in $N_2(C)$ is applied. The local search phase of the GRASP algorithm is depicted in Algorithm 3.2.

3.3. Path relinking procedure

Path relinking was first proposed in Glover (1996). It is an intensification strategy that explores trajectories connecting elite solutions previously obtained by a heuristic method. This set of elite solutions is frequently called a reference set. The strategy works with two solutions at a time, the starting and the guiding solution. A path of intermediate solutions connecting the starting and the guiding solutions is generated from the starting solution by selecting moves in a neighbourhood space, progressively incorporating the attributes of the guiding solution.

GRASP is a memoryless heuristic; the addition of path relinking provides a memory mechanism to enhance the results obtained by the heuristic. Laguna and Marti (1999) were the first in using path relinking within a GRASP procedure. Resendel and Ribeiro (2005) present different approaches to incorporate path relinking in a GRASP heuristic. Their results suggest that when the two initial solutions are very similar, the path relinking strategy procedure usually does not improve the solution. Therefore, it is convenient to use a measure of the similarity of the solutions in order to apply the path relinking strategy only when the solutions are sufficiently different.

The relinking process implemented in this work uses the edges $e \in E$ of the guiding solution that does not

Algorithm 3.2 Local search procedure of hybrid algorithm

Let $C = \{C_1, C_2, \ldots, C_T\}$ be the current solution, $C^\star$ the incumbent and $z(C)$ the objective function value.
$C^\star \leftarrow C$
Stop ← **false**
repeat
  Let $C' \in \arg\min_{x \in N_1(C)}\{z(x)\}$
  if $z(C') < z(C^\star)$ **then**
    $C^\star \leftarrow C'$
  end if
  if $C^\star \neq C$ **then**
    $C \leftarrow C^\star$
  else
    Stop ← **true**
  end if
until *Stop*
Stop ← **false**
repeat
  Let $C' \in \arg\min_{x \in N_2(C)}\{z(x)\}$
  if $z(C') < z(C^\star)$ **then**
    $C^\star \leftarrow C'$
  end if
  if $C^\star \neq C$ **then**
    $C \leftarrow C^\star$
  else
    Stop ← **true**
  end if
until *Stop*

belong to the cut-set, that is, $\{e \in E : e \notin \delta_{guide}(C_1, C_2, \ldots, C_T)\}$, where $\delta_{guide}(C_1, C_2, \ldots, C_T) = \{\{i,j\} \in E : i \in C_s, j \in C_t, s, t, \in \{1, \ldots, T\}, s \neq t\}$. These edges represent the existing flow among machines that are together in a cell in the guiding solution. The edges are sorted in non-increasing order with respect to their weights w_{ij} and stored in a list L. The path relinking is an iterative procedure. At each iteration a new edge $\{i^\star, j^\star\}$ is selected and processed. To process an edge means to fix it. That is, this edge must not belong to the cut-set in any solution generated in next iterations of the procedure. Therefore, the edge $\{i^\star, j^\star\}$ is added to the set *Fixed* which contains the edges that have already being processed. The edge $\{i^\star, j^\star\}$ is processed in the following way. If $\sigma(i^\star) = \sigma(j^\star)$ in the intermediate solution, the edge is added to the *Fixed* set. Otherwise, we compute $\Delta(i^\star, C_{\sigma(j^\star)})$ and $\Delta(j^\star, C_{\sigma(i^\star)})$, where $\Delta(i, C_t)$ is the change in the objective function value of moving node i to cell C_t. If $\Delta(i^\star, C_{\sigma(j^\star)}) \leq \Delta(j^\star, C_{\sigma(i^\star)})$ node $i^\star$ is moved to cell $C_{\sigma(j^\star)}$ and we make $C^\star = C_{\sigma(j^\star)}$. Otherwise, node $j^\star$ is moved to cell $C_{\sigma(i^\star)}$ and we make $C^\star = C_{\sigma(i^\star)}$. In both cases edge $\{i^\star, j^\star\}$ is added to the *Fixed* set. Every time a reassignment is made the feasibility of the solution with respect to the maximum cell size constraint must be checked. When the feasibility is destroyed, the cell $C^\star$ that violates the maximum size is splitted in the following way. All nodes in the set $\{s \in C^\star : \{i^\star, s\} \in \mathit{Fixed} \vee \{j^\star, s\} \in \mathit{Fixed}\}$ are left in C_t and a new cell is formed with all the remaining nodes. If the objective function value of any intermediate solution is better than the incumbent, it is updated. The relinking process ends when the solution is equal to the guiding solution.

In order to measure the *similarity* of two solutions, the number of edges that belong to the cut-set of one solution and does not belong to the cut-set of the other solution are counted. This number is divided by the total number of edges $|E|$.

In the heuristic proposed in this work, a reference set of elite solutions is first formed using the GRASP heuristic. After this, in each GRASP iteration, after the local search procedure, the path relinking strategy is applied using as guiding solution all solutions in the reference set with a *similarity* measure greater than a threshold. If the new solution is better than the worst solution of the reference set, the set is updated. The algorithm finishes when the stop criterion is met.

4. Computational results

In order to evaluate the performance of the proposed hybrid heuristic for the Manufacturing Cell Formation Problem considering part processing sequence, it is tested with six different data-sets from the literature and a data-set of larger size random generated instances:

(1) The data-set 1 used in Sofianopoulou (1993), Sofianopoulou (1997) and in Spiliopoulos and Sofianopoulou (2003), Spiliopoulos and Sofianopoulou (2008), that contains 10 cost matrices with 16 machines and, for each cost matrix, 8 different values (that range from 5 to 12) for the maximum cell size parameter are used, making a total of 80 problem instances.
(2) The data-set 2 used in Sofianopoulou (1993), Sofianopoulou (1997) and in Spiliopoulos and Sofianopoulou (2003), Spiliopoulos and Sofianopoulou (2008), that contains 10 cost matrices with 20 machines and, for each cost matrix, 6 different values (that range from 6 to 10) for the maximum cell size parameter are used, making a total of 50 problem instances.
(3) The data-set 3 used in Sofianopoulou (1993), Sofianopoulou (1997) and in Spiliopoulos and Sofianopoulou (2003), Spiliopoulos and Sofianopoulou, (2008), that contains 7 cost matrices with 25 machines and, for each cost matrix, 5 different values (that range from 8 to 12) for the maximum cell size parameter are used, making a total of 35 problem instances.
(4) The data-set 4 used in Sofianopoulou (1993), Sofianopoulou (1997) and in Spiliopoulos and Sofianopoulou (2003), Spiliopoulos and Sofianopoulou

(2008), that contains 9 cost matrices with 30 machines and, for each cost matrix, 5 different values (that range from 8 to 12) for the maximum cell size parameter are used, making a total of 45 problem instances.

(5) The data-set 5 consisting of graph partitioning problem instances used in Spiliopoulos and Sofianopoulou (2003), Spiliopoulos and Sofianopoulou (2008) and in Sorensen (1999), that contains 10 cost matrices where the number of nodes ranges from 35 to 40 and, for each cost matrix, 4 different values (4, 5, 7, and 10) for the maximum cell size parameter are used, making a total of 40 problem instances.

(6) The data-set 6 consisting of graph partitioning problem instances used in Spiliopoulos and Sofianopoulou (2003), Spiliopoulos and Sofianopoulou (2008) and in Sorensen (1999), that contains 5 cost matrices with 50 nodes and, for each cost matrix, 3 different values (4, 7, and 10) for the maximum cell size parameter are used, making a total of 15 problem instances.

(7) The data-set 7 consists of larger size random generated instances of the problem. Instances in this set were generated using the procedure described in Sofianopoulou (1997). This set contains 6 cost matrices with 60 machines and 6 cost matrices with 80 machines. For cost matrices with 60 machines, three different values were used for the number of parts, 200, 400, and 1000 parts. For cost matrices with 80 machines, three different values were used for the number of parts, 300, 600, and 1500 parts. For all instances of this set 3 different values (8, 10, and 12) for the maximum cell size parameter are used.

Optimal solutions are known for all problem instances from data-sets 1 to 4. For the data-set 5, optimal solutions are known for 10 out of the 40 problem instances, for the data-set 6, optimal solutions are known for 5 out of the 15 problem instances and for data-set 7 no optimal solution is known for any of the 36 problem instances.

To have an idea of the difficulty of solving data-set instances from the literature, we use the commercial optimisation software FICO Xpress Optimisation Suite to solve the mathematical programming formulation presented in Section 2 for all test instances, specifying a cpu time limit of 7200 s. A summary of the results obtained is presented in Table 1. The information presented in this table is as follows. The first column shows the number of the data-set, the second column depicts the number of instances of the problem within the data-set, the third column shows the number of instances in which the best upper bound obtained within the cpu time limit specified is the optimal solution or the best known solution. Finally, the fourth and fifth columns show, respectively, the average percentage gap of the best upper bound with respect to the best lower bound, and the average percentage deviation of the best upper bound found with the solver with respect to optimal or best known solution. These values are computed as follows for each data-set,

Table 1. Results obtained with FICO Xpress optimisation suite.

Data-set	Instances	Opt./ best	Average percentage gap (%)	Average percentage deviation to opt./best (%)
1	80	80	0.00	0.00
2	50	50	0.00	0.00
3	35	12	13.84	0.48
4	45	2	48.65	1.60
5	40	6	165.53	3.68
6	15	1	456.17	6.13

$$\text{Avg. percentage gap}_{set_j} = \frac{\sum_{i=1}^{|set_j|} \frac{(UB_i - LB_i)}{LB_i} * 100}{|set_j|}$$

$$\text{Avg. percentage deviation to opt./best}_{set_j} = \frac{\sum_{i=1}^{|set_j|} \frac{(UB_i - opt./best_i)}{opt./best_i} * 100}{|set_j|}$$

where j is the index of data-set j, and UB_i and LB_i are, respectively, the best upper and lower bounds found by the solver within the cpu time limit specified for the i^{th} instance of data-set j.

We can observe that the best upper bound found is the same as the optimal or best known solution only for 151 out of 265 instances. For data-sets 1 and 2, the solver is able to solve all instances to optimality. For data-set 3, the solver is able to solve 10 out of 35 instances to optimality and for 12 out of 35 instances, the best upper bound found is the optimal solution. For data-set 4, the solver is not able to optimally solve any of the instances, and only for 2 out of 45 instances, the best upper bound found is the optimal solution. Only 3 out of 40 instances are solved to optimality for data-set 5, and in 6 out of 40 instances the best upper bound coincide with the optimal or the best known solution. Finally, none of instances of data-set 6 is solved to optimality an only for 1 out of 15 instances the best upper bound is the same as the optimal or best known solution. From Table 1, we can only observe that the average percentage gap tends to increase considerably with the size (number of machines) of the instances. Although the optimisation software is capable of optimally solving all instances of data-sets 1 and 2, containing instances with 16 and 20 machines, respectively, for data-sets 3, 4, 5, and 6 the average percentage deviation increases considerably as the number of machines in the data instances increases. With respect to the best feasible solutions found by the commercial software within the cpu time limit specified, it is observed that, with the exception of data-sets

1 and 2, where all instances are solved to optimality, the average percentage deviation of best solutions found, with respect to the optimal or best known solutions, are 0.48%, 1.60%, 3.68 and 6.13% for data-sets 3, 4, 5, and 6, respectively. Again we can observe that these deviations tend to increase with the size of the instances and that the hybrid GRASP with path relinking approach presented in this study provides very good feasible solutions with a small computational burden. It is worth noting that the instances of data-sets 3, 4, 5, and 6 are much more difficult to solve and that the commercial solver could only find the optimal solution or the best known solution within the cpu time limit specified, of 7200 s, in 21 out of 135 instances of these data-sets.

The proposed heuristic was coded in C language and was run on a PC with an Intel Core i7-5500U, 2.40 GHz, processor. We set the stop criterion to 500 iterations without improvement.

An initial experimentation is carried out to tune up α parameter. For this purpose, each problem instance is run with a fixed α value 10 times. We consider five different values for α: 0.10, 0.20, 0.30, 0.40, and 0.50. For each data-set we compute the average deviation of the average solution value with respect to the optimal solution (or the best known solution value for those instances where the optimal is not known) and the number of problem instances where the optimal or best known solution is found at least in one of the 10 runs. In Table 2, we can observe that depending on the data-set, the proposed algorithm performs better with different values of α. It is an indication that solution quality depends strongly on choosing the appropriate α value for each cost matrix. Therefore, we perform another series of experiments where the α value ranges from 0.05 to 0.50 with increments of 0.05 each 5 iterations. As can be observed in Table 2, except for data-set six, the GRASP algorithm is more robust when the α value is fixed to 0.4. Not only the average deviation is smaller but also optimal or best known solutions are obtained for all but three instances.

In order to evaluate the improvement of using the path relinking strategy, another set of tests was carried on using the same seeds that those used for the GRASP heuristic. Tables 3 and 4 show the results, with greater detail, for the experiment using the GRASP heuristic and the GRASP heuristic using the path relinking strategy, respectively. In both heuristics the α value is set equal to 0.4 and the path relinking uses a similarity of 0.1. The information shown in the two tables is as follow. The first two columns show the data-set number and the number of instances on the corresponding data-set. The third column under the heading "Opt./best", shows the number of problem instances where the optimal or the best known solution is found. The next three columns show the average percentage deviation of the best, average and worst solution value with respect to the optimal or the best known solution, respectively. Finally, the last two columns show the average number of iterations needed to obtain the best solution value and the cpu time in seconds. As can be observed in Table 3 for the first four data-sets, the GRASP algorithm always gets the optimal solution on the 10 runs. Additionally, the average number of iterations to get the optimal solution is never greater than 24. What is more, never were needed more than 148 iterations to get the optimal solution for problem instances of data-sets 1 through 4. Also, for data-set 5 all optimal or best known solutions were obtained at least in one out of the 10 runs. Therefore, in order to evaluate the improvement of using the path relinking strategy, we need to analyse the results for data-set 6. As we can observe in Tables 3 and 4, the path relinking strategy improves the results for data-set 6. For this data-set, the GRASP heuristic only obtained 13 of the 15 optimal or best known solutions in at least one out of the ten runs, while using the path relinking strategy all but one optimal or best known solutions were obtained. Also, the average percentage deviation of the best, average and worst solution values with respect to the optimal or the best known solutions were reduced for this data-set. It is also important to highlight that this improvement was obtained with a very small increase in the average CPU time. Table 4 shows the results for data-set 7, where we observe that for instances of larger size, the algorithm always obtains the best known solutions in at least one out of the 10 runs.

Table 4 also shows that for the GRASP algorithm with the path relinking strategy, the average number of iterations to get the optimal or best known solutions was less than 157 for data-set 5 and 325 for data-set 6 and the number of iterations needed to get the optimal solution never exceed 482 and 483 iterations for data-sets 5 and 6, respectively. For data-set 7 the average number of iterations to get the best known solution is always less than 320. Also, the average percentage deviation of the average solution value with respect to the optimal or best known values is 0.017% and 0.158%, respectively, for data-sets 5 and 6. For data-set 7, only in three out of the 36 instances, the algorithm does not always obtain the best known solution and in these cases the average percentage deviation of the worst solution value with respect to the best known values is almost zero. This clearly shows the robustness of the proposed algorithm. What is more, the average percentage deviation of the worst solution with respect to the optimal or best known values never exceed 0.094% and 0.310% for data-sets 5 and 6, respectively. Average CPU time never exceeds 3 s for data-sets 1–6 and never exceeds 12 s for data-set 7. This shows that the required computation effort is extremely small. Detailed results for all problem instances are shown in Tables A1–A8 of

Table 2. Results with different α values.

Data-set	α value	Average percentage deviation with respect to optimal or best known solution (%)	Number of problem instances where optimal solutions where found at least in one of the 10 runs
Data-set 1	0.10	0.142	76
	0.20	0.110	77
	0.30	0.057	79
	0.40	0.000	80
	0.50	0.073	80
	varies from 0.05 to 0.50	0.000	80
Data-set 2	0.10	0.211	31
	0.20	0.006	49
	0.30	0.000	50
	0.40	0.000	50
	0.50	0.141	50
	varies from 0.05 to 0.50	0.000	50
Data-set 3	0.10	0.063	29
	0.20	0.000	35
	0.30	0.000	35
	0.40	0.000	35
	0.50	0.000	35
	varies from 0.05 to 0.50	0.000	35
Data-set 4	0.10	0.013	44
	0.20	0.002	45
	0.30	0.000	45
	0.40	0.000	45
	0.50	0.000	45
	varies from 0.05 to 0.50	0.000	45
Data-set 5	0.10	0.323	35
	0.20	0.055	39
	0.30	0.042	40
	0.40	0.017	40
	0.50	0.026	40
	varies from 0.05 to 0.50	0.042	39
Data-set 6	0.10	0.273	12
	0.20	0.178	11
	0.30	0.156	12
	0.40	0.165	12
	0.50	0.170	12
	varies from 0.05 to 0.50	0.192	11

Appendix 1. In these tables the information shown is as follows. The first and second columns under the heading "Cost matrix" and "Number of machines" show the number of cost matrix (part flows between pairs of machines) and the maximum number of machines allowed in a machine-cell, respectively. The next five columns of the table depict detailed results for the hybrid GRASP with path relinking algorithm proposed in this study. Columns under the heading "Solution values", show, respectively, the minimum, average, and maximum objective function values obtained within the 10 executions of the heuristic for each problem instance. Column under the heading "Avg. iter. to best" shows the average number of iterations needed for the heuristic algorithm to find the best solution. Column under the heading "CPU time" shows the average cpu time needed by the GRASP with path relinking algorithm. Finally, the next four columns show results obtained by the FICO Xpress Optimisation Suite when solving the mathematical programming formulation of the problem. Columns under the headings "Lower bound" and "Upper bound" show, respectively, the best lower and upper bounds obtained by the solver withing the cpu time limit of 7200 s. Column under the heading "Percentage gap" show the percentage gap between the lower and upper bounds, computed as follows,

$$\frac{(\text{Upper bound} - \text{Lower bound})}{\text{Lower bound}} \times 100.$$

The last column, under the heading "CPU time", shows the time required by the solver to optimally solve the problem instance. Values of 7200 mean that the solver cannot optimally solve the corresponding problem instance within the time limit.

We also compare our results with those obtained in Spiliopoulos and Sofianopoulou (2008). They obtained similar results because as in this work they obtained optimal or best known solutions in all but one instance. Since they only report the number of problem instances where the optimal or best known solution are obtained a further comparison is not possible. They report that the largest average computational time, for the test instances with 50 nodes, with the proposed ACO algorithm was 18 s, while in the case of the proposed GRASP algorithm the largest average cpu time never exceeded 4 s. However, results obtained in Spiliopoulos and Sofianopoulou (2008) where obtained in a different

Table 3. GRASP results.

Data-set	Instances	Opt./best	Avg. percentage deviation			Avg. iter. to best solution	Avg. CPU time (s)
			min (%)	avg (%)	max (%)		
1	80	80	0	0	0	2.46	0.02
2	50	50	0	0	0	7.08	0.06
3	35	35	0	0	0	11.63	0.11
4	45	45	0	0	0	23.35	0.24
5	40	40	0.00	0.017	0.094	142.05	0.77
6	15	13	0.007	0.165	0.351	320.60	2.62

Table 4. GRASP and path relinking results.

Data-set	Instances	Opt./best	Avg. percentage deviation			Avg. iter. to best solution	Avg. CPU time (s)
			min	avg	max		
1	80	80	0	0	0	4.33	0.02
2	50	50	0	0	0	13.10	0.07
3	35	35	0	0	0	15.57	0.14
4	45	45	0	0	0	30.24	0.27
5	40	40	0.00	0.017	0.094	157.00	0.84
6	15	14	0.003	0.158	0.310	325.13	2.67
7	36	36	0	0	0	86.53	5.04

computer and a rigourous comparison of computational CPU times is not possible. Finally, we compare our results with those obtained in Díaz et al. (2013). Their hybrid algorithm finds all optimal or best known solutions in at least one of the 10 runs for each instance of sets 1–6. In that sense, their algorithm outperforms the algorithm proposed in this work. However, for data-set 7, the algorithm proposed in this study, outperforms that hybrid algorithm in three instances. Therefore, our results are still very competitive, especially when considering the simplicity of the proposed algorithm.

5. Conclusions

In this study, we propose a hybrid heuristic for the Manufacturing Cell Formation Problem considering part processing sequence. The proposed heuristic is based on a GRASP heuristic that uses path relinking as an intensification strategy. At each iteration of the proposed heuristic a two-stage greedy procedure is used to construct initial feasible solutions that are then improved using a local search procedure that explores two different neighbourhoods: (1) reassignment of one machine from one machine-cell to another and (2) interchange of two machines assigned to two different machine cells. These solutions are further improved using a path relinking strategy. The most distinctive features of the proposed algorithm are its simplicity and ease of implementation. In order to test the performance of the proposed heuristic we use different data-sets from the literature and a data-set of larger instances randomly generated. To evaluate the difficulty of the instances of the literature, a computational test is made with a mathematical programming model using a general purpose commercial software. The results clearly show that as the number of machines in the instances increases, the gap between the upper and lower bounds increases substantially. The algorithm is run 10 times with each instance to evaluate solution quality and algorithm robustness. According to the obtained results we observe that the proposed heuristic provides optimal or best known solutions for all but one instance with a very small computational burden.

ORCID

Juan A. Díaz http://orcid.org/0000-0001-6247-8470
Dolores E. Luna http://orcid.org/0000-0002-0601-5006

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Díaz, J. A., Luna, D. E., & Zetina, C. A. (2013). A hybrid algorithm for the manufacturing cell formation problem. *Journal of Heuristics, 19*(1), 77–96.

Díaz, J. A., Luna, D., & Luna, R. (2012). A grasp heuristic for the manufacturing cell formation problem. *TOP, 20*(3), 679–706.

Faigle, U., Schrader, R., & Suletzki, R. (1987). A cutting plane algorithm for optimal graph partitioning. *Methods of Operations Research, 57*, 109–116.

Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*(2), 109–133.

Glover, F. (1996). Tabu search and adaptive memory programming: advances, applications and challenges. *Interfaces in Computer Science and Operations Research, 7*(1), 1–75.

Gonçalves, J. F., & Resende, M. G. (2004). An evolutionary algorithm for manufacturing cell formation. *Computers & Industrial Engineering, 47*(2), 247–273.

Karoum, B., & Elbenani, B. (2016). A hybrid clonal algorithm for the cell formation problem with variant number of cells. *Production Engineering, 1*(11), 19–28.

Laguna, M., & Marti, R. (1999). Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing, 11*(1), 44–52.

Martins, I. C., Pinheiro, R. G., Protti, F., & Ochi, L. S. (2015). A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem. *Expert Systems with Applications, 42*(22), 8947–8955.

Muruganandam, A., Prabhaharan, G., Asokan, P., & Baskaran, V. (2005). A memetic algorithm approach to the cell formation problem. *The International Journal of Advanced Manufacturing Technology, 25*(9), 988–997.

Noktehdan, A., Seyedhosseini, S., & Saidi-Mehrabad, M. (2015). A metaheuristic algorithm for the manufacturing cell formation problem based on grouping efficacy. *The International Journal of Advanced Manufacturing Technology, 82*, 25–37.

Papaioannou, G., & Wilson, J. M. (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*(3), 509–521.

Raja, S., & Anbumalar, V. (2016). An effective methodology for cell formation and intra-cell machine layout design in cellular manufacturing system using parts visit data and operation sequence data. *Journal of the Brazilian Society of Mechanical Sciences and Engineering, 3*(38), 869–882.

Resendel, M. G., & Ribeiro, C. C. (2005). *GRASP with path-relinking: Recent advances and applications* (pp. 29–63). Boston, MA: Springer, US.

Saeedi, S., Solimanpur, M., Mahdavi, I., & Javadian, N. (2010). Heuristic approaches for cell formation in cellular manufacturing. *Journal of Software Engineering and Applications, 3*(07), 674.

Sherali, H. D., & Smith, J. C. (2001). Improving discrete model representations via symmetry considerations. *Management Science, 47*(10), 1396–1407.

Sofianopoulou, S. (1993, October). *A mathematical programming approach to the manufacturing systems cell formation problem*. Proceedings, 2nd Balkan Conference on OR, Salonica

Sofianopoulou, S. (1997). Application of simulated annealing to a linear model forthe formulation of machine cells ingroup technology. *International Journal of Production Research, 35*(2), 501–511.

Sorensen, M. (1995). A polyhedral approach to graph partitioning. PhD thesis, Aarhus School of Business.

Sorensen, M. M. (1999). An adaptation of the kernighan-lin heuristic to the simple graph partitioning problem (Working paper 99-1). Department of Management Science and Logistics, Aarhus School of Busines.

Spiliopoulos, K., & Sofianopoulou, S. (2003). Designing manufacturing cells: A staged approach and a tabu search algorithm. *International Journal of Production Research, 41*(11), 2531–2546.

Spiliopoulos, K., & Sofianopoulou, S. (2008). An efficient ant colony optimization system for the manufacturing cells formation problem. *The International Journal of Advanced Manufacturing Technology, 36*(5), 589–597.

Wu, T.-H., Chang, C.-C., & Yeh, J.-Y. (2009). A hybrid heuristic algorithm adopting both boltzmann function and mutation operator for manufacturing cell formation problems. *International Journal of Production Economics, 120*(2), 669–688.

Wu, T.-H., Chung, S.-H., & Chang, C.-C. (2010). A water flow-like algorithm for manufacturing cell formation problems. *European Journal of Operational Research, 205*(2), 346–360.

Appendix 1

Table A1. Detailed results for data-set 1.

		GRASP-PR hybrid algorithm					FICO XPRESS			
Cost	Number of	Solution values			Avg. iter.	CPU	Lower	Upper	Percentage	CPU
matrix	machines	min	avg	max	to best	time	bound	bound	gap	time
1	5	62	62	62	13	0.03	62	62	0.00	10.86
2	5	62	62	62	22	0.03	62	62	0.00	6.46
3	5	56	56	56	1	0.02	56	56	0.00	0.98
4	5	60	60	60	3	0.04	60	60	0.00	4.31
5	5	62	62	62	2	0.04	62	62	0.00	2.78
6	5	50	50	50	1	0.03	50	50	0.00	4.22
7	5	54	54	54	1	0.03	54	54	0.00	5.00
8	5	67	67	67	4	0.03	67	67	0.00	7.27
9	5	60	60	60	3	0.03	60	60	0.00	4.69
10	5	51	51	51	2	0.03	51	51	0.00	4.52
1	6	55	55	55	1	0.03	55	55	0.00	1.17
2	6	56	56	56	2	0.03	56	56	0.00	1.31
3	6	51	51	51	1	0.02	51	51	0.00	0.94
4	6	53	53	53	5	0.03	53	53	0.00	1.11
5	6	58	58	58	4	0.03	58	58	0.00	1.09
6	6	46	46	46	52	0.03	46	46	0.00	1.22
7	6	51	51	51	2	0.04	51	51	0.00	1.86
8	6	61	61	61	9	0.03	61	61	0.00	1.89
9	6	53	53	53	1	0.02	53	53	0.00	1.39
10	6	47	47	47	5	0.03	47	47	0.00	1.02
1	7	48	48	48	1	0.03	48	48	0.00	1.30
2	7	48	48	48	1	0.03	48	48	0.00	1.08
3	7	43	43	43	1	0.02	43	43	0.00	0.81
4	7	48	48	48	60	0.03	48	48	0.00	1.28
5	7	52	52	52	1	0.03	52	52	0.00	1.39
6	7	40	40	40	1	0.03	40	40	0.00	1.19
7	7	44	44	44	2	0.03	44	44	0.00	1.30
8	7	54	54	54	2	0.03	54	54	0.00	1.34
9	7	45	45	45	1	0.03	45	45	0.00	1.20
10	7	42	42	42	3	0.03	42	42	0.00	0.89
1	8	41	41	41	1	0.02	41	41	0.00	0.41
2	8	42	42	42	2	0.02	42	42	0.00	0.33
3	8	37	37	37	1	0.01	37	37	0.00	0.25
4	8	41	41	41	2	0.02	41	41	0.00	0.39
5	8	43	43	43	1	0.02	43	43	0.00	0.38
6	8	34	34	34	1	0.02	34	34	0.00	0.34
7	8	37	37	37	1	0.02	37	37	0.00	0.33
8	8	45	45	45	4	0.02	45	45	0.00	0.41
9	8	39	39	39	1	0.02	39	39	0.00	0.34
10	8	37	37	37	1	0.02	37	37	0.00	0.31

Table A2. Detailed results for data-set 1.

		GRASP-PR hybrid algorithm					FICO XPRESS			
Cost	Number of	Solution values			Avg. iter.	CPU	Lower	Upper	Percentage	CPU
matrix	machines	min	avg	max	to best	time	bound	bound	gap	time
1	9	39	39	39	26	0.02	39	39	0.00	0.28
2	9	39	39	39	6	0.03	39	39	0.00	0.31
3	9	36	36	36	1	0.01	36	36	0.00	0.23
4	9	39	39	39	2	0.02	39	39	0.00	0.28
5	9	40	40	40	1	0.02	40	40	0.00	0.30
6	9	30	30	30	1	0.02	30	30	0.00	0.23
7	9	35	35	35	2	0.02	35	35	0.00	0.31
8	9	43	43	43	1	0.02	43	43	0.00	0.31
9	9	38	38	38	4	0.02	38	38	0.00	0.28
10	9	34	34	34	1	0.02	34	34	0.00	0.28
1	10	37	37	37	1	0.02	37	37	0.00	0.27
2	10	35	35	35	6	0.02	35	35	0.00	0.34
3	10	32	32	32	1	0.01	32	32	0.00	0.22
4	10	34	34	34	1	0.01	34	34	0.00	0.28
5	10	38	38	38	1	0.02	38	38	0.00	0.31
6	10	28	28	28	1	0.02	28	28	0.00	0.28
7	10	32	32	32	1	0.02	32	32	0.00	0.30
8	10	39	39	39	3	0.02	39	39	0.00	0.33
9	10	36	36	36	1	0.01	36	36	0.00	0.31
10	10	30	30	30	1	0.02	30	30	0.00	0.31
1	11	33	33	33	1	0.02	33	33	0.00	0.31
2	11	30	30	30	1	0.02	30	30	0.00	0.27
3	11	25	25	25	1	0.01	25	25	0.00	0.24
4	11	28	28	28	1	0.01	28	28	0.00	0.28
5	11	36	36	36	1	0.02	36	36	0.00	0.33
6	11	24	24	24	1	0.01	24	24	0.00	0.28
7	11	27	27	27	1	0.01	27	27	0.00	0.33
8	11	32	32	32	1	0.01	32	32	0.00	0.31
9	11	29	29	29	1	0.01	29	29	0.00	0.30
10	11	26	26	26	1	0.02	26	26	0.00	0.25
1	12	25	25	25	1	0.01	25	25	0.00	0.27
2	12	24	24	24	1	0.02	24	24	0.00	0.27
3	12	17	17	17	1	0.01	17	17	0.00	0.19
4	12	22	22	22	1	0.01	22	22	0.00	0.28
5	12	30	30	30	1	0.02	30	30	0.00	0.33
6	12	17	17	17	1	0.01	17	17	0.00	0.23
7	12	22	22	22	1	0.01	22	22	0.00	0.24
8	12	26	26	26	1	0.01	26	26	0.00	0.31
9	12	23	23	23	1	0.01	23	23	0.00	0.34
10	12	22	22	22	43	0.01	22	22	0.00	0.27

Table A3. Detailed results for data-set 2.

		GRASP-PR hybrid algorithm					FICO XPRESS			
Cost	Number of	Solution values			Avg. iter.	CPU	Lower	Upper	Percentage	CPU
matrix	machines	min	avg	max	to best	time	bound	bound	gap	time
1	6	472	472	472	22	0.08	472.00	472.00	0.00	3499.42
2	6	495	495	495	4	0.08	495.00	495.00	0.00	4198.93
3	6	481	481	481	21	0.08	481.00	481.00	0.00	4599.06
4	6	461	461	461	1	0.08	461.00	461.00	0.00	5206.69
5	6	452	452	452	20	0.08	452.00	452.00	0.00	2260.17
6	6	476	476	476	11	0.08	476.00	476.00	0.00	6901.37
7	6	456	456	456	5	0.08	456.00	456.00	0.00	5964.72
8	6	479	479	479	54	0.08	465.87	479.00	2.82	7200.00
9	6	486	486	486	2	0.08	465.55	486.00	4.39	7200.00
10	6	480	480	480	45	0.07	480.00	480.00	0.00	4246.44
1	7	433	433	433	19	0.07	433.00	433.00	0.00	156.46
2	7	457	457	457	5	0.07	457.00	457.00	0.00	216.02
3	7	442	442	442	48	0.08	442.00	442.00	0.00	244.03
4	7	418	418	418	2	0.08	418.00	418.00	0.00	90.84
5	7	416	416	416	27	0.07	416.00	416.00	0.00	82.97
6	7	439	439	439	1	0.08	439.00	439.00	0.00	139.86
7	7	419	419	419	10	0.08	419.00	419.00	0.00	172.82
8	7	437	437	437	12	0.08	437.00	437.00	0.00	177.03
9	7	444	444	444	5	0.08	444.00	444.00	0.00	235.41
10	7	438	438	438	1	0.08	438.00	438.00	0.00	162.02
1	8	413	413	413	23	0.07	413.00	413.00	0.00	93.64
2	8	434	434	434	25	0.07	434.00	434.00	0.00	151.08
3	8	424	424	424	15	0.07	424.00	424.00	0.00	198.29
4	8	402	402	402	42	0.07	402.00	402.00	0.00	86.67
5	8	393	393	393	1	0.07	393.00	393.00	0.00	66.81
6	8	415	415	415	1	0.07	415.00	415.00	0.00	71.14
7	8	401	401	401	5	0.08	401.00	401.00	0.00	124.93
8	8	416	416	416	1	0.07	416.00	416.00	0.00	137.18
9	8	422	422	422	18	0.07	422.00	422.00	0.00	113.69
10	8	420	420	420	1	0.07	420.00	420.00	0.00	150.71
1	9	372	372	372	2	0.06	372.00	372.00	0.00	47.52
2	9	394	394	394	1	0.06	394.00	394.00	0.00	123.36
3	9	386	386	386	8	0.06	386.00	386.00	0.00	74.19
4	9	365	365	365	18	0.06	365.00	365.00	0.00	54.95
5	9	359	359	359	26	0.06	359.00	359.00	0.00	86.47
6	9	375	375	375	3	0.06	375.00	375.00	0.00	35.86
7	9	366	366	366	33	0.06	366.00	366.00	0.00	119.41
8	9	374	374	374	6	0.07	374.00	374.00	0.00	84.89
9	9	383	383	383	11	0.06	383.00	383.00	0.00	122.32
10	9	380	380	380	1	0.06	380.00	380.00	0.00	51.21
1	10	318	318	318	14	0.05	318.00	318.00	0.00	3.48
2	10	336	336	336	1	0.05	336.00	336.00	0.00	3.80
3	10	326	326	326	25	0.05	326.00	326.00	0.00	1.99
4	10	311	311	311	1	0.05	311.00	311.00	0.00	2.88
5	10	308	308	308	1	0.05	308.00	308.00	0.00	1.55
6	10	325	325	325	18	0.05	325.00	325.00	0.00	3.45
7	10	314	314	314	23	0.05	314.00	314.00	0.00	3.50
8	10	316	316	316	4	0.05	316.00	316.00	0.00	1.70
9	10	324	324	324	10	0.05	324.00	324.00	0.00	3.48
10	10	326	326	326	1	0.05	326.00	326.00	0.00	5.83

Table A4. Detailed results for data-set 3.

		GRASP-PR hybrid algorithm					FICO XPRESS			
Cost	Number of	Solution values			Avg. iter.	CPU	Lower	Upper	Percentage	CPU
matrix	machines	min	avg	max	to best	time	bound	bound	gap	time
1	8	1207	1207	1207	7	0.18	897.79	1229.00	36.89	7200.00
2	8	913	913	913	6	0.15	687.97	924.00	34.31	7200.00
3	8	878	878	878	26	0.16	649.33	886.00	36.45	7200.00
4	8	928	928	928	8	0.17	693.43	938.00	35.27	7200.00
5	8	769	769	769	1	0.16	584.18	774.00	32.49	7200.00
6	8	870	870	870	4	0.17	643.90	878.00	36.36	7200.00
7	8	859	859	859	7	0.16	651.78	865.00	32.71	7200.00
1	9	1154	1154	1154	33	0.16	971.13	1162.00	19.65	7200.00
2	9	871	871	871	57	0.15	744.50	876.00	17.66	7200.00
3	9	836	836	836	1	0.15	706.54	844.00	19.46	7200.00
4	9	888	888	888	11	0.15	754.50	895.00	18.62	7200.00
5	9	736	736	736	3	0.15	644.12	739.00	14.73	7200.00
6	9	832	832	832	34	0.15	719.50	839.00	16.61	7200.00
7	9	816	816	816	21	0.15	703.00	823.00	17.07	7200.00
1	10	1118	1118	1118	2	0.15	956.17	1125.00	17.66	7200.00
2	10	837	837	837	58	0.14	735.51	846.00	15.02	7200.00
3	10	805	805	805	21	0.13	697.60	813.00	16.54	7200.00
4	10	853	853	853	7	0.15	748.21	854.00	14.14	7200.00
5	10	706	706	706	5	0.15	690.12	706.00	2.30	7200.00
6	10	793	793	793	17	0.15	720.63	795.00	10.32	7200.00
7	10	783	783	783	8	0.14	703.86	786.00	11.67	7200.00
1	11	1040	1040	1040	1	0.13	956.04	1047.00	9.51	7200.00
2	11	779	779	779	18	0.12	744.50	780.00	4.77	7200.00
3	11	753	753	753	32	0.12	698.79	753.00	7.76	7200.00
4	11	792	792	792	16	0.13	746.67	794.00	6.34	7200.00
5	11	658	658	658	36	0.13	658.00	658.00	0.00	5785.98
6	11	735	735	735	17	0.13	735.00	735.00	0.00	4876.90
7	11	726	726	726	1	0.12	726.00	726.00	0.00	5443.06
1	12	930	930	930	1	0.12	930.00	930.00	0.00	1277.24
2	12	697	697	697	33	0.11	697.00	697.00	0.00	834.60
3	12	676	676	676	1	0.11	676.00	676.00	0.00	1221.22
4	12	708	708	708	24	0.12	708.00	708.00	0.00	887.32
5	12	585	585	585	13	0.11	585.00	585.00	0.00	398.90
6	12	665	665	665	13	0.11	665.00	665.00	0.00	588.57
7	12	643	643	643	2	0.12	643.00	643.00	0.00	514.12

Table A5. Detailed results for data-set 4.

		GRASP-PR hybrid algorithm					FICO XPRESS			
Cost	Number of	Solution values			Avg. iter.	CPU	Lower	Upper	Percentage	CPU
matrix	machines	min	avg	max	to best	time	bound	bound	gap	time
1	8	578	578	578	45	0.31	361.79	593.00	63.91	7200.00
2	8	1152	1152	1152	49	0.31	677.89	1187.00	75.10	7200.00
3	8	3017	3017	3017	66	0.33	1675.25	3064.00	82.90	7200.00
4	8	370	370	370	14	0.27	260.79	379.00	45.33	7200.00
5	8	719	719	719	74	0.30	437.64	736.00	68.17	7200.00
6	8	2018	2018	2018	103	0.35	1167.21	2069.00	77.26	7200.00
7	8	772	772	772	4	0.30	472.96	787.00	66.40	7200.00
8	8	1745	1745	1745	80	0.33	986.44	1786.00	81.06	7200.00
9	8	4358	4358	4358	125	0.36	2376.37	4421.00	86.04	7200.00
1	9	551	551	551	3	0.28	354.85	566.00	59.50	7200.00
2	9	1111	1111	1111	34	0.29	627.24	1145.00	82.55	7200.00
3	9	2894	2894	2894	4	0.32	1577.04	2962.00	87.82	7200.00
4	9	356	356	356	1	0.26	247.43	365.00	47.52	7200.00
5	9	688	688	688	2	0.28	433.74	696.00	60.46	7200.00
6	9	1936	1936	1936	46	0.31	1067.29	1985.00	85.99	7200.00
7	9	737	737	737	6	0.29	465.17	754.00	62.09	7200.00
8	9	1676	1676	1676	41	0.31	931.74	1724.00	85.03	7200.00
9	9	4193	4193	4193	20	0.31	2229.34	4269.00	91.49	7200.00
1	10	509	509	509	1	0.27	407.75	513.00	25.81	7200.00
2	10	1028	1028	1028	67	0.27	776.94	1036.00	33.34	7200.00
3	10	2669	2669	2669	6	0.30	1912.36	2705.00	41.45	7200.00
4	10	325	325	325	56	0.24	295.23	325.00	10.08	7200.00
5	10	636	636	636	1	0.26	491.50	645.00	31.23	7200.00
6	10	1791	1791	1791	55	0.29	1326.86	1806.00	36.11	7200.00
7	10	679	679	679	2	0.25	540.37	686.00	26.95	7200.00
8	10	1547	1547	1547	21	0.28	1133.13	1571.00	38.64	7200.00
9	10	3872	3872	3872	36	0.28	2573.95	3917.00	52.18	7200.00
1	11	504	504	504	7	0.25	400.18	509.00	27.19	7200.00
2	11	1014	1014	1014	5	0.24	733.78	1029.00	40.23	7200.00
3	11	2639	2639	2639	47	0.28	1825.73	2678.00	46.68	7200.00
4	11	322	322	322	4	0.23	291.07	322.00	10.63	7200.00
5	11	624	624	624	1	0.25	480.06	635.00	32.28	7200.00
6	11	1766	1766	1766	21	0.27	1295.18	1792.00	38.36	7200.00
7	11	668	668	668	1	0.26	523.94	680.00	29.79	7200.00
8	11	1528	1528	1528	49	0.27	1100.50	1553.00	41.12	7200.00
9	11	3831	3831	3831	56	0.27	2640.54	3870.00	46.56	7200.00
1	12	486	486	486	6	0.23	400.77	492.00	22.76	7200.00
2	12	976	976	976	11	0.25	746.86	986.00	32.02	7200.00
3	12	2556	2556	2556	1	0.26	1827.90	2577.00	40.98	7200.00
4	12	310	310	310	39	0.22	285.26	312.00	9.37	7200.00
5	12	600	600	600	1	0.24	481.79	608.00	26.20	7200.00
6	12	1699	1699	1699	16	0.27	1284.83	1718.00	33.71	7200.00
7	12	643	643	643	1	0.25	519.51	653.00	25.70	7200.00
8	12	1479	1479	1479	30	0.25	1086.31	1497.00	37.81	7200.00
9	12	3710	3710	3710	103	0.27	2611.43	3745.00	43.41	7200.00

Table A6. Detailed results for data-set 5.

		GRASP-PR hybrid algorithm					FICO XPRESS			
Cost	Number of	Solution values			Avg. iter.	CPU	Lower	Upper	Percentage	CPU
matrix	machines	min	avg	max	to best	time	bound	bound	gap	time
1	4	3483.00	3483.00	3483.00	44	0.44	2084.82	3501.00	67.93	7200.00
2	4	8743.00	8744.30	8756.00	84	0.52	2725.98	9289.00	240.76	7200.00
3	4	14265.00	14265.00	14265.00	109	0.60	4420.33	14923.00	237.60	7200.00
4	4	20153.00	20153.00	20153.00	162	0.67	6042.23	20772.00	243.78	7200.00
5	4	25788.00	25788.60	25791.00	332	0.87	6928.24	26490.00	282.35	7200.00
6	4	1582.00	1582.00	1582.00	39	0.57	791.74	1582.00	99.81	7200.00
7	4	8277.00	8277.00	8277.00	109	0.81	2445.85	8648.00	253.58	7200.00
8	4	15186.00	15186.00	15186.00	141	0.95	3499.15	16143.00	361.34	7200.00
9	4	22728.00	22729.50	22731.00	366	1.24	4574.96	23374.00	410.91	7200.00
10	4	30067.00	30068.30	30073.00	359	1.33	5993.16	31203.00	420.64	7200.00
1	5	3162.00	3162.00	3162.00	488	0.49	2067.77	3223.00	55.87	7200.00
2	5	8113.00	8113.00	8113.00	79	0.54	3436.26	8510.00	147.65	7200.00
3	5	13443.00	13444.90	13462.00	190	0.70	5279.64	14108.00	167.22	7200.00
4	5	19135.00	19135.60	19138.00	344	90.00	6902.70	19855.00	187.64	7200.00
5	5	24591.00	24591.00	24591.00	149	0.75	7536.37	24792.00	228.96	7200.00
6	5	1397.00	1398.10	1408.00	275	0.82	1175.54	1397.00	18.84	7200.00
7	5	7827.00	7827.00	7827.00	169	0.95	2755.19	8300.00	201.25	7200.00
8	5	14332.00	14332.00	14332.00	127	1.03	4324.23	15166.00	250.72	7200.00
9	5	21717.00	21736.00	21773.00	356	1.38	5157.82	22569.00	337.57	7200.00
10	5	28729.00	28729.00	28729.00	154	1.18	6203.83	30112.00	385.38	7200.00
1	7	2794.00	2794.00	2794.00	116	0.49	2460.52	2794.00	13.55	7200.00
2	7	7223.00	7223.00	7223.00	20	0.53	3942.90	7515.00	90.60	7200.00
3	7	12090.00	12090.00	12090.00	61	0.59	6582.18	12670.00	92.49	7200.00
4	7	17382.00	17382.00	17382.00	74	0.64	7528.90	18079.00	140.13	7200.00
5	7	22417.00	22417.00	22417.00	37	0.67	9217.47	23206.00	151.76	7200.00
6	7	1070.00	1071.20	1082.00	195	0.79	1070.00	1070.00	0.00	369.38
7	7	7153.00	7155.70	7179.00	436	1.34	3255.71	7525.00	131.13	7200.00
8	7	13094.00	13094.00	13094.00	212	1.27	4965.88	14025.00	182.43	7200.00
9	7	20082.00	20089.60	20123.00	482	1.68	6450.91	21383.00	231.47	7200.00
10	7	26851.00	26857.40	26879.00	296	1.46	7826.63	28410.00	262.99	7200.00
1	10	2379.00	2385.20	2393.00	287	0.56	2379.00	2379.00	0.00	1309.40
2	10	6297.00	6297.00	6297.00	32	0.49	4517.16	6580.00	45.67	7200.00
3	10	10668.00	10668.00	10668.00	34	0.57	6434.90	11016.00	71.19	7200.00
4	10	15561.00	15561.00	15561.00	18	0.56	8712.15	16099.00	84.79	7200.00
5	10	20267.00	20267.00	20267.00	29	0.60	9575.24	21227.00	121.69	7200.00
6	10	929.00	929.00	929.00	184	0.71	929.00	929.00	0.00	80.83
7	10	6076.00	6076.00	6076.00	55	0.84	4324.31	6234.00	44.16	7200.00
8	10	11258.00	11258.00	11258.00	4	0.97	6051.24	12008.00	98.44	7200.00
9	10	17575.00	17575.00	17575.00	20	0.95	8426.83	18637.00	121.16	7200.00
10	10	23500.00	23500.00	23500.00	12	1.00	10228.01	24326.00	137.84	7200.00

Table A7. Detailed results for data-set 6.

		GRASP-PR hybrid algorithm					FICO XPRESS			
Cost	Number of	Solution values			Avg. iter.	CPU	Lower	Upper	Percentage	CPU
matrix	machines	min	avg	max	to best	time	bound	bound	gap	time
1	4	2955.00	2965.60	2979.00	483	2.04	1037.84	3003.00	189.3509597	7200.00
2	4	13602.00	13618.10	13649.00	220	1.85	1755.96	14895.00	748.2539466	7200.00
3	4	24567.00	24611.40	24645.00	380	2.51	3302.44	26285.00	695.9266482	7200.00
4	4	37299.00	37343.30	37383.00	304	2.50	3130.15	38481.00	1129.366005	7200.00
5	4	48366.00	48434.40	48465.00	477	3.21	5218.66	50061.00	859.2692377	7200.00
1	7	2330.00	2356.30	2370.00	433	2.45	1180.64	2415.00	104.5500745	7200.00
2	7	11689.00	11692.70	11725.00	195	2.41	2859.19	13088.00	357.7520207	7200.00
3	7	21747.00	21747.00	21747.00	310	3.08	4473.49	23827.00	432.6266517	7200.00
4	7	33550.00	33571.30	33655.00	368	3.56	5049.63	35754.00	608.0518771	7200.00
5	7	44059.00	44109.80	44146.00	334	3.47	6543.53	46644.00	612.8262574	7200.00
1	10	1923.00	1923.30	1924.00	283	1.95	1710.09	1923.00	12.45022192	7200.00
2	10	10396.00	10396.00	10396.00	208	2.22	4222.06	11483.00	171.9762391	7200.00
3	10	19670.00	19676.60	19681.00	337	2.81	6034.26	21160.00	250.6643731	7200.00
4	10	30302.00	30302.00	30302.00	79	2.37	7809.19	32339.00	314.1146521	7200.00
5	10	40227.00	40233.80	40247.00	466	3.60	9249.88	42128.00	355.4437463	7200.00

Table A8. Results obtained with FICO Xpress optimisation suite.

Data-set	Instances	Opt./best	Avg. Percentage deviation
1	80	80	0.00%
2	50	50	0.00%
3	12	35	13.84%
4	2	45	48.65%
5	6	40	165.53%
6	1	15	456.17%

Table A9. Detailed results for data-set 7.

Cost	Number of	Solution values			Avg. iter.	CPU
matrix	machines	min	avg	max	to best	time
1	8	695	695	695	64	0.9878
2	8	633	633	633	6	1.7979
3	8	1835	1835	1835	13	1.9922
4	8	1796	1796	1796	32	1.9144
5	8	4515	4515	4515	82	2.3909
6	8	3142	3142	3142	8	2.2513
7	8	1332	1332	1332	66	5.2322
8	8	1561	1561.18	1562	166	3.2231
9	8	2837	2837	2837	82	5.6467
10	8	1943	1943	1943	107	6.3674
11	8	6769	6769.08	6770	123	6.3132
12	8	6973	6973	6973	140	6.9512
1	10	786	786	786	2	2.9345
2	10	742	742	742	3	3.0623
3	10	2152	2152	2152	8	2.7323
4	10	2270	2270	2270	1	2.1707
5	10	5564	5564	5564	1	3.2347
6	10	3766	3766	3766	2	3.2856
7	10	1761	1761	1761	2	2.7998
8	10	1732	1732	1732	157	6.7707
9	10	3255	3255	3255	51	9.1589
10	10	2216	2216	2216	84	10.154
11	10	8701	8701	8701	14	6.567
12	10	8321	8321	8321	125	9.8312
1	12	827	827	827	79	2.4235
2	12	792	792	792	7	2.8889
3	12	2187	2187	2187	68	2.7916
4	12	2320	2320	2320	58	2.2365
5	12	5665	5665	5665	210	2.9205
6	12	3920	3920	3920	152	2.9724
7	12	1801	1801	1801	1	8.1072
8	12	1768	1768	1768	245	8.1449
9	12	3309	3309	3309	318	11.4766
10	12	2318	2318.6	2319	275	11.5145
11	12	8819	8819	8819	75	6.9845
12	12	8445	8445	8445	288	11.3808