CS 684: Approximation and Network Algorithms	**11/02/01**
Image Recovery	**Ranjith Rajagopalan**

1 Problem Statement

The problem of image recovery stems from the fact that observed data often differs from the truth. We would like to change the observed data in some way that we believe will more closely approximate reality. Specifically, we have an image that consists of an array of pixels, each of which has a certain color value. In this case, we're going to assume that in reality, nearby pixels shouldn't have a large difference in value. However, we also assume that the image isn't too far from the truth, i.e. the actual value of any pixel is close to the observed value.

More formally, we have a graph $G = (V, E)$ and a label set $L = \{1 \cdots k\}$. We want to find a labeling of G, $f : V \to L$. For each node v, label a, we have a cost $c_v(a) \geq 0$ which is the cost of labeling node v with label a. For each edge (u, v), we have a cost $w_{uv} \geq 0$ which is the cost incurred if $f(u) \neq f(v)$. The cost of a labeling f is

$$E(f) = \sum_{v \in V} c_v(f(v)) + \sum_{(u,v) \in E} w_{uv}(1 - \delta_{f(u)f(v)}) \tag{1}$$

where $\delta_{ij} = 1$ for $i = j, 0$ otherwise. The goal is to find a labeling of minimum cost.

This is equivalent to the notion of a Markov Random Field for people who know what that means (I am not one of said people).

This model isn't restricted to image restoration. For example, it can also be used for web page classification. Each web page represents a node in the graph, and has edges to web pages that are linked to it. The web page can be labeled by using keywords found in its text, but that may not be an accurate representation of the actual content. However, it's likely that its neighbors have similar content, so finding a min cost labeling as above may give a better picture of the page's true topic.

2 Local Search

Unfortunately, we can't hope for extremely good solutions to this problem as it is NP-complete, since it contains the Multiway Cut problem. This problem involves a graph $G = (V, E)$ with three specified nodes s, t, u, as well as edge costs c_{ij}. The goal is to find a minimum cost set of edges that separate s, t, and u.

The reduction to image recovery is as follows: Use the same graph and edge costs, use the label set $L = \{s, t, u\}$, make label costs $c_v(a) = 0$, $\forall a \in L, v \neq s, t, u$, and make label cost $c_v(a) = (1 - \delta_{av})\,\infty$, $\forall v \in \{s, t, u\}$, that is, force s, t, and u to have labels s, t, and u respectively. Then a solution to the image recovery problem will be a partition of the graph into three sets containing s, t, and u, and the cost of the solution will be exactly the cost of the edges crossing the sets.

So we can't expect to find exact solutions, instead our goal will be to find an approximation algorithm. Specifically, we're going to use local search to find a good labeling. In order to use local search, we need to define a set of moves that can modify one feasible solution into another. The algorithm starts with some feasible solution, and repeatedly finds better solutions by using moves in our restricted set that improve the cost. Once it can't find any such moves, we're guaranteed a local optimum, with respect to our move set.

The idea behind local search is to balance our move set appropriately. If we choose too many possible moves, it will be difficult to guarantee that no move in our set results in an improvement. If we were to choose all possible moves to any feasible solution as our move set, we would reach the optimal solution in one move, but we wouldn't be able to calculate efficiently what that move was. However, if our move set is too weak, then local optima may be much more expensive than the global optimum. We could get stuck at a solution far from the optimum that can't be improved by minor changes.

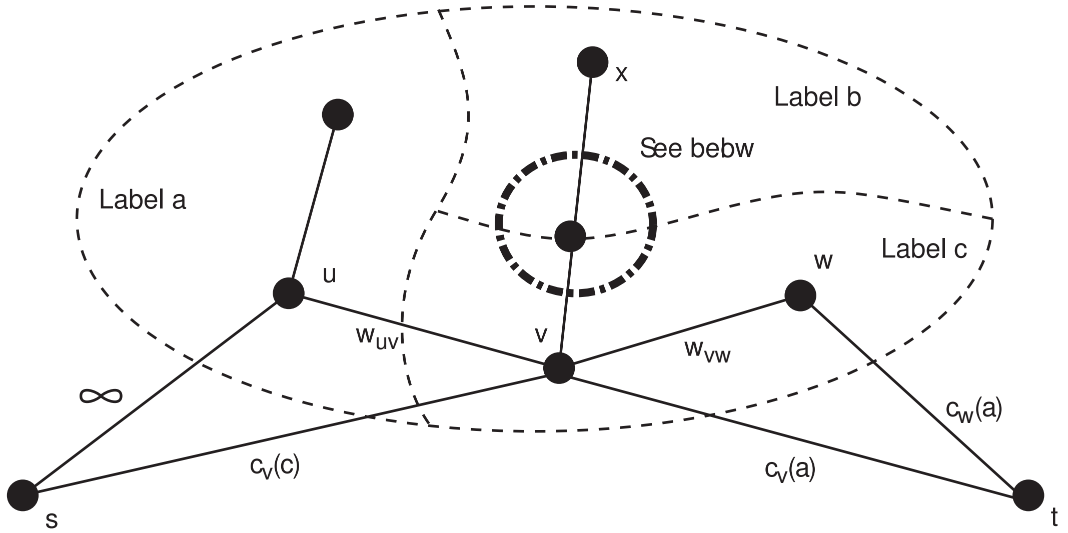
3 Expansion Moves

We need to find a set of moves that gives significant improvement, but whose improvements can be calculated easily. If the problem only consists of two labels, the cost of a labeling seems very close to a min cut, and in fact can be solved with a max flow computation using the right modifications. One natural generalization of this idea is to pick two labels a and b, and find the min cost labeling on the subset of the graph with these two labels, using these two labels. However, we're going to use something slightly different.

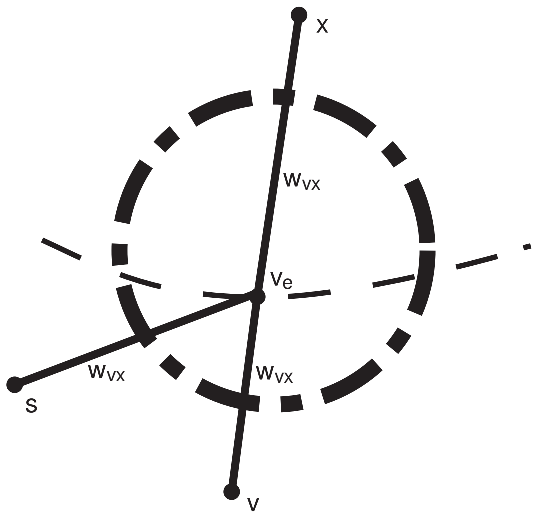
Given a labeling f, an a-expansion move is a labeling f' such that $\forall v \in V, f'(v) \in \{f(v), a\}$. That is, expand the set of nodes that has label a, but don't change a node's label if it is already labeled a. In order to use these moves, we need to show that we can calculate the min cost a-expansion move efficiently.

Lemma 3.0.1 *The optimal a-expansion move can be calculated using a max flow computation.*

Proof: The general idea is as follows. Create an artificial source and sink, s and t. A min cut will separate the graph into nodes changed to label a and nodes that we've left the same. Edge costs between actual nodes will represent edge costs for differing labels, and edge costs between an actual node and an artificial node will represent the cost of changing labels. The only problem will occur when two nodes not labeled a differ. Then there is a cost associated with their difference, but we'll need an extra gadget to represent that in our graph.



Calculating an a-expansion move



Close up of subdivided edge e=(v,x)

Specifically, add two nodes s and t, and add edges to all nodes $v \in V$. If there is an edge $e = (u, v), f(u) \neq f(v)$, and $f(u), f(v) \neq a$, then subdivide that edge and let the intermediate node be v_e, and add an edge from s to v_e. The edge costs will be as follows:

$$cost_{ij} = \begin{cases} w_{ij}, & \text{if } i, j \neq s, t, \\ c_v(a), & \text{if } i = t \text{ and } f(j) \neq a, \\ c_v(b), & \text{if } i = s \text{ and } f(j) = b, \\ \infty, & \text{if } i = s \text{ and } f(j) = a, \\ 0, & \text{if } i = t \text{ and } f(j) = a, \\ w_e, & \text{if } j = v_e, \end{cases}$$

Consider the min $s - t$ cut, let S be the side that contains s. The ∞ edges force all nodes whose label was originally a to be in S. Thus, the set S will correspond to the set of nodes we label a in the a-expansion move. Nodes not in S will retain the same label. Nodes created when edges were subdivided will be ignored for the purpose of determining new node labels. We need to show that the cost of a cut corresponds to the cost of the corresponding labeling.

For any $v \in S$ the edges of cost $c_v(a)$ are in the cut, and pay the labeling cost for these nodes. For any node v such that $v \notin S, f(v) = b$, there is an edge to s of cost $c_v(b)$ which pays the labeling cost of v. So the labeling costs are all accounted for.

Next, we need to show that the edge costs are accounted for.

Case 1: For nodes i, j with $f(i) = a, f(j) = b, j \notin S$ we know that $i \in S$, so (i, j) is in the cut, and its cost is w_{uv}.

Case 2: For nodes i, j with $f(i) = f(j) = b, i \in S, j \notin S$, the edge (i, j) is in the cut and has cost w_{uv}.

Case 3: For nodes i, j with $f(i) \neq a, f(j) \neq a, f(i) \neq f(j), i, j \notin S$, we subdivided the edge $e = (i, j)$, so we need to consider where the node v_e lies. It only has degree three, and the incident edges all have cost w_{uv}. Two of those edges are cut if $v_e \in S$, and only one is cut if $v_e \notin S$, so we must have the latter if this is a min-cut, and the edge cost is paid.

Case 4: For nodes i, j with $f(i) \neq a, f(j) \neq a, f(i) \neq f(j), i, j \in S$, then $v_e \in S$ since that ensures that no edges incident to v_e are cut, which is what we want since i, j are both labeled a.

Case 5: For nodes i, j with $f(i) \neq a, f(j) \neq a, f(i) \neq f(j), i \in S, j \notin S$, assigning $v_e \in S$ will add only one edge of cost w_{uv} to the cut, paying for the edge cost from i to j.

Thus, both the edge costs and labeling costs are paid for, so cuts correspond exactly to a-expansions. Thus, the min cut will give the optimal a-expansion. □

4 Algorithm

We know now that a-expansion moves can be calculated efficiently. Thus, the algorithm is as follows: Start with an arbitrary labeling. Find the a-expansion move that decreases cost as much as possible, and use it. Repeat until no a-expansion move decreases the cost. This will terminate since there are a finite number of labelings, and strict decreases in cost ensure that none are repeated. We want to prove two things about this algorithm. First, we want to show that the solution produced isn't too much worse than optimal. Second, we want to show it doesn't take too long to find it.

First, we show that the local optimum isn't too much worse than the global optimum. Since the solution is a local optimum, no a-expansion move will improve the cost. Thus, in order to relate the cost of the local optimum to the global optimum, we can consider a-expansion moves which use the structure of the global optimum.

Specifically, let f be the current labeling at the local optimum. Let f^* be the labeling of the global optimum. Let f_a be the labeling obtained by an a-expansion move where all nodes i such that $f^*(i) = a$ get changed to label a. Let $E_S(f)$ be the cost of f on the subgraph induced by S. Let $E'_S(f)$ be the cost of f on edges that cross S. Finally, let S_a be $f^{*-1}(a)$, that is, the set of nodes whose label in the optimal labeling is a.

Theorem 4.0.2 $E(f) \leq 2E(f^*)$
Proof: We need some intermediate facts before proving the main result.

First, we know $E(f_a) = E_{S_a}(f_a) + E_{\bar{S}_a}(f_a) + E'_{S_a}(f_a)$. This is because every vertex is either in S_a or $\bar{S}_a$, so the labeling cost for every vertex is either paid in $E_{S_a}(f_a)$ or $E_{\bar{S}_a}(f_a)$. Every edge either lies in S_a, $\bar{S}_a$, or crosses from one to the other. Thus, every edge cost paid in exactly one of the three terms.

Second, we know $E_{\bar{S}_a}(f_a) = E_{\bar{S}_a}(f)$ since $\forall v \in \bar{S}_a, f_a(v) = f(v)$. Similarly, $E_{S_a}(f_a) = E_{S_a}(f^*)$ since $\forall v \in S_a, f_a(v) = f^*(v)$.

Third, we know that $E'_{S_a}(f_a) \leq E'_{S_a}(f^*)$. This is because S_a is exactly the set of vertices with label a in f^*. Thus, every edge crossing S_a incurs its edge cost in f^*, since these edges contain one vertex with label a and one vertex without label a. Any other labeling may incur some subset of these edge costs, so specifically, we have $E'_{S_a}(f_a) \leq E'_{S_a}(f^*)$. Note that this fact requires a 0-1 metric on edge costs. With a more complicated metric, $E_{S_a}(f^*)$ would still incur every edge cost crossing S_a, but those costs might be very low for the labels used in f^* as compared with the edge costs for f.

Fourth, we know that for any labeling f, $E(f) \geq \sum_a E_{S_a}(f)$ since each label cost gets considered once, and each edge cost is considered at most once in the right hand term. Similarly, we know that $E(f) \leq \sum_a (E_{S_a}(f) + E'_{S_a}(f)) \leq 2E(f)$. This is because the sum counts each label cost exactly once, and counts each edge cost either once or twice. An edge gets counted once if it lies within some S_a, and gets counted twice if it crosses from some S_a to some S_b.

Finally, we know that $\forall a \in L, E(f) \leq E(f_a)$ since f is a local optimum and no a-expansion move will improve the cost.

Using these facts,

$$E(f) = E_{S_a}(f) + E_{\bar{S}_a}(f) + E'_{S_a}(f) \leq E_{S_a}(f_a) + E_{\bar{S}_a}(f_a) + E'_{S_a}(f_a) = E(f_a) \tag{2}$$

since $E(f) \leq E(f_a)$. This gives,

$$E_{S_a}(f) + E'_{S_a}(f) \leq E_{S_a}(f_a) + E'_{S_a}(f_a) \leq E_{S_a}(f^*) + E'_{S_a}(f^*) \tag{3}$$

since $E_{S_a}(f_a) = E_{S_a}(f^*)$ and $E'_{S_a}(f_a) \leq E'_{S_a}(f^*)$. Using the fourth fact, this gives

$$E(f) \leq \sum_a (E_{S_a}(f) + E'_{S_a}(f)) \leq \sum_a (E_{S_a}(f^*) + E'_{S_a}(f^*)) \leq 2E(f^*) \tag{4}$$

Thus, we have that $E(f) \leq 2E(f^*)$ □

The last thing we need to do is show that the algorithm only takes a polynomial number of steps. Let $W = \sum_{(u,v) \in E} w_{uv}$ and let $C = \sum_{a \in L, v \in G} c_v(a)$. If all values are integral, we're guaranteed an improvement of 1 at every step, and each step only takes time polynomial in n, so we have that the algorithm takes $\text{poly}(n, W, C)$. We want to improve this to $\text{poly}(n, \log W, \log C)$.

We're only guaranteed to reach a solution of cost $2E(f^*)$, so we'll show that if the current labeling is f, the next step will improve the cost by a constant factor of $E(f) - 2E(f^*)$. This will give a number of iterations that is logarithmic in W and C. Note that while we're guaranteed a local optimum has cost at most $2E(f^*)$, this analysis only guarantees that the algorithm will reach a $2+\epsilon$ factor in a polynomial number of iterations, with a constant that grows with ϵ.

Theorem 4.0.3 *$\exists a$ such that $E(f) - E(f_a) \geq \frac{1}{k}(E(f) - 2E(f^*))$*
Proof: Consider $E(f) - E(f_a)$. Breaking the costs down as before, we have

$$E(f) - E(f_a) = E_{S_a}(f) + E'_{S_a}(f) - (E_{S_a}(f_a) + E'_{S_a}(f_a)) \tag{5}$$

since $E_{\bar{S}_a}(f) = E_{\bar{S}_a}(f_a)$ and those terms cancel. Using the facts that $E'_{S_a}(f^*) \geq E'_{S_a}(f_a)$ and $E_{S_a}(f^*) = E_{S_a}(f_a)$,

$$E(f) - E(f_a) \geq E_{S_a}(f) + E'_{S_a}(f) - (E_{S_a}(f^*) + E'_{S_a}(f^*)) \tag{6}$$

Summing over a, and using $E(f) \leq \sum_a (E_{S_a}(f) + E'_{S_a}(f))$ and $\sum_a (E_{S_a}(f^*) + E'_{S_a}(f^*)) \leq 2E(f^*)$,

$$\sum_a E(f) - E(f_a) \geq E(f) - 2E(f^*) \tag{7}$$

Thus, since there are k labels, there must be some label a such that

$$E(f) - E(f_a) \geq \frac{1}{k}(E(f) - 2E(f^*)) \tag{8}$$

□

This guarantees that we have a constant factor improvement at each step, which means the algorithm reaches $2 + \epsilon$ times the optimal in a number of iterations logarithmic in the numerical parameters. So, the local search algorithm reaches a solution with the desired guarantee (plus an ϵ) in polynomial time.