# A Greedy Approximation Algorithm for the Uniform Metric Labeling Problem Analyzed By a Primal-Dual Technique

EVANDRO C. BRACHT, LUIS, A. A. MEIRA, and F. K. MIYAZAWA
Universidade Estadual de Campinas

We consider the uniform metric labeling problem. This NP-hard problem considers how to assign objects to labels respecting assignment and separation costs. The known approximation algorithms are based on solutions of large linear programs and are impractical for moderate- and large-size instances. We present an $8 \log n$-approximation algorithm that can be applied to large-size instances. The algorithm is greedy and is analyzed by a primal-dual technique. We implemented the presented algorithm and two known approximation algorithms and compared them at randomized instances. The gain of time was considerable with small error ratios. We also show that the analysis is tight, up to a constant factor.

Categories and Subject Descriptors: G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph labeling*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Approximation algorithms, graph labeling

## 1. INTRODUCTION

In a traditional classification problem, we assign each of $n$ objects to one of $m$ labels (or classes). This assignment must be consistent with some observed data including pairwise relationships among the objects to be classified. More precisely, the classification problem can be defined as follows. Let $P$ be a set of objects, $L$ a set of labels, $w : P \times P \to \mathbb{R}^+$ a symmetric weight function, $d : L \times L \to \mathbb{R}^+$ a symmetric distance function, and $c : P \times L \to \mathbb{R}^+$ an assignment cost function. *Labeling* of $P$ over $L$ is a function $\phi : P \to L$. The *assignment cost* of labeling $\phi$ is the sum $\sum_{i \in P} c(i, \phi(i))$ and the *separation cost*

Research supported in part by FAPESP (Proc. 03/13815-0, 02/05715-3), CNPq (Proc. 306526/04–2, 478818/03–3, 471460/04–4, 490333/04–4) and ProNEx–FAPESP/CNPq (Proc. 2003/09925-5).
Authors' address: Evandro C. Bracht, Luis, A. A. Meira, and F. K. Miyazawa, Instituto de Computação, Universidade Estadual de Campinas, Caixa Postal 6176, 13084-971, Campinas–SP, Brazil; email: {evandro,meira,fkm}@ic.unicamp.br.

of labeling $\phi$ is the sum $\sum_{i,j \in P} d(\phi(i), \phi(j))w(i, j)$. The function $w$ indicates the strength of the relation between two objects and the function $d$ indicates the similarity between two labels. The cost of labeling $\phi$ is the sum of the assignment and separation costs. The *Metric Labeling Problem (MLP)* consists of finding a labeling of objects into labels with minimum total cost. Throughout this paper, we denote the size of the sets $P$ and $L$ by $n$ and $m$, respectively.

We focus our attention on the *Uniform Metric Labeling Problem (UMLP)*, where the distance $d(i, j)$ is 1 if $i \neq j$ and 0 otherwise, for all $i, j \in L$.

The MLP has several applications, many listed by Kleinberg and Tardos [1999]. Some applications occur in image processing [Besag 1986; Cohen 1986; Dubes and Jain 1989], biometrics [Besag 1974], text categorization [Chakrabarti et al. 1998], etc. An example of an application is the restoration of images degenerated by noise. In this case, an image can be seen as a grid of pixels, each pixel of which is an object that must be classified with a color. The assignment cost is given by the similarity between the new and old coloring, and the separation cost given by the color of a pixel and the color of its neighbors.

The MLP generalizes the Multiway Cut Problem, a known NP-hard problem [Dahlhaus et al. 1994], which was first introduced by Kleinberg and Tardos [1999]. It presents an $O(\log m \log\log m)$-approximation algorithm for the MLP, and a 2-approximation algorithm for the UMLP using the randomized rounding technique over a solution of a linear program. This linear program is too large to be used in large-size instances. It has $\Theta(mn^2)$ constraints and $\Theta(mn^2)$ variables. Because the UMLP also generalizes the Multiway Cut Problem, it is also NP-hard.

More recently, Gupta and Tardos [2000] presented a formulation for the truncated labeling problem, an MLP, where the labels are positive integers and the metric distance between labels $i$ and $j$ is given by the truncated linear norm, $d_{ij} = \min\{M, |i - j|\}$, where $M$ is the maximum value allowed. They presented an algorithm that is a 4-approximation algorithm for the truncated labeling problem and a 2-approximation for the UMLP. This algorithm generates a network flow problem instance where the weights of edges come from the assignment and separation costs of the original problem. The resulting graph has $O(n^2m)$ edges and the *Min Cut* algorithm is applied $O((m/M)(\log Q_0 + \log \varepsilon^{-1}))$ times in order to obtain a $(4 + \varepsilon)$-approximation, where $Q_0$ is the cost of the initial solution. This is also a high-time complexity to be used in practical instances.

Chekuri et al. [2001] developed a new linear programming formulation that is better for the nonuniform case. However, for the uniform case, it maintains a 2-approximation factor and has a higher time complexity. This is a consequence of solving a large linear program with $\Theta(n^2m^2)$ constraints and $\Theta(n^2m^2)$ variables.

In this paper, we present a fast approximation algorithm for the UMLP and prove that it is an $8 \log n$-approximation algorithm. We also compare the practical performance of this algorithm with the linear programming-based algorithms of Chekuri et al. [2001] and Kleinberg and Tardos [1999]. Although this algorithm has a higher approximation factor, the computational tests indicated

that the obtained solutions have values that are within 60% of the optimum. Moreover, the algorithm could obtain solutions for large instances in small amount of time.

In Section 2, we present a proof for a greedy algorithm for the Set Cover Problem via a primal-dual analysis. We then analyze the case when the greedy choice is relaxed to an approximated one. In Section 3, we present a general algorithm for the UMLP using an algorithm for the Set Cover Problem. This algorithm uses as a subroutine, an approximation algorithm for the Quotient Cut Problem. In Subsection 3.3, we show that the analysis is tight, up to a constant factor. In Section 4, we compare the presented algorithm with a linear programming-based algorithms. In Section 5, we present the concluding remarks.

## 2. A PRIMAL-DUAL ANALYSIS FOR A GREEDY ALGORITHM FOR THE SET COVER PROBLEM

The Set Cover Problem is a well-known optimization problem that generalizes many others. An instance of this problem consists of: a set $E_{\mathrm{SC}} = \{e_1, e_2, \ldots, e_n\}$ of elements, a family of subsets $\mathcal{U}_{\mathrm{SC}} = \{U_1, U_2, \ldots, U_m\}$, where $U_j \subseteq E_{\mathrm{SC}}$, and a cost $w_j$, for each $j \in \{1, \ldots, m\}$. A set $S \subseteq \{1, \ldots, m\}$, such that $\cup_{j \in S} U_j = E_{\mathrm{SC}}$ is a set cover of $E_{\mathrm{SC}}$. The goal of the problem is to find a set cover $Sol \subseteq \{1, \ldots, m\}$ of $E_{\mathrm{SC}}$ that minimizes $\sum_{j \in Sol} w_j$.

Chvátal [1979] presented a greedy $H_g$-approximation algorithm for the Set Cover Problem, where $g$ is the number of elements in the largest set in $\mathcal{U}_{\mathrm{SC}}$ and $H_g$ is the value of the harmonic function of degree $g$. We denote this algorithm by Greedy. This algorithm iteratively chooses the set with minimum amortized cost, which is the cost of the set divided by the number of noncovered elements. Once a set is chosen to be in the solution, all the elements in this set are considered as covered. (For more details on the Greedy algorithm, see Chapter 3 in Vazirani [2001].)

The Greedy algorithm can be rewritten as a primal-dual algorithm, with similar set of events. To present this algorithm, we first consider a formulation for the Set Cover Problem using binary variables $x_j$ for each set $U_j$, where $x_j = 1$ if, and only if, $U_j$ is chosen in the solution. The formulation consists on finding $x$ that

$$\begin{aligned} \text{minimize} \quad & \sum_j w_j x_j \\ \text{s.t.} \quad & \sum_{j : e \in U_j} x_j \geq 1 \qquad \forall e \in E_{\mathrm{SC}} \\ & x_j \in \{0, 1\} \qquad \forall j \in \{1, \ldots, m\} \end{aligned}$$

and the dual of the relaxed version consists on finding $\alpha$ that

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E_{\mathrm{SC}}} \alpha_e \\ \text{s.t.} \quad & \sum_{e \in U_j} \alpha_e \leq w_j \qquad \forall j \in \{1, \ldots, m\} \\ & \alpha_e \geq 0 \qquad \forall e \in E_{\mathrm{SC}} \end{aligned}$$

ALGORITHM PD-SC($E_{\mathrm{SC}}, \mathcal{U}_{\mathrm{SC}}, w$)
*Input:* Set of elements $E_{\mathrm{SC}}$, family of sets $\mathcal{U}_{\mathrm{SC}}$, each set $U_j \in \mathcal{U}_{\mathrm{SC}}$ with weight $w_j$.
*Output:* A set cover $Sol \subseteq \{1, \ldots, m\}$ of $E_{\mathrm{SC}}$.

**1.** $U \leftarrow E_{\mathrm{SC}}$; $T \leftarrow 0$ ; $Sol \leftarrow \emptyset$.
**2.** Let $\alpha_e \leftarrow 0$, for all $e \in E_{\mathrm{SC}}$.
**3.** While $U \neq \emptyset$ do
**4.** increase $T$ and $\alpha_e$ for each $e \in U$, uniformly, until we have
$$\sum_{e \in U_j \cap U} \alpha_e = w_j, \text{ for some } j.$$
**5.** $Sol \leftarrow Sol \cup \{j\}$.
**6.** $U \leftarrow U \setminus U_j$.
**7.** Return $Sol$.

Fig. 1. Primal–dual algorithm for the Set Cover Problem.

The primal-dual algorithm, which we denote by PD-SC, uses a set $U$ containing the elements not covered in each iteration and a variable $T$ with a notion of time associated with each event. At each iteration, the algorithm increases the variable $T$ and the (dual) variables $\alpha_e$ uniformly, for each element in $U$, until the inequality, corresponding to a set $U_j$, becomes tight. At this point, the algorithm chooses $j$ and declares each element of $U_j$ as covered. Algorithm PD-SC is presented in Figure 1.

We first present a proof of the approximation factor of algorithm PD-SC using a factor revealing primal-dual analysis. The idea of the proof is proposed in Jain et al. [2003], Section 9, and described in Vazirani [2001]. For completeness, we included the proof here, adapting the presentation in such a way that we can easily extend theorems 3 to 5.

LEMMA 1. *The sequence of events executed by the* Greedy *and* PD-SC *algorithms is the same.*

PROOF. Note that the value of $\alpha_e$ when $\sum_{e \in U_j \cap U} \alpha_e = w_j$ is equal to the amortized cost. Because $\alpha_e$ grows uniformly, it is clear that algorithm PD-SC, at each iteration, chooses a set with minimum amortized cost. □

LEMMA 2. *Let $U_j = \{e_1, e_2, \ldots, e_k\}$ and $\alpha_i$ the dual variable associated with $e_i$, generated by algorithm* PD-SC. *If $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k$, then $\sum_{i=l}^{k} \alpha_l \leq w_j$, for any $l$.*

PROOF. At the moment just before $\alpha_l$, all $\alpha_i$, $l \leq i \leq k$, have the same value, $\alpha_l$, and $e_1, \ldots, e_k$ are all uncovered. Suppose the lemma is false. In this case, $\sum_{i=l}^{k} \alpha_l > w_j$ and, in an instant strictly before $\alpha_l$, the set $U_j$ would enter in the solution and all of its elements would have $\alpha < \alpha_l$. This is a contradiction, because $U_j$ has at least one element greater than or equal to $\alpha_l$. Therefore, the lemma is valid. □

Algorithm PD-SC returns a primal solution $Sol$, with value $val(Sol) := \sum_{j \in Sol} w_j$, such that $val(Sol) = \sum_{e \in E_{\mathrm{SC}}} \alpha_e$. Note that $\alpha$ may not be dual feasible. If there exists a value $\gamma$ such that $\alpha/\gamma$ is dual feasible, i.e., $\sum_{e \in U_j} \alpha_e/\gamma \leq w_j$ for

each $j \in \{1, \ldots, m\}$, then, by the weak duality theorem, $val(Sol) \leq \gamma OPT$. The idea of the proof is to find a value $\gamma$ for which $\alpha/\gamma$ is dual feasible.

THEOREM 3. *The* PD-SC *Algorithm, for the Set Cover Problem, is an $H_g$-approximation algorithm, where $g$ is the size of the largest set in $\mathcal{U}_{\mathrm{SC}}$.*

PROOF. Consider an arbitrary set $U_j = \{e_1, \ldots, e_k\}$ with $k$ elements and cost $w$. Let $\alpha_i$ be the dual variable associated with element $e_i$ for $i = 1, \ldots, k$. Without loss of generality, assume that $\alpha_1 \leq \ldots \leq \alpha_k$.

If $\gamma$ is a value that corrects $\alpha$, such that $\alpha/\gamma$ is dual feasible, then

$$\sum_{i=1}^{k} \alpha_i/\gamma \leq w \tag{1}$$

Thus, a sufficient condition is

$$\gamma \geq \frac{\sum_{i=1}^{k} \alpha_i}{w} \tag{2}$$

Applying Lemma 2 for each value of $l$, we have

$$\begin{cases} l = 1, & k\alpha_1 \leq w, & \Rightarrow & \alpha_1/w \leq 1/k \\ l = 2, & (k-1)\alpha_2 \leq w, & \Rightarrow & \alpha_2/w \leq 1/(k-1) \\ & & \vdots & \\ l = k, & \alpha_k \leq w, & \Rightarrow & \alpha_k/w \leq 1 \end{cases}$$

Summing up the inequalities above, we have

$$\frac{\sum_{i=1}^{k} \alpha_i}{w} \leq H_k \tag{3}$$

Therefore, when $\gamma = H_g$, we obtain that $\alpha/\gamma$ is dual feasible. □

Now, let us consider a small modification in the previous algorithm. Instead of choosing the set with minimum amortized cost, we choose a set $U_{j'}$ with amortized cost at most $f$ times greater than the minimum. That is, if $U_{j*}$ is a set with minimum amortized cost in a given iteration, then the following inequality is valid

$$\frac{w_{j'}}{|U_{j'} \cap U|} \leq f \frac{w_{j*}}{|U_{j*} \cap U|}$$

This modification can be understood, in the primal-dual version of the algorithm, as a permission that $\sum_{e \in U_j \cap U} \alpha_e$ becomes greater than $w_j$ by a factor of, at most, $f$. We denote the algorithm with this modification by $\mathcal{A}_f$.

LEMMA 4. *Let $U_j = \{e_1, e_2, \ldots, e_k\}$ and $\alpha_i$ the time variable associated to the item $e_i$ generated by algorithm $\mathcal{A}_f$. If $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k$, then $\sum_{i=l}^{k} \alpha_l \leq f\, w_j$, for any $l$.*

PROOF. Suppose the lemma is false. In this case, $\sum_{i=l}^{k} \alpha_l > f\, w_j$ and, in an instant $T < \alpha_l$, the set $U_j$ would have entered in the solution, which is a contradiction. □

The following theorem can be proved analogously to Theorem 3.

THEOREM 5. *Algorithm $\mathcal{A}_f$ is an $f$ $H_g$-approximation, where $g$ is the size of the largest set in $\mathcal{U}_{\mathrm{SC}}$.*

## 3. A NEW ALGORITHM FOR THE UNIFORM METRIC LABELING PROBLEM

The algorithm for the UMLP uses some ideas presented by Jain et al. [2003] for the Facility Location Problem. To present these ideas, we use the notion of a star. A *star* is a connected graph where only one vertex, denoted as *center* of the star, can have a degree greater than one. Jain et al. [2003] present an algorithm that iteratively selects a star with minimum amortized cost, where the center of each star is a facility.

In the UMLP, we consider a labeling $\phi : P \to L$ as a set of stars, each one with a label in the center. Thus, a star for UMLP is a pair $(l, U_S)$, where $l \in L$ and $U_S \subseteq P$. The algorithm for the UMLP iteratively selects stars of reduced amortized cost, until all objects have been covered.

Given a star $S = (l, U_S)$ for the UMLP, where $l \in L$ and $U_S \subseteq P$, we denote by $\mathcal{C}(S)$ the cost of the star $S$, which is defined as

$$\mathcal{C}(S) = c(U_S) + \frac{1}{2}w(U_S)$$

where $c(U_S) = \sum_{u \in U_S} c_{ul}$ and $w(U_S) = \sum_{u \in U_S, v \in (P \setminus U_S)} w_{uv}$. That is, $\mathcal{C}(S)$ is the cost to assign each element of $U_S$ to $l$ plus half the cost to separate each element of $U_S$ from each element of $P \setminus U_S$. We pay just half of the separation cost, because the other half will appear when we analyze the elements in $P \setminus U_S$ against the elements in $U_S$.

Denote by $\mathcal{S}_{\mathrm{UMLP}}(L, P)$ the set of all possible stars of an instance. The algorithm uses $U$ as the set of unassigned objects, $(E_{\mathrm{SC}}, \mathcal{S}_{\mathrm{SC}}, w')$ as an instance for the Set Cover Problem, $\mathcal{U}$ as a collection, and $\phi$ the labeling being produced. Without loss of generality, we consider $\mathcal{S}_{\mathrm{SC}}$ as a multiset and, given a set $U_S \in \mathcal{S}_{\mathrm{SC}}$, we can obtain the associated star $S = (l, U_S)$, and vice versa. In Figure 2, we describe the algorithm more formally.

### 3.1 Analysis of the Algorithm

To analyze the algorithm we use the following notation:

- $val_{\mathrm{UMLP}}(\phi)$: the value, in the UMLP, of a labeling $\phi$.
- $val_{\mathrm{SC}}(\mathcal{U})$: the value, in the Set Cover Problem, of a collection $\mathcal{U}$.
- $\phi_{\mathrm{OPT}}$: an optimum labeling for UMLP.
- $\mathcal{U}_{\mathrm{OPT}}$: an optimum solution for the Set Cover Problem.
- $SC(\phi)$: a collection $\{U_{S_1}, U_{S_2}, \ldots, U_{S_k}\}$ related to a labeling $\phi = \{S_1, S_2, \ldots, S_k\}$.

LEMMA 6. *If $\phi$ is the solution returned by algorithm* GUL *and $\mathcal{U}$ the solution returned by algorithm $\mathcal{A}_{\mathrm{SC}}$ at step 4 of algorithm* GUL*, then*

$$val_{\mathrm{UMLP}}(\phi) \leq val_{\mathrm{SC}}(\mathcal{U})$$

ALGORITHM GUL$(L, P, c, w, \mathcal{A}_{\mathrm{SC}})$

*Input:* Set $L$ of labels; set $P$ of objects; assignment costs $c_{ui}$ for $i \in L$ and $u \in P$; separation costs $w_{uv}$ for $u, v \in P$; and an algorithm $\mathcal{A}_{\mathrm{SC}}$ for Set Cover.

*Output:* A labeling $\phi$ of $P$.

**1.** $E_{\mathrm{SC}} \leftarrow P$.
**2.** $\mathcal{S}_{\mathrm{SC}} \leftarrow \{U_S : S = (l, U_S) \in \mathcal{S}_{\mathrm{UMLP}}(L, P)\}$.
**3.** $w'_S \leftarrow \mathcal{C}(S) + \frac{1}{2}w(U_S)$, for all $S = (l, U_S) \in \mathcal{S}_{\mathrm{UMLP}}(L, P)$.
**4.** $\mathcal{U} \leftarrow \mathcal{A}_{\mathrm{SC}}(E_{\mathrm{SC}}, \mathcal{S}_{\mathrm{SC}}, w')$, let $\mathcal{U} = \{U_{S_1}, U_{S_2}, \ldots, U_{S_t}\}$.
**5.** $U \leftarrow P$.
**6.** For $k \leftarrow 1$ to $t$ do
**7.** Let $l \in L$ be such that $S_k = (l, U_{S_k})$;
**8.** $\phi(i) \leftarrow l$, for all $i \in U_{S_k} \cap U$;
**9.** $U \leftarrow U \setminus U_{S_k}$.
**10.** Return $\phi$.

Fig. 2. Greedy algorithm for the UMLP.

PROOF

$$
\begin{aligned}
val_{\mathrm{UMLP}}(\phi) &= \sum_{S \in \phi} \mathcal{C}(S) = \sum_{S \in \phi} \left( c(U_S) + \frac{1}{2} w(U_S) \right) && (4)\\
&\leq \sum_{U_S \in \mathcal{U}} (c(U_S) + w(U_S)) && (5)\\
&= \sum_{U_S \in \mathcal{U}} \left( \mathcal{C}(S) + \frac{1}{2} w(U_S) \right)\\
&= \sum_{U_S \in \mathcal{U}} w'_S = val_{\mathrm{SC}}(\mathcal{U})
\end{aligned}
$$

Equality (4) is valid because the stars in $\phi$ are disjoints and this can be checked by counting. Inequality (5) is valid because an assignment cost in $val_{\mathrm{UMLP}}(\phi)$ also appears in $val_{\mathrm{SC}}(\mathcal{U})$,

$$\sum_{S \in \phi} c(U_S) \leq \sum_{U_S \in \mathcal{U}} c(U_S)$$

and the separation cost $w_{uv}$ for any $u$ and $v$ such that $\phi(u) \neq \phi(v)$, that appears in $val_{\mathrm{UMLP}}(\phi)$ must appear, once or twice, in $val_{\mathrm{SC}}(\mathcal{U})$,

$$\sum_{S \in \phi} \frac{1}{2} w(U_S) = \sum_{u < v : \phi(u) \neq \phi(v)} w_{uv} \leq \sum_{U_S \in \mathcal{U}} w(U_S) \qquad \square$$

LEMMA 7. $val_{\mathrm{SC}}(\mathcal{U}_{\mathrm{OPT}}) \leq 2 val_{\mathrm{UMLP}}(\phi_{\mathrm{OPT}})$

PROOF

$$
\begin{aligned}
2 val_{\mathrm{UMLP}}(\phi_{\mathrm{OPT}}) &= 2 \sum_{S \in \phi_{\mathrm{OPT}}} \mathcal{C}(S) \geq \sum_{S \in \phi_{\mathrm{OPT}}} \left( \mathcal{C}(S) + \frac{1}{2} w(U_S) \right) && (6)\\
&= \sum_{t \in SC(\phi_{\mathrm{OPT}})} w'_t
\end{aligned}
$$

$$\geq \sum_{t \in \mathcal{U}_{\mathrm{OPT}}} w'_t \tag{7}$$
$$= val_{\mathrm{SC}}(\mathcal{U}_{\mathrm{OPT}})$$

where inequality (6) is valid because $\mathcal{C}(S) \geq \frac{1}{2} w(U_S)$ and inequality (7) is valid because $SC(\phi_{\mathrm{OPT}})$ is a solution for the Set Cover Problem, but not necessarily with optimum value. □

THEOREM 8. *If $I$ is an instance for the UMLP, $\phi$ is the labeling generated by algorithm* GUL *when running on $I$ and $\phi_{\mathrm{OPT}}$ is an optimum labeling for $I$ then*

$$val_{\mathrm{UMLP}}(\phi) \leq 2\beta\, val_{\mathrm{UMLP}}(\phi_{\mathrm{OPT}})$$

*where $\beta$ is the approximation factor of algorithm $\mathcal{A}_{\mathrm{SC}}$ used at Step 4 of algorithm* GUL.

PROOF. Let $\mathcal{U}$ be the solution returned by algorithm $\mathcal{A}_{\mathrm{SC}}$ (Step 4 of algorithm GUL) and $\mathcal{U}_{\mathrm{OPT}}$ an optimal solution for the corresponding Set Cover instance. In this case, we have

$$val_{\mathrm{UMLP}}(\phi) \leq val_{\mathrm{SC}}(\mathcal{U}) \tag{8}$$
$$\leq \beta\, val_{\mathrm{SC}}(\mathcal{U}_{\mathrm{OPT}}) \tag{9}$$
$$\leq 2\beta\, val_{\mathrm{UMLP}}(\phi_{\mathrm{OPT}}) \tag{10}$$

where inequality (8) is valid by Lemma 6, inequality (9) is valid because $\mathcal{U}$ is found by a $\beta$-approximation algorithm, and inequality (10) is valid by Lemma 7. □

To obtain an $8 \log n$-approximation algorithm for the UMLP, we need to present an algorithm $\mathcal{A}_{\mathrm{SC}}$ that is a $4 \log n$-approximation algorithm for Set Cover. The algorithm is based on an approximation algorithm for the Quotient Cut Problem, which we describe in the following subsection. The algorithm PD-SC cannot be directly applied in Step 4 of algorithm GUL, because it is polynomial in the size $\mathcal{S}_{\mathrm{SC}}$, but exponential in the size of $P$.

### 3.2 Algorithm $\mathcal{A}_{\mathrm{SC}}$

In this section, we show how to obtain a greedy algorithm for Set Cover without the explicit generation of all possible sets. The algorithm basically generates a graph $G_l$ for each possible label $l \in L$ and obtains a set $U_S$ with reduced amortized cost. The set, which must enter in the solution, is the smallest one considering the sets $U_S$ from $G_l$ for each $l \in L$.

Consider a label $l \in L$. We wish to find a set $U_S$ that minimizes $w_S / |U_S \cap U|$, where $U$ is the set of unassigned objects in the iteration. Denote by $G_l$ the complete graph with vertex set $V(G_l) = P \cup \{l\}$ and edge costs $w_{G_l}$ defined as follows

$$w_{G_l}(u, v) := w_{uv} \qquad \forall u, v \in P, u \neq v$$
$$w_{G_l}(l, u) := c_{ul} \qquad \forall u \in P$$

In other words, the cost of an edge between the label and an object is the assignment cost and the cost of an edge between two objects is the separation cost.

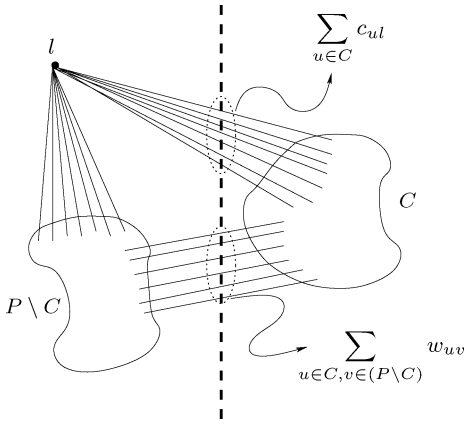


Fig. 3. Example of a cut $C$ that has the same cost $w'_S$ that the star $S = (l, C)$.

LEMMA 9. *For each set $C \subseteq P$, the cost $c(C) = \sum_{u\in C, v\notin C} w_{G_l}(u, v)$ has value $w'_S$, where $w'_S$ is the cost attributed in Step 3 of algorithm* GUL *to set indexed by star $S = (l, C)$.*

PROOF. The lemma can be proved by counting. In the Set Cover Problem, $w'_S$ is equal to

$$\mathcal{C}(S) + \frac{1}{2} w(U_S)$$

that is equal to the cost of the cut $C$. (See Figure 3). □

The problem of finding a set with smallest amortized cost $c(C)/|C|$ in $G_l$ is the Minimum Quotient Cut (QC) Problem, which can be defined as follows.

> *QC Problem* (*QC*). Given a graph $G$, weights $w_e \in \mathbb{R}^+$ for each edge $e \in E(G)$, and costs $c_v \in \mathbb{Z}^+$ for each vertex $v \in V(G)$, find a cut $C$ that minimizes $w(C)/\min\{\pi(C), \pi(\bar{C})\}$, where $w(C) := \sum_{u\in C, v\notin C} w_{uv}$ and $\pi(C) := \sum_{v\in C} c_v$.

If we define a cost function $c_l$ in each vertex of $G_l$ as

$$c_l(v) = \begin{cases} 1 & \text{if } v \in P \cap U \\ 0 & \text{if } v \in P \setminus U \\ n & \text{if } v = l \end{cases}$$

where $U$ contains the elements not covered in a given iteration, then a cut $C := \min_{l\in L}\{QC(G_l, w_l, c_l)\}$, returned by some $f$-approximation algorithm for the $QC$ Problem, corresponds to a set $U_S$, $S = (l, C)$, that is, at most, $f$ times greater than the set with minimum amortized cost. Algorithm $\mathcal{A}_f$, as stated in Theorem 5, can be implemented choosing this set instead of the set with minimum amortized cost. The following result follows from Theorems 5 and 8.

THEOREM 10. *If there exists an $f$-approximation algorithm for the QC Problem with time complexity $T(n)$, then there exists a $2f\,H_n$-approximation algorithm for the UMLP, with time complexity $O(n\,m\,T(n))$.*

The QC Problem is NP-hard and the best approximation algorithm has an approximation factor of 4 due to Freivalds [2003]. This algorithm has experimental time complexity $O(n^{1.6})$ when the degree of each vertex is a small constant and $O(n^{2.6})$ for dense graphs. Although this algorithm has polynomial time complexity estimated experimentally, it is not proved to run on polynomial time, in the worst case.

THEOREM 11. *There is an* $8 \log n$*-approximation algorithm for UMLP, with time complexity estimated in* $O(m\, n^{3.6})$.

It is important to observe that the size of an input $I$ for UMLP, $size(I)$, is $O(n(n+m))$. Thus, the estimated complexity of our algorithm is $O(size(I)^{2.3})$.

### 3.3 The Approximation Factor is $\Theta(\log n)$

In this subsection, we will show that the approximation factor is tight, up to a constant factor. We present a family of instances where the factor given by the algorithm is $H_n$. We suppose that $\mathcal{A}_{\mathrm{SC}}$ is an algorithm that selects the set with minimum amortized cost at each iteration.

Given a positive integer $n$ and positive values $\epsilon_1$ and $\epsilon_2$, where $0 < \epsilon_1 \ll \epsilon_2 \ll 1/n^2$, define an instance $\mathcal{I}_n = (L, P, c, w)$ for UMLP as follows

- $L = \{1, \ldots, n\}$
- $P = \{1, \ldots, n\}$
- for each $i \in P$ and $l \in L$, define $c_{il}$ as follows

$$c_{il} = \begin{cases} \epsilon_1 & \text{if } i \in \{1, \ldots, n-1\} \text{ and } l = i \\ \epsilon_2 & \text{if } i \in \{1, \ldots, n-1\} \text{ and } l = n \\ 1 & \text{if } i = n \text{ and } l = n \\ \infty & \text{otherwise} \end{cases}$$

- for each $i, j \in P$, define $w_{ij}$ as follows

$$w_{ij} = \begin{cases} \dfrac{1}{n-i+1} & \text{if } i \in \{1, \ldots, n-1\} \text{ and } j = n \\ 0 & \text{otherwise} \end{cases}$$

Instance $\mathcal{I}_n$ is presented in Figure 4. In the next theorem we show that algorithm GUL can obtain a solution of $\mathcal{I}_n$ with a factor of $\Theta(\log n)$ of the optimum.

THEOREM 12. *If algorithm* GUL *uses a greedy algorithm as subroutine for the Set Cover Problem, then* $\frac{\mathrm{GUL}(\mathcal{I}_n)}{\mathrm{OPT}(\mathcal{I}_n)} \to H_n$ *as* $\epsilon_1, \epsilon_2 \to 0$*, where* $\mathrm{OPT}(\mathcal{I}_n)$ *is the value of an optimum solution of* $\mathcal{I}_n$ *for UMLP.*

PROOF. The optimal solution for instance $\mathcal{I}_n$ is the star $(n, P)$ with cost $1 + \epsilon_2(n-1)$. We will now prove that the solution obtained by algorithm GUL can be formed by the stars $(1, \{1\}), (2, \{2\}), \ldots, (n, \{n\})$; whose cost is $H_n + \epsilon_1(n-1)$. The theorem can be proved by induction on the number of iterations.

Given a star $S = (l, U_S)$, for $l \in L$ and $U_S \subseteq P$, let $W_i(l, U_S)$ denote the amortized cost of $S$ at iteration $i$. That is, $W_i(l, U_S) = \frac{w'_S}{|U_S \cap U|}$, where $U$ is the set of unassigned objects at iteration $i$.

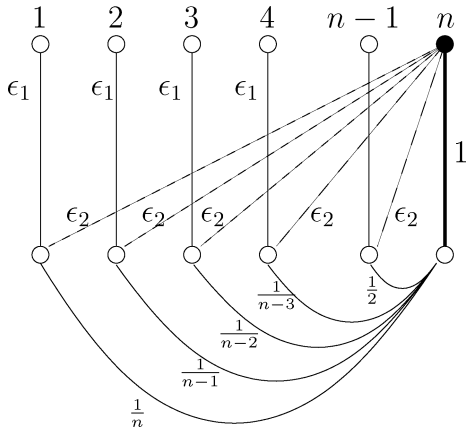


Fig. 4. Instance $\mathcal{I}_n$.

Although the number of stars may be large, we can consider few stars, as shown by the next fact.

FACT 13. *All stars with bounded cost are of the form*

- $(i, \{i\})$ *for* $i = 1, \ldots, n$
- $(n, U_S)$ *for any set* $U_S \subseteq P,\ U_S \neq \emptyset$

Consider the first iteration. In this case we will prove that star $(1, \{1\})$ has the smallest amortized cost. The following inequalities are immediate:

$$W_1(1, \{1\}) < W_1(2, \{2\}) < \ldots < W_1(n, \{n\}) \tag{11}$$

$$W_1(i, \{i\}) < W_1(n, \{i\}) \text{ for all } i \in \{1, \ldots, n-1\} \tag{12}$$

From inequalities (11) and (12), we can conclude that $W_1(1, \{1\})$ is smaller than or equal to the amortized cost of all stars containing one object.

Because $W_1(1, \{1\}) = 1/n + \epsilon_1$ and $W_1(n, P) = 1/n + \frac{n-1}{n}\epsilon_2$, we have

$$W_1(1, \{1\}) < W_1(n, P) \tag{13}$$

Consider stars with objects $Q$, such that $n \in Q \subseteq P$. Because adding an object $i \in P \setminus Q$ to $Q$ decreases the separation cost and it does not incur a significant increase in the assignment cost (just $\epsilon_2$ is added), we have

$$W_1(n, P) \leq W_1(n, Q) \tag{14}$$

From inequalities (13) and (14) we can conclude that any star that contains the object $n$ has cost greater than $W_1(1, \{1\})$.

Now, consider stars with object set $Q \subseteq P \setminus \{n\}, Q \neq \emptyset$. An minimality argument says that

$$W_1(n, \{i\}) \leq W_1(n, Q),\ \text{for } i = \min\{Q \cap U\} \tag{15}$$

Inequality (15) is valid, because

$$W_1(n, \{i\}) = \min_{i' \in Q \cap U} \left\{ \frac{1}{n - i' + 1} \right\} + \epsilon_2$$

and

$$W_1(n, Q) \geq \text{average}_{i' \in Q \cap U} \left\{ \frac{1}{n - i' + 1} \right\} + \epsilon_2$$

From the previous inequalities, we can conclude that all stars have an amortized cost greater than $W_1(1, \{1\})$. Thus, star $(1, \{1\})$ enters in the solution obtained by algorithm GUL in the first iteration.

Updating $U$, the set of unassigned objects, we note an invariant property. In the second iteration it is clear that

$$W_2(2, \{2\}) < W_2(3, \{3\}) < \cdots < W_2(n, \{n\}) \tag{16}$$

and because $W_2(2, \{2\}) = 1/(n-1) + \epsilon_1$ and $W_2(n, P) = 1/(n-1) + \frac{n-1}{n-1}\epsilon_2$ we have

$$W_2(2, \{2\}) < W_2(n, P) \tag{17}$$

In a similar way, we can conclude that all stars have an amortized cost greater than $W_2(2, \{2\})$ in the second iteration. Thus, the greedy algorithm will select the star $(2, \{2\})$ to enter in the solution.

The induction step is straightforward. Therefore, the solution produced by algorithm GUL consists of $\{\phi(1) = 1, \phi(2) = 2, \ldots, \phi(n) = n\}$. □

## 4. COMPUTATIONAL EXPERIENCE

We performed several tests with the presented algorithm, denoted by GUL, the algorithm presented by Kleinberg and Tardos [1999], denoted by $A_K$, and the algorithm presented by Chekuri et al. [2001], denoted by $A_C$. In the implementation of algorithm GUL, we do not consider intersections between sets in the solution of the subroutine $\mathcal{A}_{\text{SC}}$ in Step 4 of the algorithm. The factor 2 that appears in Lemma 7 was diminished in the experiments when the assigned objects are removed from the current graph in the QC subroutine of GUL.

We observe that the computational resources needed to solve an instance with the presented algorithm is very small compared with the other two algorithms cited above. All the implemented algorithms and the instances are available on the web [Bracht et al.]. The tests were performed in an Athlon XP with 1.2 GHz and 700 MB of RAM and the linear programs were solved by the Xpress-MP solver [2002].

We considered five different sets of instances. We denote these sets by *A, B, C, D*, and *E*.

The instances of set $A$ have the following characteristics: The number of labels is given by $n = \lceil \frac{2k}{3} \rceil$ and the number of objects by $m = \lfloor \frac{k}{3} \rfloor$ for different values of $k = n + m$. The assignment costs are given by $c_{ui} = random(1000)$ for all $u \in P$ and $i \in L$, and separation costs by $w_{uv} = random(10)$ for all $u, v \in P$, where $random(i)$ returns a random value between 0 and $i$.

The largest instance we could solve with algorithm $A_C$ has 46 objects and 24 labels and it took 16,722 s. The time is basically that of solving the linear program. When the primal–dual algorithm is tested with this instance, we obtained in 7 s a solution that is 0.25% worse. Concerning algorithm $A_K$, the largest instance that we could test has 80 objects and 40 labels and it took 15,560 s.

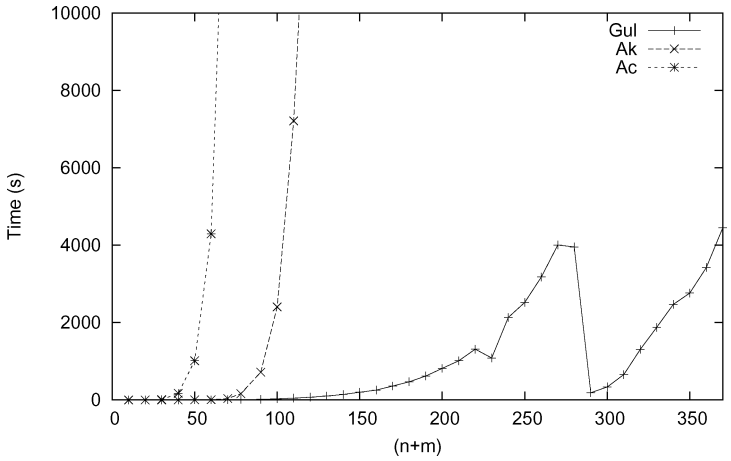


Fig. 5. Running times for instances of set $A$.

This is basically the time to solve the associated linear program. For this instance, algorithm GUL obtained in 73 s a solution that is 1.01% worse. The major limitation to use $A_K$ was the time complexity, while the major limitation to using algorithm $A_C$ was the time and memory complexities. It is important to note that algorithm GUL does not use a linear programming solver.

In Figure 5, we can observe a strong reduction in the function time associated with the GUL algorithm when $n+m$ achieves 290. In this case, we observed that when the separation costs are large, there is a high probability that the size of the star becomes large, thus, the number of objects assigned to a label in each iteration is also large and the running time is reduced. The expectation of the separation costs, in our instances, for stars with a small number of objects, is proportional to $n$, while the expectation of the assignment costs is a constant. Thus, increasing $n+m$ also increases the separation cost. The average number of objects assigned per iteration just before the time reduction was 1.2, that is near to the worst case, while for the instance just after the decrease, the average number of objects assigned per iteration was 32.3.

For all comparable instances of set $A$, algorithm GUL produced solutions that are, at most, 1.27% worse than the value of the linear programming relaxation.

The instances of set $B$ were generated as follows. We maintain the same ratio between the number of classes and objects. The assignment costs are given by $c_{ui} = random(5n)$ and the separation costs by $w_{uv} = random(5)$ for all $u, v \in P$ and $i \in L$, where $n = |P|$.

In these instances, the expected ratio between assignment and separation costs keeps constant while varying $n+m$. Thus, the size of the star that enters the solution is almost the same and the time complexity changes smoothly with the size of the instances.

The largest instance that we could solve with algorithm $A_C$ has 46 objects and 24 labels, and it took 18,720 s. Memory was the main restriction to execute large instances with this algorithm. When we tested algorithm GUL with this instance, we obtained in 7 s a solution that is 1.25% worse. The largest

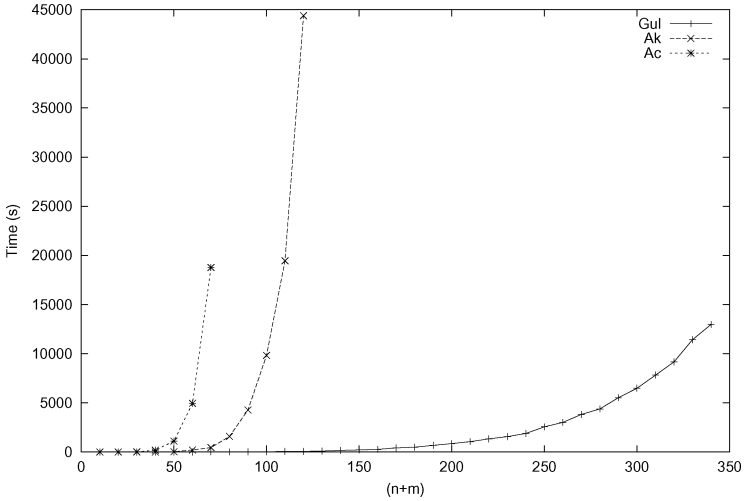


Fig. 6. Running times for instances of set $B$.

instance solved by $A_K$ had 120 objects and 65 labels, and it took 44,400 s. For this instance, algorithm GUL produced a solution in 68 s that is 0.48% worse. The worst ratio of a solution produced by GUL, when compared with the $A_K$ relaxation, was 11.53%. Considering all instances of this set, the number of objects per star was 1.03, on average. The running times for this set of instances can be compared in Figure 6.

To generate the instances of set $C$, we fix the number of labels (objects) to 40 and vary the number of objects (labels) from 5 to 65. The assignment and separation costs were generated in the same way as done for the instances of set $B$. Because algorithm $A_C$ spent more time, we did not use this algorithm for the remaining tests.

In Table I, we show the results for the instances of set $C$. The column $Ratio(\%)$ corresponds to the value $(\text{GUL}(I)/A_K(I)-1)\,100$, for each instance $I$. The largest error ratio was 1.25%. When the number of labels is fixed and the number of objects increases, the gain in time of algorithm GUL, compared to algorithm $A_K$, was considerable.

On average, algorithm GUL associated 1.05 objects in each iteration. When the number of objects, $n$, increases, algorithm $A_K$ takes substantially more time, compared to GUL. This confirms the time complexity of algorithm $A_K$ (for solving a linear program with $O(n^2m)$ constraints by $O(n^2m)$ variables) and algorithm GUL, experimentally estimated in $O(m\,n^{3.6})$.

For all instances in set $D$, the number of objects is $n = 40$ and the number of labels $m = 20$. The assignment costs are given by $c_{ui} = random(1000)$, and the separation costs are given by $w_{uv} = random(10\rho)$, where $\rho$ is an integer varying from 1 to 100. The comparison of algorithms GUL and $A_K$ is presented in Table II.

Because the separation costs increase as $\rho$ increases, larger stars become promising. When large stars enter the solution, the number of iterations decreases. Notice that the hardest instance occurs in the transition between large and small stars. When the separation costs are large ($\rho > 80$), the problem

Table I. Results of $A_K$ and GUL Over the Instances of Set $C$

| Instance | | Time (s) | | |
|---|---|---|---|---|
| Objects ($n$) | Labels ($m$) | GUL | $A_K$ | Ratio (%) |
| 40 | 5 | 0 | 1 | 0.18 |
| 40 | 10 | 2 | 3 | 0.23 |
| 40 | 15 | 3 | 6 | 0.67 |
| 40 | 20 | 4 | 9 | 0.29 |
| 40 | 25 | 5 | 12 | 0.47 |
| 40 | 30 | 5 | 23 | 0.19 |
| 40 | 35 | 6 | 30 | 0.66 |
| 40 | 40 | 7 | 36 | 0.27 |
| 40 | 45 | 8 | 49 | **1.25** |
| 40 | 50 | 9 | 56 | 0.78 |
| 40 | 55 | 10 | 75 | 0.77 |
| 40 | 60 | 11 | 94 | 0.44 |
| 40 | 65 | 11 | 68 | 0.37 |
| 5 | 40 | 0 | 1 | 0 |
| 10 | 40 | 1 | 0 | 0 |
| 15 | 40 | 0 | 1 | 0 |
| 20 | 40 | 1 | 2 | 0.87 |
| 25 | 40 | 2 | 3 | 0.83 |
| 30 | 40 | 3 | 7 | 0.46 |
| 35 | 40 | 4 | 17 | 0.70 |
| 45 | 40 | 11 | 77 | 0.62 |
| 50 | 40 | 16 | 131 | 0.56 |
| 55 | 40 | 22 | 400 | 0.38 |
| 60 | 40 | 26 | 912 | 0.47 |
| 65 | 40 | 33 | 2198 | 0.60 |

Table II. Result of $A_K$ and GUL Over the Instances of Set $D$

| Time (s) | | | | |
|---|---|---|---|---|
| GUL | $A_K$ | Ratio (%) | $\rho$ | Number of Iterations |
| 3 | 5 | 0.34 | 1 | 36 |
| 4 | 7 | 0.39 | 5 | 37 |
| 4 | 7 | 0.57 | 10 | 38 |
| 4 | 11 | 0.75 | 15 | 39 |
| 4 | 11 | 0.43 | 20 | 38 |
| 3 | 25 | 0.66 | 25 | 36 |
| 3 | 21 | 0.47 | 30 | 37 |
| 4 | 23 | 0.51 | 35 | 38 |
| 3 | 35 | 0.51 | 40 | 37 |
| 4 | 41 | 0.88 | 45 | 36 |
| 4 | 67 | 0.68 | 50 | 36 |
| 4 | 114 | 1.06 | 55 | 36 |
| 4 | 168 | 1.75 | 60 | 36 |
| 4 | 203 | 1.26 | 65 | 32 |
| 4 | 439 | 2.26 | 70 | 32 |
| 4 | 831 | 4.32 | 75 | 29 |
| 4 | 1182 | 13.33 | 80 | 23 |
| 2 | 790 | 0 | 85 | 1 |
| 0 | 549 | 0 | 90 | 1 |
| 0 | 262 | 0 | 55 | 1 |
| 0 | 114 | 0 | 100 | 1 |

becomes easy, because it consists in connecting all objects to one label. When $\rho = 80$, the corresponding instance was "difficult" for algorithm GUL, because the approximation factor increases and was also "difficult" for algorithm $A_K$, because its running time also increased. We can conclude, from this experiment, that the ratio between assignment and separation costs is an important characteristic in hard instances.

We observe that less than 1% of the instances of this set led to linear programs with fractional solutions, both for algorithm $A_K$ and $A_C$. The larger ratio between the value of the solution produced by algorithm $A_K$ and its associated linear program was 24.17% using 1020 s. In this case, algorithm GUL produced a solution within a ratio of 7.15% in 4 s. For algorithm $A_K$ the average error for the instances where the linear programming solution was not integral was 7.02% and the average running time was 836 seconds. The average error ratio of algorithm GUL when applied to the same instances was 5.8% in 3.02 seconds, average.

For all instances in the sets $A, \ldots, D$, the associated graph is complete. That is, each object has a separation cost with all other objects and each label has an assignment cost with each object.

Based on the instances of set $D$, we generate a set $E$ of instances associated with sparse graphs, where the degree of each vertex, in $P$, is 2, on average. More precisely, these instances have 20 objects and 10 labels and 40 edges, on average. The assignment costs are given by $c_{ui} = random(20)$ and the separation cost by $w_{uv} = random(8)$. On average, 8% of these instances produced linear programs, which lead to fractional solutions. The average (largest) error ratio of a solution produced by algorithm $A_K$, from the fractional solution, was 6.47% (21.34%). The largest error ratio of a solution produced by algorithm GUL was 50.89% and the average error ratio was 13.46%.

The computational results we have obtained with these instances gave us more knowledge to generate hard instances. This led us to the instance $\mathcal{I}_n$, which proves that GUL analysis is tight up to a constant factor (see Section 3.3).

To illustrate the applicability of the primal-dual algorithm in practical instances, we have applied the algorithm for the image-restoration problem with an image degenerated by noise. The image has pixels in gray scale and dimension $60 \times 60$ with 3600 objects to be classified in two colors, black and white. To define the assignment and separation costs, we consider that each color is an integer between 0 and 255. Each pixel corresponds to an object. The assignment cost of an object $u$ to a label $i$ is given by $c_{ui} = |clr(u) - clr(i)|$, where $clr(u)$ is the actual color of $u$ and $clr(i)$ is the color associated to label $i$. The separation cost of an object $u$ and an object $v$ is given by $w_{uv} = 255 - |clr(u) - clr(v)|$.

The following images (Figures 7–10), present the results obtained applying the primal–dual algorithm. In each figure, the image on the left is obtained inserting some noise and the image on the right is the image obtained after the classification of the objects.

We note that images with more noise have more pixels with different colors (labels). This explains the higher running time for these instances.

Although these times are large for image-processing applications, it illustrates the performance of algorithm GUL and solutions quality. In real

Fig. 7. Image with 25% of noise. Time needed is 288 s.

Fig. 8. Image with 50% of noise. Time needed is 319 s.

Fig. 9. Image with 75% of noise. Time needed is 511 s.

Fig. 10. Image with 100% of noise. Time need is 973 s.

instances, it is possible to define a small window over larger images and the processing time may decrease. Clearly, the classification problem is more general and the primal–dual algorithm is appropriate for moderate- and large-size instances.

## 5. CONCLUDING REMARKS

We presented a primal–dual approximation algorithm with approximation factor $8 \log n$ for the UMLP. We compared the primal–dual algorithm with linear programming-based approximation algorithms. We proved that our analysis is tight, up to a constant factor. The previous approximation algorithms for this problem have high-time complexity and are adequate only for small- and moderate-size instances. In all experimental tests, the presented algorithm obtained solutions within 60% of the optimum and could obtain solutions for moderate- and large-size instances.

ACKNOWLEDGMENTS

We would like to thank K. Freivalds for making available his code for the QC Problem and Cristina G. Fernandes for helpful suggestions to this article.

REFERENCES

BESAG, J. 1974. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society B36*, 2, 192–236.

BESAG, J. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B48*, 3, 259–302.

BRACHT, EVANDRO C., MEIRA LUIS, A. A., AND MIYAZAWA, F. K. 2002. Instances and programs to *Uniform Metric Labeling Problem*. `http://www.ic.unicamp.br/~aprox/labeling`.

CHAKRABARTI, S., DOM, B., AND INDYK, P. 1998. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 307–318.

CHEKURI, C., KHANNA, S., NAOR, J. S., AND ZOSIN, L. 2001. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*. 109–118.

CHVÁTAL, V. 1979. A greedy heuristic for the set-covering problem. *Math. of Oper. Res. 4*, 3, 233–235.

COHEN, F. S. 1986. Markov random fields for image modelling and analysis. *Modelling and Applications of Stochastic Processes*. 243–272.

DAHLHAUS, E., JOHNSON, D. S., PAPADIMITRIOU, C. H., SEYMOUR, P. D., AND YANNAKAKIS, M. 1994. The complexity of multiterminal cuts. *SIAM Journal on Computing 23*, 4, 864–894.

DASH OPTIMIZATION. 2002. *Xpress-MP Manual. Release 13*.

DUBES, R. C. AND JAIN, A. K. 1989. Random field models in image analysis. *Journal of Applied Statistics 16*.

FREIVALDS, K. 2003. A nondifferentiable optimization approach to ratio-cut partitioning. In *Proceedings of the 2nd Workshop on Efficient and Experimental Algorithms*. Lectures Notes on Computer Science, vol. 2647. Springer-Verlag, New York.

GUPTA, A. AND TARDOS, E. 2000. A constant factor approximation algorithm for a class of classification problems. In *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing*. 652–658.

JAIN, K., MAHDIAN, M., MARKAKIS, E., SABERI, A., AND VAZIRANI, V. 2003. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM (JACM) 50*, 6 (Nov.), 795–824.

KLEINBERG, J. AND TARDOS, E. 1999. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *Proceedings of the 40th Annuall IEEE Symposium on Foundations of Computer Science*. 14–23.

Vazirani, V. 2001. *Approximation Algorithms*. Springer-Verlag, New York.

Received February 2005; revised December 2005; accepted February 2006