A Constant Factor Approximation Algorithm for a Class of Classification Problems

Anupam Gupta* Éva Tardos†

Abstract

In a traditional classification problem, we wish to assign labels (or classes) from a set L of labels to each of n objects, in a way that is consistent with some observed data that we have about the problem that includes pairwise relationships among the objects to be classified. Kleinberg and Tardos recently formulated a general classification problem of this type which they called the "metric labeling problem" and gave an $O(\log |L| \log \log |L|)$-approximation algorithm for the metric labeling problem. Their approximation algorithm is based on solving a linear programming relaxation of a natural integer program and then randomized rounding.

In this paper we consider an important case of the metric labeling, the truncated linear metric. This is the simplest non-uniform and robust metric, and it naturally arises in a number of applications. We give a 4-approximation algorithm for this metric. Our algorithm is a natural local search method, where the local steps are based on minimum cut computations in an appropriately constructed graph. Our method extends previous work by Boykov, Veksler and Zabih on a more restricted classes of metrics.

*Computer Science Division, UC Berkeley, Berkeley, CA 94720. Email: `angup@cs.berkeley.edu`. Supported by NSF grant CCR-9820951

†Department of Computer Science, Cornell University, Ithaca NY 14853. Email: `eva@cs.cornell.edu`. Research done in part while a visiting Miller Professor at the Computer Science Division, UC Berkeley, Berkeley, CA 94720. Partially supported by a Guggenheim Fellowship, NSF grant CCR-9700163 and ONR grant N00014-98-1-0589.

1 Introduction

In a traditional classification problem, we wish to assign one of l labels (or classes) to each of n objects, in a way that is consistent with some observed data that we have about the problem that includes pairwise relationships among the objects to be classified [**?**, **?**]. Kleinberg and Tardos [**?**] recently formulated a general classification problem which they called the "metric labeling problem". They also gave an $O(\log l \log\log l)$-approximation algorithm for the metric labeling problem, where l is the number of labels. Their approximation algorithm is based on solving a linear programming relaxation of a natural integer program and then rounding.

Classification and the Metric Labeling Problem. The traditional classification problem can give rise to the following labeling problem. Consider a set P of n objects that we wish to classify, and a set L of l possible labels. A *labeling* of P over L is simply a function $f : P \to L$; we choose a label for each object. The quality of our labeling is based on the contribution of two sets of terms.

- For each object $p \in P$ and label $i \in L$, we have a non-negative *assignment cost* $c(p, i)$ associated with assigning the label i to the object p.
- We have a graph G over the vertex set P, with edge set E indicating the objects that are related; each edge $e = \{p, q\}$ will have a non-negative weight w_e, indicating the strength of the relation. In addition, we have a distance function $d(\cdot, \cdot)$ on the set L of labels. If we assign label i to object p and label j to object q, and $e = \{p, q\}$ is an edge of G, then we pay a *separation cost* $w_e\, d(i, j)$.

Thus, the *total cost* of a discrete labeling f is given by

$$Q(f) = \sum_{p \in P} c(p, f(p)) + \sum_{e=\{p,q\} \in E} w_e d(f(p), f(q)).$$

The *labeling problem* asks for a discrete labeling of minimum total cost. Recall that a *distance* $d : L \times L \to \mathbb{R}^+$ is a function that is symmetric and $d(i, i) = 0$ for all $i \in L$. If d also satisfies the triangle inequality, then d is a *metric* [**?**]. The labeling problem is called *metric* if the distance function $d(\cdot, \cdot)$ is a metric on the label set L.

The labeling problem is closely related to the theory of *Markov Random Fields* (MRFs). Suppose that we have observed data $f'(p)$ for each object $p \in P$, and we wish to decide on the most probable "true" labeling f of P given this data. We assume that f' was obtained from a "true" labeling f by the introduction of independent random noise at each object. Using the theory of Markov Random Fields we need to assume that in the original labeling f the probability an object has a certain label depends only on the labels of the neighboring nodes. Now the labeling problem above can be restated as the problem of finding the f that maximizes the *a posteriori* probability $\Pr[f|f']$, if the underlying MRF model satisfies two standard assumptions of Homogeneity and Pairwise Interactions. (See [**?**] for more details on the connection to MRFs.) Next we give two examples from vision when this framework is used.

Labeling with the Truncated Linear Metric. In this paper we give a 4-approximation algorithm for the important case of the metric labeling problem where the labels are the positive integers, and the metric distance between labels i and j is given by the truncated linear norm, $d(i, j) = \min\{M, |i - j|\}$. Here we discuss why this is a natural metric to be singled out for study.

To motivate the formulation and the metric we use, we consider two well-studied problems from vision: the problem of "restoring" an image that has been degraded by noise [**?**, **?**], and the visual correspondence problem [**?**] that is the basis of determining depth and motion in an image.

In both examples we are given a large grid of pixels as our graph. In the *image restoration problem* we want to label each pixel with its "true" intensity. We are given an "observed" intensity $f'(p)$ for each pixel, that is the result of corruption by noise. We would like to find the best way to label each pixel with a (true) intensity value, based on the observed intensities and the special structure of the image. Our determination of the "best" labeling is a labeling problem based on the trade-off between two competing influences: We would like to give each pixel an intensity close to what we have observed; furthermore, since real images are mainly smooth, with occasional boundary regions of sharp discontinuity, we would like spatially neighboring pixels to receive similar intensity values. The labels for this problem are the possible intensities, and we assign a penalty of $w_e\, d(i, j)$ if neighboring pixels are assigned labels i and j respectively.

Now consider the desired properties of a distance function $d(\cdot, \cdot)$. Small differences in intensity can reflect, for example, gradual changes in lighting, and hence should receive a small penalty. Large changes in intensity are likely to reflect object boundaries, and hence should receive a large penalty. In addition, we want the assigned penalty to be *robust*: once the difference of the assigned labels is large enough to suggest an object boundary, the penalty should be constant. If the penalty function is not robust, there is a possibility of oversmoothing at the boundaries: Consider a small fuzzy region at the between two objects which have been assigned far-apart labels. If the metric is robust, the fuzzy region is likely to receive the label of one of the objects, otherwise it might receive labels inbetween the labels of the two objects, and hence using a such metric will not result in sharp object boundaries. In summary, we want a metric on the labels with the following two properties.

- The metric should not be uniform, if two labels $i, j \in L$ are similar, than $d(i, j)$ should be small.
- The metric should be robust, once two labels $i, j \in L$ are sufficiently different to suggest object boundaries, the distance $d(i, j)$ the distance $d(i, j)$ should be constant.

Now consider the image restoration problem for a grey-scale image. In this case the possible intensities are integers, and the simplest, and most natural metric function that satisfies our properties is the truncated linear metric of $d(i, j) = \min\{M, |i - j|\}$ that we consider in this paper.

An example where non-uniform, rubust metrics are even more important arises in the visual correspondence problem, which is the basis of determining motion and depth from the camera in an image. In the *visual correspondence problem*, we are given two images of the same scene, a pixel in one image corresponds to a pixel in the other if both pixels are projections along lines of sight of the same physical scene element. The problem is to determine this correspondence between pixels of two images. To apply the labeling problem to solve this problem we arbitrarily select one of the images to be the primary image. The quantity to be estimated for each pixel is the difference of its place on the image, and the place of the corresponding pixel on the other image (called the *disparity*). This disparity is in one-to-one correspondence with the depth of the corresponding point in the scene from the camera. Small differences in depth can reflect sloping objects (i.e., an object that starts close to the camera, but extends to some distance), while large differences in depth are likely to reflect object boundaries. The possible depths values are integer multiples of some small shift, and the truncated linear metric is a extremely natural robust metric to use in this case.

Previous Results. Traditional local search methods (such as simulated annealing) with simple local steps (such as relabeling one object) have been extensively used previously to solve such classification problems.

However, the algorithms performed poorly in practice, and no bounds were known for the performance.

Some special cases of the labeling problem are known to be solvable in polynomial time using max-flow computations. Besag [**?**] and Greig et al. [**?**] showed that the case of $l = 2$ labels is polynomially solvable as a two-terminal minimum cut problem. Another special case, when the metric labeling problem reduces to two-terminal minimum cuts, has been discovered independently by Boykov et al. [**?**] and Ishikawa and Geiger [**?**]. They consider the special case of metric labeling when the label set is $L = \{0, 1, \dots, l\}$, and the distance between two labels i and i is $d(i, j) = |i - j|$. Using this linear metric however does not lead to a robust labeling and hence can lead to oversmoothing. The work of Karzanov [**?**] studies some other polynomially solvable special cases of the labeling problem.

Kleinberg and Tardos [**?**] provided the first polynomial-time approximation algorithm for the metric labeling problem. They give an $O(\log |L| \log \log |L|)$-approximation algorithm for this problem, with respect to an arbitrary graph on the set of objects, and an arbitrary metric on the set of labels. For the special case of the *uniform metric*, in which all distances are equal to 1, they give a 2-approximation algorithm. For the general case, they use a result of Bartal [**?**, **?**] which says that every metric can be approximated by hierarchically well-separated tree metric. The best approximation of the truncated linear metric with a random hierarchically well-separated tree metric had distortion $O(\log M \log \log M)$, and hence for the case of the truncated linear metric the method gives an $O(\log M \log \log M)$-approximation. Furthermore, this approximation algorithm is based on solving a linear program (that corresponds to a fractional classification problem), and rounding its solution. The linear program involved is quite large, and this causes the method to be too slow to be practical for tmagem. re, the

The metric labeling problem is an extension of the well-studied *multiway cut problem* [**?**, **?**, **?**, **?**, **?**], in which one must find a partition of a graph into l sets in a way that is consistent with an initial assignment of certain vertices to these sets, and which cuts as few edges as possible. Boykov et al. developed the connection between uniform labelings, with $l > 2$ labels, and multiway cuts in a graph, showing a direct reduction from labelings to multiway cuts [**?**]. However, their reduction required the use of edges of enormously large weight, and so it is not approximation-preserving. In fact, it is not difficult to see that once we allow highly variable costs $c(\cdot, \cdot)$ (or infinite costs), direct extensions of either the *Isolation Heuristic* of Dahlhaus et al. [**?**], or the algorithm of Calinescu, Karloff, and Rabani [**?**] no longer provide reasonable approximation guarantees. This is because these methods do not deal well with objects that *cannot* receive certain labels, and they can produce results that are arbitrarily far from optimal. Boykov et al. [**?**, **?**, **?**] have developed flow-based local search heuristics for labeling problems using the uniform metric. For one of their heuristics [**?**], they prove that any local optimum is a 2-approximation. In fact, it is possible to adapt their proof to show that the local improvement heuristic provides a $2 + \epsilon$-approximation in polynomial time. In [**?**], they show that the heuristics also perform very well in practice. They also consider other metrics, and in [**?**], they show that a similar local improvement step can be described for any metric, but in general the locally optimal solution can be arbitrarily far from optimum. In particular, for the truncated linear metric, their methods provide a $2M$-approximation algorithm [**?**], where M is the upper-bound used in defining the metric.

Our Results. In this paper we consider the metric labeling problem the *truncated linear metric*, that is the simplest and most natural, non-uniform and robust metric. We assume that the metric distance between labels i and j is given by the truncated linear norm $d(i, j) = \min\{M, |i - j|\}$.

Our first observation is that the truncated linear metric can be approximated by a random tree metric with expected stretch at most 3 (See Figure 2.1). The trees produced by this approximation are however not hierarchically well-separated, and hence cannot be used directly in the algorithm of [**?**]. However,the ideas of linear programming and rounding of that algorithm can be adapted to give a constant factor approxima-

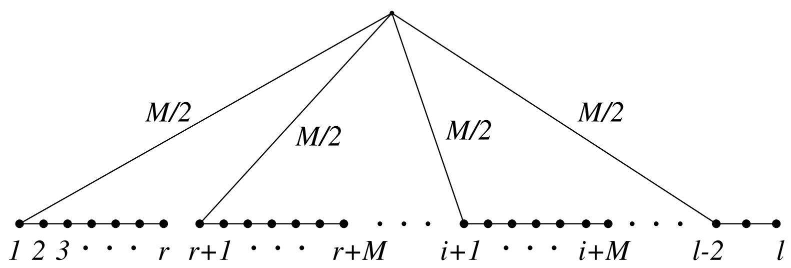

Figure 2.1: The embedding for the truncated linear metric

tion algorithm. Unfortunately, this constant is large, and furthermore, solving the large LP still remains practically infeasible.

Our main result is a flow based local improvement algorithm that gives a factor 4-approximation to the optimum. The key insight this algorithm comes from the above tree embedding. Any tree in the distribution consists of roughly l/M edges leaving the root, each of these has length $M/2$, and there is a linear segment of length at most M edges of length 1 each attached to each of them. (See figure 2.1.) Thus the tree metrics in the embedding can be thought of as a combination of the uniform metric and a set of (untruncated) linear metrics. Now recall that the metric labeling problem on the (untruncated) linear metric can be solved optimally using flows and cuts, and further that the uniform metric can be approximated well using a flow based local search. Our main result is a local search method for the truncated linear metric that is obtained by a clever combination the two techniques. We prove that our methods result in a 4-approximation algorithm for this problem.

2 Approximation with tree-metrics and random partitioning

Consider a path of length l, where the vertices are the labels $L = \{1, 2, \dots, l\}$ and edges $\{i, i+1\}$ are of unit length. To this, we add edges of length M between all pairs of labels $\{i, j\}$ with $|i - j| > M$. The geodesic metric generated by this graph is exactly the truncated linear metric $\mathcal{L} = (L, d)$.

To embed this into a random tree, consider picking a random offset $r \in \{1, 2, \dots, M\}$, and for each position $1 \leq k \leq l$ which is equal to $r \pmod m$, deleting all edges that connect a pair of labels $i \leq k$ and $j > k$. This results in a set of *intervals*, each of length at most M. Let us call this set of intervals $\mathcal{S}_r$, the *partition* associated with r. Now we add the lowest numbered vertex of each interval in $\mathcal{S}_r$ to a new *root* vertex with an edge of length $M/2$ to get a tree. (See figure 2.1 for an example.)

Theorem 2.1 *For any pair of labels $i, j \in L$, the distance between them in any of the (random) trees is at least $d(i, j)$, and the expected distance between them is at most $3d(i, j)$.*

Let us also record an easy fact about the chance that two vertices lie in different intervals of T.

Fact 2.2 *For any pair of labels $i, j \in L$, the probability that i and j lie in different intervals of the a random tree T is exactly $d(i, j)/M$.*

We shall not use this tree embedding directly. However, it will influence the structure of both the algorithm and the analysis in the following sections.

3 The local search algorithm

The local search algorithm starts with an arbitrary labeling f of the vertices of the graph G. It then makes a simple "local" move whose goal is to decrease the cost of the labeling. If making the move causes it to reach a new labeling whose cost is lower than the cost of the current labeling, it moves to the new labeling and continues.

Boykov et al. [**?**] use the following local improvement step. Consider a label i. In a single "local" improvement step Boykov et al. allow relabeling any subset of the vertices with the label i. They call this local move an *i-expansion move*. They prove that if the distance function on the labels is actually a metric, then the best i-expansion move can be found via a single min-cut computation in an appropriately constructed graph [**?**, **?**]. Further, they prove that in the case of the uniform metric, any local optimum is a 2-approximation to the global optimum[**?**]. For metrics other than the uniform metric, however, a locally optimal solution can be arbitrarily far from optimum. In particular, for the truncated linear metric, their methods provide a $2M$-approximation algorithm [**?**], where M is the upper-bound used in defining the truncated metric.

In this section we develop an analogous, but much more general local improvement method. In a single local step we consider an interval I of labels of length at most M, and allow any subset of vertices to be relabeled to any of the labels in this interval. Given a labeling f we call another labeling f' a *local relabeling* if it can be obtained from f by such a local move, i.e., if for all objects $f(p) \neq f'(p)$ implies that $f'(p) \in I$. Unfortunately, we will not be able to find the best possible such local move. However, we will show that if the current labeling has cost sufficiently far above the minimum possible cost, then our method will find a move that significantly decreases the cost of the labeling.

In the algorithm, we repeatedly pick a random interval I (of length at most M) and try to relabel some subset of objects to the labels in I to decrease the cost of our labeling. After this local step, each object will either have its label unchanged, or will have a label in the interval I. To perform this relabeling, we will create a flow network and find a minimum $s-t$ cut in this network. This minimum cut can be associated with a new labeling f', and if this new labeling f' has a lower cost than the cost of the original labeling f, we move to the new labeling. In summary, the algorithm is the following:

Algorithm Local-Search:

> **repeat**
>   pick a random interval I
>   build the flow network N_I associated with I
>   **if** labeling given by the minimum cut on N_I has lower cost
>   **then** move to new labeling.
> **until** some stopping rule.

More formally, we will pick the random intervals I in the following manner: we pick a random integer $-M < r < l$, and set I to be the part of the interval of length M starting from offset r that lies in L. I.e., $I = \{r+1, r+2, \ldots, r+M\} \cap \{1, 2, \ldots, l\}$. Note that the probability of any fixed interval J being picked by this method is almost the same as the chance of being picked by first choosing a random tree from the tree embedding of the previous section, and then randomly picking one of the intervals in it.

In the next subsection, we will describe how to build the flow network associated an interval I to which the labels can be changed. We then analyze this algorithm in subsection 3.2, showing, for instance, that running the algorithm until we reach a local optimum (i.e., a state where none of the intervals leads to an improvement) implies that we are close to the optimal value.

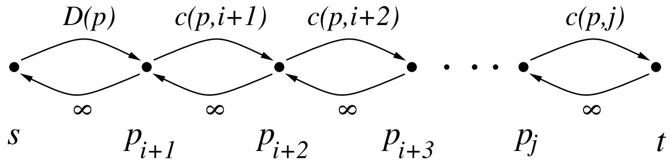

Figure 3.2: The chain for vertex p

3.1 The flow network

Let us consider an interval I of labels. Assume that the labels in the interval I are $\{i+1, i+2, \dots, j\}$, where the number of labels in I, $j-i$ is at most M. (The intervals we consider have length exactly M except when they are some initial or final portion of the line.) In this move, each vertex in G will either retain its label or move to a label in I. We now describe a flow network $N_I = (V, A)$ associated with I: it will be a directed graph with a source s and a sink t, and with capacities on the edges. We describe the construction in two steps.

For each vertex p of the original graph G, we add $j-i$ nodes $\{p_{i+1}, \dots, p_j\}$ to N_I. (Figure 3.2 shows this construction.) We add directed edges (p_k, p_{k+1}) and (p_{k+1}, p_k) for all $k = i+1, \dots j-1$. We then attach p_{i+1} to s and p_j to t by directed edges in both directions. For the following discussion, it will be convenient to think of the vertex s as being p_i, and t as being p_{j+1} also.

Now we shall assign the capacities. For $i+1 \leq k \leq j$, each edge (p_k, p_{k+1}) is assigned a capacity equal to the assignment cost $c(p,k)$, while the edge (p_{k+1}, p_k) is assigned an infinite capacity. Finally, the edge (p_{i+1}, s) is assigned an infinite capacity, while (s, p_{i+1}) is assigned a capacity $D(p)$ which is defined as follows: if $f(p) \in I$, then we set $D(p) = \infty$, else $D(p) = c(p, f(p))$.

To see the rationale for this construction, consider any minimum $s-t$ cut in N_I: the infinite capacity edges ensure that this cut will include exactly one edge from the chain corresponding to each vertex p. We interpret the edge (p_k, p_{k+1}) being cut to mean that the vertex p is assigned the label k (if $k \neq i$); if the edge (s, p_{i+1}) is cut and $f(p) \notin I$, then we retain the original label $f(p)$ for the vertex p. It is clear to see that the assignment cost will exactly the capacity of the edge in the cut, and thus the chain for vertex p captures the assignment costs.

We also need to model the separation costs; for this we will add edges between the chains. Consider an edge $e = \{p, q\}$ in the original graph. There will be different cases depending on the original labels of p and q. One of the cases, when neither p nor q have their original label in I is shown on Figure 3.3. For each of the corresponding nodes p_k and q_k, where $(i+1) < k \leq j$, we add a pair of oppositely directed edges between them, each with capacity w_e. If both p and q are labeled with vertices in I, then we do nothing more. If $f(p) \notin I$ but $f(q) \in I$, then we add an edge (p_{i+1}, q_{i+1}) with capacity $w_e\, d(f(p), i+1)$. Finally, if both the labels do not lie in I, then we add a new vertex v_{pq}. We then add two oppositely directed edges between p_{i+1} and v_{pq} having capacities $w_e\, d(f(p), i+1)$, and opposite edges between v_{pq} and q_{i+1} having capacities $w_e\, d(f(q), i+1)$. To complete the construction, we attach an edge (s, v_{pq}) with capacity $w_e\, d(f(p), f(q))$. (A view of this third case is given in Figure 3.3, where we have dropped some of the infinite capacity edges, and used undirected edges to indicate a pair of oppositely directed edges with identical costs.)

To see that this captures the separation costs, let us focus on the edge $e = \{p, q\}$. If both the vertices retain their original labels, then the cut will minimized when it passes through (s, v_{pq}), and incurs a cost of $w_e\, d(f(p), f(q))$, which is exactly the separation cost in this case. In case both the vertices are labeled with labels in I, then the cut value will be exactly $w_e|f(p) - f(q)|$, which is again the right separation cost.

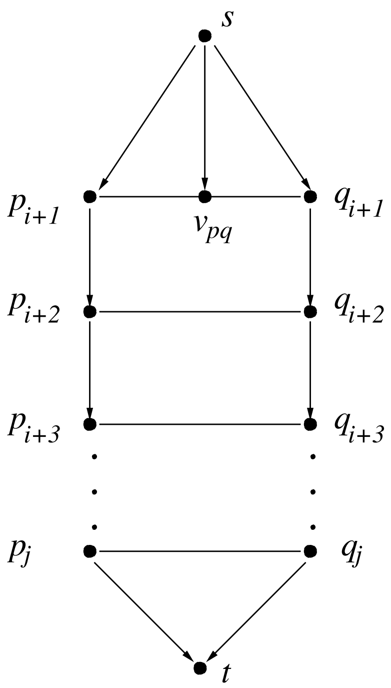

Figure 3.3: Construction for the edge $\{p, q\}$

However, if the cut indicates that one of the vertices (say p) retains its label $f(p) \notin I$, and q is labeled with some element $k \in I$, then the cut will incur a separation cost of $w_e[d(f(p), i+1) + (k-(i+1))]$. Note that this value possibly overestimates the actual separation cost in the new labeling.

Before we formalize the relation between cuts and labeling in N_I we will consider two simpler special cases: (1) when the interval I consists of all labels, and (2) when the interval has a single label $i+1$. In case (1) the construction is the same as the one discovered independently by Boykov et al. [**?**] and Ishikawa and Geiger [**?**] for finding the optimal labeling for the line metric.

Theorem 3.1 (Boykov et al. [?], Ishikawa and Geiger [?]) *The minimum cut of the flow network associated with an interval I consisting of all the labels gives the optimal labeling for the line metric on I.*

The other extreme case is when the interval consists of a single label $I = \{i+1\}$. In this case, the construction above is the same as the one by Boykov et al. [**?**] for finding the best $(i+1)$-expansion move, i.e., relabeling where labels can only change to the label $(i+1)$.

Theorem 3.2 (Boykov et al. [?, ?]) *The minimum cut of the flow network N_I associated with an interval $I = \{i+1\}$ consisting of a single label is the optimal $(i+1)$-expansion move.*

We now summarize the discussion of the properties of the labeling obtained by the minimum cut in the flow network for the general case, when interval I has more than a single node, but does not contain all labels in L. This theorem will be used to show that a sequence of such moves leads to a 4-approximation.

First we note that a minimum cut will cut at most one of the edges between the pairs of nodes p_{i+1} and v_{pq}, or q_{i+1} and q_{i+1}, and the edge (s, v_{pq}) for any edge $e = \{p, q\}$ in the graph G. This follows from the fact that $d(\cdot, \cdot)$ satisfies the triangle inequality, and hence the cost of any two of these edges is more than the cost of the third one. We shall call a cut *simple* if it has finite capacity, and it does not cut more than one of the above five edges associated with any edge $e \in E$.

Theorem 3.3 *The simple cuts in the flow network N_I are in one-to-one correspondence with local relabelings f'. The cost of the relabeling $Q(f')$ is no more than the cost of the associated cut, and the cost of the cut overestimates the cost of the labeling by replacing the separation cost $w_e d(f'(p), f'(q))$ for edges $e = \{p, q\}$ where exactly one end receives a label in I by a possibly larger term $w_e[d(f'(p), i+1) + d(i+1, f'(q))]$. However, we have that*

$$d(f'(p), f'(q)) \ \leq \ d(f'(p), i+1) + d(i+1, f'(q)) \ \leq \ 2M. \tag{3.1}$$

3.2 The analysis

In this subsection, we show that when the algorithm reaches a local minimum, then the cost of the algorithm's solution is no more than 4 times the optimal cost. Further, we also show that the algorithm converges to a such solution in polynomial time.

Let us fix an optimal labeling f^*, and let the algorithm's current labeling be f. For any subset $X \subseteq P$, let $A^*(X)$ and $A(X)$ be the assignment cost that the optimum and the current labeling pay respectively for the vertices in X, and for a set of edges $Y \subseteq E$, let $S^*(Y)$ and $S(Y)$ be the separation cost for those edges paid by the optimum and the current solution respectively. Clearly, $Q(f) = A(P) + S(E)$ and the optimum $Q(f^*) = A^*(P) + S^*(E)$.

Consider the case when the algorithm chooses an interval I. Let P_I be the set of vertices of G to which f^* assigns labels from the interval I. Let E_I be the set of edges in $E(G)$ such that the f^*-labels of both endpoints lie in I. Let $\partial_I^- \subset E_I$ be the set of edges such that only the end of the edge with higher f^*-label has an f^*-label in I, and ∂_I^+ be the set of edges such that only the end with lower f^*-label has an f^*-label in I.

In the proof we will consider a random partition $\mathcal{S}_r$ of the labels, as defined in Section 2. Clearly, $P = \cup_{I \in \mathcal{S}_r} P_I$ and $\cup_{I \in \mathcal{S}_r} \partial_I^- = \cup_{I \in \mathcal{S}_r} \partial_I^+$. We will use ∂_r to denote this union of the *boundary edges* in the partition. Also note that $E = \partial_r \cup (\cup_{I \in \mathcal{S}_r} E_I)$.

The edges in ∂_r will play a special role in the analysis. The following lemma will be useful in the analysis later.

Lemma 3.4 *For a random partition $\mathcal{S}_r$, the expected value of $M \sum_{e \in \partial_r} w_e$ is $S^*(E)$.*

Proof: The probability that edge $e = \{p, q\}$ will be part of ∂_r is exactly $d(f^*(p), f^*(q))/M$, hence the edge e contributes $w_e d(f^*(p), f^*(q))$ to the expectation of the sum. This is exactly the same as its contribution to $S^*(E)$. ■

Consider the situation when the algorithm picks the segment I. One possible local move is changing the labels of vertices in the region P_I to their labels in the optimum labeling f^*. We will use this move to bound the improvement obtained in our algorithm.

Lemma 3.5 *For a labeling f, and an interval I, the local relabeling move that corresponds to the minimum* *cut in N_I decreases the cost of the solution by at least*

$$\left(A(P_I) + S(E_I \cup \partial_I^- \cup \partial_I^+)\right) - \left(A^*(P_I) + S^*(E_I \cup \partial_I^-) + M \sum_{e \in \partial_I^-} w_e + 2M \sum_{e \in \partial_I^+} w_e\right).$$

Proof: The algorithm selects the minimum cut in N_I. The labeling corresponding to this cut has cost at most the capacity of the cut in N_I. To prove the bound we have to show a cut with small capacity. Consider the cut that corresponds to the local move of relabeling all nodes in P_I to their f^*-label. The capacity corresponding to the assignment cost for nodes in P_I will be exactly $A^*(P_I)$, and the capacity corresponding to separation cost for the edges in E_I will be $S^*(E_I)$. For nodes and edges that do not change their label in this move, the capacity corresponding to their assignment and separation cost will be unchanged.

Consider the edges in $\partial_I^- \cup \partial_I^+$. These are the edges that possibly change the label of exactly one of their ends. We estimate the capacity corresponding to the separation cost of these edges by considering the edges in ∂_I^- and ∂_I^+ separately. For edges $e \in \partial_I^+$, we use (3.1) to get a bound of $2Mw_e$. Now consider an edge $e = \{p, q\} \in \partial_I^-$, and assume that $p \in P_I$. The bound on the capacity corresponding to the separation of this edge is $w_e[d(f^*(p), i+1) + d(i+1, f(q))]$ from (3.1). Note that because e is in ∂_I^- we have $f^*(q) \leq i$, and hence $d(f^*(p), i+1) \leq d(f^*(p), f^*(q))$. Bounding the other term $d(i+1, f(q))$ by M, we get that the capacity corresponding to the separation of edge e is at most $w_e[d(f^*(p), f^*(q)) + M]$. Summing this over all edges in $\partial_I^- \cup \partial_I^+$, we get exactly the claimed bound. ∎

Let us first assume that we have reached a local optimum, i.e., no local move with any interval decreases the cost of the labeling. This implies that for all I, the quantity in the above lemma is non-positive. This implies the following theorem.

Theorem 3.6 *If the labeling f is a local optimum, its cost $Q(f)$ is at most 4 times the optimal cost $Q(f^*)$.*

Proof: The fact that f is a local optimum implies the improvement guaranteed by Lemma 3.5 is non-positive for any intervals I.

$$A(P_I) + S(E_I \cup \partial_I^- \cup \partial_I^+) \leq A^*(P_I) + S^*(E_I \cup \partial_I^-) + M \sum_{e \in \partial_I^-} w_e + 2M \sum_{e \in \partial_I^+} w_e.$$

Now consider a partition $\mathcal{S}_r$ and sum these inequalities for each interval $I \in \mathcal{S}_r$. On the left hand side we get $A(P) + S(E) + S(\partial_r)$ as edges in ∂_r occur in the boundary of two intervals. This is at least $Q(f)$, the cost of the labeling f. Summing the right hand side, we get exactly $A^*(P) + S^*(E) + 3M \sum_{e \in \partial_r} w_e$. So we have that

$$Q(f) \leq A^*(P) + S^*(E) + 3M \sum_{e \in \partial_r} w_e$$

for any partition $\mathcal{S}_r$. Now we can take expectations. The left side is a constant, and by Lemma 3.4, the expected value of the right hand side is at most $A^*(P) + 4S^*(E)$. Thus we get $Q(f) \leq A^*(P) + 4S^*(E) \leq 4Q(f^*)$, as claimed. ∎

In fact, we can also bound the number of steps we have to run the algorithm to reach a near-optimal solution. Let Q_0 be the cost of the initial labeling.

Theorem 3.7 *If the main loop of the algorithm is repeated $O((l/M)(\log Q_0 + \log \epsilon^{-1})$ times, then the expected cost of the resulting labeling is at most $(4 + \epsilon)\, Q(f^*)$.*

Proof: To estimate the expected decrease of the labeling cost in one iteration, consider the following alternate process for selecting a random interval. First select a partition $\mathcal{S}_r$ at random, and then select an interval $I \in \mathcal{S}_r$. This process selects each interval with roughly the same probability as the one our algorithm used. The difference is that some partitions have one more interval than others, and intervals in such partitions have a slightly smaller chance of being selected.

Consider a labeling f and a partition $\mathcal{S}_r$. We use Lemma 3.5 to estimate the improvement obtained if interval I is used. As in the previous proof, we will sum the estimates over the set of intervals in the partition. There are at most $\lceil l/M \rceil + 1$ intervals in $\mathcal{S}_r$, hence the expected decrease in the labeling cost when a random interval is selected from $\mathcal{S}_r$ is at least

$$\frac{1}{\lceil l/M \rceil + 1} \left(Q(f) - Q(f^*) - 3M \sum_{e \in \partial_r} w_e \right).$$

But by Lemma 3.4 for a randomly chosen partition $\mathcal{S}_r$ the expected value of this decrease is at least

$$\Omega \left(\frac{M}{l} (Q(f) - 4Q(f^*)) \right),$$

so in $O(l/M)$ iterations the difference $Q(f) - 4Q(f^*)$ is expected to decrease by a constant factor. This implies the claimed bound on the running time. ■

It is easy to see that the algorithm can be trivially derandomized at a cost of a factor $(l + M)$ increase in the running time. This can be done by considering all possible $(l + M)$ intervals and making the moves corresponding to the best possible interval, or making local moves for each of the $(l + M)$ intervals in turn in a round robin fashion.