UNIVERSITÀ DEGLI STUDI DI ROMA TRE
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Exact algorithms for a discrete metric labeling problem

GAIA NICOSIA[1], ANDREA PACIFICI[2]

RT-DIA-90-04 **Aprile 2004**

(1) Dipartimento di Informatica e Automazione
Università di Roma Tre,
Via della Vasca Navale, 79
00146 Roma, Italy.
`(nicosia@dia.uniroma3.it)`

(2) Dipartimento di Ingegneria dell'Impresa,
Università di Roma "Tor Vergata",
Via del Politecnico 1
I-00133 Roma, Italy.
`(pacifici@disp.uniroma2.it)`

ABSTRACT

We are given a edge-weighted undirected graph $G = (V, E)$ and a set of labels/colors $C = \{1, 2, \dots p\}$. A nonempty subset $C_v \subseteq C$ is associated with each vertex $v \in V$. A coloring of the vertices is feasible if each vertex v is colored with a color of C_v. A coloring uniquely defines a subset $E' \subseteq E$ of edges having different colored endpoints. The problem of finding a feasible coloring which defines a minimum weight E' is, in general, NP-complete. In this work we first propose polynomial time algorithms for some special cases, namely when the input graph is a tree, a cactus or with bounded treewidth. Then, an implicit enumeration scheme for finding an optimal coloring in the general case is described and computational results are presented.

1 Introduction

The problem we address in this work was inspired by the following application in Flexible Manufacturing Systems. A set of assembly operations, with precedence constraints among them, must be processed by a set of multi-purpose machines with different capabilities, i.e., every operation may be processed only on a subset of the machines. A *part transfer* occurs every time a part (subassembly) completes its processing on one machine and must be transferred to another machine for the next processing operation [10]. Obviously, it is desirable to assign each operation to a feasible machine minimizing the number of part transfers, in order to reduce possible machines setup and transportation costs.

The problem may be modelled as follows: a graph $G = (V, E)$ (representing the n operations and their pairwise-relationships, e.g., precedences, connections, etc.) and a set of colors $C = \{1, 2, \dots p\}$ (representing the flexible machines) are given together with nonnegative weights $w : E \rightarrow \mathbb{Z}_+$ on the edges. Each vertex (operation) $v \in V$ is associated to a set $C_v \subseteq C$ of *feasible colors* (machines) for that vertex. A coloring $\phi : V \rightarrow C$ of the vertices is feasible if $\phi(v) \in C_v$. The objective is to find a feasible coloring of the vertices, so that the total weight of edges with multi-colored endpoints is minimum. Observe that, due to the particular objective function, we do not need to consider orientation for precedences/edges. When only assembly operations are considered, the resulting graph G is a tree, while other types of relations, like dis-assembly operations, require considering a general undirected graph.

Any coloring uniquely defines a subset $E' \subseteq E$ of edges having different colors assigned to the endpoints. E' is a multiway cut, in the sense that its removal disconnects vertices with different colors. Our problem consists of finding a coloring which defines a multiway cut having minimum cost. When a color $i \in \bigcap_{v \in V} C_v$ exists, our problem is trivial: simply color all the vertices with i thus obtaining an empty set of edges with different colored endpoints. When $p = 2$ ($C = \{1, 2\}$), the problem reduces to a standard maximum $\{s, t\}$-flow/minimum $\{s, t\}$-cut computation on the graph obtained by merging all the vertices having $C_u = \{1\}$ (resp. $C_u = \{2\}$), in one vertex s (resp. one vertex t), and setting the edge capacities for the new graph. Unfortunately, the problem is NP-hard, for any $p \geq 3$, since it is a generalization of the Multiway Cut problem, whose complexity was investigated by Dahlhaus *et al.* in [4].

It is possible to show that our problem admits a 2-approximation algorithm, as shown in [9], where Kleinberg and Tardos introduce the *Metric Labeling Problem*. This may be formulated as follows: given an an undirected graph $G = (V, E)$ with nonnegative edge weights, and a set L of *labels*, a *labeling* is an assignment of one label to each vertex. The cost of a labeling is based on the contribution of two sets of terms. A nonnegative *assignment cost* $c(v, i)$ that one pays when assigning label i to vertex v; and a *separation cost* such that, for all edges $e = uv$, if we assign $u \in V$ and $v \in V$ to labels $i \in L$ and $j \in L$ then one pays $w_e d(i, j)$, where d is a metric on the set of labels L. As already mentioned, our problem is a special case of the metric labeling problem introduced in [9], when the metric d is discrete (or uniform), i.e. $d(i, j) = \delta_{ij}$ (where $\delta_{ij} = 0$ if $i = j$ and $\delta_{ij} = 1$, otherwise). As a consequence, we will refer to our problem as *Discrete Metric Labeling Problem* or DMLP .

Another problem related to DMLP is the *Colored* Multiway Cut problem studied in [6, 7]. Erdös and Székely introduce a generalization of multiway cut where a partial coloring of the vertices (i.e., a subset $V' \subseteq V$ and an assignment of colors to the elements of V') is given and the problem consists of finding an extension of the coloring to all the vertices V in such a way that the total cost of the multiway cut thus defined is minimized. It is easy to see that this problem is a special case of DMLP. The authors also provide a $O(np^2)$ algorithm for trees.

The paper is organized as follows. In the next section, some polynomial algorithms are presented for special classes of instances of DMLP. In particular, the cases corresponding to G being a (*i*) tree in Section 2.1, (*ii*) a cycle in Section 2.2, (*iii*) a cactus in Section 2.3, and (*iv*) a graph with bounded treewidth in Section 2.4. In Section 3 we describe a combinatorial branch and bound algorithm for finding an optimal coloring in the general case. Subsection 3.1 illustrates the lower bound used for the enumeration scheme, while the results of an extensive computational experience are reported in Section 4. Finally, in Section 5, some conclusions are drawn.

2 Polynomial time algorithms

In this section we present dynamic programming procedures for solving DMLP when G is: (*i*) a tree, (*ii*) a cycle, (*iii*) a cactus, and (*iv*) a graph with bounded treewidth.

2.1 DMLP on trees

We now describe a dynamic programming algorithm for finding an optimal solution of DMLP , when G is a tree, in $O(|V|p)$ time.

Let the root of the tree $T = (V, E)$ be any fixed vertex $r \in V$. We denote by $S_u(r)$, or simply by S_u, the sub-tree rooted in u with respect to the root r, i.e, the connected component that contains u, in the graph obtained by removing the edges of the path between u and the root. A vertex v adjacent to u on this path is called the *parent* of u and u is a *child* of v. A *leaf*, with respect to the root r, is any vertex $u \neq r$ whose degree is 1.

The validity of a dynamic programming algorithm is guaranteed by the following lemma.

Lemma 2.1 *Let $f : V \to C$ be an optimal coloring of the tree $T = (V, E)$ such that $f(v) = i$, then any optimal coloring of S_v obtained by setting $C_v = \{i\}$, is also optimal in the whole tree.*

Proof. Trivial by contradiction. In fact, if a better solution $\bar{f}$ exists for the sub-tree S_v, a better solution for the whole tree can be obtained by simply exchanging the coloring f relative to this sub-tree, with $\bar{f}$. □ □

Lemma 2.1 suggests the following procedure: for each vertex $v \in V$ and $i \in C$ we compute the quantity $F_v(i)$ as the *minimum weight of the coloring relative to the sub-tree S_v, when vertex v is colored i.* The values of $F_v(i)$ are initialized as follows. For each leaf v of T, let

$$F_v(i) = \begin{cases} 0 & \text{if } i \in C_v; \\ +\infty & \text{otherwise.} \end{cases} \tag{1}$$

The following recursive relation holds:

$$F_v(i) := \sum_{u \text{ child of } v} \left(\min_{h \in C_u} \{F_u(h) + w(uv)\delta_{h,i}\} \right) \tag{2}$$

The minimum cost for DMLP is therefore:

$$z^* = \min_{h \in C_r} \{F_r(h)\}\,. \tag{3}$$

Starting from the optimal coloring of the root

$$f(r) := \arg\min_{h \in C_r} \{F_r(h)\} \tag{4}$$

an optimal coloring of all the vertices may be obtained by simple backtracking:

$$f(u) = \arg\min_{h \in C_u} \left\{F_u(h) + w(uv)\delta_{h,f(v)}\right\}. \tag{5}$$

Based on the Recursion 2 we may derive an optimal solution in $O(|V|p)$. In fact, it is easy to see that in $O(|V|p^2)$ iterations we are able to find $F_v(i)$ for all $v \in V$, $i \in C$. Computation of the minima in the above equations 2 and 5 may be devised in time $O(1)$ by keeping track of two additional data for each vertex v of T, during the computation of $F_v(i)$.

Namely, the minimum

$$F_v^{\min} = \min_{i \in C_v}\{F_v(i)\}$$

and its argument $i^{\min}$. Therefore Equation 2 and 5 become

$$F_v(i) := \sum_{u \text{ child of } v} \left(\min\left\{F_u^{\min} + w(uv); F_u(i)\right\}\right) \tag{6}$$

and

$$f(u) = \begin{cases} i^{\min} & \text{if } F_v(i) = F_u^{\min} + w(uv) \\ i & \text{if } F_v(i) = F_u(i). \end{cases} \tag{7}$$

We now show how to extended the dynamic programming procedure to deal with a more general problem that includes vertex assignment costs, as in the Metric Labeling Problem [9]. Assume there is an additional cost $c(i, v)$ for assigning color i to vertex v, then it is enough to modify Expression 6 as follows.

$$F_v(i) := \sum_{u \text{ child of } v} \left(\min\left\{F_u^{\min} + w(uv); F_u(i)\right\}\right) + c(i, v) \tag{8}$$

The procedure for trees can be easily extended to cycles and then, by taking into account vertex assignment costs, to a more general class of graphs, namely *cacti*, as shown in the next sections.

2.2 DMLP on cycles

In the following we present an algorithm for solving DMLP on a cycle. Let $C = \{v_1, v_2, \ldots, v_n, v_1\}$ be a cycle. The basic idea is to exploit the algorithm for trees in the following way: pick any vertex v of the cycle and duplicate it in order to obtain a path (see Figure 1). In a feasible solution, the resulting pair of vertices, as they are the same vertex, need to be assigned the same color: we may force this condition by imposing the set of feasible colors of the two vertices to be the same single color $\{i\}$ and iterating the procedure for finding an optimal coloring of the path for all feasible colors $i \in C_v$. The best solution among the $|C_v|$ alternatives is then chosen.

Since we have to solve $|C_v|$ problems on a path, the complexity of this procedure is $O(p^2 n)$ and, as in the case of trees, it can be immediately extended to deal with vertex assignment costs.

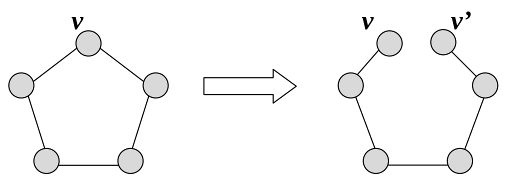

Figure 1: Duplicating vertex v

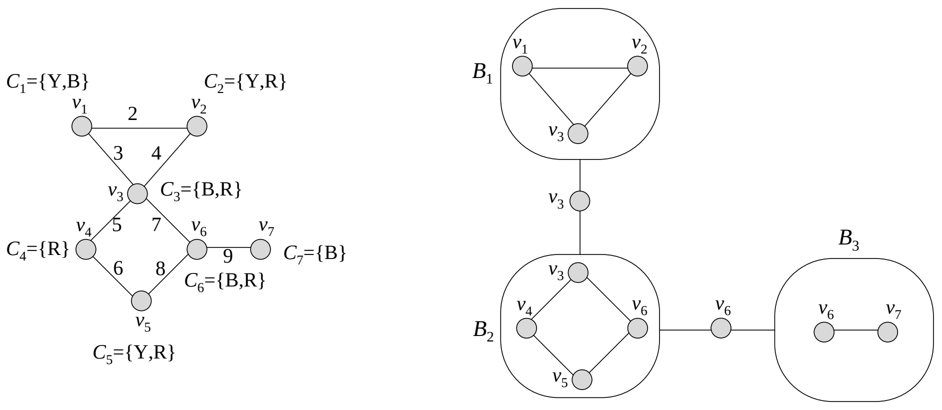

Figure 2: A cactus and its block-cutvertex tree.

2.3 DMLP on cacti

Based on the procedures for trees and cycles, an algorithm for a more general class of graphs, namely *cacti*, may be devised. A *cactus* is a simple connected graph with the property that every edge belongs to at most one cycle.

Before illustrating the procedure, we need some additional notation. A connected graph without any cutvertex is called *block*. A *block of a graph G* is a maximal subgraph of G such that it is a block. Every graph is the union of its (edge-disjoint) blocks (and two blocks intersect in a cutvertex). In a cactus, every block is either an edge or a cycle. It is also useful to introduce a tree called *block-cutvertex tree* describing the block structure of any given graph. The vertices are given by the union of the set of blocks and the set of cutvertices. Two vertices are adjacent if one corresponds to a block B_h and the other to a cutvertex c_j such that c_j is in B_h. An *end-block* is a block of a graph such that it correspond to a leaf in its block-cutvertex tree (that is, a block with only one cut vertex).

We solve DMLP on a cactus G by using the structure of the block-cutvertex tree T. At each step, the algorithm solves a special instance of the problem on one end-block of G (which, we recall, is either an edge or a cycle) and determines a certain cost, which is then associated to the corresponding cutvertex as an assignment cost. The algorithm proceeds by eliminating the blocks one by one from G until a "root cutvertex" is reached.

The algorithm works as follows.

1. Initialization: let the *root* of G be any fixed cut-vertex $r \in V$ and set the *artificial* vertex assignment costs $c(v, i) = 0$ for each vertex v and each color $i \in C_v$.

2. Repeat the following until there are no more end-blocks:

   (a) Pick an end-block B and let v be the corresponding cutvertex.

   (b) Evaluate $F_B(i)$, the minimum cost coloring of end-block B when cutvertex v is colored with i, by using one of the DP procedures described in the Sections 2.1 and 2.2;

   (c) Set the cost of assigning color i to vertex v as $c(v,i) = c(v,i) + F_B(i)$. Eliminate all vertices of block B but v.

3. Return $\min_{i \in C_r} c(r,i)$.

We illustrate the algorithm with a simple example.

Example 2.2 *Consider the cactus depicted in Figure 2, where the edges weights and the feasible colors are reported. We apply the above described procedure for finding an optimal coloring. We choose the cutvertex v_3 to be the root r. With reference to the depicted block-cutervertex tree, we evaluate the cost of coloring end-block B_1 and obtain: $F_{B_1}(B) = 6$ and $F_{B_1}(R) = 5$. Therefore, $c(v_3, B) = 6$ and $c(v_3, R) = 5$. Once block B_1 has been eliminated, the only end-block remaining is B_3 and its corresponding cutvertex is v_6. In this case, $F_{B_3}(B) = c(v_6, B) = 0$ and $F_{B_3}(R) = c(v_6, R) = 9$. Now, the only end-block is B_2, and its corresponding cutvertex is v_3. We then compute $F_{B_2}(B) = 13$ and $F_{B_2}(R) = 9$. Finally, we obtain $c(v_3, B) = 6 + 13 = 19$ and $c(v_3, R) = 5 + 9 = 14$. Therefore, the optimal solution has value 14 and corresponds to coloring all the vertices with R except v_1 and v_7, which are colored with B.*

Since for each block B_j with n_j vertices the time required to compute the values $F_{B_j}(i)$ is $O(p^2 n_j)$, then the whole procedure requires $O(p^2 n)$ time.

2.4 DMLP on graphs with bounded treewidth

In this section we sketch how our algorithm may be extended to solve DMLP in polynomial time for a more general class of graphs: those having *bounded treewidth*. Bounded treewidth graphs, with treewidth at most k, are also known as *partial k-trees*. Indeed the definitions of these equivalent classes of graphs are different. For further details on treewidth theory and algorithms and tree decomposition of a graph, we refer to [3, 12].

In order to illustrate the algorithm, we need some definitions and results on chordal graphs (those not having C_4 as an induced subgraph) and (partial) k-trees.

Definition 2.3 *Let $G = (V, E)$ be a graph. A vertex $v \in V$ is* simplicial *if $N(v)$ is a clique in G. An ordering $\{v_1, \ldots, v_n\}$ is a* perfect *(or* simplicial*) elimination ordering if, for all $i = 1, \ldots, n$, v_i is simplicial on the subgraph of G induced by $\{v_i, \ldots, v_n\}$.*

A well known result on chordal graphs, is that a graph is chordal if and only if it has a perfect elimination ordering [5]. Given a chordal graph, a perfect elimination ordering, may be easily found by a simple *Maximum Cardinality Search* algorithm (see [8]).

Definition 2.4 *Let $G = (V, E)$ be an undirected graph. G is a k-tree if it satisfies one the following:*

- *G is a complete graph K_k with k vertices;*
- *It has a (simplicial) vertex v with $N(v) = K_k$ and $G \setminus \{v\}$ is a k-tree.*

Note that any *k-tree* is a chordal graph with $\omega \leq k+1$.

Definition 2.5 *A partial k-tree is a subgraph of a k-tree.*

We call *embedding* of a graph G, a k-tree such that G is a subgraph of the k-tree, and k is minimum. Given k and an arbitrary graph G, an embedding of G, if it exists, and a corresponding perfect elimination ordering, can be found in time $O(n^{k+2})$, which is polynomial for fixed k [2].

In the following we describe a procedure, adapting the general idea indicated in [1], for solving DMLP on k-trees. This procedure can clearly be used for solving DMLP on partial k-trees, since once an embedding has been found, it is sufficient to set the costs of the added edges equal to zero and apply the algorithm on the resulting k-tree. Note that this procedure works in fact for any chordal graph with bounded clique size.

Given a perfect elimination ordering $\{v_1, \ldots, v_n\}$ of the vertices of G, we proceed in a similar fashion to what has been done for trees. It is important to emphasize the following trivial property.

Remark 2.6 *An optimal coloring of a clique of k vertices may be obtained by explicitly enumerating all the feasible labelings in $O(p^k)$ time; this is polynomially bounded for fixed k.*

Let G_j be the subgraph of G induced by $\{v_j, \ldots, v_n\}$. At the generic step j, we remove the simplicial vertex v_j (a *k-leaf*) from G_j. We then store suitable information associated to the k-clique[1] $K(v_j)$ formed by the neighbors of v_j in G_j. The procedure keeps removing vertices $v_{j+1}, v_{j+2}, \ldots$, until what is left of G is a k-clique (the *root* clique).

As in [1], if v is a vertex which is not in the root clique, we denote by $K(v)$ the set of neighbors of v in G_j (see Figure 3). That is, if $v = v_j$, $K(v) = N(v_j) \cap \{v_{j+1}, \ldots, v_n\}$. Vertex v together with $K(v)$ forms a $(k+1)$-clique K' containing $k+1$ k-cliques, namely $K_u = K' \setminus \{u\}$, where u is any vertex of $K(v)$.

For simplicity, let us denote by K be the k-clique $K(v_j)$ of G_j. A vertex is a *descendant* of K if it is adjacent to either vertices of K or descendants of K in G_j. From the definition of perfect elimination ordering, if v_j is a vertex of K, any descendant v_h of K is such that $h < j$.

Recalling the algorithm in Section (2.1), in the procedure for k-trees, a k-clique K plays a role similar to that of a generic vertex u of the tree and the descendants of K a role similar to that of the subtree S_u rooted at u.

When removing vertex v_j, what kind of information should we store, associated to the k-clique K? Suppose we fix the colors of the vertices of the k-clique $K = \{u_1, \ldots, u_k\}$. A *configuration* $\ell_K = \{l_{u_1}, \ldots, l_{u_k}\}$ is a feasible coloring of the vertices $\{u_1, \ldots, u_k\}$ of K. We want to store *the least costly contribution $A_K(\ell_K)$ to the objective function due to all the descendants of K when the vertices of K are colored according to a certain configuration ℓ_K*. Due to the adjacency relations of the descendants of K we may compute $A_K(\ell_K)$ recursively. Similarly to Lemma 2.1, it is not hard to show the following result.

[1]Note that here a k-clique is a set of k pairwise adjacent vertices and —differently from the standard notation —it is possibly not maximal.

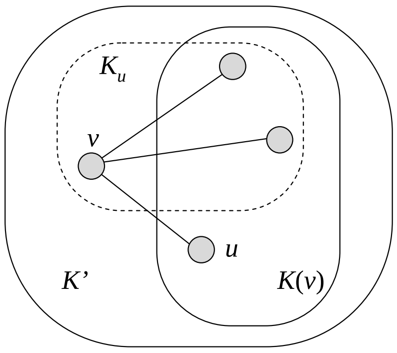

Figure 3: Simplicial vertex v.

Lemma 2.7 *Let $f : V \to C$ be an optimal coloring of the k-tree G such that the vertices of a k-clique $K = \{u_1, \ldots, u_k\}$ of G_j are colored $l_{u_1}, \ldots, l_{u_k}$. Then any optimal coloring of the descendants of K obtained by setting $C_u = \{l_u\}$ for any vertex u of K, is also optimal in the the whole tree.*

Observe that, differently from the tree case, we do not want to store in $A_K(\cdot)$, information about the cost of the edges of K. For a fixed configuration ℓ of the k-clique $K(v)$, we may compute $A_{K(v)}(\ell)$ as follows:

$$A_{K(v)}(\ell) = A_{K(v)}(\ell) + \min_{i \in C_v} \left\{ \sum_{u \in K(v)} (w(uv)\delta(i, l_u) + A_{K_u}(\ell_u)) \right\} \tag{9}$$

where ℓ_u is a configuration of K_u that assigns all $x \in K(v) \setminus \{u\}$ with l_x, as in ℓ, and vertex v colored with i.

On the ground of the above recursion we devise a procedure that finds an optimal solution to DMLP . Suppose the vertices are indexed according to a perfect elimination ordering, then the algorithm proceeds as follows.

1. Initialize: for any k-clique K of G and for any feasible color configuration ℓ of k-clique K, set $A_K(\ell) = 0$.

2. For $j = 1$ to $n - k$ do the following:

   (a) let $v = v_j$;

   (b) for any feasible color configuration ℓ of $K(v)$, use Expression 9 to compute $A_{K(v)}(\ell)$;

   (c) eliminate vertex v from G.

3. For any feasible color configuration ℓ of the root k-clique $K = \{v_{n-k}, \ldots, v_n\}$, compute the final cost

$$F(\ell) = A_K(\ell) + \sum_{uv \in E(K)} w(uv)\delta(l_u, l_v).$$

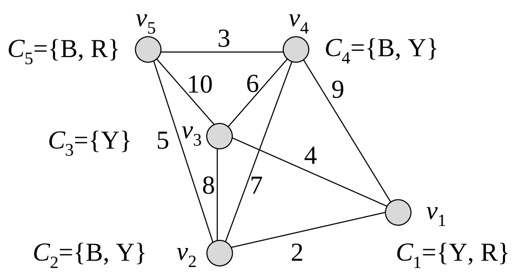

Figure 4: A k-tree with $k = 3$.

4. Return $\min\{F_K(\ell)$ s.t. ℓ is a feasible configuration of $K\}$.

We avoid description of the backtracking procedure to determine the optimal coloring of the vertices. Regarding the computational complexity, we note that (i) the number of k-cliques in a k-tree is $O(nk)$, (ii) the number of configurations for each k-clique is $O(p^k)$. Therefore, the time complexity of the initialization step is $O(nkp^k)$, while that of Step 2 is $O(nkp^{k+1})$. Thus, for fixed k, DMLP on k-trees can be solved in polynomial time.

Example 2.8 *Consider the* 3*-tree depicted in Figure 4, where the edges weights and feasible colors are reported. Let us apply the coloring algorithm to this instance. Observe that we need to store information associated to the following cliques:* $\{1,2,3\}$, $\{1,2,4\}$, $\{1,3,4\}$, $K(1) = \{2,3,4\}$, $\{2,3,5\}$, $\{2,4,5\}$, *and* $K(2) = \{3,4,5\}$, *which is the root clique with descendant vertices* 1 *and* 2*.*

Initially, we set $A_K(\ell_K) = 0$ *for each clique* K *and feasible color configuration* ℓ_K*. The first vertex to eliminate is* 1*, and for each feasible configuration of* $K(1) = \{2,3,4\}$ *according to Recursion 9 we have:*

$$\begin{aligned} A_{K(1)}(B,Y,B) &= A_{K(1)}(B,Y,B) + \\ &\quad \min\{9+2+A_{K_2}(Y,Y,B)+A_{K_3}(Y,Y,Y)+A_{K_4}(Y,B,Y); \\ &\quad 9+4+2+A_{K_2}(R,Y,B)+A_{K_3}(R,Y,Y)+A_{K_4}(R,B,Y)\} \\ &= 0+\min\{11+0;15+0\} = 11. \end{aligned}$$

Here, the minima are evaluated over the values corresponding to the feasible colors of vertex 1 *(i.e. Y and R) and* $K_2 = \{1,3,4\}$, $K_3 = \{1,2,4\}$, *and* $K_4 = \{1,2,3\}$*. Similarly, for the other feasible configurations we obtain:*

$$\begin{aligned} A_{K(1)}(Y,Y,B) &= 9; \\ A_{K(1)}(B,Y,Y) &= 2; \\ A_{K(1)}(Y,Y,Y) &= 0. \end{aligned}$$

All minima in the above equations are obtained coloring vertex 1 *with Y. In the next step we eliminate vertex* 2*, thus obtaining for* $K(2) = \{3,4,5\}$*:*

$$\begin{aligned} A_{K(2)}(Y,B,B) &= A_{K(2)}(Y,B,B) + \\ &\quad \min\{8+A_{K_3}(B,B,B)+A_{K_4}(B,Y,B)+A_{K_5}(B,Y,B), \\ &\quad 12+A_{K_3}(Y,B,B)+A_{K_4}(Y,Y,B)+A_{K_5}(Y,Y,B)\} \\ &= 0+\min\{8+0+0+11;12+0+0+9\} = 19. \end{aligned}$$

Here, the minima are evaluated over the values corresponding to the feasible colors of vertex 2 *(i.e. B and Y) and* $K_3 = \{2,4,5\}$, $K_4 = \{2,3,5\}$, *and* $K_5 = \{2,3,4\}$. *Similarly, for the other feasible configurations we obtain:*

$$\begin{aligned}
A_{K(2)}(Y,Y,B) &= \min\{17,5\} = 5;\\
A_{K(2)}(Y,B,R) &= \min\{24,21\} = 21;\\
A_{K(2)}(Y,Y,R) &= \min\{22,5\} = 5.
\end{aligned}$$

All minima in the above equations are obtained by coloring vertex 2 *with Y. Eventually, we are left with the root clique* $K = K(2) = \{3,4,5\}$ *and, using the equation in Step 3, we may compute the final cost for any feasible color configuration* ℓ_K *for the clique.*

$$\begin{aligned}
A_K(Y,B,B) &= 16+19 = 35\\
A_K(Y,Y,B) &= 13+5 = 18\\
A_K(Y,B,R) &= 19+21 = 40\\
A_K(Y,Y,R) &= 13+5 = 18
\end{aligned}$$

There are two optimal solutions corresponding to coloring vertices 3, 4, *and* 5 *with Y, Y, B or Y, Y, R, respectively. By standard backtracking, we derive the optimal coloring for vertices* 1 *and* 2 *(i.e., Y) for both the optimal solutions.*

3 An enumeration scheme for DMLP

In this section we present a branch-and-bound scheme for solving DMLP on arbitrary graphs.

In the branch-and-bound enumeration tree, which illustrates successive decompositions of the original problem, every node of the enumeration tree represents a particular instance of DMLP in which the set C_u of feasible colors for each vertex u has been changed with respect to the original instance. In particular, any node i of the enumeration tree represents a feasible partial coloring $f_i : S_i \to C$ of the vertices of $S_i \subseteq V$. The root node is associated to the original problem where the coloring of no vertex has been decided yet (i.e., $S_{\text{root}} = \emptyset$.) The children of any node i of the enumeration tree, that represent successive decomposition of the subproblem associated to i, are obtained as follows. Pick a vertex $u \in V \setminus S_i$ and generate $|C_u|$ children of i, one for every feasible color for u. Let $C_u = \{q_1, q_2, \ldots, q_h\}$ and let $j(1), j(2), \ldots, j(h)$ be the children of i in the enumeration tree. For each $j(l)$ child of i, $f_{j(l)}$ is an extension of f_i. More precisely, $f_{j(l)} : S_i \cup \{u\} \to C$ with $f_{j(l)}(u) = q_l$. Then, any child of i represents a feasible partial coloring of $S_i \cup \{u\}$.

We next show how to obtain two different lower bounds for DMLP.

3.1 Tree lower bound

Consider two instances I and I' of problem DMLP differing only in the weights of the edges: in particular, for each edge weight w_e in instance I, the corresponding edge-weight w'_e in instance I' is such that $w_e \geq w'_e$. It is trivial to show that the following holds.

Lemma 3.1 *Let* z *and* z' *be the optimal solution values for the two instances* I *and* I', *then* $z \geq z'$.

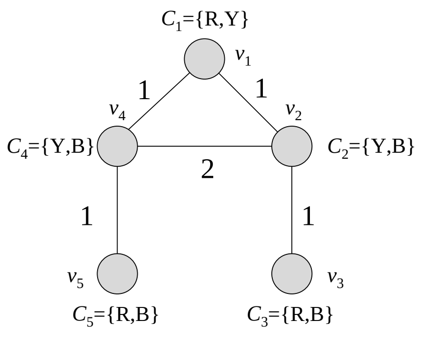

Figure 5: The maximum weight spanning tree does not provide the best bound.

Based on the preceding Lemma 3.1, we may easily derive a lower bound for the optimal solution value of DMLP as follows. Consider any spanning tree T of G and let $w'_e = w_e$ for all $e \in T$, else let $w'_e = 0$. Then, any optimal solution for DMLP on T with weights w and feasible colors $\mathcal{C}$ is feasible for the original instance and its value z_T is equal to the optimal solution value z^* of G with weights w'. Then, by Lemma 3.1, z_T is not greater than z^*.

Clearly, it would be desirable to know the spanning tree T^* producing the largest lower bound, i.e. $\max\{z_T$ such that T is a spanning tree of $G\}$. Observe that T^* is not, in general, a maximum weight spanning tree. Consider, for instance, the graph in Figure 5, in which the lower bound obtained via the maximum weight spanning tree has value 1, while any other spanning tree T provides a bound of $z_T = 2$.

3.2 Neighborhood lower bound

Another lower bound on the optimal solution to DMLP can be derived by estimating the contributions of all vertices of the graph to the objective function. In particular, let the contribution z_u of each vertex u correspond to the optimal solution value of DMLP on the star defined by u and its neighbors in $N(u)$. Given an optimal coloring $f : V \to C$, we have clearly that $z_u \leq \sum_{v \in N(u)} w(uv)\delta_{f(u),f(v)}$. Then, summing up the contributions z_u of all the vertices $u \in V$, since each edge of graph G is considered at most twice, $\frac{1}{2}\sum_{u \in V} z_u$ is a lower bound on the value of the optimal solution of DMLP.

For instance consider the graph depicted in Figure 6. Here $z_u = \min\{5, 6\} = 5$, obtained by assigning Y, Y, Y, R, R to vertices u, v_1, v_2, v_3, v_4, respectively.

3.3 Generalized tree lower bound

A generalization of the lower bounds previously described is illustrated hereafter. Let $\mathcal{T} = \{T^{(1)}, T^{(2)}, \ldots, T^{(q)}\}$ be a family of subtrees of G. Let us denote with $\gamma_e(\mathcal{T})$ the number of occurrences of edge $e = uv$ in the family $\mathcal{T}$ (i.e. the number of trees that include e). Let μ^i_e be the contribution of edge e in an optimal solution f^i restricted to subtree T^i, i.e., $\mu^i_e = w_e$ if $e = uv \in T^{(i)}$ and $f^i(u) \neq f^i(v)$ or $\mu^i_e = 0$, otherwise, $i = 1, \ldots, q$. Let μ^*_e be the contribution of edge e in the optimal solution of DMLP (on the whole graph G). Finally, let γ_e denote the number of trees of $\mathcal{T}$ including edge e and $\gamma = \max_{e \in E}\{\gamma_e\}$.

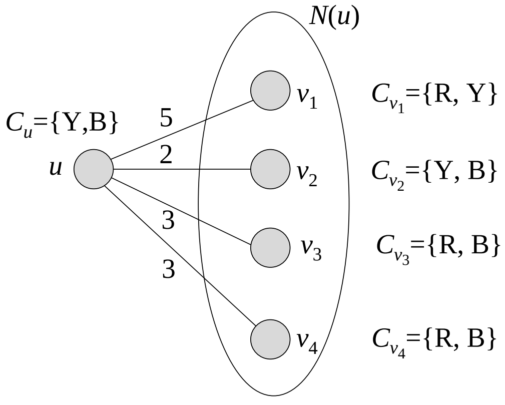

Figure 6: Neighborhood lower bound for vertex u.

Clearly,

$$\sum_{e \in T^{(i)}} \mu_e^i \leq \sum_{e \in T^{(i)}} \mu_e^*.$$

Therefore,

$$\sum_{i=1}^{q} \sum_{e \in T^{(i)}} \mu_e^i \leq \sum_{i=1}^{q} \sum_{e \in T^{(i)}} \mu_e^*.$$

The left hand side of the previous inequality is equal to $\sum_{e \in E} \sum_{i: e \in T^{(i)}} \mu_e^i$; while the right hand side is $\sum_{e \in E} \gamma_e \mu_e^*$.

As a consequence:

$$\sum_{e \in E} \frac{\sum_{i: e \in T^{(i)}} \mu_e^i}{\gamma} \leq \sum_{e \in E} \frac{\gamma_e \mu_e^*}{\gamma_e} = z^*.$$

Note that when $\gamma_e = 1$ for all $e \in E$ then $\mathcal{T}$ is a partitioning of E, while if $\gamma_e \leq 1$ for all $e \in E$ then $\mathcal{T}$ is a packing of E. The tree lower bound is an example of the latter case. The neighborhood lower bound corresponds to choosing the $T^{(i)}$'s as the stars induced by u and its neighbors in $N(u)$ for all $u \in V$. Observe that in this case $\gamma_e = 2$ for all $e \in E$.

4 Computational results

In this section we present the results of preliminary computational experiments on different classes of randomly generated instances.In particular, we compared the quality of the two lower bounds described in Sections 3.1 (tree lower bound) and 3.2 (neighborhood lower bound). We randomly generated forty instances where (i) the number of vertices varies from 50 to 300, (ii) the number of edges from 1,100 to less than 40,000, and (iii) the number of colors from 3 to 15. All the experiments were run on a PC with a 800 MHz clock, 256 MB RAM and coded in Fortran 90.

We measured the gaps of the two lower bounds with respect to the same upper bound. Gap$_i$ is defined as $\frac{UB-LB_i}{UB}$ for $i = 1, 2$. The benchmark upper bound UB is the cost of the solution obtained by coloring the vertices according the optimal algorithm on a maximum weight spanning tree. The results show that the first lower bound based on the spanning tree cost is highly ineffective (Section 3.2), while the second based on the neighborhood search has an average gap equal to 0,369.

5 Conclusions

In this work we propose exact polynomial time algorithms for DMLP when the given graph is a tree, a cactus, or with bounded treewidth. Note that our algorithm for trees outperforms the algorithm presented in [6, 7] for the special case of the colored multiway cut problem. Additionally, for the general NP-hard case, we present an implicit enumeration scheme.The main task for future research on this subject, due to its relevance in the applications, should concern the study of the effectiveness of the proposed approach including the following points.

1. Designing and testing dominance rules to be used at each node of the enumeration tree.

2. Designing and testing new lower bounds and/or enhancing the quality of the proposed lower bounds.

3. Comparing "combinatorial" bounds and bounds obtained by integer programming relaxations.

References

[1] Arnborg S., A. Proskurowski, (1989) Linear time algorithms for NP-hard problems restricted to partial k-trees, *Discr. Appl. Math.* 23, 11–24

[2] Arnborg S., D.G. Corneil, A. Proskurowski, (1987) Complexity of finding embeddings in a k-tree, *SIAM J. Algebraic Discr. Methods* 8, 277–284

[3] Brandstädt A., L. Van Bang, J.P. Spinrad, (1999) *Graph classes: a survey*, SIAM Monographs on Discrete Mathematics and Applications. ISBN 0-89871-432-X.

[4] Dahlaus E., D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, M. Yannakakis, (1994), The Complexity of Multiterminal Cuts, *SIAM J. of Comp.*, 23, 864–894.

[5] Dirac, G. (1961), On rigid circuit graphs, *Abh. Math. Sem. Univ. Hamburg*, 25, 71-76.

[6] Erdös P., L. Székeli (1992), Evolutionary trees: An integer multicommodity max-flow min-cut, *Adv. in Appl. Math.*, 13, 375–389.

[7] Erdös P., L. Székeli (1994), On weighted multiway cuts in trees, *Math. Progr.: Series A*, v.65 n.1, 93–105.

[8] Golumbic M.C., (1984) Algorithmic aspects of perfect graphs, in *Topics on perfect graphs*, (ed. C. Berge and V. Chvatal) North-Holland, 301–323

[9] Kleinberg J., É. Tardos, (1999) Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields, *Proc. FOCS '99*, 14–15.

[10] Lucertini M., D. Pacciarelli, A. Pacifici, (1996), Optimal flow Management in Assembly Processes: The Minimal Part Transfer Problem, *Syst. Sc.*, 22, 69-80.

[11] Nicosia G., A. Pacifici (2004), Exact algorithms for a discrete metric labeling problem, *Tech. rep. RT-DIA-90-2004*, Dip. Inf. e Autom., Univ. "Roma Tre".

[12] West D.B., (2001) *Introduction to Graph Theory*, 2nd Edition, Prentice Hall, NJ. ISBN 0-13-014400-2.