ISSN 2007-9737

Memetic Algorithm with Hungarian Matching Based Crossover and Diversity Preservation

Emmanuel Romero Ruiz[1], Carlos Segura[2]

[1] Universidad de Guanajuato, Guanajuato, Mexico

[2] Centro de Investigación en Matemáticas (CIMAT), Área de Computación, Guanajuato, Mexico

{emmanuel.romero, carlos.segura}@cimat.mx

Abstract. The Graph Partitioning Problem (GPP) is a well-known NP-hard combinatorial problem that involves the finding of a partition of vertexes that minimizes the number of cut edges while fulfilling a set of constraints. This paper presents a newly designed optimizer for the GPP: the Memetic Algorithm with Hungarian Matching Based Crossover and Diversity Preservation (MAHMBCDP). MAHMBCDP is a population-based scheme that incorporates an explicit mechanism to control the diversity with the aim of making a proper use of resources when dealing with long-term executions. Among the novelties of our proposal, the design of a crossover operator that is based on the Hungarian Algorithm to calculate a maximum matching is particularly important. Experimental validation with a set of well-known instances of the graph partitioning archive shows the proper performance of our proposal. In fact, new best-known solutions could be attained in ten test cases.

Keywords. Graph partitioning problem, memetic algorithm, diversity preservation, maximum matching.

1 Introduction

The Graph Partitioning Problem (GPP), is a well-known NP-hard [10], combinatorial optimization problem that has been addressed with a large number of different techniques [4]. One of the reasons why the GPP is probably so popular is because it arises in several practical cases [6]. For instance, it has been used for the mesh partitioning in simulators that apply the finite element and/or the finite volume paradigms, in the implementation of parallel software, and in the design of circuit layouts, among others.

Solving the GPP means finding a partition of the vertexes of a given graph that fulfill a balance constraint among the sizes of the different subsets and at the same time minimizes the number of cut edges, i.e. the number of edges that run between vertexes of different subsets. Note that, in the related literature, the terms subset, parts and classes are used to refer to each one of the groups of vertexes established by the partition. Thus, any of these concepts are used indifferently in this paper.

In many of the applications of the GPP quite large graphs arise [14]. Given the complexity class of the GPP, exact solvers, such as some based on quadratic programming formulations [11], are nowadays not applicable to practical graphs. As a result, several heuristics and metaheuristics have been proposed with the aim of obtaining high-quality approximate solutions for such large cases. Initially, several constructive and improvement heuristics were devised [15]. While they are fast, the quality of the obtained partitions is not so large. Subsequently, trajectory-based metaheuristics were adapted to deal with the GPP [3], reaching much higher quality solutions at the cost of an increase in the required computational budget. Finally, nowadays hybrid population-based metaheuristics lead the attainment of high-quality solutions. In fact, most of the best-known solutions that have been attained for the instances that are maintained in the graph partitioning archive [13], have been obtained with population-based approaches. Particularly, memetic algorithms that combine an Evolutionary Algorithm (EA) with local search or with trajectory-based metaheuristics seem to be the most promising approaches [8].

There are several problems that are closely related to the GPP. One of them is the Frequency Assignment Problem (FAP). In the FAP, a set of partitions must also be identified. However, the objective is related to minimizing the number of edges that appear between vertexes that belong to the same class, instead of vertexes that belong to different classes [1]. In any case, some of the ideas that are used in the optimization of the FAP can be applied to the GPP. In a recent research [25] a novel memetic algorithm could provide an important advance in the solving of the FAP. In fact, several new best-known solutions could be attained for some well-known benchmarks. The most important novelty of the solver was the application of a mechanism to explicitly control the diversity of the population. Particularly, the main aim is to lose the diversity in a controlled way that depends on the stopping criterion set by the user. In this way, shorter stopping criteria induce a faster convergence.

The hypothesis behind the development of this research is that the kinds of mechanisms that provide benefits in the FAP might also help in the development of optimizers for the GPP. We based our hypothesis on two aspects. First, on the similarities that exist between the current optimizers for the FAP and GPP. Second, in the fact that state-of-the-art population-based approaches reach convergence of the population in relatively few generations. For instance, in [8], only 200 generations are evolved because after such number of generations all the individuals of the population are located in a similar region and no further improvements are obtained with larger executions. Since the appearance of premature convergence in typical EAs seems to be a key for the success of the schemes that control the diversity in an explicit way, applying variants of this proposal seems truly promising. Note that another particular feature of the method developed in [25], is that it applies a computationally expensive local search with a large neighborhood and ad-hoc crossover operators. As a result, the memetic algorithm developed in this paper also considers a hill climber with a large neighborhood [22] and a newly designed crossover operator adapted to the GPP.

Experimental validation has been performed with a set of test cases of the well-known graph partitioning archive. Particularly, 15 graphs with up to $16,840$ nodes have been taken into account. The partitions obtained with these graphs for different number of subsets show the proper performance of our proposal. In fact, new best-known solutions could be obtained in ten test cases. The most important advances appear in not too large graphs and with few subsets. The increase in the number of edges, vertexes and/or subsets implies a drastic decrease in the number of generations that can be evolved, meaning that the advantages of controlling the diversity in an explicit way diminish.

The rest of the paper is organized as follows. The mathematical formulation of the GPP is presented in Section 2. Related work is discussed in Section 3. Our novel proposals are detailed in Section 4. Section 5, is devoted to our experimental validation. Finally, our conclusions and future work are presented in Section 6.

2 Graph Partitioning Problem: Mathematical Definition

This section provides a formal definition of the GPP [2]:

— Let $G = (V, E, w)$, be an undirected graph where V is the set of nodes, E the set of edges and $w : E \to \mathbb{R}_{>0}$, a weight function.

— In the following $n = \|V\|$ and $m = \|E\|$.

— Let k be a fixed positive integer that represents the number of classes in the desired partition and $\epsilon > 0$ a real number that is related to the balance constraint.

— The feasible search space for the problem is the set of partitions $\Delta = \{S = \{S_1, S_2, ..., S_k\}$ such that S is a partition into k sets of V and $\|S_i\| \leq L_{max} := (1 + \epsilon)\lceil n/k \rceil$ for all $i \in \{1, 2, .., k\}$ }.

— Finally the function to minimize in the GPP is:$f(S) = \sum_{i<j} w(E_{ij})$ where $E_{ij} = \{(u, v) \in E : u \in S_i, v \in S_j\}$ and $w(C) = \sum_{e \in C} w(e)$.

The experimental validation of this paper takes into account the instances present in the Chris Walshaw's Graph Partitioning Archive [13]. In these instances, all the weights are equal to 1, so the objective is to minimize the number of conflicting edges. Moreover, the archive reports results for distinct values of ϵ. However, this paper focuses in the case where $\epsilon = 0$. Some minor modifications are required to deal with different ϵ values. However, performing these adaptations is out of the scope of this work.

3 Related Work

This section is devoted to reviewing some of the works that are closely related to our approaches. First, among the large set of EAs that have been used to tackle the GPP, some of the ones that have obtained the most promising results, as well as the ones that share several features with our proposal are discussed. In addition, since our proposal is based on controlling the diversity explicitly, the most popular techniques that have been devised with the aim of alleviating the effects of premature convergence are summarized. Some of these techniques are used to validate our proposal.

3.1 EAs for the GPP

A large amount of population-based metaheuristics has been designed to deal with the GPP [4]. Among them, EAs are probably the most popular approaches. Since the initial proposals [5], it was clear that incorporating a procedure to intensify in the regions located by the EA was quite important. In most of the initial algorithms, the heuristic proposed by Kernighan and Li was taken into account [15].

This heuristic is an inexpensive process that attains competitive partitions. More complex ways of intensifying have been defined with the increase of the computational power. In some cases, EAs are integrated with trajectory-based strategies [8], whereas in other ones, hill-climbers with larger neighborhoods are applied [22]. These kinds of approaches imply the use of additional computational resources, but they have allowed improving further the best-known solutions in several large graphs. Thus, in our proposal, the relatively complex neighborhood defined in [22], is taken into account.

The design of proper crossover operators has also implied large research efforts. In the initial approaches, general operators such as the uniform and/or the n-point crossover were applied [17]. However, it was soon clear that these operators were quite disruptive. Particularly, they had a tendency to create quite unbalanced partitions, meaning that after the balancing mechanisms, the offspring might share just a few features with their parents [4]. An attempt to avoid this issue was devised in [6]. In this case, the vertexes that share the same class in both parents maintain their class in the offspring.

Then, the remaining vertexes are set by using a greedy constructive heuristic that ensures a proper balance. One fact that is not taken into account in the previous operator is that the specific class identifier of a vertex is not really important. The important features are related to the set of vertexes that share the same class. Taking this into account, more complex crossover operators that are based on inheriting these kinds of features have been devised [9, 8]. The principle behind these schemes is to transmit large sets of vertexes that share the same class between parents and offspring. In [8], a greedy approach based on maximizing intersections is used. More complex non-greedy proposals were previously suggested in [9]. However, in this last case, it was applied to the graph coloring problem and not to the GPP. Since these kinds of operators have reported really promising results, our proposal also takes this principle into account. Particularly, some of the ideas suggested in [9], inspired the development of our novel crossover operator.

However, our operator also induces the creation of relatively balanced partitions in the offspring, which is an important feature in the GPP but not in the graph coloring problem.

Finally, an important issue that has not been studied in depth is the management of diversity. However, some of the analyses show that the proper control of diversity might be an issue for the proper performance. In [8], the number of generations was limited to 200. Authors argue that after this number of generations, all the individuals are located in the same region and that further improvements are not expected with larger executions. Additionally, in many cases, some actions to delay the convergence are included. For instance, in [5] offspring substitute the most similar parent, whereas resorting to restarting mechanisms when convergence is detected is another typical approach [19].

3.2 Control of Diversity in EAs

Since our approach incorporates a novel way of controlling the diversity, this subsection discusses some of the most popular strategies that have been designed with the aim of avoiding premature convergence. Particularly, the techniques that have been used in the experimental validation of our proposal are described. Readers are referred to [28] for additional techniques and for more complete descriptions. Note that, a large number of techniques to deal with premature convergence have been devised [20].

The methods are usually classified in base of the component of the EA that they modify in the following groups [28]: *selection-based*, *population-based*, *crossover/mutation-based*, *fitness-based*, and *replacement-based*. Additionally, depending on the number of altered components, they are referred to as *uniprocess-driven* or *multiprocess-driven*. Although the development of multiprocess-driven schemes might result in higher-quality solutions, currently most of the efforts have been placed on the design of uniprocess-driven strategies. Controlling several related stochastic components simultaneously is much more complex, so probably for this reason multiprocess-driven schemes are not so popular. As a result, our proposal is a uniprocess-driven strategy. Among the different strategies, the replacement-based methods have reported quite promising results with several optimization problems [26, 23]. Since our novel proposal belongs to this category, three additional replacement-based schemes are taken into account in this paper:

— The *Restricted Tournament Selection* [12], (RTS) is a popular steady-state scheme in which after each new individual (O) is created, CF individuals from the current population are selected at random. Then, the best individual between O and its most similar individual in the selected set survives. Note that this scheme belongs to the group of crowding strategies [18].

— In the *Hybrid Genetic Search with Adaptive Diversity Control* [29] (HGSADC), individuals are sorted by their contribution to diversity and by their original cost. Then, the rankings of the individuals are used to calculate a score using two parameters (N_{Close} and N_{Elite}) which are used to determine the survivors.

— The *clearing* strategy (CLR) [21] is an extension of fitness sharing and it alters both the fitness assignment procedure and the replacement selection phase. In the clearing procedure, individuals are grouped into niches — defined via the parameter σ — and the resources of a niche are attributed to the best W elements in each niche. Moreover, the winners of each niche are copied to the next population. In order to avoid a large immobilization of the population, the winners with a fitness lower than the mean are discarded.

Note that, in addition to these techniques, a generational scheme with elitism [7], (GEN_ELIT) is also incorporated in our experimental validation. This replacement phase does not incorporate any special mechanism to delay convergence. However, since it is one of the mostly applied strategies, it was taken into account. In GEN_ELIT the new population contains the offspring and the best solution of the previous generation.

4 Proposal

Our proposal (MAHMBCDP) is an EA that incorporates a local search strategy. The general algorithm is a quite standard Memetic Algorithm (see Algorithm 1). The only modification to the general approach is the incorporation of a balancing strategy to fulfill the GPP constraints. In order to fully define our proposal, the genetic operators, balancing operator, local search and replacement strategy must be described. The next subsections details each of these components.

Algorithm 1 Diversity-based Lamarckian Memetic Algorithm

```
1: **Initialization**: Generate an initial population $P_0$ with $N$ individuals. Assign $t = 0$.
2: **Local Search**: Perform a local search for every individual in the population.
3: **while** (not stopping criterion) **do**
4:     **Mating selection**: Perform binary tournament selection on $P_t$ in order to fill the mating pool.
5:     **Variation**: Apply genetic operators to the mating pool to create a child population $CP$.
6:     **Balancing** Apply a balancing phase because of the problem's balance restrictions.
7:     **Local Search**: Perform a local search for every individual in the offspring.
8:     **Survivor selection**: Combine $P_t$ and $CP$, and apply the BNP survivor selection strategy.
9:     $t = t + 1$
10: **end while**
```

Algorithm 2 Hungarian Based Crossover

```
1: Let $elg\_r$ a boolean vector for the "eligible" rows.
2: Let $elg\_c$ a boolean vector for the "eligible" columns.
3: Let $blc\_r$ a boolean vector for the "blocked" rows.
4: Let $blc\_c$ a boolean vector for the "blocked" columns.
5: The $elg$ vectors are set to true.
6: The $blc$ vectors are set to false.
7: **for** $i$ in $\{1, 2, ..., n\}$ **do**
8:     **if** $i \equiv 0, mod(2)$ **then**
9:         $L = NULL$
10:        **for** $l$ such that $elg\_c[l]$ is true (in random order) **do**
11:            $B_l = \bigcup_{k,m}\{$ $\{P_{k,l}$ not erased$\} \cup \{P_{\sigma^{-1}(l),m}$ such that $blc\_c[m]$ is true $\}$ $\}$.
12:            If $L == NULL$ or $|B_l| > |B_L|$ set $L = l$.
13:        **end for**
14:        $T_j = B_L$.
15:        Erase from $P$ all the elements of $T_j$.
16:        $elg\_c[L] =$ false.
17:        $elg\_r[\sigma^{-1}(L)] =$ false.
18:        $blc\_r[\sigma^{-1}(L)] =$ true.
19:    **else**
20:        $L = NULL$
21:        **for** $l$ such that $elg\_r[l]$ is true (in random order) **do**
22:            $B_l = \bigcup_{k,m}\{$ $\{P_{l,k}$ not erased$\} \cup \{m, P_{\sigma(l)}$ such that $blc\_r[m]$ is true $\}$ $\}$.
23:            If $L == NULL$ or $|B_l| > |B_L|$ set $L = l$.
24:        **end for**
25:        $T_j = B_L$.
26:        Erase from $P$ all the elements of $T_j$.
27:        $elg\_r[L] =$ false.
28:        $elg\_c[\sigma(L)] =$ false.
29:        $blc\_c[\sigma(L)] =$ true.
30:    **end if**
31: **end for**
```

4.1 Crossover Operator

As usual, the crossover operator takes two individuals and generates two new ones with some similar features to the parents. Many of the crossover operators that are used for the GPP are inspired in those developed for the graph coloring problem. Our proposal considers a newly designed crossover operator, the Hungarian Based Crossover (HBX) — see Algorithm 2 — that is inspired by the one devised in [9]. Particularly, it is based on the principle of maximizing the number of nodes that share the same class both in the parents and offspring.

The HBX operator works as follows. Let $I_1 = \{S_1^1, S_2^1, ..., S_k^1\}$ and $I_2 = \{S_1^2, S_2^2, ..., S_k^2\}$ be the parents selected for the crossover operator. The Hungarian method is a well-known combinatorial optimization algorithm that solves, in polynomial time, the assignment problem in weighted bipartite graphs, i.e. it calculates the maximum or minimum weight matching. In this case, a complete bipartite graph (G) with $2 \times k$ vertexes is built. The first k vertexes are associated to the subsets of I_1, whereas the last ones correspond the subsets of I_2. The edges connect any of the first k vertexes with any of the last k vertexes. The weight of each edge is established as the size of the intersection of the sets associated with each vertex.

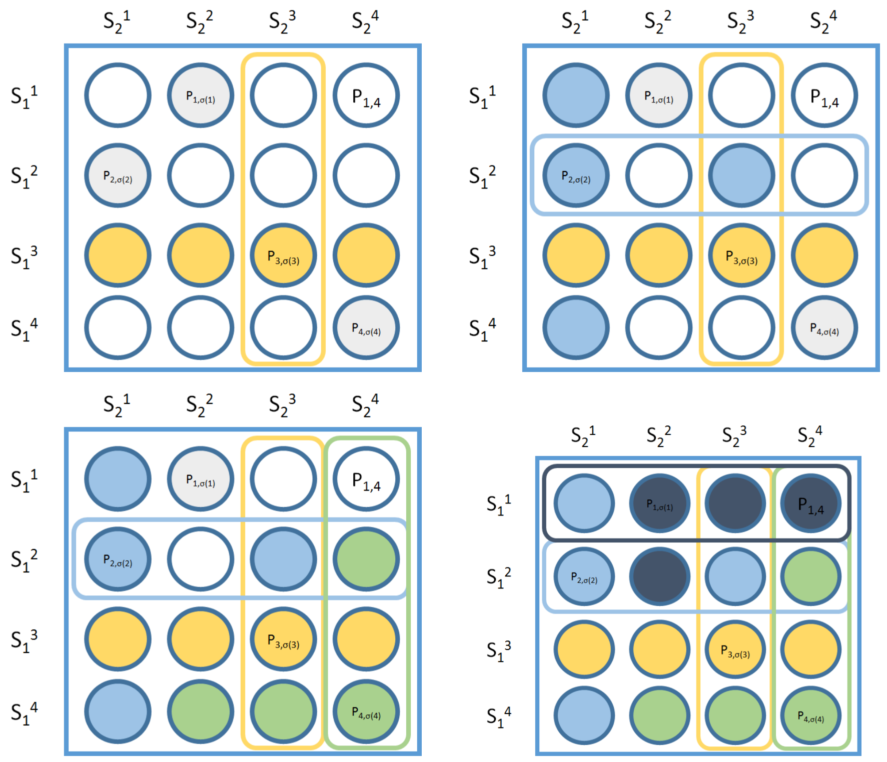


Fig. 1. Illustration of the HBX

The permutation σ obtained with the application of the Hungarian Algorithm to G maximizes: $\sum_{i=1}^{k} \|S_i^1 \cap S_{\sigma(i)}^2\|$.

4.1.1 Main Properties of HBX

Previously to giving the specific crossover procedure, it is important to discuss some of the properties that share the offspring. Let P be an intersection matrix defined in a way that $P_{i,j} = S_i^1 \cap S_j^2$, and let $T = \{T_1, T_2, ..., T_k\}$ be the offspring. Then, T has the following properties:

1. For each j, there is a unique i_j such that $S_{i_j}^1 \cap S_{\sigma(i_j)}^2 \subset T_j \subset S_{i_j}^1 \cup S_{\sigma(i_j)}^2$.
2. $P_{i,j} \subset T_l$ for all the pairs (i, j) and some l.
3. T_l is the union of exactly k distinct $P_{i,j}$.

The reason behind the use of σ is that the Hungarian Algorithm gives the maximum matching between the sets in both partitions. This means that S_i^1 is somewhat similar to $S_{\sigma(i)}^2$ and their intersection is susceptible to have a lot of elements. The property $1)$ establishes that these intersections are used as the core of the sets in the new partition. It also establishes that the elements are selected exclusively from the unions of the considered subsets.

Taking these properties into account, it can be stated that operators that share these properties are not very destructive operators. This because property $2)$ focuses on inheriting together large subsets of vertexes that were assigned to the same subset in both parents. Finally, the principle behind the third property, is to avoid an excessive imbalance in the sizes of the newly generated subsets. However, note that some imbalance usually appears, so a method to properly balance the subsets is required.

ISSN 2007-9737

Algorithm 3 Mutation based on Random Connected Component

```
1: Let $w$ be a random node in the graph.
2: Let $set$ be a set that contains $w$.
3: $k = n_{iter}$
4: while $k > 0$ do
5:     for $u$ in $set$ do
6:         for $v$ neighbor of $u$ do
7:             Take $p$ random in $[0, 1]$.
8:             if $p < p_m$ then
9:                 Add $v$ to $set$
10:            end if
11:        end for
12:    end for
13:    $k - -$
14: end while
15: Move all the vertexes in $set$ to a randomly selected
    subset
```

4.1.2 Construction

This section is devoted to present the specific algorithm used to build two offspring that fulfill the aforementioned properties. Let P be the intersection matrix previously defined. Then Algorithm 2 is the process required to create the first new individual. This process is illustrated in Fig. 1. The main principle is to select in each step the position $(i, \sigma(i))$ of the matrix, where the union of some elements of the corresponding row and column (see lines 11 and 22 for details) is maximized. Note that it can be proved that in line 11 there are exactly $k - j/2$ distinct $P_{i,l}$ not erased from P and $j/2$ $P_{\sigma^{-1}(l),m}$ such that the column m is "blocked". Something similar is true in line 22 and this proves the property 3 of HBX.

In order to generate the second offspring, the order in which rows and columns are considered is exchanged, i.e. the even iterations take into account the columns, whereas the odd iterations take into account the rows. Note that regardless of the order considered, both individuals share the same properties described above.

In Fig. 1 the evolution of HBX for a specific individual is shown. In this case, $k = 4$, $\sigma = (2, 1, 3, 4)$. Each of the four images in Fig. 1 can be considered as a matrix, P. Each entry, $P_{i,j}$, of these matrices represent the intersection of the corresponding sets in the parents S^1, S^2.

In the last image, the union of the sets with the same color represents an element in the new partition, T. The first image shows how T_1 is chosen as a row from P and how column 3 is blocked. Note that the row was chosen randomly and the column considered the Hungarian permutation. In the second image, part of T_2 is taken from a column of P and the remaining elements are taken from sets in row 2 that are in a blocked column ($P_{2,3}$). Similarly in the image 3 and 4, T_3 and T_4 are created. Everything follows the procedure described in Algorithm 2

Another important fact of the novel operator is that no information belonging to the GPP definition was used in the design. Thus, this crossover operator might be used in any optimization problem in which sharing the same value in different variables is an important feature.

4.2 Mutation Operator

In the GPP it is desirable to keep connected components together in one subset of the partition. The reason is that connected components encapsulates common edges, meaning that lower cut edges might be induced. Taking this property into account, the principle of the mutation operator is to move parts of connected components from one subset to another one. In order to select this subcomponent, the same approach than the one presented in [25] is applied. Then, the selected vertexes are moved to a set S_i, where i is selected in a random way. Algorithm 3 describes the process. In this work we use $p_m = 0.1$ and $n_{iter} = 5$.

4.3 Balancing Operator

The balancing routine consists of two phases and it is very important because of the problem's balancing restriction. This restriction might not be valid after the application of the crossover and/or mutation operator so it has to be restored. The procedure that is in charge of this process is given in Algorithm 4.

In the first phase, nodes are moved from larger to smaller sets by taking into account the implications of the move on the edge cut. This process

Algorithm 4 Balancing Operator

```
1: Let $ind$ the input individual to balance.
2: $k = n_{iterBal}$
3: while $k > 0$ and $ind$ is not balanced do
4:     Select $i$ randomly in $\{i$ such that $\|S_i\|$ is not biggest set of $ind\}$.
5:     Find $u$ the node of maximum gain among $\cup S_j$ such that $\|S_j\| > \|S_i\|$.
6:     Move $u$ to $S_i$.
7:     $k - -$
8: end while
9: while $ind$ is not balanced do
10:    Choose $u \in \cup S_j$ such that $\|S_j\| > S_{max}$.
11:    Move $u$ to a random set $S_k$ with $\|S_k\| < S_{max}$.
12: end while
```

is repeated $n_{iterBal}$, which is set to $|V|$ in our experimental validation.

The second phase, which ensures the attainment of a properly balanced solution, moves randomly selected nodes from sets that do not fulfill the given restriction to subsets where there is space available.

4.4 Local Search

In order to create individuals with better fitness, a local search algorithm was added. This routine is performed after the balancing procedure and it consists of two phases.

In the first phase, the strategy that is called the "perfectly balanced local search by negative cycle detection" in [22] is carried out. The principle behind this algorithm is to find a cycle of negative gain and move the nodes around it. In this way the balance is kept and the fitness is diminished.

The second phase is a stochastic hill climbing. For this routine two individuals are neighbor if their only difference is a swap in an edge. It means, if (u, v) is an edge and individual I_1 fulfill that $u \in S_{c_u}$ and $v \in S_{c_v}$, with $c_u \neq c_v$, then individual I_2 such that $u \in S_{c_v}$ and $v \in S_{c_u}$ is neighbor of I_1 if and only if the previous one is their only difference. Once the notion of neighborhood is defined, a usual stochastic hill climbing is applied [27].

Algorithm 5 BNP survivor selection technique

```
Require: Population, Offspring
1: for all I $\in$ Offspring do
2:     I.cost = edge cut associated with individual I
3: end for
4: Penalized = $\emptyset$
5: CurrentIndividuals = Population $\cup$ Offspring
6: Best = Individual with lowest edge cut in CurrentIndividuals
7: NewPop = { Best }
8: CurrentIndividuals = CurrentIndividuals $\setminus$ { Best }
9: Update D taking into account the elapsed time ($T_e$), stopping criterion ($T_s$) and initial value of D ($D_I$): $D = D_I - D_I * \frac{T_e}{T_s}$. We took $D_I = distInitFactor * M$, where $M$ is the mean distance in the first population and $distInitFactor \in [0, 1]$.
10: while (|NewPop| $<$ N) do
11:     for all I $\in$ CurrentIndividuals do
12:         I.DCN = distance to the closest individual of $I$ in NewPop
13:         if $I$.DCN $< D$ then
14:             I.cost = Infinity
15:             Penalized.insert(I)
16:             CurrentIndividuals.erase(I)
17:         end if
18:     end for
19:     for all I $\in$ Penalized do
20:         I.DCN = distance to the closest individual of $I$ in NewPop
21:     end for
22:     if CurrentIndividuals is empty then
23:         Selected = I with largest I.DCN in Penalized
24:         Penalized = Penalized $\setminus$ {Selected}
25:     else
26:         Selected = I with lowest I.cost in CurrentIndividuals
27:         CurrentIndividuals = CurrentIndividuals $\setminus$ {Selected}
28:     end if
29:     NewPop = NewPop $\cup$ Selected
30: end while
31: Population = NewPop
```

4.5 Survivor Selection

Our memetic algorithm applies a novel replacement phase which is called the Best-Non-Penalized (BNP) survivor selection strategy (see Algorithm 5). One of the principles of the BNP strategy is to avoid the selection of too close individuals. Specifically, the approach tries to avoid the selection of pairs of individuals that are closer than a D value. Since in the initial phases it is important to explore, whereas in the last phases the process should intensify, the D value varies during the execution. Particularly, the variable D is initialized in base of the content of the first population (see line 9) and it is decreased in a linear way.

In each execution of the survivor operator N individuals from the previous population and offspring are selected to survive. BNP iteratively selects the best element that is not penalized. Then, all the remaining individuals with distance at most D to the previously selected ones are penalized. If it is not possible to find a non-penalized individual, the individual with larger distance to the currently selected individuals is taken.

Note that the principles of BNP are similar to the ones that guided the design of the *Replacement with Multi-objective based Dynamic Diversity Control strategy*(RMDDC) [25]. The main difference is that in RMDDC the same importance is given to quality and diversity, so a multi-objective selection is used. In the case of GPP, due to the high computational cost associated with some components, not too many generations can be evolved. Thus, the incorporation of an additional bias towards high-quality solutions that is performed in the BNP strategy is mandatory to obtain high-quality solutions with the stopping criterion set in our analyses.

An important fact that should be noticed is that the survivor operator requires a distance-like function between individuals. In this case, a function based on the Hungarian Algorithm is used. Let τ be a permutation of k elements, then $d(S^1, S^2, \tau) = |V| - \sum_{i=1}^{k} \|S_i^1 \cap S_{\tau(i)}^2\|$. Our distance is defined as the minimum $d(S^1, S^2, \tau)$ over all the τ's. This value can be calculated efficiently by using the Hungarian Best Matching Algorithm.

5 Experimental Validation

In this section, the experiments conducted with our proposal (MAHMBCDP) are described. Since our proposal is a stochastic algorithm, each execution was repeated 30 times and comparisons were carried out by applying a set of statistical tests. A similar guideline as the one applied in [24] was used. Specifically, the following tests were applied, assuming a significance level of 5%. First, a *Shapiro-Wilk test* was performed to check whether or not the values of the results followed a Gaussian distribution.

If so, the *Levene test* was used to check for the homogeneity of the variances. If samples had equal variance, an ANOVA *test* was done; if not, a *Welch test* was performed. For non-Gaussian distributions, the non-parametric *Kruskal-Wallis* test was used to test whether samples are drawn from the same distribution. The following section describes the set of benchmarks and experiments that have been taken into account to validate our proposal.

Our approach involves the setting of some parameters. The population size was set to 50, the crossover probability (p_c) was set to 0.85 and the mutation probability (p_m) was set to 0.1. These values are quite standard values and we are aware that some improvements might be obtained by applying parameter setting schemes [16]. However, since very large executions were performed (the stopping criterion was set to 48 hours) and they provided promising results no further experimentation was carried out.

Our validation involves the application of several replacement strategies in addition to the BNP strategy used in MAHMBCDP. Depending on the replacement phase, additional parameters might be required. In order to set them, some initial experiments were developed by considering some values that have reported promising results [23] in other problems, as well as some modifications of these values. The ones that obtained the best performance are the following. In the RTS, CF was set to 25. In HGSADC, N_{Close} and N_{Elit} were set to 3 and 8, respectively. In CLR, σ was set to $0.20 \times M$, where M is the mean distance among individuals of the first population.

Finally, in the BNP strategy, $distInitFactor$ was set to 0.6. More information on the sensitivity of this parameter is given in the following subsections.

Table 1. Features of the graphs selected for the experimental validation

Graph	$\|V\|$	$\|E\|$	Graph	$\|V\|$	$\|E\|$
add20	2395	7462	data	2851	15093
3elt	4720	13722	uk	4824	6837
add32	4960	9462	bcsstk33	8738	291583
whitaker3	9800	28989	crack	10240	30380
wing_nodal	10937	75488	fe_4elt2	11143	32818
vibrobox	12328	165250	bcsstk29	13992	302748
4elt	15606	45878	fe_sphere	16386	49152
cti	16840	48232			

5.1 Benchmark

In order to evaluate the performance of our proposal, test cases from the graph partitioning archive (TGPA) of Chris Walshaw [13], were used. This is a site that has been maintained since 2000 and includes results from most of the important partitioning software packages. The whole set is composed of 34 graphs.

The smallest graph has $2,395$ nodes and $7,462$ edges and the biggest has $448,695$ nodes and $3,314,611$ edges. The number of edges in these graph are usually between two and twenty times the number of nodes. For each of these graphs, the best known partitions for $k \in \{2,4,8,16,32,64\}$ and $\epsilon \in \{0.0,0.01,0.03,0.05\}$ are available. The values $k \in \{4,8,16,32,64\}$ and $\epsilon = 0.0$ were taken into account for testing our algorithm. We chose $k = 2^n$ in order to compare our results with the ones in TGPA, but the scheme can be applied for any arbitrary k.

In this work, the first 15 graphs in TGPA were selected. The reason is that we identified that our algorithm was not able to evolve a reasonable number of generations (more than $1,000$), in graphs with more than $17,000$ edges, which is quite important for the proper performance of diversity-based memetic algorithms. Table 1 details the number of nodes and edges of the selected graphs.

5.2 Analysis of Diversity

One of the most important features of our approach is that it considers the diversity explicitly. Our proposal depends on a parameter, $distInitFactor \in [0,1]$. In order to use the same parameter value in every case, a preliminary experiment was performed.

Particularly, five different equidistributed values were used with two instances and two different values of k. The mean and best results obtained for 30 executions are shown in Table 4 for each of the tested values. The values 0.6 and 0.8 reported quite competitive results. Since the value 0.6 reported a lower mean in more cases, this value is used in all the remaining experiments.

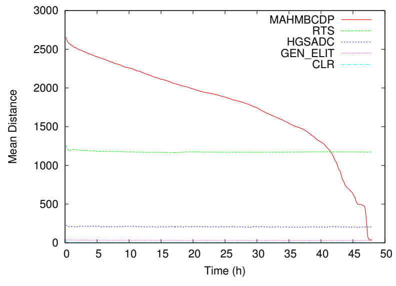
Table 2. Comparison among different schemes for the 3elt graph

3elt						
	$k=4$			$k=64$		
Scheme	Best	Mean	Worst	Best	Mean	Worst
MAHMBCDP	**201**	**201**	**201**	**1535**	**1537.1**	**1539**
RTS	201	210	463	1538	1551.4	1569
HGSADC	201	262.2	463	1536	1546.8	1558
CLR	291	389.3	517	1601	1692	1796
GEN_ELIT	237	402.4	532	1545	1555.6	1569

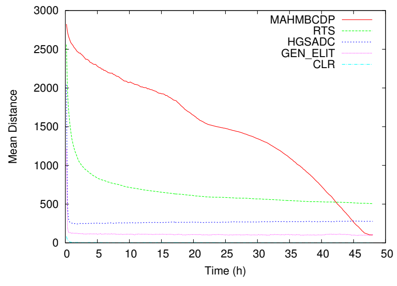
For validating the BNP survivor selection integrated in MAHMBCDP, some additional popular schemes were also tested. Specifically, results were obtained with the following methods: RTS, HGSADC, CLR and GEN_ELIT. Table 2 summarizes the results obtained with the *3elt* graph. Specifically, the best, mean and worst values are reported for $k = 4$ and $k = 64$. The same information is shown in Table 3 for the *uk* instance. It is clear that MAHMBCDP obtained the most promising results. In fact, the statistical tests described before were applied and showed the superiority of MAHMBCDP in every case. Particularly, the maximum p value was equal to 0.00013 in the comparisons.

In order to properly understand the reasons behind the superiority of the BNP technique it is important to analyze the evolution of the diversity and fitness (cut edge). Fig. 2 shows the evolution of the diversity whereas Fig. 3 shows the evolution of the fitness for the aforementioned cases. The diversity was measured as the mean distance among all the individuals in the population.

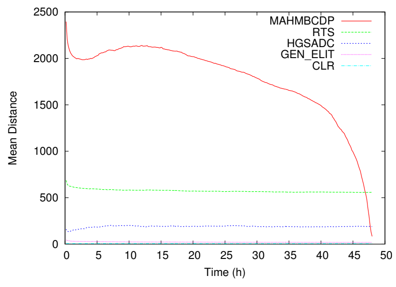
It is noticeable that the only scheme that induces a gradual decrease in diversity is MAHMBCDP, i.e. the scheme that applies the BNP strategy.



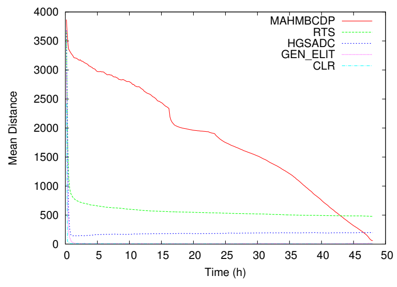
(a) *3elt*, k = 4



(b) *3elt*, k = 64

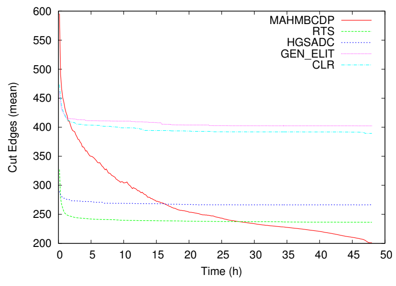


(c) *uk*, k = 4

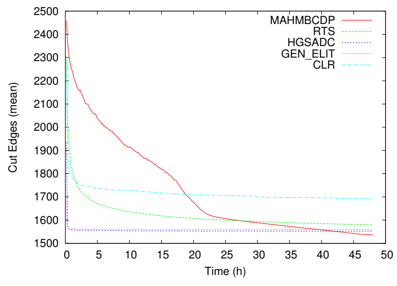


(d) *uk*, k = 64

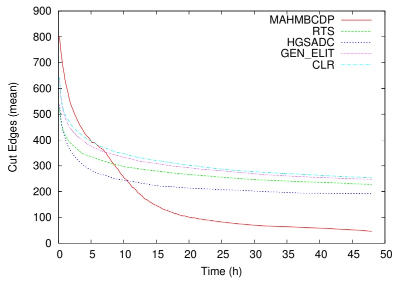
Fig. 2. Evolution of diversity in four different test cases



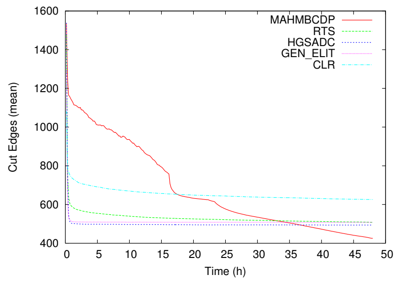
(a) *3elt*, k = 4



(b) *3elt*, k = 64



(c) *uk*, k = 4



(d) *uk*, k = 64

Fig. 3. Evolution of fitness in four different test cases

Table 3. Comparison among different schemes for the uk graph

uk						
	$k = 4$			$k = 64$		
Scheme	Best	Mean	Worst	Best	Mean	Worst
MAHMBCDP	**41**	**45.8**	**73**	**420**	**425.2**	**430**
RTS	153	198.73	249	444	460	489
HGSADC	119	187.13	249	450	488.1	532
CLR	195	253.2	329	585	625.7	686
GEN_ELIT	194	247.5	322	466	507.1	489

The large diversity in the initial phases explains why MAHMBCDP does not attain very promising partitions initially. In this phase, it is looking for promising regions in the variable space but intensification is not promoted. At the end of the execution, the diversity is low, meaning that intensification is promoted in the most promising regions identified. This results in solutions of higher quality in the long term.

An interesting thing is that the decrease of diversity was not linear as we expected by decreasing D in Algorithm 5 in that way.

This is specially clear in the cases with $k = 4$. Thus, creating an adaptive algorithm to manage the decrease of diversity seems promising, but that work is beyond the reach of this paper.

5.3 Validation with the Graph Partitioning Archive

In order to illustrate the effectiveness of the proposed memetic algorithm, test with the 15 first graphs of the Graph Partitioning Archive were performed. MAHMBCDP was tested with the values $k = \{4, 8, 16, 32, 64\}$. Tables 5, 7 and 6 summarize the results obtained with MAHMBCDP. In addition to the best, mean and worst results obtained, the best-known solution found by any solver previous to this paper is shown in the column TGPA. MAHMBCDP reaches the best known solution for almost every instance for $k = 4$ and $k = 8$. Moreover, in five test cases it could generate a new best-known solution. The only cases where the best-known solution could not be attained are *bcsstk29* and *vibrobox*. The number of edges for these two instances and bcsstk33 is at least twice than for the rest. This reduced hundreds of times the number of generations evolved by our algorithm, meaning that in these cases the way of managing the diversity of MAHMBCDP might not be adequate because it usually requires a lot of generations to reach high-quality solutions.

For the larger values of k, the performance of MAHMBCDP also degraded. Again, we could identify that the number of generations evolved was drastically reduced with the increase of k. In fact, in many cases, less than $1,000$ generations could be evolved. Thus, the MAHMBCDP is really useful for those cases where many generations can be evolved. However, taking into account that several computationally expensive operators are used, the proposal is not scalable to large graphs (greater than about $17,000$ edges) or to large values of k. In any case, for values of k larger than 8, MAHMBCDP could identify five new best-known solutions.

As a summary, it is quite important to remark that best-known solution could be improved further in ten cases. Taking into account that these graphs have been tackled with a very large amount of proposals, this achievement is an important proof of the proper performance of the solver designed in this paper.

6 Conclusions and Future Work

The Graph Partitioning Problem (GPP) is a well-known NP-hard combinatorial optimization problem that has been addressed with numerous optimizers. Among them, hybrid metaheuristics that combine population-based approaches with trajectory-based schemes have reported the most promising results. Recently, the state-of-the-art of some related problems could be advanced significantly by taking into account the diversity of the population in an explicit way. This paper studies the applicability of these kinds of approaches in the GPP. Specifically, a memetic algorithm that incorporates a hill-climber and a relatively large neighborhood has been designed. The proposal (MAHMBCDP) incorporates a novel replacement strategy, the best-non-penalized scheme (BNP), that is based on considering both the diversity and quality in the survivor selection phase. Additionally, an important novelty is the design of a crossover operator that is based on the principle

Table 4. Summary of the results obtained with different $distInitFactor$ values

	0.0		0.2		0.4		0.6		0.8		1.0	
Instance	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best
uk, $k = 4$	193.83	135	132.9	44	73.33	41	45.8	41	43.46	41	43.467	42
uk, $k = 64$	539.2	492	445.83	429	427.2	421	425.26	420	425.767	418	425.97	421
3elt, $k = 4$	375.9	270	201	201	201	201	201	201	201	201	201	201
3elt, $k = 64$	1558.23	1544	1538.56	1534	1537.4	1535	1537.13	1535	1537.63	1535	1538.1	1536

Table 5. MAHMBCDP results for k = 4, 8

	$k = 4$				$k = 8$			
Instance	TGPA	Best	Mean	Worst	TGPA	Best	Mean	Worst
add20	1154	**1151**	1153.1	1155	1686	**1681**	1682.73	1685
data	382	382	382	382	668	668	668.07	669
3elt	201	201	201	201	345	345	345	345
uk	41	41	47.37	81	84	**83**	86.6	94
add32	34	34	34	34	67	67	67.33	68
bcsstk33	21717	21717	21717	21717	34437	34437	34444.7	34454
whitaker3	381	381	381	381	656	656	657.1	658
crack	366	366	366	366	679	679	679.07	680
wing_nodal	3575	3575	3578.2	3582	5435	5435	5436.93	5439
fe_4elt2	349	349	349	349	607	**606**	607.6	609
vibrobox	18976	18988	19116.27	19268	24484	**24479**	24479.57	24481
bcsstk29	8035	8069	8086.7	8106	13986	14001	14034.57	14084
4elt	326	326	326	768	545	545	565.23	807
fe_sphere	768	768	768.07	769	1156	1156	1205.47	1383
cti	954	954	954	1178	1788	1788	1796.13	1848

Table 6. MAHMBCDP results for k = 64

	$k = 64$			
Instance	TGPA	Best	Mean	Worst
add20	2949	2952	2961.23	2968
data	2839	2841	2852.03	2856
3elt	1532	1535	1537.07	1539
uk	408	416	424.53	437
add32	485	485	493.43	502
bcsstk33	107185	107413	107632.23	107814
whitaker3	2491	2505	2525.43	2538
crack	2535	2549	2560	2568
wing_nodal	15775	15781	15801.37	15821
fe_4elt2	2478	2490	2503.27	2518
vibrobox	46571	46692	46820	46952
bcsstk29	55241	55807	56203.93	56742
4elt	2565	2572	2588.57	2606
fe_sphere	3543	3594	3634.03	3671
cti	5629	5697	5737.3	5787

of maximizing the number of nodes that share the same class both in the parents and offspring.

The experimental validation has been performed with test cases of the Chris Walshaw's Graph Partitioning Archive. Comparisons against other strategies that were devised with the aim of facing premature convergence show the important benefits of the BNP strategy. It is also remarkable that in ten test cases, new best-known solutions could be attained. Taking into account the large number of techniques that have been used to deal with the GPP, this is a quite important achievement. The main drawback of the proposal is that due to the incorporation of computationally expensive components, there are issues with the scalability. Both the increase on the number of edges and the increase on the number of classes of the desired partitions provoke degradation on the performance.

Several lines of future work might be explored. First, some actions to improve the scalability might be studied. Particularly, taking into account more simple neighborhood definitions but with more complex trajectory-based schemes to intensify, seems a promising approach. Second, applying adaptive ways to manage the diversity might bring additional benefits. Finally, parallelizing the

Table 7. MAHMBCDP results for k = 16, 32

	$k = 16$				$k = 32$			
Instance	TGPA	Best	Mean	Worst	TGPA	Best	Mean	Worst
add20	2047	**2041**	2047.03	2052	2362	**2360**	2362.8	2367
data	1127	1127	1127.8	1128	1799	1799	1803.2	1813
3elt	573	573	573	573	960	960	961.53	962
uk	146	148	153.63	161	254	**253**	259	264
add32	118	118	118	122	213	213	213.77	218
bcsstk33	54680	54687	54721.27	54800	77414	77508	77616.93	77824
whitaker3	1088	1089	1090.47	1094	1668	1672	1684.8	1697
crack	1088	1088	1089.8	1091	1679	1682	1687.7	1697
wing_nodal	8334	**8333**	8341.77	8377	11768	11774	11793.13	11825
fe_4elt2	1007	1008	1008.77	1010	1614	1619	1628.67	1642
vibrobox	31892	32428	32744.6	32920	39477	**39443**	39699.7	39975
bcsstk29	21958	22136	22217.73	22332	34968	35130	35283.63	35533
4elt	934	937	957.13	1084	1551	1554	1566.9	1597
fe_sphere	1714	1719	1875.63	2068	2490	2500	2561.3	2659
cti	2793	2799	2846.37	2895	4046	4069	4114.2	4153

proposal to reduce the time required to obtain high-quality solutions seems a plausible approach.

Acknowledgements

The authors acknowledge funding from Conacyt project no. 285599 ”Toma de decisiones multiobjetivo para sistemas altamente complejos”.

References

1. **Aardal, K. I., van Hoesel, S. P. M., Koster, A. M. C. A., Mannino, C., & Sassano, A. (2007).** Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, Vol. 153, No. 1, pp. 79–129.

2. **Bader, D. A., Meyerhenke, H., Sanders, P., & Wagner, D.**, editors **(2013).** *10th DIMACS Implementation Challenge Workshop on Graph Partitioning and Graph Clustering Proceedings*, volume 588 of *Contemporary Mathematics*. American Mathematical Society, Atlanta, GA, USA.

3. **Bichot, C.-E. (2013).** *Local Metaheuristics and Graph Partitioning*. John Wiley & Sons, Inc., pp. 137–161.

4. **Bichot, C.-E. (2013).** *Population-Based Metaheuristics, Fusion-Fission and Graph Partitioning Optimization*. John Wiley & Sons, Inc., pp. 163–199.

5. **Bui, T. N. & Moon, B. R. (1996).** Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, Vol. 45, No. 7, pp. 841–855.

6. **Chardaire, P., Barake, M., & McKeown, G. P. (2007).** A probe-based heuristic for graph partitioning. *IEEE Transactions on Computers*, Vol. 56, No. 12, pp. 1707–1720.

7. **Eiben, A. & Smith, J. (2003).** *Introduction to Evolutionary Computing*. Natural Computing Series. Springer.

8. **Galinier, P., Boujbel, Z., & CoutinhoFernandes, M. (2011).** An efficient memetic algorithm for the graph partitioning problem. *Annals of Operations Research*, Vol. 191, No. 1, pp. 1–22.

9. **Galinier, P. & Hao, J.-K. (1999).** Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, Vol. 3, No. 4, pp. 379–397.

10. **Garey, M. R. & Johnson, D. S. (1990).** *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

11. **Hager, W. W., Phan, D. T., & Zhang, H. (2013).** An exact algorithm for graph partitioning. *Mathematical Programming*, Vol. 137, No. 1, pp. 531–556.

12. **Harik, G. R. (1995).** Finding multimodal solutions using restricted tournament selection. *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 24–31.

13. **Hendrickson, B., Leland, R., Karypis, G., Kumar, V., Barnard, S., & Simon, H. (2017).** The graph partitioning archive. http://chriswalshaw.co.uk/partition/ Last visited 03/03/17.

14. **Kahng, A. B., Lienig, J., Markov, I. L., & Hu, J. (2011).** *VLSI Physical Design: From Graph*

Partitioning to Timing Closure. Springer Publishing Company.

15. **Kernighan, B. & Lin, S. (1970).** An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell Systems Technical Journal*, Vol. 49, No. 2, pp. 291–307.
16. **Lobo, F. G., Lima, C. F., & Michalewicz, Z.**, editors **(2007).** *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer.
17. **Maini, H., Mehrotra, K., Mohan, C., & Ranka, S. (1994).** Genetic algorithms for graph partitioning and incremental graph partitioning. *Proceedings of Supercomputing '94*, pp. 449–457.
18. **Mengshoel, O. J. & Goldberg, D. E. (2008).** The crowding approach to niching in genetic algorithms. *Evol. Comput.*, Vol. 16, No. 3, pp. 315–354.
19. **Merz, P. & Freisleben, B. (1998).** Memetic algorithms and the fitness landscape of the graph bi-partitioning problem. *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - PPSN*, Springer, pp. 765–774.
20. **Pandey, H. M., Chaudhary, A., & Mehrotra, D. (2014).** A comparative review of approaches to prevent premature convergence in GA. *Applied Soft Computing*, Vol. 24, pp. 1047 – 1077.
21. **Petrowski, A. (1996).** A clearing procedure as a niching method for genetic algorithms. *Proceedings of IEEE International Conference on Evolutionary Computation 1996 (CEC'96)*, pp. 798–803.
22. **Sanders, P. & Schulz, C. (2013).** *Think Locally, Act Globally: Highly Balanced Graph Partitioning*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 164–175.
23. **Segura, C., Aguirre, A. H., Peña, S. I. V., & Rionda, S. B. (2015).** *The Importance of Proper Diversity Management in Evolutionary Algorithms for Combinatorial Optimization*. Springer International Publishing, pp. 121–148.
24. **Segura, C., Coello, C. A. C., Segredo, E., & Aguirre, A. H. (2016).** A novel diversity-based replacement strategy for evolutionary algorithms. *IEEE Transactions on Cybernetics*, Vol. 46, No. 12, pp. 3233–3246.
25. **Segura, C., Hernandez, A., Luna, F., & Alba, E. (2017).** Improving diversity in evolutionary algorithms: New best solutions for frequency assignment. *IEEE Transactions on Evolutionary Computation*, Vol. In Press, No. 99.
26. **Segura, C., Peña, S. I. V., Rionda, S. B., & Aguirre, A. H. (2016).** The importance of diversity in the application of evolutionary algorithms to the Sudoku problem. *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 919–926.
27. **Talbi, E.-G. (2009).** *Metaheuristics: From Design to Implementation*. Wiley Publishing.
28. **Črepinšek, M., Liu, S.-H., & Mernik, M. (2013).** Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys*, Vol. 45, No. 3, pp. 35:1–35:33.
29. **Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013).** A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, Vol. 40, No. 1, pp. 475–489.

Article received on 01/02/2017; accepted on 16/05/2017.
Corresponding author is Emmanuel Romero Ruiz.