The Labeled Maximum Matching Problem

Francesco Carrabs, Raffaele Cerulli, Monica Gentili
Department of Mathematics and Computer Science
University of Salerno
Via Ponte Don Melillo, 84084 Fisciano (SA), Italy.
{fcarrabs, raffaele, mgentili }@unisa.it

Abstract

Given a graph G where a label is associated with each edge, we address the problem of looking for a maximum matching of G using the minimum number of different labels, namely the *Labeled Maximum Matching Problem*. It is a relatively new problem whose application is related to the timetabling problem [15]. We prove it is NP-complete and present four different mathematical formulations. Moreover, we propose an exact algorithm based on a branch-and-bound approach to solve it. We evaluate the performance of our algorithm on a wide set of instances and compare our computational times with the ones required by CPLEX to solve the proposed mathematical formulations. Test results show the effectiveness of our procedure, that hugely outperforms the solver.

Key words: Matching, Label, Color, Exact Approach, Branch and Bound.
PACS:

1. Introduction

Matching problems are considered among the most important combinatorial problems. The classic application of matching deals with pairing of objects from two disjoint sets, however, pairings do not always deal with disjoint sets. Matchings have applications also in connection with relaxation and heuristics for many well-known problems (e.g., traveling salesman problem, the postman problem, among others). In this paper we deal with a variant of the classical maximum matching problem, namely the *Labeled Maximum Matching Problem* (LMM). Given an undirected and labeled graph $G = (V, E, C)$ where a label $c \in C$ is assigned with each edge, we look for a maximum matching of G using the minimum number of different labels. A practical application of the problem is presented in [15] and [16] where its relationship with the timetabling problem is shown. Indeed, a solution of the timetabling problem may be seen as a matching between classes and teachers that satisfies additional restrictions. For example, we can assume the classes to belong to different schools spread out on a given area and a professor may teach in different schools. Let us represent a school with a given label. An inspector needs to maximize the number of assessed teachers during their lectures without visiting the same class more than once. Hence, the lectures to be attended would form a maximum matching. For convenience the inspector would like these lectures to take place in the smallest possible number of different schools. Then, clearly the inspector has to construct a maximum matching meeting a minimum number of labels in the graph associated with the lectures.

The class of labeled problems is a recent research area. Problems in this class may be used to model many real world problems arising, for example, in communication networks, multimodal transportation networks,

etc. The Minimum Labeled Spanning Tree Problem (MLST) was the first problem introduced in this area and defined by [1] and [8]. It is shown to be NP-hard and many heuristics and approximation results are provided in the literature to solve it [2, 4, 14, 17, 18, 19]. Many other labeled problems are recently addressed in the literature: the Minimum Labeled Steiner Problem [5], the Minimum Labeled Hamiltonian Problem [6, 7, 20], the Labeled Shortest Path Problem [3].

To the best of our knowledge, the existing papers in the literature studying the specific problem addressed in this paper are [15] and [16], where its complexity and approximability was studied on bipartite graphs. In this paper, we study the problem on general graphs. We assess its complexity and provide different mathematical formulations. Since the problem is proved to be NP-complete, an exact branch and bound approach is proposed to solve it. We compare our results with those provided by a solver. Our extensive experimentation shows the effectiveness of the proposed approach.

The sequel of the paper is organized as follows. Section 2 introduces the basic notation. The complexity of the problem is addressed in Section 3, and, the proposed mathematical models are described in Section 4. Section 5 contains the detailed description of our branch and bound algorithm. Computational results are reported in Section 6. Finally, concluding remarks are discussed in Section 7.

2. Basic Notation

Let $G = (V, E, C)$ be an undirected and labeled graph where V is the vertex set, E the edge set and C a set of labels. A label $\mathscr{C}(i,j) \in C$ is associated with each edge $(i,j) \in E$. Given a subset $S \subseteq E$ of edges the set $\mathscr{C}(S) = \bigcup_{(i,j)\in S} \mathscr{C}(i,j)$ is the corresponding set of labels. The subgraph induced by a given label $c \in C$ is denoted by G^c, i.e. $G^c = (V, E^c, c)$ with $E^c = \{(i,j) \in E : \mathscr{C}(i,j) = c\}$. Similarly, we define $G^{\overline{c}}$ the subgraph of G without edges whose label is c, i.e $G^{\overline{c}} = (V, E \setminus E^c, C \setminus \{c\})$. The set of edges incident to a vertex $i \in V$ is denoted by $\delta(i) = \{j : (i,j) \in E\}$. Given a set of vertices $X \subseteq V$, we denote by $G[X] = (V \setminus X, E[X], \mathscr{C}(E[X]))$ the subgraph of G induced by X where $E[X] = \{(i,j) : i, j \in V \setminus X\}$.

A *matching* M of G is a set of edges that meets each vertex of G at most once. We denote by $\mathcal{V}(M)$ the set of vertices covered by M. A matching is called a *perfect matching* if all vertices are covered by M, i.e $|\mathcal{V}(M)| = |V|$, while it is called *near-perfect* if it covers all vertices but one. We say an edge $(i,j) \in E$ is *feasible* for M if and only if $i, j \notin \mathcal{V}(M)$, otherwise it is *infeasible*. A *maximum matching* of G, denoted by $\mathcal{M}(G)$, is a matching of maximum size. A *minimum labeled maximum matching* of G, denoted by $\hat{\mathcal{M}}(G)$, is a maximum matching with the minimum number of labels, i.e. $|\mathscr{C}(\hat{\mathcal{M}}(G))|$ is minimum.

3. Complexity

In this section we prove the addressed problem is NP-complete. Let us define the decision version of our problem, namely, the *Bounded Labeled Maximum Matching Problem (BLMM)*:

Bounded Labeled Maximum Matching Problem*: Given an undirected and edge labeled graph $G = (V, E, C)$ and a positive integer k: is there a maximum matching $\mathcal{M}(G)$ of G such that the total number of different labels in $\mathcal{M}(G)$ is less than or equal to k, i.e. $|\mathscr{C}(\mathcal{M}(G))| \leq k$?*

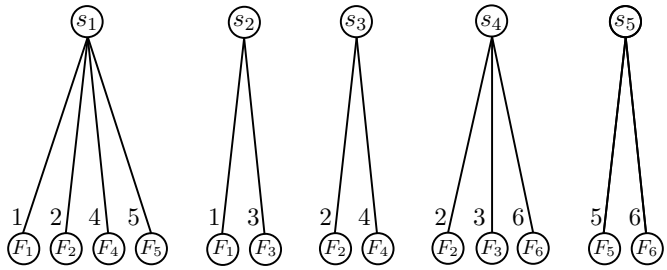
Theorem 1 *The Bounded Labeled Maximum Matching Problem is NP-Complete.*

Proof. It is easy to see that BLMM $\in$ NP since a nondeterministic algorithm needs only to guess a subset M of edges and to check in polynomial time whether each vertex has degree less than or equal to 1, the size of M is equal to the size of a maximum matching and $\mathscr{C}(M)$ has size less than or equal to k.

We prove the theorem by reduction from the well known Minimum Set Covering Problem (MSC). Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n elements, $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ be a family of m subsets of S, i.e. $F_i \subseteq S, i = 1, 2, \ldots, m$, and k be a positive integer. The decisional version of MSC consists in selecting no

S	$\mathcal{F}$
s_1	$F_1 = \{s_1, s_2\}$
s_2	$F_2 = \{s_1, s_3, s_4\}$
s_3	$F_3 = \{s_2, s_4\}$
s_4	$F_4 = \{s_1, s_3\}$
s_5	$F_5 = \{s_1, s_5\}$
	$F_6 = \{s_4, s_5\}$

(a)



(b)

Fig. 1. *(a) A generic instance of the Minimum Set Covering Problem, and, (b) the corresponding instance of the Bounded Labeled Maximum Matching Problem.*

more than k subsets in $\mathcal{F}$ that cover all the elements in S. We now define from the generic instance of MSC a graph $G = (V, E)$, a labeling function of the edges and show that there exists a covering of S with at most k subsets if and only if there exists a maximum matching of G using at most k labels.

Let us denote by $\mathcal{F}(s_i) = \{F_j \in \mathcal{F} : s_i \in F_j\}$ the collection of sets in $\mathcal{F}$ containing the element s_i and let n_i be its size. For each element s_i we define in G a vertex s_i and the set of n_i vertices corresponding to the subsets $F_j \in \mathcal{F}(s_i)$. We define an edge in G between each vertex s_i and the corresponding vertices $F_j \in \mathcal{F}(s_i)$ with associated label j (Figure 1). Therefore, G contains $|V| = n + \sum_{i=1}^{n} n_i$ vertices, and, $|E| = \sum_{i=1}^{n} n_i$ edges. This construction can be accomplished in polynomial time.

Note that, any maximum matching $\mathcal{M}(G)$ of G contains exactly n edges, one for each vertex s_i. Consider now a cover of S of size k, that is a selection of k subsets $\{F_{j_1}, F_{j_2}, \ldots, F_{j_k}\}$, such that $\bigcup_{h=1}^{k} F_{j_h} = S$. We can define a maximum matching $\mathcal{M}(G)$ that uses at most k different labels, by selecting for each vertex s_i an edge (s_i, F_j) whose label belongs to the set $\{j_1, j_2, \ldots, j_k\}$. On the contrary, let $\mathcal{M}(G)$ be a maximum matching of G using k different labels. We can build a cover of S of size at most k by selecting for each label $j \in \mathscr{C}(\mathcal{M}(G))$ the corresponding set F_j. □

4. Mathematical Formulations

In this section we provide four different mathematical formulations for the problem. In particular, we propose three integer linear programming (ILP) formulations and a lagrangean relaxation one (LR). All of them will be used to evaluate the performance of our branch and bound approach.

Let x_e be a set of binary variables, associated with each edge $e \in E$, whose value is equal to 1 if edge e is selected and 0 otherwise. Given an edge $e = (i, j)$, we denote its extremes by $\overleftarrow{e} = i$ and $\overrightarrow{e} = j$. Let y_k, $k \in C$, be a set of binary variables whose value is equal to 1 if label k is selected and 0 otherwise. We define a binary parameter a_e^k that is equal to 1 if label k is associated with edge e and 0 otherwise. Finally, let θ be the size of a maximum matching of G, i.e. $|\mathcal{M}(G)| = \theta$.

The first integer linear programming formulation we propose is the following:

$$(ILP_0) \qquad z_{ILP_0} = \min \sum_{k \in C} y_k \tag{1}$$

subject to

$$\sum_{e\in\delta(i)} x_e \leq 1 \qquad \forall\, i \in V \tag{2}$$

$$\sum_{e\in E} x_e = \theta \tag{3}$$

$$y_k \geq x_e a_e^k \qquad \forall\, k \in C, \forall\, e \in E \tag{4}$$

$$y_k \in \{0,1\} \qquad \forall\, k \in C \tag{5}$$

$$x_e \in \{0,1\} \qquad \forall\, e \in E \tag{6}$$

The objective function (1) requires the minimization of the total number of used labels. Constraints (2) and (3) ensure the selected edges form a maximum matching. Constraints (4) are logical constraints linking the two set of binary variables.

To strengthen the formulation we considered other three mathematical models: (i) a lagrangean relaxation (LR) of ILP_0, (ii) an alternative integer linear programming formulation (ILP_1) and (ii) a formulation based on the binding degree and blossom inequalities (ILP_2).

By introducing a lagrangian multiplier $\lambda \in R$, and relaxing constraint (3), we obtain the following relaxed problem:

$$(LR) \qquad z_{LR}(\lambda) = \min \sum_{k\in C} y_k + \lambda(\sum_{e\in E} x_e - \theta) \tag{7}$$

subject to constraints (2), (4), (5) and (6).

From lagrangian duality theory, we know $z_{LR}(\lambda) \leq z_{ILP_0}$, for each $\lambda \in R$, i.e. the optimum solution of the lagrangean relaxation is a valid lower bound to the optimum solution value z_{ILP_0} for any $\lambda \in R$. The best attainable lower bound is the solution of the following lagrangian dual problem:

$$(LD) \qquad z_{LD} = \max_{\lambda\in R} z_{LR}(\lambda) \tag{8}$$

The following theorem holds.

Theorem 2 *The optimum solution λ^* of (LD) is such that $\lambda^* < 0$.*

Proof. It is easy to see that, when $\lambda \geq 0$ the optimum solution of (LR) is obtained by setting $x_e = 0, \forall\, e \in E$. This implies from constraints (4) that $\sum_{k\in C} y_k = 0$, and it follows $z_{LR} = -\lambda\theta \leq 0$. On the contrary, when $\lambda < 0$, we have $z_{LR} = \sum_{k\in C} y_k + \lambda(\sum_{e\in E} x_e - \theta) > 0$. Indeed, constraints (2) impose $\sum_{e\in E} x_e - \theta \leq 0$. □

Let $\{x_e(\lambda), y_k(\lambda)\}$ be any pair of optimal solutions to LR. If this solution is feasible for the original problem ILP_0, it is also ϵ-optimal for ϵ defined as:

$$\epsilon = \theta - \sum_{e\in E} x_e$$

The next theorem shows that when $\lambda < -\theta$ then $\epsilon = 0$, that is $\{x_e(\lambda), y_k(\lambda)\}$ is feasible for ILP_0 and therefore is also optimal.

Theorem 3 *If $\lambda < -\theta$ then $\sum_{e\in E} x_e(\lambda) = \theta$.*

Proof. Let $\{x_e^*, y_k^*\}$ be the optimum solution of ILP_0, and $z_{ILP_0} = \sum_{k\in C} y_k^*$ the corresponding objective function value. Let us suppose there exists an optimum solution of (LR), say $\{\hat{x}_e(\lambda), \hat{y}_k(\lambda)\}$ such that $\sum_{e\in E} \hat{x}_e(\lambda) < \theta$, and, let $\hat{z}_{LR}(\lambda)$ be the corresponding objective function value. By lagrangean theory we know $\hat{z}_{LR}(\lambda) \leq z^*_{ILP_0}$, that is:

$$\sum_{k\in C} \hat{y}_k(\lambda) + \lambda(\sum_{e\in E} \hat{x}_e(\lambda) - \theta) \leq \sum_{k\in C} y_k^* \leq \theta$$

by binary constraints, $\sum_{e\in E} \hat{x}_e(\lambda) - \theta < -1$ that implies $\hat{z}_{LR}(\lambda) > \theta$, a contradiction. □

Therefore, we have the following:

Corollary 4 *If $\lambda < -\theta$ then $z_{LR}(\lambda) = z_{ILP_0}$.*

An alternative integer linear programming formulation is the following:

$$(ILP_1) \qquad z_{ILP_1} = \max \varphi \sum_{e\in E} x_e - \sum_{k\in C} y_k \tag{9}$$

subject to constraints (2), (4), (5) and (6).

Equation (9) defines the objective function as the weighted difference between the cardinality of a matching and the total number of used labels. Note that, by opportunely setting the value of the weight φ, the optimum solution of ILP_1 is achieved with a maximum matching. Let $\mathcal{M}(G)$ be a maximum matching of G, i.e. $|\mathcal{M}(G)| = \theta$, and $|\mathscr{C}(\mathcal{M}(G))| = h$ the size of its label set. The corresponding objective function value is $z_{ILP_1}(\mathcal{M}(G)) = \varphi\theta - h$. Let M be a matching of G with $|M| < \theta$, $\mathscr{C}(M)$ the corresponding label set, and, $z_{ILP_1}(M) = \varphi|M| - |\mathscr{C}(M)|$ the objective function value. We want to define a value for φ such that $z_{ILP_1}(\mathcal{M}(G)) > z_{ILP_1}(M)$, for each matching M of G such that $|M| < \theta$. The maximum value of $z_{ILP_1}(M)$ is equal to $\varphi(\theta - 1) - 1$, obtained considering a matching M such that $|M| = \theta - 1$ and a corresponding label set with size equal to 1. Therefore, $z_{ILP_1}(\mathcal{M}(G)) > z_{ILP_1}(M)$ for each matching M of $G \Leftrightarrow \varphi\theta - h > \varphi(\theta-1)-1 \Leftrightarrow \varphi > h-1$. The following results are then proved. Let $\{x_e^*, y_k^*\}$ be an optimum solution of ILP_1.

Theorem 5 *If $\varphi > h - 1$ then $\sum_{e\in E} x_e^* = \theta$.*

Corollary 6 *If $\varphi > h - 1$ then $\sum_{k\in C} y_k^* = z_{ILP_0}$.*

The last integer linear programming formulation is obtained by replacing constraint (3) with binding degree and blossom inequalities. These constraints can be derived analyzing the dual solution computed by Edmonds Cardinality Matching Algorithm [10]. The following theorem shows how to generate the new constraints from a dual solution:

Theorem 7 *[13] Given any graph $G = (V, E)$, denote by Y the set of vertices not covered by at least one maximum matching of G, by X the neighbours of Y in $V \setminus Y$, and by W all other vertices. Then:*
any maximum matching of G contains a perfect matching of $G[W]$, near-perfect matchings of the connected components of $G[Y]$, and, matches all vertices in X to distinct connected components of $G[Y]$.

The partitioning of V in W, X, Y is known as the Gallai-Edmonds decomposition of G. This decomposition is produced by Edmonds' Cardinality Matching Algorithm at the end of the computation. Let us denote by $\mathbb{CC}(G) = \{\mathbb{C}_1, \ldots, \mathbb{C}_p\}$ the set of connected component of G, and by $|\mathbb{C}_i|$ the number of vertices in the component. The mathematical formulation that uses the Gallai-Edmonds decomposition is:

$$(ILP_2) \qquad z_{ILP_2} = \min \sum_{k\in C} y_k \tag{10}$$

subject to

$$\sum_{e \in \delta(i)} x_e \leq 1 \qquad \forall\, i \in V \tag{11}$$

$$y_k \geq x_e a_e^k \qquad \forall\, k \in C, \forall\, e \in E \tag{12}$$

$$\sum_{e \in \delta(i),\, \overrightarrow{e} \in W} x_e = 1 \qquad \forall\, i \in W \tag{13}$$

$$\sum_{e \in \delta(i),\, \overrightarrow{e} \notin W} x_e = 0 \qquad \forall\, i \in W \tag{14}$$

$$\sum_{e \in \delta(i),\, \overrightarrow{e} \in Y} x_e = 1 \qquad \forall\, i \in X \tag{15}$$

$$\sum_{e \in \delta(i),\, \overrightarrow{e} \notin Y} x_e = 0 \qquad \forall\, i \in X \tag{16}$$

$$\sum_{e \in E[\mathbb{C}_i]} x_e = \left\lfloor \frac{|\mathbb{C}_i|}{2} \right\rfloor \qquad \forall\, \mathbb{C}_i \in \mathbb{CC}(G[Y]) \tag{17}$$

$$y_k \in \{0,1\} \qquad \forall\, k \in C \tag{18}$$

$$x_e \in \{0,1\} \qquad \forall\, e \in E \tag{19}$$

Constraints (13)-(17) are the binding degree and blossom inequalities derived from the Gallai-Edmonds decomposition. These constraints guarantee exactly θ edges are selected, i.e. a maximum matching of G is selected. In particular, constraints (13) and (14) guarantee that for each vertex $i \in W$ exactly one edge (i, j) with $j \in W$ is selected while all the other edges (i, j), where $j \notin W$ are not selected. Constraints (15) and (16) ensure that for each vertex $i \in X$ exactly one edge (i, j) such that $j \in Y$ is selected, while all the other edges (i, j) with $j \notin Y$ are not selected. Finally, constraints (17) ensure that for each connected component in $G[Y]$ a near-perfect matching is selected.

We compare the computational times required by CPLEX to solve LR, ILP_1 and ILP_2 models in section 6. Table 1 shows the ILP_2 model, on average, turned out to have the best computational times. For the sake of clearness of the exposition we decided to show only the most significant results. Therefore, due to the very poor performance of the ILP_0 model, the corresponding computational times are not reported in section 6.

5. The Branch and Bound Algorithm

In this section we describe our branch and bound approach to solve the problem. Note that, the branching strategy is not based on the integer linear programming formulations provided in the previous sections, as it will be clear in the sequel. The following subsections describe in details the basic choices for the implementation of our branch-and-bound algorithm, that is:

(*i*) the branching strategy and the characterization of the search tree (subsection 5.1);
(*ii*) the exploration strategy (subsection 5.2);
(*iii*) the pruning operations (subsection 5.3);
(*iv*) the edge choice policy (subsection 5.4).

5.1. *The Branching Strategy and the Search Tree*

Our algorithm explores a search tree $\mathcal{T}$ that represents all the matchings of a given graph G and selects a maximum matching of G using the minimum number of labels. To avoid confusion, from now on, we use the term "vertex" to denote a vertex in G, and, "node" to denote a decision node of $\mathcal{T}$.

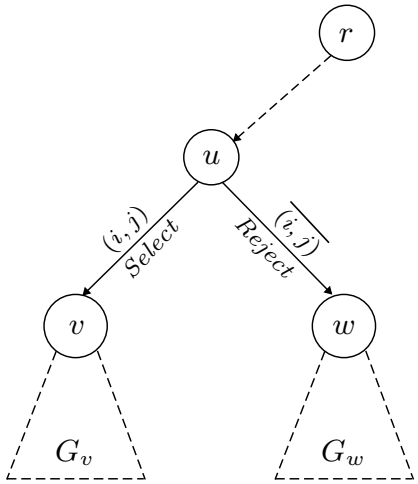


Fig. 2. *A branch operation on a generic node u carried out by choosing edge $(i, j) \in \ell_u$.*

The following structures are associated with each node $u \in \mathcal{T}$:

- the set $\rho(u) \subseteq E$ corresponding to the selected edges along the path in $\mathcal{T}$ from the root r to u. By construction, $\rho(u)$ defines a matching of G;

- the set $\overline{\rho}(u) \subseteq E$ corresponding to the rejected edges along the path in $\mathcal{T}$ from the root r to u;

- the set $Z_u \in C$ of labels associated with the selected edges $\rho(u)$, i.e. $Z_u = \mathscr{C}(\rho(u))$;

- the *candidate edge set* ℓ_u, that is the set of feasible edges that can be chosen to perform a branch operation from u;

- a residual graph $G_u = (V_u, E_u, C_u)$ such that:

  - vertex set $V_u \subseteq V$ contains all the vertices not covered by $\rho(u)$, i.e. $V_u = V \setminus \mathcal{V}(\rho(u))$;

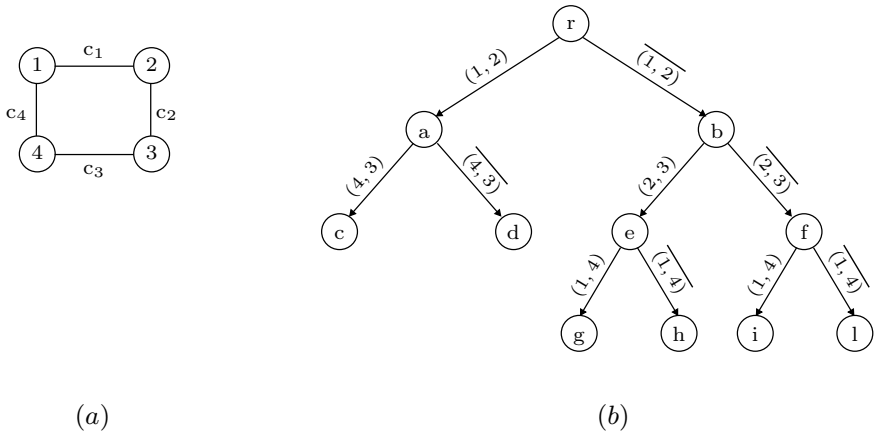
  - edge set E_u contains all the not rejected edges induced by V_u, i.e $E_u = \{(i, j) \in E : i, j \in V_u \text{ and } (i, j) \notin \overline{\rho}(u)\}$;

  - label set C_u contains all the labels associated with edges in E_u, i.e. $C_u = \bigcup_{(i,j)\in E_u} \mathscr{C}(i, j)$.

On the root r we have: $\rho(r) = \overline{\rho}(r) = \emptyset$, $Z_r = \emptyset$, $\ell_r = E$ and $G_r = G$. Given a generic node $u \in \mathcal{T}$ a branching on it consists in either selecting a new edge $(i, j) \in \ell_u$ to be part of a matching or not. This branching generates two children, the left-child v and the right-child w (see Figure 2). The former corresponds to the selection of the edge (i, j), the latter to the rejection of (i, j). For this reason, we define v as a *selection node* (s-node) and w as a *rejection node* (r-node).

In general, left children in $\mathcal{T}$ are s-nodes, while right children are r-nodes. On s-node v, the matching $\rho(v)$ is obtained by adding (i, j) to $\rho(u)$, the set $\overline{\rho}(v)$ is equal to $\overline{\rho}(u)$ and, the residual graph G_v, is derived from G_u by removing vertices i and j. On the other hand, for r-node w we have: $\rho(w) = \rho(u)$, $\overline{\rho}(w) = \overline{\rho}(u) \cup \{(i, j)\}$ and G_w is obtained from G_u by deleting edge (i, j).

Consider for example the simple graph of 4 vertices shown in Figure 3a, where a label c_i is associated



(a) (b)

Fig. 3. *(a) A simple graph with 4 vertices, and, (b) the associated decision tree $\mathcal{T}$.*

with each edge. The associated decision tree is shown in Figure 3b. With node a the sets $\rho(a) = \{(1,2)\}$, $\overline{\rho}(a) = \emptyset$, $Z_a = \{c_1\}$ and $\ell_a = \{(4,3)\}$ are associated. With node b these sets are $\rho(b) = \emptyset$, $\overline{\rho}(b) = \{(1,2)\}$, $Z_b = \emptyset$ and $\ell_b = \{(2,3),(1,4),(4,3)\}$. With node h we have $\rho(h) = \{(2,3)\}$, $\overline{\rho}(h) = \{(1,2),(1,4)\}$, $Z_h = \{c_2\}$ and $\ell_h = \emptyset$. Note that, by construction, for a generic node u of the search tree, $\rho(u)$ is a matching of G and all the edges in G_u are feasible edges for it, then $\ell_u = E_u$.

To better understand the interactions among the next subsections we give now an outline of the algorithm. A more detailed description will be given in subsection 5.7.

5.1.1. *Outline of the Algorithm*

At a given node u, if $\rho(u)$ is a maximum matching of G and $|Z_u| < |\mathscr{C}(M^*)|$, where M^* is the incumbent solution, then the algorithm sets $M^* = \rho(u)$. At this point a new node of the tree is exploited according to the exploration strategy, described in subsection 5.2. On the contrary, if $\rho(u)$ is not a maximum matching of G, then the size of a maximum matching $\mathcal{M}(G_u)$ of G_u is computed to check whether a feasible solution of the original problem can be obtained. Indeed, observe that the subset of edges $\rho(u) \cup \mathcal{M}(G_u)$ is a matching of G. If the cardinality of this matching is lower than the cardinality of a maximum matching of G, then it is not possible to find a feasible solution of the problem in the subtree rooted in u: u is pruned and a new node is explored. Otherwise, a lower bound lb_u (see subsection 5.3) is computed. If this lower bound is such that there does not exist a better solution than the incumbent one on the subtree rooted in u, then u is pruned. Otherwise a branching from u is carried out by selecting an edge in ℓ_u. The algorithm stops when there are no more nodes to be explored.

5.1.2. *Reduction of the Candidate Edge Set*

In the previous section we showed the candidate edge set ℓ_u coincides with the edge set of the residual graph G_u. The number of branchings performed in the subtree $\mathcal{T}_u$ rooted in u is at most equal to $|\ell_u|$. Therefore, lower the size of ℓ_u, lower the size of $\mathcal{T}_u$. In this section, we define two rules, namely the *used label rule* and the *necessary label rule*, to reduce the size of ℓ_u. Theorem 12, in subsection 5.3, will prove this reduction does not affect the correctness of the algorithm.

The first rule consists in choosing edges in ℓ_u whose label has not yet been used:

*R*1. (*Used label Rule*): *Choose* $(i,j) \in \ell_u$ *such that* $\mathscr{C}(i,j) \notin Z_u$.

The application of $R1$ has two main consequences. The former is that we can remove from ℓ_u all those edges whose label belongs to Z_u, i.e. $\ell_u = E_u \setminus \{(i,j) : \mathscr{C}(i,j) \in Z_u\}$. Moreover, the number of s-nodes along a path from r to any node u can be upper bounded to the number $|\mathscr{C}(M^*)|$ of labels associated with the incumbent solution M^*. Indeed, by applying $R1$, Z_u is increased by one every time an edge is selected, that is every time an s-node is generated. Hence, the selected edges $\rho(u)$ in the path from r to any node u have all distinct label, i.e. $|\rho(u)| = |Z_u|$. Since at any node u the condition $|Z_u| < |\mathscr{C}(M^*)|$ holds (otherwise u would be pruned) then it follows $|\rho(u)| < |\mathscr{C}(M^*)|$. Therefore, the size of $\mathcal{T}$ mainly depends on the number of generated r-nodes.

The second rule used to reduce the set ℓ_u is based on the following definition of a *necessary label*.

Definition 8 *Given an undirected and labeled graph $G = (V, E, C)$, a label $c \in C$ is necessary if all the maximum matchings of G use it.*

Note that, to check whether a given label c is necessary we need to compute the size $|\mathcal{M}(G^{\overline{c}})|$ of a maximum matching on the subgraph $G^{\overline{c}}$ obtained from G by removing all those edges whose label is c. If $|\mathcal{M}(G^{\overline{c}})| < |\mathcal{M}(G)|$ then c is necessary.

Given a node $u \in \mathcal{T}$, we denote by $\mathcal{I}_u$ the necessary label set associated with the residual graph G_u. Note that some of the labels in Z_u could be necessary, then for performance reasons, we compute $\mathcal{I}_u$ considering only those labels not belonging to Z_u, i.e. $\mathcal{I}_u \subseteq \mathscr{C}(\ell_u) \setminus Z_u$. This will make Z_u and $\mathcal{I}_u$ two disjoint sets , i.e. $Z_u \cap \mathcal{I}_u = \emptyset$, and $|Z_u \cup \mathcal{I}_u|$ a lower bound to the value of any feasible solution generated from u. Recall that, these solutions can be obtained joining the set $\rho(u)$ with a maximum matching $\mathcal{M}(G_u)$. Since $\mathcal{I}_u \subseteq \mathscr{C}(\mathcal{M}(G_u))$, a solution produced by any maximum matching algorithm contains at least one edge for each label in $\mathcal{I}_u$. Therefore, it is convenient to remove from ℓ_u all the edges whose label is necessary. We perform this reduction by applying the following necessary label rule:

R2. (*Necessary label Rule*): *Choose $(i,j) \in \ell_u$ such that $\mathscr{C}(i,j) \notin \mathcal{I}_u$.*

Because of this new rule the candidate edge set becomes $\ell_u = E_u \setminus \{(i,j) : \mathscr{C}(i,j) \in \{Z_u \cup \mathcal{I}_u\}\}$.

Finally, we describe how we computed the necessary label set $\mathcal{I}_u$ associated with node u. From Definition 8, we need to compute the maximum matching $\mathcal{M}(G_u^{\overline{c}})\ \ \forall c \in \{C \setminus Z_u\}$ and to check whether $|\mathcal{M}(G_u^{\overline{c}})| < |\mathcal{M}(G_u)|$. However, to compute the necessary label set $\mathcal{I}_u$, we can restrict the search of the necessary labels to those ones belonging to $\{\mathscr{C}(\mathcal{M}(G_u)) \setminus Z_u\}$ as stated by the following remark, derived from Definition 8:

Remark 9 *Given a node $u \in \mathcal{T}$, the necessary label set $\mathcal{I}_u$ is included in the label set of any maximum matching of G_u, i.e. $\mathcal{I}_u \subseteq \{\mathscr{C}(\mathcal{M}(G_u)) \setminus Z_u\}$ for any $\mathcal{M}(G_u)$.*

5.2. *The Exploration Strategy*

Our branch and bound approach applies a *depth-first strategy* (DFS) which explores the tree by visiting at each step the left-child of the last visited node. After reaching a leaf node or after the execution of a pruning operation, this strategy backtracks to the father node and then visits its right-child. Once both children are visited, the exploration restarts from their grandfather and so on.

There are two main reasons for choosing such an exploration strategy. First of all, it is computationally more efficient than either a *best-first* strategy or a *breadth-first* one. By applying the DFS, the structures associated with any children of u (see subsection 5.1) are quickly computed. These updating operations are computationally more expensive when the other two alternative strategies are applied.

Moreover, we decided not to apply the best-first strategy since it turned out to be not efficient for our

specific search tree $\mathcal{T}$. Usually, this strategy is better than other ones in reducing the size of the search tree, since, the choice of the most promising node to be explored, speeds up the identification of an optimal solution. However, in our case, as we will see in subsection 5.3, the lower bounds associated with the nodes are in many cases very close to each other.

5.3. *Pruning Operations*

In this section we give a detailed description of the four pruning operations (steps) we apply at each node of the search tree when it is exploited.

Let M^* be the incumbent solution. Initially, M^* is a generic maximum matching of G that is updated, during the computation, each time a maximum matching with fewer labels is found. From now on, we say that a pruning step, applied on a node u, fails if it does not prune u. Our four pruning steps namely $P1$, $P2$, $P3$ and $P4$ are carried out subsequently: a step is performed if the previous one failed.

Let u be a node of $\mathcal{T}$ and $\rho(u)$ be the associated matching. Any feasible solution $\rho(u)$ contains all the labels in $Z_u \cup \mathcal{I}_u$. If $|Z_u \cup \mathcal{I}_u| \geq |\mathscr{C}(M^*)|$ then u is pruned. On the contrary, if $|Z_u \cup \mathcal{I}_u| < |\mathscr{C}(M^*)|$, our algorithm checks whether it is possible to obtain a feasible solution, containing $\rho(u)$, by using the edges of G_u. Then, a maximum matching $\mathcal{M}(G_u)$ is computed. If $|\mathcal{M}(G_u)| = |M^*| - |\rho(u)|$ then $\mathcal{M}(G_u) \cup \rho(u)$ is a maximum matching of G otherwise u is pruned. We denote $P1$ this first pruning step that is formally defined as follows:

$P1$. *If* $|\mathcal{M}(G_u)| < |M^*| - |\rho(u)|$ *then node* u *is pruned.*

If $P1$ fails, next pruning step $P2$, checks whether it is possible to build a feasible solution by using only labels in $Z_u \cup \mathcal{I}_u$.

$P2$. *Build the subgraph* $G_u^{Z_u \cup \mathcal{I}_u}$ *and compute* $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u})$. *If* $|\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u})| + |\rho(u)| = |M^*|$ *then* $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u}) \cup \rho(u)$ *is a maximum matching of* G *with* $|Z_u \cup \mathcal{I}_u|$ *labels. Since* $|Z_u \cup \mathcal{I}_u| < |\mathscr{C}(M^*)|$, *update* M^* *and prune node* u.

If $P2$ fails, then at least a new label $c \in C \setminus \{Z_u \cup \mathcal{I}_u\}$ needs to be added to $Z_u \cup \mathcal{I}_u$ to obtain a feasible solution. Pruning step $P3$ tries to build such a feasible solution.

$P3$. *If* $|Z_u \cup \mathcal{I}_u| + 1 \geq |\mathscr{C}(M^*)|$ *then prune* u, *otherwise compute* $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u \cup c})$, *for each* $c \in C \setminus \{Z_u \cup \mathcal{I}_u\}$. *If* $|\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u \cup c})| + |\rho(u)| = |M^*|$, *for some* $c \in C \setminus \{Z_u \cup \mathcal{I}_u\}$, *then* $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u \cup c}) \cup \rho(u)$ *is a maximum matching of* G *with* $|Z_u \cup \mathcal{I}_u| + 1$ *labels:* M^* *is updated and* u *is pruned.*

Let us define lb_u as the lower bound, associated with node u, to the number of labels in $C \setminus \{Z_u \cup \mathcal{I}_u\}$ needed to obtain any maximum matching $\mathcal{M}(G_u)$. Formally, $lb_u \leq |\mathscr{C}(\mathcal{M}(G_u)) \setminus \{Z_u \cup \mathcal{I}_u\}|$ for any $\mathcal{M}(G_u)$. Note that when $P2$ fails $lb_u = 1$, while when $P3$ fails $lb_u = 2$. Pruning step $P4$ to follow checks whether it is possible to obtain a feasible solution better than M^*. Formally:

$P4$. *If* $|Z_u \cup \mathcal{I}_u| + lb_u \geq |\mathscr{C}(M^*)|$ *then* u *is pruned.*

When all the four pruning steps fail, the algorithm performs a branch operation on u. Therefore, any branching in the search tree is carried out if and only if $lb_u \geq 2$.

Obviously, the greater the value of lb_u, the greater the chance node u is pruned after the execution of $P4$. We will see how the value of lb_u can be increased by the application of the heredity property, the lower bounding technique and the upper bounding technique described in subsections 5.5 and 5.6.

We can now introduce, as mentioned in subsection 5.4, Theorem 12 to prove the correctness of our algorithm when the candidate edge set ℓ_u is reduced to be equal to $E_u \setminus \{(i,j) : \mathscr{C}(i,j) \in (Z_u \cup \mathcal{I}_u)\}, \ \forall u \in \mathcal{T}$.

We focus now on special nodes of the search tree $\mathcal{T}$, we refer to as *closing* nodes.

Definition 10 *Let $G = (V, E, C)$ be an undirected and labeled graph and let $\hat{\mathcal{M}}(G)$ be a maximum matching of G with the minimum number of labels (i.e. an optimum solution). A node $u \in \mathcal{T}$ is a **closing** node if the following two conditions are satisfied:*

1. *$\rho(u) \subseteq \hat{\mathcal{M}}(G)$ and $\overline{\rho}(u) \cap \hat{\mathcal{M}}(G) = \emptyset$;*
2. *$|Z_u \cup \mathcal{I}_u| = |\mathscr{C}(\hat{\mathcal{M}}(G))| - 1$;*

Closing nodes are such that, if explored, the application of the pruning steps updates M^* to the optimum solution. This is proved in the following Lemma.

Lemma 11 *Let $G = (V, E, C)$ be an undirected and labeled graph, $\hat{\mathcal{M}}(G)$ be a optimum matching of G and u a closing node of $\mathcal{T}$. If node u is explored then $M^* = \hat{\mathcal{M}}(G)$.*

Proof. From condition 1 it follows: (i) the edge set E_u contains all the edges of $\hat{\mathcal{M}}(G)$ that have not yet been selected, and, (ii) all the labels in Z_u and $\mathcal{I}_u$ are used by the optimum solution $\hat{\mathcal{M}}(G)$. Hence, both $P1$ and $P4$ fail on u. Consequently, from condition 2, it follows that $\mathscr{C}(\hat{\mathcal{M}}(G)) \setminus \{Z_u \cup \mathcal{I}_u\} = \{c'\}$, that is, exactly a single additional label needs to be added to $\{Z_u \cup \mathcal{I}_u\}$ to obtain an optimum solution. Then, the application of $P3$ on $G_u^{Z_u \cup \mathcal{I}_u \cup c'}$ returns a maximum matching $\mathcal{M}(G_u)$ of G_u that contains label c' and such that $\mathcal{M}(G_u) \cup \rho(u)$ is a maximum matching of G. Moreover, since $|\mathscr{C}(\mathcal{M}(G_u) \cup \rho(u))| = |\mathscr{C}(\hat{\mathcal{M}}(G))|$ then $\mathcal{M}(G_u) \cup \rho(u)$ is an optimum solution of G. Note that such a matching $\mathcal{M}(G_u)$ exists because we could always set $\mathcal{M}(G_u) = \hat{\mathcal{M}}(G) \setminus \rho(u)$. $\square$

We prove now the correctness of our algorithm by the following Theorem.

Theorem 12 *Let $G = (V, E, C)$ be an undirected and labeled graph, M^* be the initial feasible solution and $\hat{\mathcal{M}}(G)$ be an optimum matching of G. Our branch and bound algorithm finds the optimum solution of G.*

Proof. Obviously, if $|\mathscr{C}(M^*)| = |\mathscr{C}(\hat{\mathcal{M}}(G))|$ no closing node exists in $\mathcal{T}$ and the algorithm stops without updating M^*. Otherwise if $|\mathscr{C}(M^*)| > |\mathscr{C}(\hat{\mathcal{M}}(G))|$, we need to show our algorithm defines a search tree $\mathcal{T}$ containing at least one closing node u. That is, we need to prove there exists a path $P(r,u)$ in $\mathcal{T}$ that connects the root r to u, where u is the first closing node. Let us suppose this path does not exist, i.e. there exists a node $x \in \mathcal{T}$ such that $P(r,x) \subset P(r,u)$, where a pruning operation is carried out.

Note that each node in $P(r,u)$ satisfies condition 1 of Definition 10, therefore, $\rho(x) \subseteq \hat{\mathcal{M}}(G)$ and $\overline{\rho}(x) \cap \hat{\mathcal{M}}(G) = \emptyset$. A pruning operation on node x is carried out either because one of the four pruning steps successes or because $\ell_x = \emptyset$. Let us analyze first the pruning steps at node x.

Condition 1 implies $P1$ fails on x. Moreover, since x does not satisfy condition 2, then $|Z_x \cup \mathcal{I}_x| < |\mathscr{C}(\hat{\mathcal{M}}(G))| - 1$. Hence, $P2$ and $P3$ fail, since at least two additional labels are needed to obtain a feasible solution from x. Additionally, $P4$ fails on x too. Indeed, by definition: $lb_x \leq |\mathscr{C}(\hat{\mathcal{M}}(G_x)) \setminus \{Z_x \cup \mathcal{I}_x\}|$. Therefore, we have the following relationships:

$$\begin{aligned} |\{Z_x \cup \mathcal{I}_x\}| + lb_x &\leq |\{Z_x \cup \mathcal{I}_x\}| + |\mathscr{C}(\hat{\mathcal{M}}(G_x)) \setminus \{Z_x \cup \mathcal{I}_x\}| \\ &= |\mathscr{C}(\hat{\mathcal{M}}(G_x))| \\ &\leq |\mathscr{C}(\hat{\mathcal{M}}(G))| \\ &< |\mathscr{C}(M^*)| \end{aligned}$$

Finally, the application of rules $R1$ and $R2$, implies $\ell_x = E_x \setminus \{(i,j) : \mathscr{C}(i,j) \in \{Z_x \cup \mathcal{I}_x\}\}$. Since, condition 2 does not hold on x, then $|Z_x \cup \mathcal{I}_x| < |Z_u \cup \mathcal{I}_u|$ that is, there exists a label set $C' \neq \emptyset$ such that $C' = (Z_u \cup \mathcal{I}_u) \setminus (Z_x \cup \mathcal{I}_x)$. Therefore, ℓ_x contains at least all the edges whose label is in C'. □

5.4. *Edge Choice Policy*

In this section we describe the rule we applied to choose edges from the candidate edge set ℓ_u. Since the search tree $\mathcal{T}$ represents *all* the matchings of G, the idea consists of choosing first the edges that have more chances to belong to a maximum matching of G. In this way, when such an edge is selected, the convergence of the incumbent solution M^* to the optimum is faster. On the other hand, when it is rejected, the success probability of $P1$ (on the r-nodes) is increased since building feasible solutions results more difficult.

The chance of a given edge to belong to a maximum matching of G is evaluated by considering its *incidence* value defined as follows:

Definition 13 *Given an undirected and labeled graph $G = (V, E, C)$ and an edge $(i,j) \in E$, we denote by $\pi(i,j)$ its incidence in G, corresponding to the degree sum of its vertices. Formally, $\pi(i,j) = |\delta(i)| + |\delta(j)|$.*

Note that, if (i,j) belongs to a maximum matching and it is rejected then any other maximum matching must cover at least one vertex between i and j. Therefore, the lower $\pi(i,j)$, the lower the chance to find such an alternative maximum matching. For instance, if $\pi(i,j) = 2$ then it is not possible to produce a maximum matching if (i,j) is rejected. Hence, we apply the following rule to choose edges in ℓ_u:

R3. (*Incidence Rule*): *Choose edge $(i,j) \in \ell_u$ such that $\pi(i,j) \leq \pi(i',j')$, $\forall (i',j') \in \ell_u$.*

5.5. *Heredity property*

In this section we introduce the *heredity property* that shows the strict relationship between the lower bound value lb_u of a node u, the necessary label set $\mathcal{I}_u$ and the corresponding elements of its children. We will show that when the heredity property can be applied then the operations needed to compute both the necessary label set $\mathcal{I}_u$ and the lower bound on any node u of the search tree can be hugely reduced (Theorem 15).

Given a node $u \in \mathcal{T}$, let v and w be its left and right children, respectively. The heredity property holds on v and w when the pruning step $P1$ fails on them. In particular, we recall that if $P1$ fails on v, then condition $|\mathcal{M}(G_v)| = |\mathcal{M}(G_u)| - 1$ is verified; on the other hand, if $P1$ fails on w, then condition $|\mathcal{M}(G_w)| = |\mathcal{M}(G_u)|$ is verified.

The heredity property states that the necessary label set $\mathcal{I}_u$ is a subset of $\mathcal{I}_v$ and $\mathcal{I}_w$, i.e. $\mathcal{I}_u \subseteq \mathcal{I}_v$ and $\mathcal{I}_u \subseteq \mathcal{I}_w$. Consequently, the children of u can directly inherit the necessary label set of their father, that is $\mathcal{I}_v = \mathcal{I}_u \cup \mathcal{N}_v$ and $\mathcal{I}_w = \mathcal{I}_u \cup \mathcal{N}_w$, where $\mathcal{N}_v$ and $\mathcal{N}_w$ are the additional necessary labels to be computed on v and w, respectively. We saw in subsection 5.3 that the necessary label sets are used into the pruning steps, to reduce the size of the search tree. However, we experimentally verified that this reduction is consistent only on the r-nodes. Therefore, we decided to compute the set of additional necessary labels only on these nodes and to set $\mathcal{N}_v = \emptyset$, that is $\mathcal{I}_v = \mathcal{I}_u$. We formally state this by the following rule:

R4. (*Necessary label Set Rule*): *Let $u \in \mathcal{T}$ be a node in the search tree, v and w be its right and left children, respectively: set $\mathcal{I}_v = \mathcal{I}_u$ and $\mathcal{I}_w = \mathcal{I}_u \cup \mathcal{N}_w$.*

Moreover, the heredity property states that $lb_v \geq lb_u - 1$ and $lb_w \geq lb_u - |\mathcal{N}_w|$. That is, the lower bound of a given node in the search tree can be computed by considering the lower bound of its father, reducing,

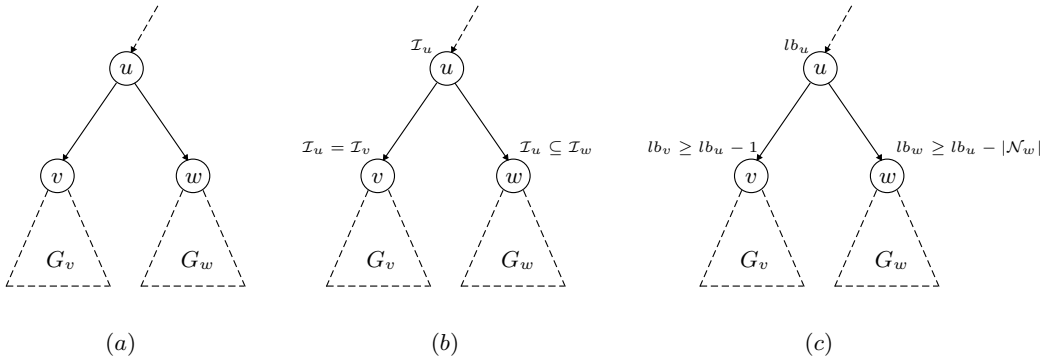


Fig. 4. *(a) A node $u \in \mathcal{T}$ of the search tree and its right and left children. (b) The relationships among the necessary label sets $\mathcal{I}_u$, $\mathcal{I}_v$ and $\mathcal{I}_w$. (c) The relationships among the lower bound values lb_u, lb_v and lb_w.*

therefore, the number of applied pruning steps on the node.

Before introducing the heredity property Theorem, consider the following lemma:

Lemma 14 *Let G be an undirected and labeled graph and let G' be the subgraph of G obtained removing an edge (i, j) of G. If $|\mathcal{M}(G')| = |\mathcal{M}(G)|$ then $|\mathscr{C}(\hat{\mathcal{M}}(G'))| \geq |\mathscr{C}(\hat{\mathcal{M}}(G))|$.*

Proof. The proof is straightforward by observing that, from the hypothesis, every maximum matching of G' is a maximum matching of G. □

Theorem 15 (Heredity Property) *Let u be a node of $\mathcal{T}$ where a branching operation on the edge (i, j) is performed, and let v and w be its left and right children, respectively (Figure 4a). The following conditions hold:*

Case 1: If $|\mathcal{M}(G_v)| = |\mathcal{M}(G_u)| - 1$ then $\mathcal{I}_u \subseteq \mathcal{I}_v$ (Figure 4b).
Case 2: If $|\mathcal{M}(G_w)| = |\mathcal{M}(G_u)|$ then $\mathcal{I}_u \subseteq \mathcal{I}_w$ (Figure 4b).
Case 3: If $|\mathcal{M}(G_v)| = |\mathcal{M}(G_u)| - 1$, then $lb_v \geq lb_u - 1$ (Figure 4c).
Case 4: If $|\mathcal{M}(G_w)| = |\mathcal{M}(G_u)|$, then $lb_w \geq lb_u - |\mathcal{N}_w|$ (Figure 4c).

Proof.

Case 1: We need to prove that any label $c \in \mathcal{I}_u$ is necessary for v, that is $\mathcal{I}_u \subseteq \mathscr{C}(\mathcal{M}(G_v))$ for any $\mathcal{M}(G_v)$. Let us suppose, by contradiction, there exists a maximum matching $\mathcal{M}(G_v)$ such that $c \in \mathcal{I}_u$ and $c \notin \mathscr{C}(\mathcal{M}(G_v))$. Since $|\mathcal{M}(G_v)| = |\mathcal{M}(G_u)| - 1$, then $\{\mathcal{M}(G_v) \cup (i, j)\}$ is a maximum matching of G_u. However, from rule $R2$ we have that $\mathscr{C}(i, j) \neq c$ and therefore $c \notin \mathscr{C}(\mathcal{M}(G_v) \cup (i, j))$, that is a contradiction because c is necessary.

Case 2: The proof is straightforward by applying the same reasoning of the previous case.

Case 3: First we need to prove that $lb_u - 1$ is a lower bound for v, i.e. $lb_u - 1 \leq |\mathscr{C}(\mathcal{M}(G_v)) \setminus \{Z_v \cup \mathcal{I}_v\}|$ for each $\mathcal{M}(G_v)$. Let us suppose, by contradiction, $lb_u - 1 > |\mathscr{C}(\mathcal{M}(G_v)) \setminus \{Z_v \cup \mathcal{I}_v\}|$, then we have the following relationships:

$$lb_u > |\mathscr{C}(\mathcal{M}(G_v)) \setminus \{Z_v \cup \mathcal{I}_v\}| + 1 \tag{20}$$
$$= |\mathscr{C}(\mathcal{M}(G_v)) \setminus \{Z_u \cup \mathscr{C}(i,j) \cup \mathcal{I}_u\}| + 1 \tag{21}$$
$$= |\mathscr{C}(\mathcal{M}(G_v) \cup (i,j)) \setminus \{Z_u \cup \mathscr{C}(i,j) \cup \mathcal{I}_u\}| + 1 \tag{22}$$
$$> |\mathscr{C}(\mathcal{M}(G_u)) \setminus \{Z_u \cup \mathcal{I}_u\}| \tag{23}$$

Relationship (21) follows from rule $R4$, i.e. $\mathcal{I}_v = \mathcal{I}_u$, and from $R1$, i.e. $Z_v = Z_u \cup \mathscr{C}(i,j)$. Relationship (22) holds regardless of whether $\mathscr{C}(i,j)$ belongs to $\mathscr{C}(\mathcal{M}(G_v))$ or not. Finally, since $|\mathcal{M}(G_v)| = |\mathcal{M}(G_u)| - 1$ then $\mathcal{M}(G_v) \cup (i,j)$ is a maximum matching of G_u and relationship (23) follows. From this last inequality a contradiction follows since, by definition, lb_u is a lower bound of node u.

Since any branching on the search tree is performed if and only if $lb_u \geq 2$, then we need to analyze the value of lb_v in the following two cases: $lb_u - 1 \geq 2$ and $lb_u - 1 = 1$. In the former case, since the pruning steps $P2$, $P3$ and $P4$ fail, the algorithm sets $lb_v = lb_u - 1$. In the latter case, the application of $P3$ sets $lb_v = 2$. In both cases $lb_v \geq lb_u - 1$.

Case 4: First we need to prove that $lb_u - |\mathcal{N}_w|$ is a lower bound for w, i.e. $lb_u - |\mathcal{N}_w| \leq |\mathscr{C}(\mathcal{M}(G_w)) \setminus \{Z_w \cup \mathcal{I}_w\}|$ for each $\mathcal{M}(G_w)$. Let us suppose, by contradiction, $lb_u - |\mathcal{N}_w| > |\mathscr{C}(\mathcal{M}(G_w)) \setminus \{Z_w \cup \mathcal{I}_w\}|$ then we have the following relationships:

$$lb_u > |\mathscr{C}(\mathcal{M}(G_w)) \setminus \{Z_w \cup \mathcal{I}_w\}| + |\mathcal{N}_w| \tag{24}$$
$$\geq |\mathscr{C}(\hat{\mathcal{M}}(G_w)) \setminus \{Z_w \cup \mathcal{I}_w\}| + |\mathcal{N}_w| \tag{25}$$
$$= |\mathscr{C}(\hat{\mathcal{M}}(G_w)) \setminus \{Z_u \cup \mathcal{I}_u \cup \mathcal{N}_w\}| + |\mathcal{N}_w| \tag{26}$$
$$\geq |\mathscr{C}(\hat{\mathcal{M}}(G_u)) \setminus \{Z_u \cup \mathcal{I}_u \cup \mathcal{N}_w\}| + |\mathcal{N}_w| \tag{27}$$
$$\geq |\mathscr{C}(\hat{\mathcal{M}}(G_u)) \setminus \{Z_u \cup \mathcal{I}_u\}| \tag{28}$$

Relationship (25) holds since $|\mathscr{C}(\mathcal{M}(G_w))| \geq |\mathscr{C}(\hat{\mathcal{M}}(G_w))|$. Relationship (26) follows by observing that: *i*) by construction $Z_w = Z_u$ and, *ii*) from rule $R4$ we have $\mathcal{I}_w = \mathcal{I}_u \cup \mathcal{N}_w$. Inequality (27) follows from Lemma 14, i.e. $\mathscr{C}(\hat{\mathcal{M}}(G_u)) \leq \mathscr{C}(\hat{\mathcal{M}}(G_w))$. Since at most $\mathcal{N}_w$ labels can be removed from $\mathscr{C}(\hat{\mathcal{M}}(G_u))$ then relationship (28) follows, that is a contradiction.

We need, now, to analyze the value of lb_v in the following two cases: $lb_u - |\mathcal{N}_w| \geq 2$ and $lb_u - |\mathcal{N}_w| < 2$. In the former case, since the pruning steps $P2$, $P3$ and $P4$ fail, the algorithm sets $lb_w = lb_u - |\mathcal{N}_w|$. In the latter case, the application of the pruning steps sets $lb_w = 2$. In both cases $lb_w \geq lb_u - |\mathcal{N}_w|$. □

The advantages derived from the heredity property in terms of performed operations should be evident. Given a generic node $u \in \mathcal{T}$, the standard sequence of operations performed on it when it is explored is: $P1$, computation of $\mathcal{N}_u$, $P2$, $P3$ and $P4$. Pruning step $P1$ is carried out on every node of the search tree. If this step fails then the hypotheses of Theorem 15 hold, and, by using the relationships derived from the heredity property, some of the remaining steps could be avoided.

5.6. *Combined Matching Property*

In this section we introduce a simple property (Theorem 16) that outlines the relationships among the cardinality of the maximum matchings computed on the subgraphs of G induced by different subsets of labels. Two consequences derive from Theorem 16: (i) an improved lower bound lb_u can be obtained at each node u of the tree (lower bounding technique in subsection 5.6.1), and, (ii) the total number of maximum matching problems to be solved during the execution of the algorithm can be reduced (upper bounding technique in subsection 5.6.2). We show in Section 6 the effectiveness of the application of these techniques in improving the computational times.

Theorem 16 *Let $G = (V, E, C)$ be an undirected and labeled graph and $\mathcal{M}(G^{c_i})$ be a maximum matching of the subgraph G^{c_i} induced by label c_i. Then $|\mathcal{M}(G^{c_1 \cup \ldots \cup c_p})| \leq \sum_{i=1}^{p} |\mathcal{M}(G^{c_i})|$.*

Proof. Let us consider the set S_i of the edges in $\mathcal{M}(G^{c_1 \cup \ldots \cup c_p})$ whose label is c_i, that is $S_i = \mathcal{M}(G^{c_1 \cup \ldots \cup c_p}) \cap E^{c_i}$ with $1 \leq i \leq p$. Obviously, we have $\bigcup_i S_i = \mathcal{M}(G^{c_1 \cup \ldots \cup c_p})$, and $|S_i| \leq |\mathcal{M}(G^{c_i})|$. Consequently, $|\mathcal{M}(G^{c_1 \cup \ldots \cup c_p})| = \sum_{i=1}^{p} |S_i| \leq \sum_{i=1}^{p} |\mathcal{M}(G^{c_i})|$. □

5.6.1. *Lower Bounding Technique*

According to Theorem 16 we can derive a simple lower bound to the optimum solution value $|\mathscr{C}(\hat{\mathcal{M}}(G))|$ as outlined in the following remark.

Remark 17 *Let $h = |C|$ be the size of the set of labels associated with G. Consider the maximum matchings $\mathcal{M}(G^{c_i})$ ordered according to their non decreasing size, i.e. $|\mathcal{M}(G^{c_{i_1}})| \geq |\mathcal{M}(G^{c_{i_2}})| \geq, \ldots, |\mathcal{M}(G^{c_{i_h}})|$. Let k be the highest index value such that $\sum_{j=1}^{k} |\mathcal{M}(G^{c_{i_j}})| < |\mathcal{M}(G)|$, then k is a lower bound to the optimum solution value, i.e. $|\mathscr{C}(\hat{\mathcal{M}}(G))| > k$.*

By applying Remark 17, we could compute a lower bound lb_u greater than 2, for each node of the search tree. We decided, however, to compute such a lower bound only at the root node r. Indeed, the computation of the maximum matchings $\mathcal{M}(G_u^c) \ \forall c \in C \setminus \{Z_u \cup \mathcal{I}_u\}$ and their subsequent sorting results too expensive, if carried out at each node of the tree.

5.6.2. *Upper Bounding Technique*

Remark 17 can also be used to reduce the total number of maximum matching problems to be solved by our algorithm, resulting in a great improvement of its performance. Indeed, our experimental results show that the 85% of the computational time of the algorithm is spent in the resolution of maximum matching problems. The idea consists in introducing a test into the pruning step $P3$ that checks whether it is possible to obtain a feasible solution of G that contains the edges of a maximum matching $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u \cup c})$ without computing it. Let us consider a generic node u of the search tree where the pruning step $P3$ has to be carried out. By Remark 17, we can avoid the computation of $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u \cup c})$ if the following condition is satisfied:

$$|\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u})| + |\mathcal{M}(G_u^c)| < |\mathcal{M}(G_u)| \tag{29}$$

Note that, both the two matchings $\mathcal{M}(G_u)$ and $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u})$ are already known since they were computed during the execution of the pruning steps $P1$ and $P2$. Hence, to check whether condition (29) is satisfied we need only to compute the maximum matchings $\mathcal{M}(G_u^c) \ \forall c \in C \setminus \{Z_u \cup \mathcal{I}_u\}$.

This operation is expensive as above mentioned. Therefore, we evaluated an upper bound U_u^c of $|\mathcal{M}(G_u^c)|$ whose computation is cheaper. Hence, in the pruning step $P3$ the maximum matching $\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u \cup c})$ will be computed only in case inequality $|\mathcal{M}(G_u^{Z_u \cup \mathcal{I}_u})| + U_u^c \geq |\mathcal{M}(G_u)|$ holds. Obviously, the lower U_u^c, the lower the number of maximum matchings to be computed.

In our implementation, we computed two different upper bounds and we set the value of U_u^c equal to the minimum one.

A first upper bound is the size of the maximum matching $\mathcal{M}(G^c)$ computed on the root of search tree. This value will be a good upper bound when the number of edges having the same label in the graph is low. The second upper bound is obtained computing a minimum vertex cover (MVC) on the residual graph G_u. MVC is an NP-complete problem [12], therefore, we did not compute its exact solution value, but an approximate one. In particular, we applied a simple two-approximate algorithm [9] whose running time is $O(V + E)$ that is cheaper than the computation of $\mathcal{M}(G_u^c)$ that requires $O(V^3)$ time.

5.7. *The Algorithm*

Let $G = (V, A, C)$ be an undirected and labeled graph and let M^* be an initial feasible solution of our problem returned by a given heuristic. Our branch and bound algorithm starts by analyzing the root node r and setting $Z_r = \emptyset$, $\ell_r = E$ and $\mathcal{I}_r = \emptyset$. Pruning step $P3$ is then applied to check whether the optimal solution $\hat{\mathcal{M}}(G)$ is composed by a single label. If this is the case the algorithm stops and returns the optimum solution. Otherwise, by applying Remark 17, the value of the lower bound lb_r is computed and the pruning step $P4$ is carried out. If it successes then no feasible solution better than M^* can be found in the tree and the algorithm stops. Otherwise, an edge in the set ℓ_r is selected to perform the first branch on the tree.

Let us consider now a generic iteration. Let u be a generic node of the search tree and v and w be its left and right children, respectively.

When the s-node v is explored, pruning step $P1$ is carried out. If $P1$ successes, then v is pruned and the algorithm backtracks to u. Otherwise, by the heredity property, the necessary label set and the lower bound value at the node are $\mathcal{I}_v = \mathcal{I}_u$ and $lb_v = lb_u - 1$. If $lb_v \geq 2$ then an edge in ℓ_v is selected to perform the branch. Otherwise, since $lb_v = 1$, pruning steps $P2$, $P3$ and $P4$ are applied. If all these pruning steps fail, then lb_v is set equal to 2 and an edge of ℓ_v is chosen to perform the branch. Otherwise, if $P3$ successes, then a new feasible solution better than M^* is found, the incumbent solution is updated and a backtrack to u is carried out (i.e. node v is pruned). On the other hand, if $P3$ fails and $P4$ successes the algorithm backtracks to u.

Let us consider now the visit of the r-node w. Step $P1$ is carried out first. If it successes then w is pruned and a backtrack operation is carried out. Otherwise, if it fails, then by the heredity property, the set $\mathcal{N}_w$ is computed. Two cases may occur now. If $\mathcal{N}_w = \emptyset$, then $\mathcal{I}_w = \mathcal{I}_u$. Since by construction $Z_w = Z_u$, then no additional pruning steps are carried out on w (indeed, they would fail as happened on the father u). In this case, an edge in ℓ_w is chosen to carry out the branch. On the other hand, if $\mathcal{N}_w \neq \emptyset$ then lb_w is set equal to $\max\{0, lb_w - |\mathcal{N}_w|\}$. If $lb_w \geq 2$ then an edge in ℓ_w is selected to perform a branch. Otherwise, according to the value of lb_w, the remaining pruning steps are applied such that either a feasible solution is found or lb_w is set equal to 2. In the former case, if the found feasible solution is better than the incumbent one M^*, then M^* is updated, w is pruned and a backtrack is carried out. In the latter case, an edge in ℓ_w is selected to perform a branch.

The algorithm stops when all the nodes of the tree are explored.

6. Computational results

Our branch-and-bound algorithm is coded in C and run on a 2.8 GHz Intel Xeon processor. All the proposed formulations were coded in C++ and solved using the mixed-integer optimizer of ILOG CPLEX version 10.1. The computation of the classical maximum matching is performed in $O(|V|^3)$ time following the implementation described by Gabow in [11]. We performed our tests on random graphs generated according to three parameters: the number of vertices n, the density of the graph d and the number of labels C. The value of n ranges from 10 to 100 with a step equal to 10. Parameter d is equal to $0.25, 0.5, 0.75$ and 1 so that the total number of edges in the graph is $m = dn(n-1)/2$. Parameter C is equal to 25%, 50%, 75% and 100% of n. We have a total of 160 different scenarios. We fixed a maximum running time equal to 3600 seconds, when the optimal solution is not found within this threshold the term *n.d.* (not determined) is reported in the tables.

Table 1, 2 and 3 report our results for $n = 40, 60, 80, 100$. All the values are average values over five different instances.

Comparison of the computational times required by CPLEX to solve the mathematical models is given

in Table 1. The first three columns report the scenario characteristics. Column *Card* reports the size of a maximum matching, column UB contains the initial upper bound to the solution value returned by a greedy heuristic, while column OPT the optimum solution value. Subsequent three columns contain the average CPU times (in seconds) required by CPLEX to solve the corresponding model. Finally, the last three columns report the percentage gap between the computational time of any couple of models. In particular, let A_1 and A_2 be two algorithms and t_1 and t_2 their execution times, respectively. To evaluate how much A_1 is faster than A_2 we computed the following gap value: $A_1 \rightarrow A_2 = \frac{t_1 - t_2}{\max\{t_1, t_2\}}$. If A_1 is faster than A_2, then the gap is negative and represents the percentage of saved time of A_1 respect to A_2. Otherwise, the gap is positive and represents the percentage of additional time spent by A_1 respect to A_2. When the difference in time between a couple of models was less than 1 second we set the corresponding gap equal to 0%. The average values reported in the last row are computed neglecting these cases.

Table 1: *Comparison of CPLEX computational times to solve the proposed mathematical formulations.*

n	d	C	Card	UB	OPT	Time			Gap		
						LR	ILP_1	ILP_2	LR → ILP_1	LR → ILP_2	ILP_1 → ILP_2
40	0.25	10	20	3.6	3.2	0.33	0.29	0.36	0.00%	0.00%	0.00%
40	0.25	20	20	5	4.8	0.71	0.90	0.74	0.00%	0.00%	0.00%
40	0.25	30	20	6.2	5.6	1.14	1.14	0.96	0.00%	0.00%	0.00%
40	0.25	40	20	8.8	6.6	1.05	0.92	0.99	0.00%	0.00%	0.00%
40	0.50	10	20	2	2	0.14	0.27	0.11	0.00%	0.00%	0.00%
40	0.50	20	20	3.2	3	1.49	1.73	1.56	0.00%	0.00%	0.00%
40	0.50	30	20	3.8	3.2	1.06	1.85	2.50	0.00%	-57.60%	0.00%
40	0.50	40	20	4.8	3.8	2.05	2.97	2.10	0.00%	0.00%	0.00%
40	0.75	10	20	1.6	1.6	0.14	0.16	0.31	0.00%	0.00%	0.00%
40	0.75	20	20	2.4	2	0.82	1.93	1.49	-57.51%	0.00%	0.00%
40	0.75	30	20	3	3	3.47	4.69	3.73	-26.01%	0.00%	0.00%
40	0.75	40	20	3.6	2.8	2.89	3.87	2.75	0.00%	0.00%	28.84%
40	1	10	20	1.2	1	0.19	0.21	0.24	0.00%	0.00%	0.00%
40	1	20	20	2	2	0.48	0.85	1.09	0.00%	0.00%	0.00%
40	1	30	20	2.6	2	2.47	3.77	3.05	-34.48%	0.00%	0.00%
40	1	40	20	3	3	6.38	7.19	7.42	0.00%	-14.02%	0.00%
60	0.25	15	30	3.2	3	1.73	1.12	0.99	0.00%	0.00%	0.00%
60	0.25	30	30	6	4.8	7.79	5.32	4.21	31.71%	45.93%	20.83%
60	0.25	45	30	7.2	6	26.47	38.57	28.63	-31.37%	-7.55%	25.77%
60	0.25	60	30	9	7	12.84	18.03	17.72	-28.79%	-27.53%	0.00%
60	0.50	15	30	2	2	1.89	1.07	1.26	0.00%	0.00%	0.00%
60	0.50	30	30	3.2	3	7.50	10.68	7.68	-29.78%	0.00%	28.09%
60	0.50	45	30	4.4	4	22.09	29.99	82.76	-26.34%	-73.31%	-63.76%
60	0.50	60	30	5.2	4.4	352.69	377.94	114.77	-6.68%	67.46%	69.63%
60	0.75	15	30	1.8	1.8	0.95	2.05	0.54	-53.66%	0.00%	73.76%
60	0.75	30	30	2.2	2	5.09	11.56	3.87	-55.97%	24.05%	66.56%
60	0.75	45	30	3.2	3	26.23	35.79	29.73	-26.71%	-11.78%	16.92%
60	0.75	60	30	4	3	59.00	55.06	53.14	6.68%	9.94%	3.49%
60	1	15	30	1.2	1	0.86	0.99	1.07	0.00%	0.00%	0.00%
60	1	30	30	2	2	6.88	2.74	5.93	60.17%	0.00%	-53.83%
60	1	45	30	3	2.2	23.62	45.31	40.53	-47.87%	-41.72%	10.56%
60	1	60	30	3.4	3	53.87	63.95	70.72	-15.76%	-23.83%	-9.58%
80	0.25	20	40	3.4	3	6.17	6.63	5.06	0.00%	17.99%	23.68%
80	0.25	40	40	5.6	5	150.60	26.60	65.34	82.34%	56.61%	-59.29%
80	0.25	60	40	7.4	6.2	826.62	677.78	636.80	18.01%	22.96%	6.05%
80	0.25	80	40	9.4	7.6	1381.21	1484.52	2388.18	-6.96%	-42.16%	-37.84%
80	0.50	20	40	2.4	2	12.05	8.16	11.63	32.28%	0.00%	-29.82%
80	0.50	40	40	4	3	53.74	55.95	48.83	-3.95%	9.14%	12.73%
80	0.50	60	40	4.4	4	1495.27	964.45	1557.55	35.50%	-4.00%	-38.08%
80	0.50	80	40	5.6	5	2752.41	3128.96	2459.19	-12.03%	10.65%	21.41%

continued on next page

Table 1 – *continued from previous page*

80	0.75	20	40	2	2	4.09	13.12	8.72	-68.83%	-53.12%	33.51%
80	0.75	40	40	2.6	2	51.27	83.91	54.64	-38.90%	-6.17%	34.88%
80	0.75	60	40	3.4	3	133.40	315.47	113.84	-57.71%	14.66%	63.91%
80	0.75	80	40	4.2	3.8	2536.69	2944.65	2815.67	-13.85%	-9.91%	4.38%
80	1	20	40	1.4	1	10.87	12.45	17.85	-12.69%	-39.11%	-30.26%
80	1	40	40	2.2	2	55.11	105.20	37.86	-47.61%	31.29%	64.01%
80	1	60	40	2.8	2.6	150.37	224.45	115.89	-33.01%	22.93%	48.37%
80	1	80	40	3.8	3	371.85	270.27	393.42	27.32%	-5.48%	-31.30%
100	0.25	25	50	4	3.2	36.30	25.08	24.43	30.91%	32.71%	0.00%
100	0.25	50	50	7	5	1656.39	2149.56	2516.45	-22.94%	-34.18%	-14.58%
100	0.25	75	50	8.6	n.d.	3600.34	3600.32	3602.33	n.d.	n.d.	n.d.
100	0.25	100	50	9.2	n.d.	3600.43	3600.60	3602.42	n.d.	n.d.	n.d.
100	0.50	25	50	2.6	2	51.88	41.96	37.29	19.12%	28.13%	11.14%
100	0.50	50	50	3.6	3	160.15	146.32	161.79	8.64%	-1.01%	-9.56%
100	0.50	75	50	5	4	3290.95	3224.37	2260.96	2.02%	31.30%	29.88%
100	0.50	100	50	5.4	5	3600.87	3600.81	3602.83	n.d.	n.d.	n.d.
100	0.75	25	50	2	2	45.31	33.30	30.04	26.51%	33.70%	9.78%
100	0.75	50	50	3	2	200.42	203.45	136.43	-1.49%	31.93%	32.94%
100	0.75	75	50	3.8	3	690.92	918.05	925.41	-24.74%	-25.34%	-0.80%
100	0.75	100	50	4	4	3601.36	3601.26	3603.23	n.d.	n.d.	n.d.
100	1	25	50	1.4	1	46.20	47.70	53.64	-3.14%	-13.87%	-11.07%
100	1	50	50	2.6	2	200.50	308.96	88.45	-35.10%	55.89%	71.37%
100	1	75	50	3	2.6	750.60	636.88	727.82	15.15%	3.03%	-12.50%
100	1	100	50	3.4	3	976.44	3100.79	787.90	-68.51%	19.31%	74.59%
Avg.									**-10.55%**	**1.81%**	**11.02%**

Table 1 shows that CPLEX computational times grow with the number of nodes n and with density d, as it was expected. Some of the instances with $n = 100$ were not solved within the fixed maximum running time. Column $LR \rightarrow ILP_1$ shows that LR requires less CPU time than ILP_1 in 29 scenarios and greater time in 14 scenarios. On average, CPLEX requires a computational time that is 10% less to solve LR than to solve ILP_1. From column $LR \rightarrow ILP_2$ we observe that CPLEX resulted faster in solving ILP_2 than in solving LR in 20 scenarios and slower in 19 scenarios. On average, CPLEX requires 1% less of CPU time to solve ILP_2 respect to LR. However, ILP_2 seems to have a better behavior when n increases. Indeed, for 16 out of 20 scenarios where it results faster than LR the value of parameter n is greater than or equal to 80. Finally, let us analyze the results for ILP_1 and ILP_2 reported in the last column of Table 1. On the average, ILP_2 is about 11% faster than ILP_1. In 26 scenarios ILP_2 results faster than ILP_1 and slower in 14. From these results we considered ILP_2 to be the best of the proposed models.

Table 2 and Table 3 show the computational times of our branch and bound. Table 2 compares the computational times of the algorithm in case the upper bounding technique is applied or not. Table 3 evaluates the performance of our branch and bound by comparing its computational times with those required by CPLEX to solve the ILP_2 formulation.

Consider Table 2. Column BB^- corresponds to the case when the upper bounding technique is not applied and column BB when it is applied. For each implementation two values are given: the total number of solved maximum matching problems (column MM) and the total computational time (in seconds). Last column reports the percentage gap between these values, respectively. We can see there are only three scenarios where the optimum solution was not found within the given threshold ($n = 80$, $d = 0.25$, $C = 80$ and $n = 100$, $d = 0.25$ and $C = 75, 100$). We can say the implementation with the upper bounding technique hugely reduces the total number of matchings in most of the scenarios and always results in better computational times. On the average, this implementation solves a number of maximum matchings that is 17% less than the other. There are five scenarios where this percentage is greater than 50%. For those few instances with a number of matchings greater in column BB^- than in column BB, the gap is less than 3%, and, in these

scenarios, the computational time does not worsen.

Table 2: *Comparison of the computational times of our branch and bound when the upper bounding technique is applied (BB) and when it is not applied (BB^-).*

n	**d**	**C**	**Card**	**UB**	**OPT**	**BB^-**		**BB**		**BB → BB^-**	
						MM	Time	MM	Time	MM	Time
40	0.25	10	20	3.6	3.2	768.60	0.01	746.40	0.01	-2.89%	0.00%
40	0.25	20	20	5	4.8	15117.40	0.16	10063.40	0.12	-33.43%	0.00%
40	0.25	30	20	6.2	5.6	126981.00	1.26	48715.80	0.57	-61.64%	0.00%
40	0.25	40	20	8.8	6.6	2049191.60	18.18	564350.00	6.72	-72.46%	-63.04%
40	0.50	10	20	2	2	10.00	0.00	10.00	0.00	0.00%	0.00%
40	0.50	20	20	3.2	3	475.20	0.00	386.60	0.00	-18.64%	0.00%
40	0.50	30	20	3.8	3.2	3412.80	0.06	2524.20	0.04	-26.04%	0.00%
40	0.50	40	20	4.8	3.8	29684.20	0.47	14479.80	0.25	-51.22%	0.00%
40	0.75	10	20	1.6	1.6	7.40	0.00	7.40	0.00	0.00%	0.00%
40	0.75	20	20	2.4	2	40.40	0.00	41.40	0.00	2.42%	0.00%
40	0.75	30	20	3	3	1047.80	0.02	828.40	0.02	-20.94%	0.00%
40	0.75	40	20	3.6	2.8	1536.40	0.03	806.20	0.02	-47.53%	0.00%
40	1	10	20	1.2	1	2.00	0.00	2.00	0.00	0.00%	0.00%
40	1	20	20	2	2	20.00	0.00	20.00	0.00	0.00%	0.00%
40	1	30	20	2.6	2	335.00	0.01	333.40	0.01	-0.48%	0.00%
40	1	40	20	3	3	2319.80	0.06	1731.00	0.05	-25.38%	0.00%
60	0.25	15	30	3.2	3	698.40	0.01	637.80	0.01	-8.68%	0.00%
60	0.25	30	30	6	4.8	276727.40	6.67	171028.60	4.55	-38.20%	-31.78%
60	0.25	45	30	7.2	6	6206103.00	129.51	2632610.00	61.25	-57.58%	-52.71%
60	0.25	60	30	9	7	85862828.20	1643.34	41993329.80	1048.81	-51.09%	-36.18%
60	0.50	15	30	2	2	15.00	0.00	15.00	0.00	0.00%	0.00%
60	0.50	30	30	3.2	3	1124.60	0.04	973.20	0.04	-13.46%	0.00%
60	0.50	45	30	4.4	4	50377.00	2.46	36943.80	1.43	-26.67%	-41.87%
60	0.50	60	30	5.2	4.4	1479888.60	50.89	821907.00	30.81	-44.46%	-39.46%
60	0.75	15	30	1.8	1.8	14.60	0.00	14.60	0.00	0.00%	0.00%
60	0.75	30	30	2.2	2	90.80	0.00	92.20	0.00	1.52%	0.00%
60	0.75	45	30	3.2	3	2566.80	0.13	2039.80	0.10	-20.53%	0.00%
60	0.75	60	30	4	3	17107.60	1.07	13347.20	0.66	-21.98%	0.00%
60	1	15	30	1.2	1	2.60	0.00	2.60	0.00	0.00%	0.00%
60	1	30	30	2	2	30.00	0.00	30.00	0.00	0.00%	0.00%
60	1	45	30	3	2.2	1895.80	0.11	1896.60	0.12	0.04%	0.00%
60	1	60	30	3.4	3	5122.60	0.30	3977.00	0.25	-22.36%	0.00%
80	0.25	20	40	3.4	3	948.60	0.04	897.00	0.04	-5.44%	0.00%
80	0.25	40	40	5.6	5	376025.80	15.70	253360.20	12.97	-32.62%	-17.39%
80	0.25	60	40	7.4	6.2	57837961.00	2151.72	40620075.00	1620.24	-29.77%	-24.70%
80	0.25	80	40	9.4	7.6	105835006.40	3601.93	85050540.80	3601.98	n.d.	n.d.
80	0.50	20	40	2.4	2	29.80	0.00	30.40	0.00	1.97%	0.00%
80	0.50	40	40	4	3	2516.60	0.22	2262.40	0.17	-10.10%	0.00%
80	0.50	60	40	4.4	4	202484.20	14.19	166515.80	12.20	-17.76%	-14.02%
80	0.50	80	40	5.6	5	7950753.20	571.87	4774864.20	398.79	-39.94%	-30.27%
80	0.75	20	40	2	2	20.00	0.00	20.00	0.00	0.00%	0.00%
80	0.75	40	40	2.6	2	1073.20	0.13	1088.80	0.10	1.43%	0.00%
80	0.75	60	40	3.4	3	6325.20	0.62	5655.60	0.51	-10.59%	0.00%
80	0.75	80	40	4.2	3.8	505501.80	44.86	453524.40	38.29	-10.28%	-14.65%
80	1	20	40	1.4	1	6.00	0.00	6.00	0.00	0.00%	0.00%
80	1	40	40	2.2	2	45.80	0.00	46.00	0.00	0.43%	0.00%
80	1	60	40	2.8	2.6	4327.20	0.49	4345.80	0.48	0.43%	0.00%
80	1	80	40	3.8	3	10966.00	1.23	9193.20	1.03	-16.17%	0.00%
100	0.25	25	50	4	3.2	11896.40	1.46	12045.80	1.58	1.24%	0.00%
100	0.25	50	50	7	5	5513712.60	649.38	5001211.60	400.15	-9.30%	-38.38%
100	0.25	75	50	8.6	n.d.	52354040.60	3601.83	34334121.80	3601.77	n.d.	n.d.

continued on next page

Table 2 – *continued from previous page*

100	0.25	100	50	9.2	n.d.	62095015.80	3602.11	48802153.00	3601.92	n.d.	n.d.
100	0.50	25	50	2.6	2	38.40	0.00	39.20	0.01	2.04%	0.00%
100	0.50	50	50	3.6	3	4249.40	0.87	3863.40	0.88	-9.08%	0.00%
100	0.50	75	50	5	4	494050.80	101.69	419359.80	81.49	-15.12%	-19.86%
100	0.50	100	50	5.4	5	20192236.60	3339.44	13808429.60	2834.44	-31.62%	-15.12%
100	0.75	25	50	2	2	25.00	0.00	25.00	0.01	0.00%	0.00%
100	0.75	50	50	3	2	1999.00	0.52	2022.60	0.51	1.17%	0.00%
100	0.75	75	50	3.8	3	13282.20	3.27	12136.60	2.98	-8.63%	0.00%
100	0.75	100	50	4	4	1023879.60	241.29	939194.80	230.62	-8.27%	-4.42%
100	1	25	50	1.4	1	4.60	0.00	4.60	0.00	0.00%	0.00%
100	1	50	50	2.6	2	56.60	0.02	57.20	0.02	1.05%	0.00%
100	1	75	50	3	2.6	8846.60	3.02	8916.80	2.82	0.79%	0.00%
100	1	100	50	3.4	3	15489.80	5.20	12661.20	3.73	-18.26%	-28.27%
Avg.										**-17.77%**	**-24.85%**

Consider now Table 3 where the comparison between the computational times of our branch and bound and those of the solver CPLEX, is reported. Last column shows the gap between the computational time of the algorithms is huge. Our algorithm is faster than CPLEX on 48 scenarios, and, in most of the cases the percentage gap is over 95%. Our branch and bound is slower in only 5 cases that correspond to the hardest solvable instances: sparse graphs and a great number of labels (see for example, instances where $n = 80$, $d = 0.25$ and $C = 60, 80$). Indeed, the size of the branch and bound tree grows with the optimum solution value, and therefore, the total number of nodes to be explored is greater. Note that, when the optimum solution value is low, our exact approach is faster and is able to solve instances in less than 4 seconds where CPLEX requires more than 700 seconds (e.g., $n = 100$, $d = 1$, $C = 75, 100$).

Table 3: *Comparison of the computational times of our branch and bound and those of CPLEX.*

n	**d**	**C**	**Card**	**UB**	**OPT**	**ILP_2**	**BB**	**GAP**
40	0.25	10	20	3.6	3.2	0.36	0.01	0.00%
40	0.25	20	20	5	4.8	0.74	0.12	0.00%
40	0.25	30	20	6.2	5.6	0.96	0.57	0.00%
40	0.25	40	20	8.8	6.6	0.99	6.72	-85.33%
40	0.50	10	20	2	2	0.11	0.00	0.00%
40	0.50	20	20	3.2	3	1.56	0.00	100.00%
40	0.50	30	20	3.8	3.2	2.50	0.04	98.40%
40	0.50	40	20	4.8	3.8	2.10	0.25	88.10%
40	0.75	10	20	1.6	1.6	0.31	0.00	0.00%
40	0.75	20	20	2.4	2	1.49	0.00	100.00%
40	0.75	30	20	3	3	3.73	0.02	99.46%
40	0.75	40	20	3.6	2.8	2.75	0.02	99.27%
40	1	10	20	1.2	1	0.24	0.00	0.00%
40	1	20	20	2	2	1.09	0.00	100.00%
40	1	30	20	2.6	2	3.05	0.01	99.67%
40	1	40	20	3	3	7.42	0.05	99.33%
60	0.25	15	30	3.2	3	0.99	0.01	0.00%
60	0.25	30	30	6	4.8	4.21	4.55	0.00%
60	0.25	45	30	7.2	6	28.63	61.25	-53.25%
60	0.25	60	30	9	7	17.72	1048.81	-98.31%
60	0.50	15	30	2	2	1.26	0.00	100.00%
60	0.50	30	30	3.2	3	7.68	0.04	99.48%
60	0.50	45	30	4.4	4	82.76	1.43	98.27%
60	0.50	60	30	5.2	4.4	114.77	30.81	73.16%
60	0.75	15	30	1.8	1.8	0.54	0.00	0.00%
60	0.75	30	30	2.2	2	3.87	0.00	100.00%

continued on next page

Table 3 – *continued from previous page*

60	0.75	45	30	3.2	3	29.73	0.10	99.66%
60	0.75	60	30	4	3	53.14	0.66	98.76%
60	1	15	30	1.2	1	1.07	0.00	100.00%
60	1	30	30	2	2	5.93	0.00	100.00%
60	1	45	30	3	2.2	40.53	0.12	99.70%
60	1	60	30	3.4	3	70.72	0.25	99.65%
80	0.25	20	40	3.4	3	5.06	0.04	99.21%
80	0.25	40	40	5.6	5	65.34	12.97	80.15%
80	0.25	60	40	7.4	6.2	636.80	1620.24	-60.70%
80	0.25	80	40	9.4	7.6	2388.18	3601.98	-33.70%
80	0.50	20	40	2.4	2	11.63	0.00	100.00%
80	0.50	40	40	4	3	48.83	0.17	99.65%
80	0.50	60	40	4.4	4	1557.55	12.20	99.22%
80	0.50	80	40	5.6	5	2459.19	398.79	83.78%
80	0.75	20	40	2	2	8.72	0.00	100.00%
80	0.75	40	40	2.6	2	54.64	0.10	99.82%
80	0.75	60	40	3.4	3	113.84	0.51	99.55%
80	0.75	80	40	4.2	3.8	2815.67	38.29	98.64%
80	1	20	40	1.4	1	17.85	0.00	100.00%
80	1	40	40	2.2	2	37.86	0.00	100.00%
80	1	60	40	2.8	2.6	115.89	0.48	99.59%
80	1	80	40	3.8	3	393.42	1.03	99.74%
100	0.25	25	50	4	3.2	24.43	1.58	93.53%
100	0.25	50	50	7	5	2516.45	400.15	84.10%
100	0.25	75	50	8.6	n.d.	3602.33	3601.77	n.d.
100	0.25	100	50	9.2	n.d.	3602.42	3601.92	n.d.
100	0.50	25	50	2.6	2	37.29	0.01	99.97%
100	0.50	50	50	3.6	3	161.79	0.88	99.46%
100	0.50	75	50	5	4	2260.96	81.49	96.40%
100	0.50	100	50	5.4	5	3602.83	2834.44	21.33%
100	0.75	25	50	2	2	30.04	0.01	99.97%
100	0.75	50	50	3	2	136.43	0.51	99.63%
100	0.75	75	50	3.8	3	925.41	2.98	99.68%
100	0.75	100	50	4	4	3603.23	230.62	93.60%
100	1	25	50	1.4	1	53.64	0.00	100.00%
100	1	50	50	2.6	2	88.45	0.02	99.98%
100	1	75	50	3	2.6	727.82	2.82	99.61%
100	1	100	50	3.4	3	787.90	3.73	99.53%

Figure 5 compares the behavior of both the algorithms respect to the size n of the instances. The average weighted time is computed considering the ratio between the average time of our algorithm and the number of times it is faster than CPLEX.

7. Conclusions

We addressed an NP-complete variant of the well know maximum matching problem, namely the Minimum Labeled Maximum Matching: given a graph G where a label is associated with each edge, we look for the maximum matching using the minimum number of different labels. It is a relatively new problem whose application is related to the timetabling problem [15]. We assessed its complexity and provided four alternative mathematical formulations, namely ILP_0, LR, ILP_1 and ILP_2. The LR and ILP_2 models were better than the others in terms of solver computational time. The former was the one we derived from the study of the lagrangean relaxation for which we characterized the optimal value of the lagrangean multipliers. The latter was obtained by using the blossom inequalities and binding degree ones derived from the dual solution associated with a maximum matching. Moreover, we proposed a branch and bound approach that

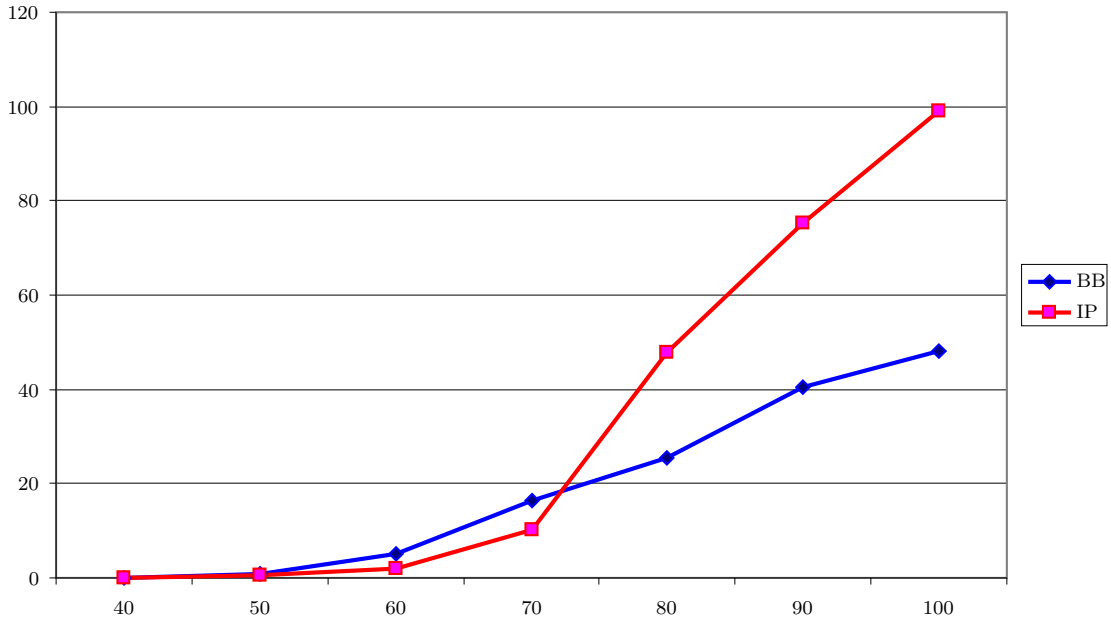


Fig. 5. *Comparison of the computational times of our branch and bound and those of CPLEX in solving* ILP_2.

deeply analyzes the relationship between the cardinality of the matching and the corresponding objective function value. Our wide experimentation shows our resolution approach finds on the average the solution in less than 25 seconds and it hugely outperforms solver CPLEX on instances with more than 70 vertices.

Acknowledgements

The authors thank the anonymous referee for his valuable comments in improving the provided mathematical formulations.

References

[1] Broersma, H. and X. Li. *Spanning trees with many or few colors in edge-colored graphs.* Discussiones Mathematicae Graph Theory, Vol. 17, pp. 259-269, 1997.

[2] Brüggemann, T. and J. Monnot and G.J. Woeginger. *Local search for the minimum label spanning tree problem with bounded color classes.* Operations Research Letters, Vol. 31, pp. 195-201, 2003.

[3] Carr, R.D. and S. Doddi and G. Konjedov and M. Marathe. *On the red-blue set cover problem.* In Proc. 11th ACN-SIAM Symposium on Discrete Algorithms, pp. 345-353, 2000.

[4] Cerulli, R. and A. Fink and M. Gentili and S. Voß. *Metaheuristics comparison for the minimum labelling spanning tree problem.* In The Next Wave on Computing, Optimization, and Decision Technologies, B.L. Golden, S. Raghavan, and E.A. Wasil, editors, Springer, New York, pp. 93-106, 2005.

[5] Cerulli, R. and A. Fink and M. Gentili and S. Voß. *Extensions of the Minimum Labeling Spanning Tree Problem.* Journal of Telecommunications and Information Technology, Vol. 4, pp. 39-45, 2006.

[6] Cerulli, R. and P. Dell'Olmo and M. Gentili and A. Raiconi. *Heuristic approaches for the Minimum Labelling Hamiltonian Cycle Problem.* Electronic Notes in Discrete Mathematics, Vol. 25, pp. 131-138, 2006.

[7] Cerulli, R. and M. Gentili and B. Golden and J.Silberholz. *Comparison of Heuristics for the Colorful Traveling Salesman Problem.* Submitted to Computers and Operations Research.

[8] Chang, R.-S. and S.-J. Leu. *The minimum labeling spanning trees.* Information Processing Letters, Vol. 63, pp. 277-282, 1997.
[9] Cormen, T.H. and C.E. Leiserson and R.L. Rivest. *Introduction to algorithms.* MIT Press, Cambridge, MA, 2001.
[10] Edmonds, J. *Paths, trees, and flowers.* Canadian Journal of Mathematics, Vol. 17, pp. 449-467, 1965.
[11] Gabow, H.N. *An efficient implementation of Edmonds' algorithm for maximum matching on graphs.* Journal of the ACM, Vol. 23(2), pp. 221-234, 1976.
[12] Karp, R.M. *Reducibility Among Combinatorial Problems.* In Complexity of Computer Computations, R. E. Miller and J. W. Thatcher (editors), New York: Plenum, pp. 85-103, 1972.
[13] Korte B. and J. Vygen. *Combinatorial Optimization: Theory and Algorithms.* 3st edition, Springer, pp. 238, 2005.
[14] Krumke, S.O. and H.-C. Wirth. *On the minimum label spanning tree problem.* Information Processing Letters, Vol. 66, pp. 81-85, 1998.
[15] Monnot, J. *The labeled perfect matching in bipartite graphs.* Information Processing Letter, Vol. 96, pp. 81-88, 2005.
[16] Monnot, J. *On Complexity and Approximability of the Labeled Maximum/Perfect Matching Problems.* ISAAC, Vol. 3827, pp. 934-943, 2005.
[17] Wan, Y. and G. Chen and Y. Xu. *A note on the minimum label spanning tree.* Information Processing Letters, Vol. 84, pp. 84-99, 2002.
[18] Xiong, Y. and B. Golden and E. Wasil. *Worst-case behavior of the MVCA heuristic for the minimum labeling spanning tree problem.* Operations Research Letters, Vol. 33, pp. 77-80, 2005.
[19] Xiong, Y. and B. Golden and E. Wasil. *A one-parameter genetic algorithm for the minimum labeling spanning tree problem.* IEEE Transactions on Evolutionary Computation, Vol. 9(1), pp. 55-60, 2005.
[20] Xiong, Y. and B. Golden and E. Wasil. *The Colorful Traveling Salesman Problem.* Operations Research-Computer Science Interfaces Series. Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, Vol. 37, pp. 115-123, 2007.