Journal of Theoretical and Applied Information Technology
31st October 2017. Vol.95. No 20
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN HYBRID GRAVITATIONAL METHOD FOR SOLVING THE CELL FORMATION PROBLEM

[1]MANAL ZETTAM
[1]Informatics, Systems and Optimization Laboratory,
Department of computer Science, Faculty of Science, Ibn Tofail University Kenitra, Morocco
E-mail: [1]zettammanal@gmail.com

ABSTRACT

The Cellular Manufacturing System has emerged as a strategy capable of improving the manufacturing cost and decreasing the lead time with no negative impacts on the product quality. One of the fundamental problems of this strategy is known as the Cell Formation Problem, which consists of defining cells including a machine group and a product (part) family. In this paper, we consider the Manufacturing Cell Formation Problem where the objective is to maximize the Grouping Efficacy measure. We introduce a hybrid gravitational method (HGM) combining a population based algorithm and a local search procedure. This hybrid metaheuristic is a variant of the Discrete Gravitational Search Algorithm (DGSA). It allows searching extensively the feasible domain of a problem during the early iterations. It also includes a well-balanced mechanism for enhancing exploration and exploitation strategies. Several modifications were necessary to adapt the DGSA in order to maintain similar performance when dealing with the CFP. The major ones consist in modifying the population of solutions and in performing the exploration phase. The Path Relinking process used in DGSA to modify the solution is replaced by a crossover strategy derived from concepts of the Gravitational Search Algorithm (GSA). The performance of this hybrid variant of the Gravitational Search Algorithm is evaluated using a set of 35 benchmarks from the literature. The numerical results confirm that our hybrid metaheuristic could be considered as one of the best algorithms.

Keywords: *Cell Formation Problem, Hybrid Gravitational Method, Gravitational Search Algorithm, Tabu Search, Grouping Efficacy.*

1. INTRODUCTION

The Group Technology (GT) is a concept which had emerged in the late 1950s; it had been first introduced by [15] and adopted for the first time by the Ford Motor Company in the 1970s according to [17]. Since then GT has been widely applied in the industrial field due to its impact on increasing both the productivity and the manufactured products quality with a reduced lead time and cost. The application of the GT concept known as Cellular Manufacturing is based on the principle that similar things should be manufactured similarly. One of the most used methods to bring the cellular manufacturing philosophy to bear is referred as the Cell Formation Problem (CFP). The CFP takes its origins from the works performed by the pioneers of GT cited in [21].

Consider a production organization (system) within a company including a set of machines and a set of parts to be produced. The CFP consists in dividing the system into subsystems, referred to as cells, in order to decrease inter-cell movements and to increase the intra-cell movements. This can be done by grouping parts into families and assign them to machine groups. To the best of our knowledge, the methods used to solve the CFP can be divided into the following categories:

a. Cluster Analysis,
b. Graph Partitioning Approaches,
c. Mathematical Programming Methods,
d. Heuristic and Metaheuristic Algorithms,
e. Artificial Intelligence Methodologies.

A survey of the methods employed to solve the CFP can be found in [16]. Some exact methods have been applied to solve CFP of small dimension, and two popular exact methods can be found in [5] and [5]. In spite of that, the CFP is still considered as an NP-complete (hard) problem according to [24]. For this particular reason, a large number of heuristics have been developed to solve the CFP using reasonable computational time.

In this paper, we introduce a hybrid gravitational method combining a population based algorithm and a local search procedure. This hybrid metaheuristic is a variant of the Discrete Gravitational Search Algorithm (DGSA) introduced in [1]. DGSA allows searching extensively the feasible domain of a problem during the early iterations of the process. DGSA also includes a well-balanced mechanism for enhancing exploration and exploitation strategies. Several modifications were necessary to adapt the DGSA in order to maintain similar performance when dealing with the CFP. The required modifications to adapt the DGSA to the CFP are described thoroughly in the paper. The major ones consist in modifying the population of solutions and in performing the exploration phase. Moreover, the Path Relinking process used in DGSA to modify the solution is replaced by a crossover strategy derived from concepts of the Gravitational Search Algorithm (GSA) proposed [18].

The reminder of this paper is organized as follows. In Section 2, the CFP and the mathematical model are introduced. The basic notions of the Gravitational Search Algorithm (GSA) required to specify the crossover of our hybrid metaheuristic are presented in Section 3. Our variant HGM of the DGSA is detailed in Section 4. First, we indicate how the solutions are encoded in the process, and how the solutions of the initial population are generated. Then we describe the Dependent Movement Search Operator Phase (DMSOP) where a solution is modified using a crossover operator driven by the other solutions in the population, and the Independent Movement Search Operator Phase (IMSOP) where the offspring is improved independently of the other using a Tabu search procedure. The numerical results are summarized in Section 5. First, the values of the parameters are determined, and the efficiency of the HGM is analyzed using 35 well-known benchmark problems. Our approach HGM outperforms the best-known value for one problem, it reaches the best-known values for 30 other problems, and it misses the best-known solution for 4 other problems by a small margin. Concluding remarks are presented in Section 6.

2. THE MATHEMATICAL FORMULATION OF THE CELL FORMATION PROBLEM

2.1 The Cell Formation Problem

Given a 0-1 machine-part incidence matrix, the main goal of the CFP is to obtain a diagonal block matrix by rearranging the rows and the columns. The columns and the rows are rearranged in such a way to construct machine groups and part families that optimize one of the performance measures defined in literature. In this paper, the Grouping Efficacy measure is selected due to its several advantages such as:

a. improving both the with-in cell machine use and to reduce the inter-cell movement,

b. allowing to differentiate between well-structured (with a high with-in cell machine and low inter-cell movement) and ill-structured (with a low with-in cell machine and high inter-cell movement) incidence matrices,

c. generating interesting block diagonal matrices,

d. simplifying its formulation.

(a)

	Parts	1	2	3	4	5	6	7	8	9	10	11
Machines	1	1	1	0	0	0	1	0	0	0	0	0
	2	0	1	0	0	0	1	0	0	1	0	0
	3	1	0	0	0	0	0	1	0	0	0	1
	4	0	0	1	0	0	0	1	0	0	0	0
	5	0	0	1	1	0	0	0	0	0	0	1
	6	0	0	0	1	1	0	0	0	0	1	0
	7	0	0	0	0	1	0	0	1	0	1	0

(b)

	Parts	4	5	8	10	1	2	6	9	3	7	11
Machines	6	1	1	0	1	0	0	0	0	0	0	0
	7	0	1	1	1	0	0	0	0	0	0	0
	1	0	0	0	0	1	1	1	0	0	0	0
	2	0	0	0	0	0	1	1	1	0	0	0
	3	0	0	0	0	1	0	0	0	0	1	1
	4	0	0	0	0	0	0	0	0	1	0	0
	5	0	0	0	0	0	0	0	0	1	1	1

Figure 1. An incidence matrix example.

Many performance measures had been cited in literature. A survey overview of performance measures can be found in [8] and [25].We select the Grouping Efficacy defined as follows:

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

Where a is the total number of 1 in the incidence matrix, a_0^{In} is the number of voids (the number of zeros within a cell), a_1^{Out} is the number of exceptional elements (the number of 1 out of all cells), and a_1^{In} is the number of 1 within a given cell. An example of a solution of the CFP is shown in Figure 1, where machines and parts were respectively grouped into cells and families such as each part family were assigned to a cell containing the machines required for the operations.

2.2 The mathematical model

In this subsection a mathematical model similar to the one presented in [2] and [26], is introduced. The mathematical programming model is formulated as below:

$$Max\ Eff = \frac{\sum_{k=1}^{C}\sum_{i=1}^{M}\sum_{j=1}^{P} a_{ij}x_{ik}y_{ik}}{a+\sum_{k=1}^{C}\sum_{i=1}^{M}\sum_{j=1}^{P}(1-a_{ij})x_{ij}y_{ik}} \quad (2)$$

Subject to:

$$\sum_{k=1}^{C} x_{ik} = 1,\ \ i=1,...,M \quad (3)$$

$$\sum_{k=1}^{C} y_{jk} = 1,\ \ j=1,...,P \quad (4)$$

$$\sum_{i=1}^{M} x_{ik} \geq 1,\ \ k=1,...,C \quad (5)$$

$$\sum_{j=1}^{P} y_{jk} \geq 1,\ \ k=1,...,C \quad (6)$$

$$x_{ik} = 0 \text{ or } 1,\ i=1,...,M,\ k=1,...,C \quad (7)$$

$$y_{jk} = 0 \text{ or } 1,\ j=1,...,P, k=1,...,C \quad (8)$$

Where x_{ik} and y_{jk} are two binary variables defined as below:

For each $i=1,...,M$, $k=1,...,C$

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

For each $j=1,...,P, k=1,...,C$

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

And where

$$a_1^{Out} = a - \sum_{k=1}^{C}\sum_{i=1}^{M}\sum_{j=1}^{P} a_{ij}x_{ik}y_{jk} \quad (9)$$

$$a_0^{In} = \sum_{k=1}^{C}\sum_{i=1}^{M}\sum_{j=1}^{P}(1-a_{ij})x_{ik}y_{jk} \quad (10)$$

a_{ij} denotes the element in the i^{th} row and the j^{th} column of the incidence matrix. The constraints (3) and (4) ensure that each part and each machine must be assigned to only one cell. The constraints (5) and (6) indicate that each cell contains at least one part and one machine. The constraints (7) and (8) indicate that decision variables are binary. In the computational experiments, the number of cells, denoted *C*, was fixed for each problem to its value in the best-known solution reported in the literature.

Our solution approach deals with a 0-1 programming problem. It is considered as hybrid metheuristic. This hybrid metaheuristic includes a population based procedure as well as a local search algorithm. The proposed hybrid metaheuristic is in line with the discrete gravitational search algorithm (DGSA) proposed in [1]. As the hybrid metaheuristic is referring to the swarm procedure gravitational search algorithm (GSA), we should first summarize this swarm procedure.

3. THE GRAVITATIONAL SEARCH ALGORITHM

The swarm procedure called Gravitational Search Algorithm (GSA) is a population based procedure first proposed by [18] to mimic the Newton's Law of Gravity and the Law of Motion when solving continuous optimization problem of the form $\underset{s\in R^n}{\text{Max}}\ fit(s)$.The value $fit(s)$ measures the fitness of a solution $s=(s_1,\dots,s_d,\dots,s_n)\in R^n$. In the GSA, each solution s corresponds to an agent having four specifications: position $(s_1,\dots,s_d,\dots,s_n)$, inertia mass, active gravitational mass, and passive gravitational mass.

Now assume that N solutions (agents) are available:

$$s^i = (s_1^i,\dots,s_d^i,\dots,s_n^i),\quad i=1,2,\dots,N. \quad (11)$$

In the context of gravitational motion, good solutions (with higher fitness $fit(s^i)$) move more slowly than ones having smaller fitness $fit(s^i)$. Hence, agents with smaller fitness navigate around agents with higher fitness. Through the iterations, the gravitational and inertia mass of each agent is regulated. After a while, the solutions will be attracted by the one having the best *fit* that represents an optimal solution in the search space.
All solutions (agents) are modified to mimic the Newton Law of Gravity. At iteration *t*, we denote the position of solution i by:

$$s^i(t)=(s_1^i(t),\dots,s_d^i(t),\dots,s_n^i(t)),\quad i=1,2,\dots,N. \quad (12)$$

It is modified according to a velocity vector $v^i(t)=(v_1^i(t),\dots,v_d^i(t),\dots,v_n^i(t))$ involving an acceleration vector $ac^i(t)$. To determine these vectors, consider the acting force on solution i by another solution j specified as follows:

$$F_d^{ij}(t)=G(t)\frac{M_{pi}(t)\times M_{aj}(t)}{R_{ij}(t)+\xi}(s_d^j(t)-s_d^i(t)),\quad d=1,2,..,n \quad (13)$$

where $M_{aj}(t)$ represents the active gravitational mass of agent j, $M_{pi}(t)$ represents the passive gravitational mass of agent i, $G(t)$ represents the gravitational constant at time t, ξ represents a small constant, and $R_{ij}(t)$ represents the Euclidian distance between the two agents i and j $\left(\text{i.e., } R_{ij}(t)=\left\|s^i(t),s^j(t)\right\|_2\right)$

To include a stochastic aspect into the GSA, the total force acting on solution (agent) i in a dimension d is determined by a randomly weighted sum of d^{th} components of the forces exerted from other agents:

$$\bar{F}_d^i(t) = \sum_{j=1, j\neq i}^{N} rand_j F_d^{ij}(t) \quad (14)$$

where $rand_j$ denotes a random number in the interval $[0,1]$. Thus, by the Law of Motion, the acceleration $ac_d^i(t)$ of the d^{th} component of agent i at a specific time t is specified as follows:

$$ac_d^i(t) = \frac{\bar{F}_i^d(t)}{M_{ii}(t)} \quad (15)$$

where $M_{ii}(t)$ denotes the i^{th} agent inertia mass. Moreover, the velocity of the solution (agent) i at the next iteration is specified as a linear combination of its current velocity and its current acceleration. Therefore, the position and the velocity of the d^{th} component of solution i are defined as follows:

$$v_d^i(t+1) = rand_i \times v_d^i(t) + ac_d^i(t) \quad (16)$$

$$s_d^i(t+1) = s_d^i(t) + v_d^i(t+1) \quad (17)$$

where $rand_i$ is a uniform random number in the interval $[0,1]$ to induce a stochastic aspect into the GSA.

We initialize the gravitational constant G at a specific value which is reduced during the search process in order to control the search accuracy. In other words, G is a function of the initial value (G_0) and time (t):

$$G(t) = G(\mathrm{G}_0, t) \quad (18)$$

The fitness of the updated solutions is used to simulate the fact that the best solutions are more attractive and that they induce a larger impact on modifying the gravitational and the inertia masses of the other solutions. At implementation stage of GSA, the three different masses take the same value:

$$M_{ai}(t) = M_{pi}(t) = M_{ii}(t) = M_i(t), \quad i = 1,2,...,N. \quad (19)$$

Moreover, they are specified as follows:

$$m_i(t) = \frac{fit(s^i(t)) - worst(t)}{best(t) - worst(t)} \quad (20)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (21)$$

where $worst(t)$ and $best(t)$ are defined as follows:

$$best(t) = \max_{j=1,2,...N} \left\{ fit(s^j(t)) \right\} \quad (22)$$

$$worst(t) = \min_{j=1,2,...N} \left\{ fit(s^j(t)) \right\} \quad (23)$$

During the early iterations of the procedure, the acting forces on an agent are specified as in equation (14) to simulate an exploration phase of the process to search more extensively the feasible domain. Later, the process should move to an exploitation phase as proposed in [18]. Thus in (14), instead of using all the other solutions to specify the acting force, we use only a set Kbest of the best solutions available. In other words, in the early stage, the set Kbest includes all the solutions, then its size is reduced during the process to include only the best solutions. Consequently, the equation (14) is modified as follows:

$$\bar{F}_d^i(t) = \sum_{j\in Kbest, j\neq i} rand_j F_d^{ij}(t) \quad (24)$$

According to [9], the GSA is a swarm intelligence algorithm usually described as a swarm based system leading to emergent behavior specified through interactions between components of the system. Moreover, all the swarm intelligence algorithms have their own specific way to explore and exploit the search space, even though they all share common principles.

4. THE HYBRID GRAVITATIONAL METHOD

In this paper, we introduce a hybrid gravitational method quite similar to the Discrete Gravitational Search Algorithm proposed in [1] to solve the CFP. Accordingly, our hybrid method includes two phases: the Dependent Movement Search Operator Phase (DMSOP) where a solution is modified according to the solutions within the population, and the Independent Movement Search Operator Phase (IMSOP) where the offspring is improved independently of the solution within the population using a local search procedure. Hence, our hybrid is composed of a population based algorithm corresponding to the Dependent Movement Search Operator that modify the population of solutions where each offspring solution can be improved using a local search method. First we summarize the hybrid metaheuristic:

Hybrid gravitational method (HGM)

(1) Generate an initial population S including N feasible solutions.

Compute the acceleration ac^s for each solution $s \in S$.

Specify the set $Kbest$ of the solutions having the best values of Eff in the population, and the parameter p_{imp}

(2) For $q_{iteration}$ generations

- Select randomly a solution $s \in (S - Kbest)$
- Modify the solutions in S using the (DMSOP) and (IMSOP) operators as follows:
  - Let the current solution $s' = s$.
  - Apply the following process for each solution $s^0 \in Kbest$ in decreasing order of their Eff values:
    - Use the (DMSOP) operator to complete a crossover between the solution s' and $s^0 \in Kbest$ to generate a solution s^3.
    - Improve solution s^3 by fixing the groups to assign parts to families according to the machine groups, and afterward fixing the new families in order to assign the machines to the groups according to the part families, and repeat sequentially the two assignment modifications until no modification is possible. Denote by s'' the best solution generated during this process.
    - If necessary, apply the repair process to the offspring solution s'' to insure that no group or no family is empty.
    - Use the (IMSOP) to improve the solution s'' using the local search procedure according to the parameter p_{imp} .
    - Let s' denotes the resulting solution.
- The resulting s' replaces the solution s in S having the worst Eff value.
- Compute the updated acceleration ac^s for each solution $s \in S$.
- Determine the new set $Kbest$.

Now let us complete the presentation of the method HGM by describing in more details the chromosomal representation, the initialization of the initial population, and the stages of the proposed HGM.

4.1 The Chromosomal representation

For our CFP problem, each chromosome (solution) $s \in S$ includes $P + M$ genes, encoded as a vector of integers. Recall that P and M are the number of parts and machines. Each integer component of the chromosome s lies in the interval $[1, C]$ (C being the number of cells). Hence, the chromosome is composed of two subsets associated with parts and machines, respectively. Figure2 illustrates a chromosome where s_j denotes the cell to which part or machine j belongs, according to the value of $j \in [1,\ldots,P]$ or $j \in [P+1,\ldots,P+M]$, respectively.

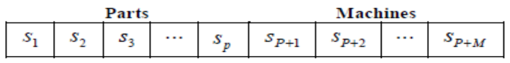
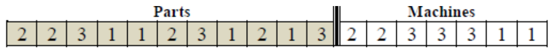

Parts
Machines
s_1 s_2 s_3 $\cdots$ s_P s_{P+1} s_{P+2} $\cdots$ s_{P+M}

Figure 2. The chromosomal representation.

Accordingly, the chromosome associated with the solution in Figure 1 is the following.



4.2 The generation of the initial population

Let N be the number of solutions in the population S. The first solution in the initial population is obtained by combining the group-and-assign method introduced in [23] and the approach presented in [19]. The group-and-assign method introduced in [23] is used to generate the machine groups. The approach presented in [19] is used to determine the part families on the basis of the machine groups. The reminder solutions of the initial population are generated randomly.

For the sake of completeness, the group-and-assign method is summarized as follows. The first machine is selected randomly among those having an index between 1 and $\lfloor M/C \rfloor$ (the floor of M/C). The selected machine (referred as the seed machine) is assigned to the first group gr_1. The reminder $(C-1)$ seed machines will be assigned to the other cells $\{gr_2,\ldots,gr_C\}$. Each machine is selected sequentially among the unassigned machines in order to reduce the number of parts that are not serviced by the current set of seed machines already selected. In the second stage, the reminder machines are assigned in order to maximize the number of parts serviced by all the machines already assigned to the cell. The procedure can be summarized as follows:

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

$$\bar{k}(i) = \underset{k=1,\ldots,C}{\text{Min}} \left\{ \frac{1}{|gr_k|} \sum_{j=1}^{n} \sum_{i_k \in gr_k} \left| a_{ij} - a_{i_k j} \right| \right\}$$

$$gr_i = \underset{k=1,\ldots,C}{\text{ArgMin}} \left\{ \frac{1}{|gr_k|} \sum_{j=1}^{n} \sum_{i_k \in gr_k} \left| a_{ij} - a_{i_k j} \right| \right\}$$

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 the group $gr_{\bar{i}}$; i.e., $gr_{\bar{i}} = gr_{\bar{i}} \cup \{\bar{i}\}$.

III. Eliminate $\bar{i}$ from INA, and repeat steps I-III until all machines are assigned.

The approach introduced in (Rojas et al., 2004) is used to assign parts to families according to the machine groups in order to generate an initial solution having a good grouping efficacy. It mainly consists in assigning each part j to the family F_k where the quotient $\frac{u_{jk}}{e_j + v_{jk}}$ is maximized. Here u_{jk} denotes the number of machines in group gr_k that process the part j, v_{jk} denotes the number of machines within group gr_k that are not processing the part j, and e_j denotes the number of machines processing the part j.

Note that whenever a new solution is generated, a correction process exists to ensure that each group and each family contains a machine or a part, respectively. Thus, if a group has no machine within it, then a machine is selected randomly from another group including more than one machine, and it is transferred to the former group. A similar process exists for the part families.

4.3 The Dependent Movement Search Operator Phase for the CFP

The first step during the DMSOP is to apply sequentially a crossover operator to a randomly selected solution $s \in (S - Kbest)$ and a solution in *Kbest* to generate a new solution s'. The solutions of *Kbest* set are picked up one by one in a decreasing order according to their *Eff*. To summarize the crossover operator, consider some step of this process where the crossover operator is applied between the current solution s' and a solution:

s'_1	s'_2	$\cdots$	s'_P	s'_{P+1}	s'_{P+2}	$\cdots$	s'_{P+M}

s^o_1	s^o_2	$\cdots$	s^o_P	s^o_{P+1}	s^o_{P+2}	$\cdots$	s^o_{P+M}

The crossover operator is governed by the acceleration vector ac^0 of s^o as specified in (15). Recall that in the CFP, ac^0 is a $P+M$ vector. The crossover is completed as follows. First, select randomly an index $r \in [1, P+M]$. Then, for each index $d \in \{1, 2, \ldots, M+P\}$, compute the value $Diff_d = ac^0_r - ac^0_d$. If $Diff_d < 0$, then $s^3_d = s'_d$, otherwise $s^3_d = s^o_d$.

s^3_1	s^3_2	$\cdots$	s^3_P	s^3_{P+1}	s^3_{P+2}	$\cdots$	s^3_{P+M}

The output solution s^3 of the crossover operation becomes the new solution s'; i.e., $s' = s^3$. Then, we improve solution s^3 by fixing the groups to assign parts to families according to the machine groups, and afterward fixing the new families in order to assign the machines to the groups according to the part families, and repeat sequentially the two assignment modifications until no modification is possible. Thereafter the best solution obtained by DMSOP is denoted s''.

4.4 The Independent Movement Search Operator Phase for the CFP

In this paper, a Tabu search algorithm with short memory, is used as the Independent Movement Search Operator Phase (IMSOP). Recall that the Tabu search algorithm was first introduced by [3] [11], and then extended in [4]. The main purpose of the Tabu search algorithm is to allow overcoming the local optima. The search starts with solution s'', and iteratively it seeks for better solutions in a neighborhood where each element is obtained by applying either the *add/drop* operator or the *swap* operator according to a probability p. The neighborhood includes only solutions that are not Tabu unless it satisfies the aspiration criteria. This process continues until the stopping criteria are met. To specify completely the Tabu search algorithm, five different concepts have to be defined: the neighborhood structure, the movement operators, the Tabu list, the aspiration criteria, and the stopping criteria.

The neighborhood size is denoted $I_{neighbor}$ (in our implementation, $I_{neighbor} = M/4$). To generate each neighbor, we select randomly one of the two operators *add/drop* and *swap* according to a probability p $\left(\text{in our implementation, } p = 1/2\right)$. Assume that s' denotes the current solution. To generate a neighbour of s' usingthe *add/drop* operator, we select randomly an index $d \in \{1,2,...,M+P\}$ and a cell $c \in C$. Then the neighbor solution is obtained by modifying the value of s'_d to c. similarly, to generate a neighbour of s' using the *swap* operator, two index $d_1, d_2 \in \{1,2,...,M+P\}$ are selected randomly. Then the neighbor solution is obtained by exchanging the values s'_{d_1} and s'_{d_2}.

The Tabu list T is stored into a matrix $C \times (P+M)$ .Whenever a machine or a part t_{ij} denotes the i^{th} row and the j^{th} column of the matrix T. If a machine or a part $d \in \{1,2,...,M+P\}$ is assigned to a cell $c \in C$, then the value of t_{ij} increases from its value to the iteration where this move will lose its tabu status. In our case, we consider a short-term memory tabu search where a move is forbidden for 10 iterations unless the aspiration criterion is met. The aspiration criterion is met if all moves except one are tabu.

5. THE COMPUTATIONAL RESULTS

In the first part of this section, a statistical study is performed to determine the parameter values of the HGM process for solving the CFP. Then, once these values are specified, we complete a comparative study to evaluate the performance of the proposed algorithm with respect to the best known solution of 35 benchmark problems.

To evaluate the algorithm performance when solving a problem, we use a *gap* specified as the difference between the value of the obtained solution denoted *A* and the best known solution *B*. The *gap* is calculated as follows:

$$gap(\%) = \frac{B-A}{B} \times 100 \qquad (25)$$

In the DGSA introduced by [1] the $Kbest$ set size and the gravitational constant *G* parameters were defined as relevant parameters, and thus we kept the same strategies to determine their values. Indeed, [1] determine their values based on experiments. Moreover, a statistical study was performed to define the other relevant parameters of the proposed HGM. Based on experiments, we notice that the value of the *gap* depends on the those of p_{imp} (the probability to apply the improving process IMSOP), *N* (the population size), and $q_{iteration}$ (the number of required iterations) variables. To verify the relevance of the mentioned parameters, we use the multiple regression method defining the relationships among variables. In our model the dependent variable is the *gap*. The independent variables in the model are p_{imp} ,*N*, and $q_{iteration}$.

Ten different executions of the HGM with different values of the independent variables were performed. We randomly picked up the p_{imp} ,*N*, and $q_{iteration}$ values for the 35 problems. The statistical study shows that the relation between the dependent variable and independent variables is statically significant. The Figure 3 indicates that the gap variable value is almost constant even if *N* varies. Therefore *N* is not included as a regression model variable. Unlike the Figure 3, the Figures 4 and 5confirm that the variations of p_{imp} and $q_{iteration}$ influence the *gap* value. Furthermore, the statistical study demonstrates that only 74.49% of the *gap* variation is explained by the variations of p_{imp} and $q_{iteration}$ .Thus, the regression model is defined as follows:

$$gap = 0.5329 - 0.152\, p_{imp} + 0{,}0012\, q_{iteration} \qquad (26)$$

The Multiple Regression process uses two statistical tests to demonstrate the significance of variables, namely the Fisher and the Student tests.

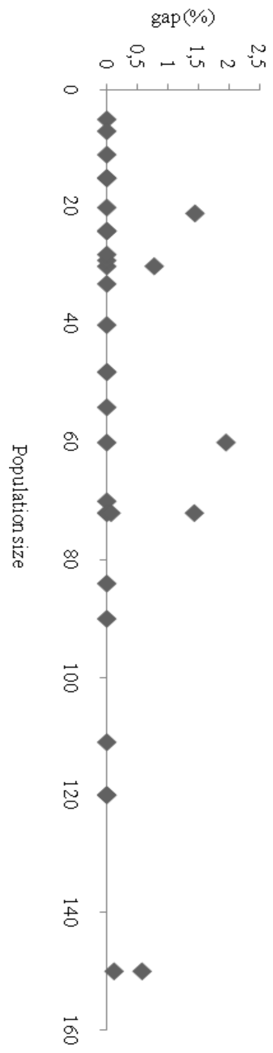


Figure 2. The Population Size And The Gap Variation Graph.

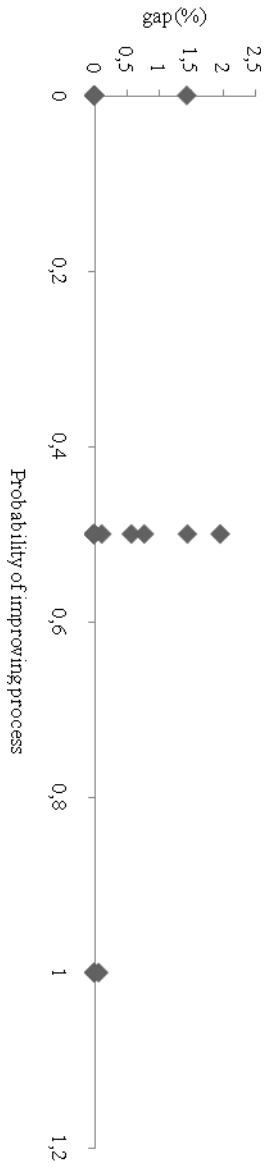


Figure 4.The Probability Of Improving Process And The Gap Variation Graph.

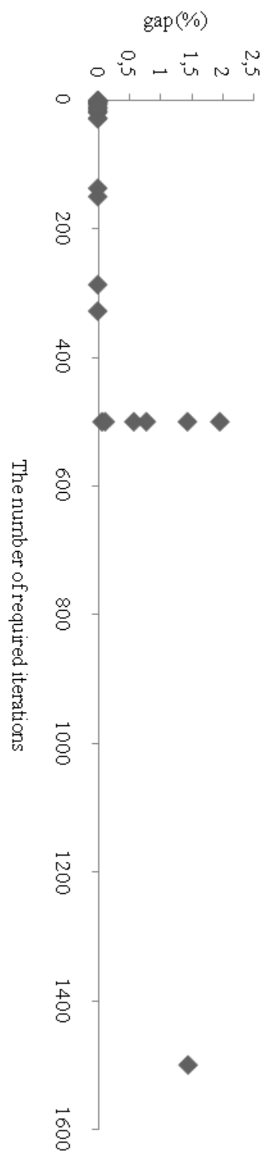


Figure 5.The Number Of Required Iterations And The Gap Variation Graph.

The fisher and student tests show that p_{imp} and $q_{iteration}$ are significant variables. Therefore, inasmuch as the value of the p_{imp} is known, we can use the equation (26) to predict the value of $q_{iteration}$, and vice-versa. The parameters which play a major role in the performance of the method are provided in Table 1.

Table 1. The Relevant Parameters Of HGM.

Parameter	Value
The size of the population N	$3M$
The number of iterations $q_{iteration}$	500
The size of the $Kbest$ set	$3M-1$
The value of the initial G	100

After adjusting the parameters, the proposed HGM was used to solve the set of 35 benchmarks from literature. The proposed heuristic is coded in JAVA language, and it is run on a personal computer with an intel Core i5-2540M 2.60GHz processor.

Each instance is solved 10 times. The results are summarized in Table 2. Each problem is specified in the first 4 columns. Then the results obtained with the method HGM are summarized in the next 4 columns: the worst, the best, the average *Eff* in columns 5, 6, and 7, respectively, and the average CPU time in column 8 for each problem. The value of the best known solution and the *gap*% for each problem are given in columns 9 and 10, respectively. Finally, the references where the best known solutions are found are also provided in column 11 of the Table 2.

The results in Table 2 indicate that the HGM procedure outperforms other solution method to determine the best known solution for problem P33, and it reaches the best known solutions for 30 other problems. The best known solutions is missed for the four problems (P18,P21, P27 and P31), but the *gap* % indicate that the gap to the best known solution is rather small. Thus the numerical results indicate the efficiency of the HGM procedure to solve the CFP.

CONCLUSION

In this paper, we introduce a hybrid gravitational method combining a population based algorithm and a local search procedure to deal with the NP-complete cell formation problem. This hybrid is a variant of the Discrete Gravitational Search Algorithm (DGSA) allowing to search extensively the feasible domain of a problem

Table 2. Computational Results

Problem	M	P	C	HGM Worst *Eff*	HGM Best *Eff*	HGM Average *Eff*	HGM Time (second)	Best known *Eff*	Gap (%)	References
P1	5	7	2	82.35	82.35	82.35	0.0219	82.35	0	[22]
P2	5	7	2	69.57	69.57	69.57	0.0156	69.57	0	[22]
P3	5	18	2	79.59	79.59	79.59	0.0124	79.59	0	[22]
P4	6	8	2	76.92	76.92	76.92	0.0141	76.92	0	[22]
P5	7	11	5	60.87	60.87	60.87	0.0937	60.87	0	[22]
P6	7	11	4	70.83	70.83	70.83	0.0234	70.83	0	[22]
P7	8	12	4	69.44	69.44	69.44	0.0266	69.44	0	[22]
P8	8	20	3	85.25	85.25	85.25	0.0188	85.25	0	[22]
P9	8	20	2	58.72	58.72	58.72	0.039	58.72	0	[22]
P10	10	10	5	75	75	75	0.0577	75	0	[6]
P11	10	15	3	92	92	92	0.0236	92	0	[6]
P12	14	24	7	72.06	72.06	72.06	0.3263	72.06	0	[7]
P13	14	24	7	71.83	71.83	71.83	0.3766	71.83	0	[7]
P14	16	24	8	53.26	53.26	53.26	4.1425	53.26	0	[22]
P15	16	30	6	69.53	69.53	69.53	0.915	69.53	0	[2]
P16	16	43	8	57.53	57.53	57.53	10.6387	57.53	0	[22]
P17	18	24	9	57.73	57.73	57.73	3.6019	57.73	0	[22]
P18	20	20	5	43.06	43.17	43.109	164.3702	43.45	0.64	[13]
P19	20	23	7	50.81	50.81	50.81	4.3093	50.81	0	[7]
P20	20	35	5	77.91	77.91	77.91	1.6562	77.91	0	[22]
P21	20	35	5	57.14	57.14	57.14	280.6444	57.98	1.45	[22]
P22	24	40	7	100	100	100	1.1683	100	0	[22]
P23	24	40	7	85.11	85.11	85.11	1.3688	85.11	0	[22]
P24	24	40	7	73.51	73.51	73.51	1.3756	73.51	0	[22]
P25	24	40	11	53.29	53.29	53.29	53.4848	53.29	0	[22]
P26	24	40	12	48.95	48.95	48.95	393.5611	48.95	0	[22]
P27	24	40	12	46.58	46.58	46.58	957.4588	47.26	1.44	[13]
P28	27	27	5	54.82	54.82	54.82	11.047	54.82	0	[22]
P29	28	46	10	47.08	47.08	47.08	584.1145	47.08	0	[13]
P30	30	41	14	63.31	63.31	63.31	790.244	63.31	0	[13]
P31	30	50	13	59.77	59.77	59.77	2033.4975	60.12	0.58	[14]
P32	30	50	14	50.83	50.83	50.83	792.9925	50.83	0	[7]
P33	36	90	17	47.9	48.00	47.97	1128.0133	47.75	-0.5	[2]
P34	37	53	3	60.64	60.64	60.64	399.5972	60.64	0	[22]
P35	40	100	10	84.03	84.03	84.03	42.2884	84.03	0	[6]

during the early iterations of the process, and including a well-balanced mechanism for enhancing exploration and exploitation strategies. Hence, at each iteration of our variant of the DGSA, two phases are completed. Moreover, the Path Relinking process used in HGM to modify the solution is replaced by a crossover strategy derived from concepts included in the gravitational search algorithm (GSA). During the Independent Movement Search Operator Phase (IMSOP) the offspring can be improved, according to some probability, independently of the other using a Tabu search procedure.

A statistical study was performed to determine the parameter values of the HGM process for solving the CFP. A comparative study to evaluate the performance of the proposed algorithm with respect to the best known solution of 35 benchmark problems was performed too. Our hybrid gravitational method outperforms the best-known value for one problem, it reaches the best-known values for 30 other problems, and it misses the best-known solution for 4 other problems by a small margin. Thus this hybrid metaheuristic can be considered as one of the best algorithms to solve the CFP.

REFRENCES:

[1] Dowlatshahi, M. B., Nezamabadi-pour, H., & Mashinchi, M. (2014). A discrete gravitational search algorithm for solving combinatorial optimization problems. Information Sciences, 258, 94-107. doi: 10.1016/j.ins.2013.09.034

[2] Elbenani, B., Ferland, J. A., & Bellemare, J. (2012). Genetic algorithm and large neighbourhood search to solve the cell formation problem. Expert Systems with Applications, 39(3), 2408-2414. doi: http://dx.doi.org/10.1016/j.eswa.2011.08.089

[3] Glover, F. (1989). Tabu Search—Part I. ORSA Journal on Computing, 1(3), 190-206. doi: 10.1287/ijoc.1.3.190

[4] Glover, F. (1990). Tabu Search—Part II. ORSA Journal on Computing, 2(1), 4-32. doi: 10.1287/ijoc.2.1.4

[5] Goldengorin, B., Krushinsky, D., & Slomp, J. (2012). Flexible PMP Approach for Large-Size Cell Formation. Operations Research, 60(5), 1157-1166. doi: 10.1287/opre.1120.1108

[6] Gonçalves, J. F., & Resende, M. G. C. (2004). An evolutionary algorithm for manufacturing cell formation. Computers & Industrial

Engineering, 47(2–3), 247-273. doi: http://dx.doi.org/10.1016/j.cie.2004.07.003

[7] James, T. L., Brown, E. C., & Keeling, K. B. (2007). A hybrid grouping genetic algorithm for the cell formation problem. Computers & Operations Research, 34(7), 2059-2079. doi: http://dx.doi.org/10.1016/j.cor.2005.08.010

[8] Keeling, K. B., Brown, E. C., & James, T. L. (2007). Grouping efficiency measures and their impact on factory measures for the machine-part cell formation problem: A simulation study. Engineering Applications of Artificial Intelligence, 20(1), 63-78. doi: 10.1016/j.engappai.2006.04.001

[9] Krause, J., Cordeiro, J., Parpinelli, R. S., & Lopes, H. S. (2013). 7 - A Survey of Swarm Algorithms Applied to Discrete Optimization Problems. In X.-S. Y. C. X. H. G. Karamanoglu (Ed.), Swarm Intelligence and Bio-Inspired Computation (pp. 169-191). Oxford: Elsevier.

[10] Krushinsky, D., & Goldengorin, B. (2012). An exact model for cell formation in group technology. Computational Management Science, 9(3), 323-338. doi: 10.1007/s10287-012-0146-2

[11] Hansen, P., 1986. The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. Presented at the Proceedings of the Congress on Numerical Methods in Combinatorial Optimization.

[12] Li, X., Baki, M. F., & Aneja, Y. P. (2010). An ant colony optimization metaheuristic for machine–part cell formation problems. Computers & Operations Research, 37(12), 2071-2081. doi: http://dx.doi.org/10.1016/j.cor.2010.02.007

[13] Luo, J., & Tang, L. (2009). A hybrid approach of ordinal optimization and iterated local search for manufacturing cell formation. The International Journal of Advanced Manufacturing Technology, 40(3-4), 362-372. doi: 10.1007/s00170-007-1346-8

[14] 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(3, Part 2), 6598-6604. doi: http://dx.doi.org/10.1016/j.eswa.2008.07.054

[15] Mitrofanov, S. P. (1966). Scientific Principles of Group Technology: National Lending Library for Science and Technology.

[16] 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. doi: 10.1016/j.ejor.2009.10.020

[17] PARASHAR, B. S. N. (2008). CELLULAR MANUFACTURING SYSTEMS: An Integrated Approach: PHI Learning.

[18] Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. Information Sciences, 179(13), 2232-2248. doi: 10.1016/j.ins.2009.03.004

[19] Rojas, W., Solar, M., Chacón, M., & Ferland, J. (2004). An Efficient Genetic Algorithm to Solve the Manufacturing Cell Formation Problem. In I. C. Parmee (Ed.), Adaptive Computing in Design and Manufacture VI (pp. 173-183): Springer London.

[20] Sarker, B. R. (2001). Measures of grouping efficiency in cellular manufacturing systems. European Journal of Operational Research, 130(3), 588-611. doi: http://dx.doi.org/10.1016/S0377-2217(99)00419-1

[21] Suresh, N. C., & Kay, J. M. (1998). Group Technology and Cellular Manufacturing: State-Of-The-Art Synthesis of Research and Practice: Kluwer Academic Pub.

[22] Tunnukij, T., & Hicks, C. (2009). An Enhanced Grouping Genetic Algorithm for solving the cell formation problem. International Journal of Production Research, 47(7), 1989-2007. doi: 10.1080/00207540701673457

[23] Wu, T. H., Low, C., & Wu, W. T. (2004). A tabu search approach to the cell formation problem. The International Journal of Advanced Manufacturing Technology, 23(11-12). doi: 10.1007/s00170-003-1766-z

[24] Batsyn, M., Bychkov, I., Goldengorin, B., Pardalos, P., & Sukhov, P. (2013). Pattern-based heuristic for the cell formation problem in group technology. In Models, Algorithms, and Technologies for Network Analysis (pp. 11-50). Springer, New York, NY.

[25] Sarker, B. R., & Li, Z. (2001). Job routing and operations scheduling: a network-based virtual cell formation approach. Journal of the Operational Research Society, 673-681.

[26] Zettam, M., & Elbenani, B. (2016). Gravitational Search Algorithm Applied to the Cell Formation Problem. In Nature-Inspired Computation in Engineering(pp. 251-266). Springer International Publishing.