

CS138 Set #1

Chris Beck

April 12, 2009

2.1

The upper bound on OPT here is simply $|E|$, the number of edges in G . Let $<$ denote the ordering in which the algorithm considers vertices of V . Each edge (i, j) is considered once, when the second vertex is added to the cut; after this round, those edges are then placed permanently either with the cut or across the cut. Thus the edges E are partitioned by the rounds. Each round places at least half of it's edges against the cut. Thus the total at the end contains at least half of every part, so at least half of all the edges are crossing the cut we end up with. We conclude $2 \cdot ALG = 2d(S, \bar{S}) \geq |E| \geq OPT$, and thus the algorithm achieves factor $1/2$ unconditionally.

To see that this analysis is tight, consider the complete bipartite graph $K_{2,n}$, and suppose that the algorithm considers the two first, and then the others. It thus places the first in S and the second in \bar{S} , and then no matter how it distributes the remaining vertices, the result will be that exactly half of the edges cross the cut. Yet the graph is bipartite, so the optimal cut contains all the edges. Thus the worst case performance is in fact $1/2$.

If we consider the complete graph K_{2n} , the optimal cut is into equal sized halves, with value n^2 , while there are $n(2n - 1)$ total edges. Thus the ratio of the true value of OPT to the upper bound of E is $\frac{n}{2n-1}$, which approaches $1/2$ asymptotically.

Aside: It is not actually possible to find a family of graphs which achieve OPT exactly as bad as $1/2 |E|$, which we prove using the algorithm just discussed. Assume that our graph is not the empty graph. Then we may run the algorithm with $v_1 \sim v_2$, so that our average yield of edges in the cut vs. edges seen so far is already strictly better than $1/2$ in the 0'th round. Then at the end of the last round, our cumulative average yield will be a weighted average of the average yield for all previous rounds, with none worse than $1/2$ and at least one strictly better, hence we find a cut with strictly more than $1/2|E|$ edges crossing it. Thus NO graph has $2 \cdot OPT = |E|$, and $OPT \rightarrow |E|/2$ in the limit is the nearest thing we can demonstrate.

The generalization to edge-weighted graphs is obvious; the new problem would be find the cut with maximum total weight crossing it, and the greedy algorithm compares, instead of $d(v, A)$ and $d(v, B)$, the sums $\sum_{a \in A} w(v, a)$ and $\sum_{b \in B} w(v, b)$. Similar arguments demonstrate that this is unconditionally $1/2$ approximating, and the same specific examples demonstrate tightness of the analysis and a similar worst case ratio of $1/2$ between OPT for this problem and the upper bound used in the analysis, which is simply the sum over all edges of the weight in this generalization.

2.2

After any flip, the value of the cut is increased by at least one. Since the value is always positive, and OPT is bounded above by n^2 , there can be at most polynomially many flips. Thus the local search heuristic finishes in polynomial time.

After termination, we have S such that the cut is locally optimal with respect to flipping. Thus $\forall v \in S, 2d(v, \bar{S}) \geq d(v)$, and $\forall v \in \bar{S}, 2d(v, S) \geq d(v)$. Adding all of this together,

$$2d(S, \bar{S}) + 2d(\bar{S}, S) \geq \sum_v d(v) = 2|E|$$

$$2d(S, \bar{S}) \geq |E|$$

Thus using the upperbound of $|E|$ on OPT , we see that this heuristic is $1/2$ approximating.

2.5

The algorithm we present here is a divide and conquer algorithm.

- Given an input graph, we start with an arbitrary cut and apply the local search algorithm of 2.2 to find a locally maximal cut in the graph, (S, \bar{S}) , with edges C crossing the cut.
Then in polynomial time, we may solve the minimum vertex cover problem perfectly on the subgraph with edgeset C , in polynomial time, since this graph is bipartite.
- If our cut C captures all of the edges, then we return the computed vertex cover and we are done.
- If there are leftover edges, i.e. $E - C$ is not empty, then the cut separates the graph on those edges into two connected components; these are the subgraphs induced by vertex sets S and \bar{S} .
- We then run the algorithm recursively on those components; what we return is the union of the results of those two calls, plus the minimum vertex cover computed in this call.

To see that this divide and conquer algorithm runs in polynomial time, we observe that algorithm 2.2 guarantees that if $N(v)$ denotes the set of edges incident on vertex v , then for any vertex v , at least half of the edges in $N(v)$ are crossing the cut it finds. If this were not true, the local search algorithm would switch vertex v to the other side of the cut, this is simply the definition of locally optimal. Since *every* vertex has half of it's edges crossing the cut, we see that if Δ is the max degree of the input graph to one call of the algorithm, the graphs passed to the two recursive calls have max degree at most $\Delta/2$. Thus the tree of all recursive calls made on a given input has depth at most $\log_2 \Delta$, since at each level, Δ is cut in half at least. The branching factor is fixed at two, so with depth $\log_2 \Delta$, there can be at most $2^{\log_2 \Delta + 1} = 2\Delta$ nodes in the tree. Each node carries out a polynomial time computation in terms of it's input, and Δ is certainly bounded above by a polynomial in terms of the input, so the whole computation runs in polynomial time.

Correctness: For any edge e , e must appear in some cut at some stage of the algorithm. Thus vertex cover produced at that stage covers e . Thus the union of all vertex covers produced at all stages covers every edge e . This is in fact the output of the algorithm, since each call of the algorithm aggregates the results of the subcalls together with the vertex cover it found, so at the top we simply have the union of all vertex covers. Therefore the algorithm produces a vertex cover of the original graph.

Approximation factor: Suppose $O \subset V$ is an optimal vertex cover for our graph. Given an induced subgraph on vertices $S \subset V$, it is clear that $O \cap S$ is a vertex cover for that subgraph. Consider that every recursive call in this algorithm operates on some induced subgraph of the original graph; in fact, since each call partitions it's vertices among it's two children, the collection of all calls made at depth i forms a partition of all the vertices V ; each vertex appears in some depth i call, and no vertex is shared. Thus if V_j is the subset handled by the j 'th call in level i , $O \cap V_j$ is a vertex cover, and since he finds an optimal one on that subgraph, his result is smaller than $|O \cap V_j|$. Thus the union of vertex covers found in depth i is smaller (by union bound) than $\sum_j |O \cap V_j| = O \cap V = O$, the equality following since the V_j are a partition of V . Now using union bound again, this time across all the levels, we see that the union of all vertex covers has size at most $|O| \cdot \log_2 \Delta$. Thus $ALG \leq OPT \cdot \log_2 \Delta$, proving our approximation factor.