CS 787: Advanced Algorithms

DC-Based Approximations on Graphs

Instructor: Dieter van Melkebeek

We now return to the one basic design paradigm we skipped in our discussion of approximation algorithms for NP-hard optimization problems, namely divide-and-conquer (DC), and apply it to problems on graphs. The basic idea is as follows: Find a balanced cut of small capacity, i.e., a cut in which both sides contain roughly the same number of vertices and such that few edges cross the cut. Ignore the cut edges, recursively solve the problem on the subgraphs induced by each side of the cut, and then use those solutions to construct a solution to the original instance. The latter will only be approximate because we ignored the edges that cross the cut.

Finding a balanced but of minimum capacity is itself an NP-hard optimization problem. We build our way towards constructing balanced cuts of small capacity by developing approximation algorithms for two problems related to multi-commodity flow: minimum multi-cut and sparsest cut. Those approximation algorithms will be based on LP relaxations and on efficient low-distortion embeddings of graph metrics into metrics on real spaces, namely L_1. We obtain a pseudo-approximation algorithm for balanced cut, and use it to develop an approximation algorithm for the minimum cut linear arrangement problem on graphs.

1 Multi-Commodity Flow

Multi-commodity flow is the generalization of the classical flow problem for a single commodity to multiple commodities that can simultaneously flow through the same pipe system (e.g., different oil products). Here is the problem formalization.
Given: Graph $G = (V, E)$ with non-negative edge weights. k pairs of vertices (s_i, t_i), $1 \leq i \leq k$ and non-negative demands, d_i, $1 \leq i \leq k$.
Goal: Find a flow for each commodity , i.e., observing

- conservation for each i individually.
- capacity constraints for the total flow.

such that either:

(1) Total flow is maximized. This is called the Max Multi Commodity problem. The d_i's are ignored for this case.

(2) Maximize λ such that for every commodity i the flow is $\geq \lambda \cdot d_i$. This is called the Max Concurrent Flow problem.

Both those problems can be cast as linear programs and be solved in polynomial time. We are interested in two problems that are closely related to the duals of the above problems. The input to those problems is the same as above.

(3) Minimum Multicut: Find a subset of edges $F \subseteq E$ (called a multicut) of minimum total weight such that after removal of those edges no pair (s_i, t_i) is connected. All the flow from the s_i's to the t_i's will have to pass through the edges in the multicut so the flow is upper bounded by the capacity of the multicut. Therefore, weak duality applies to give (1) $\leq$ (3).

(4) Sparsest Cut: Find $S \subseteq V$ such that the marginal cost of the cut $(S, \overline{S})$ is minimized. The marginal cost is expressed by the sparsity $\sigma(S)$ of the cut and formally defined as

$$\sigma(S) = \frac{\sum_{e \in \delta(S)} w_e}{\sum_{i \in I(S)} d_i},$$

where $\delta(S) \doteq E \cap (S \times \overline{S})$ denotes the edges that cross the cut, and

$$I(S) \doteq \{i \mid s_i \text{ and } t_i \text{ are on different sides of} S\}.$$

Weak duality applies here to give (2) $\leq$ (4).

Proof. Suppose that in (2), some λ is obtained. Then, for any $S \subseteq V$

$$\begin{aligned} \sum_{i \in I(S)} \lambda d_i &\leq \sum_{i \in I(S)} \text{Flow for } i \text{ that crosses } \delta(S) \\ &\leq \sum_{e \in \delta(S)} w_e \\ \lambda &\leq \frac{\sum_{e \in \delta(S)} w_e}{\sum_{i \in I(S)} d_i} \\ &= \sigma(S) \end{aligned}$$

Therefore, $\lambda \leq \min \sigma(S)$ □

Strong duality, however, does not hold, i.e., it is not true that the optimum value of (1) = optimum value of (3), or optimum value of (2) = optimum value of (4). This would imply $\mathsf{P} = \mathsf{NP}$ because (1) and (2) are representable in a LP and can be solved efficiently whereas (3) and (4) are NP-Hard problems. However, the existence of the $O(\log k)$ approximation algorithms we'll develop for both implies that (3) $\leq O(\log k)$(1) and that (4) $\leq O(\log k)$(2).

2 Minimum Multi-Cut Problem

We first develop a factor $4\ln(k+1)$ approximation algorithm for the Minimum Multi-Cut problem. The algorithm will not be used as such for the construction of balanced cuts of small cost, However, it is interesting in its own right and paves the way towards the approximation algorithm for Sparsest Cut, which will be a core ingredient in our construction of balanced cuts of small cost.

Recall that we are given a graph $G = (V, E)$ with non-negative weights on edges $w : E \mapsto [0, \infty)$ and a set of k pairs of vertices, say (s_i, t_i) for $1 \leq i \leq k$. We want to find a subset $F \subseteq E$ such that none of the (s_i, t_i) pairs is connected in $(V, E - F)$, and $w(F)$ is minimized.

We start by formulating the problem as an Integer Linear Program.

2.1 Integer linear program and its relaxation

$$min \sum_{e \in F} w_e x_e \qquad s.t. \quad \begin{cases} (\forall i)(\forall \text{path } P \text{ between } (s_i, t_i)) & \sum_{e \in P} x_e \geq 1 \\ (\forall e \in E) & x_e \in \{0, 1\} \end{cases}$$

The linear programming relaxation will be obtained by replacing the integral constraints, $x_e \in \{0, 1\}$, on the variables by $0 \leq x_e \leq 1$.

Observe that the above linear program can have exponentially many constraints, as there may be exponentially many paths between two vertices. Given a candidate solution for the linear program a separation oracle could check if it is a feasible solution. If not, it would give a constraint which was violated by the solution. A closer look at the linear program should convince you that if we let the edges have the weights x_e, then the constraints are equivalent to saying that the length of a shortest path between any pair, (s_i, t_i), of vertices to be disconnected should be greater than one. Checking this is easy. Just find shortest paths between every pair and then check if each of the pairs have distance at least one. At the same time if a solution is infeasible then we will have a pair (s_i, t_i) and a path connecting s_i and t_i of length less than one. Our separation oracle would return the constraint for this path. We also know that the interior point algorithm finds an optimal solution to a linear program, in polynomial time, provided that we have a separation oracle for this problem. As such, we can, in polynomial time, solve the LP relaxation of the above integer linear program.

3 Intuition behind the approximation algorithm

In this section we give the intuition behind the approximation algorithm. The details will be worked out later.

We first start from any feasible solution, x, of the LP relaxation. Once again we will appeal to the observation we made while designing the separation oracle. We will begin by defining a metric on the set of vertices. The distance function $d(\cdot, \cdot)$ is simply the shortest path metric, i.e., the distance $d(u, v)$ is equal to the length of a shortest path between the vertices u and v (if the "length" of an edge is x_e). The following interpretation to a feasible solution is crucial in the design of the algorithm we will discuss.

We can visualize every feasible solution to the LP relaxation of the problem as a network of pipes, the edges being the pipes and the vertices being the junctions where these pipes connect. Now suppose, we make the pipes have their cross-sectional area equal to the weight of the edge (w_e) they represent and their length equal to the value of x^*_e, then the total volume of the system is equal to the cost of the solution.

Note that the way we defined the metric ensures that any ball of radius less than half will never contain both s_i and t_i for any (s_i, t_i) pair which was to be disconnected in the original graph. This is because the diameter would be less than 1, and so the furthest distance between any two points inside the ball would have to be less than 1 as well, while any feasible solution has the length of a shortest path between any (s_i, t_i) pair greater than or equal to 1.

This gives us a very simple algorithm for finding a solution to the problem. The algorithm would iteratively pick a ball of radius at most half around an s_i which still needs to be disconnected from the corresponding t_i. It will then add all the edges that are cut by this ball to the solution set F. It would also remove everything, edges and vertices, that is inside the ball or is cut by it from the graph. Since this will never have both s_j and t_j for any j, the set F in the end will disconnect all such pairs. Note that in one iteration we could possibly disconnect more than one pair.

The next step would be to relate the volume of the ball to the cost of the cut represented by this ball. This way we will be able to relate the volume of the balls considered in a particular iteration to the weight of the edges added in F in that iteration. Since we get rid of the whole volume in the

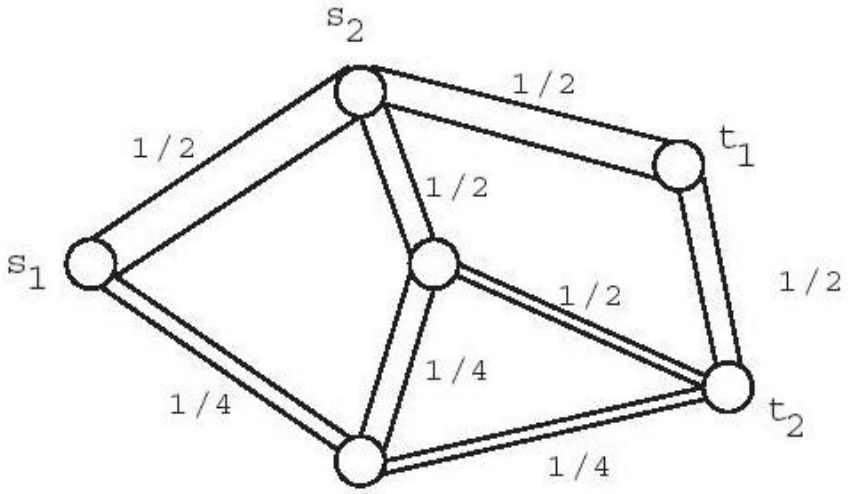


Figure 1: LP solution as a pipe system.

cut as well as the edges cut by it, the cumulative volume of the balls at each stage can not exceed the total volume of the system. Hence we have a relation between the cumulative weight of the edges in F and the total volume of the system. We have already said that the fashion in which we designed the network, will ensure that the total volume of the system will be exactly the cost of the feasible solution from which we started. Henceforth, we will have a relation between cost of the edges in the set F, found by the iterative procedure described above, and the solution from which we start. Needless to say, we will start with an optimal solution of the LP-relaxation. Therefore we will have a relation between the weight of the set F and the cost of the optimal solution of the LP-relaxation. This relation will turn out to be a linear dependence with a factor of at most $4\ln(k+1)$.

The only thing that remains is to relate the volume of a ball to the cumulative cost of the edges cut by it. Before that, let us introduce some notation.

Fix s_i. Let $B(r)$ denote a ball of radius r centered at s_i. Let $Int(B(r))$ be the set of edges completely inside $B(r)$, i.e., both the end points of the edge are at a distance at most r from s_i. Let $C(r)$ denote the set of edges cut by the ball. By cutting we mean that exactly one end point of the edge is inside $B(r)$. Unless stated otherwise, when we say $e = (u, v)$ is cut by $B(r)$, we will mean that the vertex u is inside $B(r)$ and v is outside it. Here we emphasize that the distance between two vertices is the length of a shortest path between them. It should not be confused with the euclidean distance between the vertices in the hypothetical pipeline network topology. The volume, $V(r)$, of a ball, $B(r)$, is then the total volume of edges inside the ball, and partial volumes of edges cut by the ball, based on the distance of the cut into the edge.

$$V(r) = \sum_{e \in Int(B(r))} w_e x_e + \sum_{e=(u,v) \in C(r)} w_e (r - d(s_i, u))$$

Observe that the volume function is differentiable with respect to r except for finitely many values of r where the function is discontinuous. We note that the discontinuities can occur only at the vertices, and we do not need continuity at these points for our purposes. Apart from these

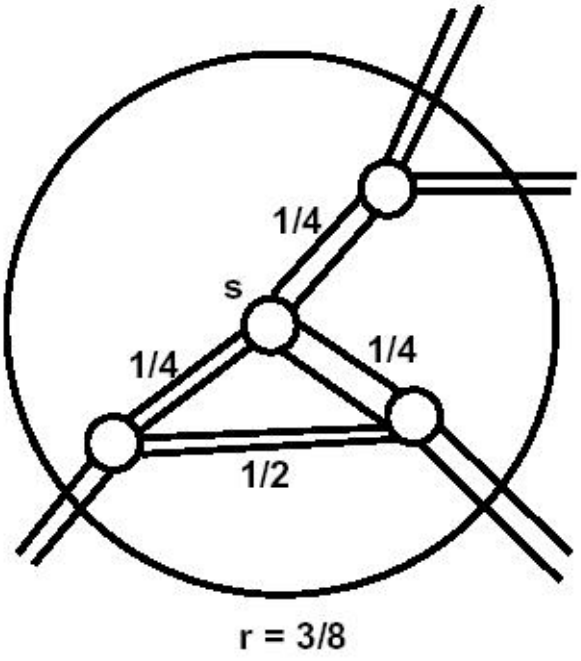


Figure 2: Example of a ball of radius 3/8.

discontinuities the rate of change of the volume is just the sum of the weights of the edges cut by it:

$$\frac{dV(r)}{dr} = \sum_{e \in C(r)} w_e$$

Since the edges have positive weights, $V(r)$ is a non-decreasing function. This holds even at the discontinuities because for $e = (u, v) \in E$ and $r = d(s_i, v)$, the triangle inequality implies that $r - d(s_i, u) \leq x_e$.

In order to guarantee the existence of a good radius less than $1/2$, we need to increase to base level of the volume by adding a point volume at the origin of the ball. That is, we consider

$$\tilde{V}(r) = V(r) + \alpha OPT_{LP}$$

for some $\alpha \geq 0$, where OPT_{LP} denotes the optimal value of the cost function obtained by solving the linear program (which is the same as the total volume of the system).

We want to choose a radius $r^* < \frac{1}{2}$ for each ball such that for some choice of α and ρ the following holds:

$$\frac{d\tilde{V}}{dr}(r^*) \leq \rho \tilde{V}(r^*)$$

In other words, the cost of the edges cut by the ball is less than a factor ρ times the volume of the system inside the ball (plus some factor of OPT_{LP}). Once we have this, we can argue that the iterative procedure we suggested above will give us a solution which has cost at most

$\rho(k\alpha + 1)OPT_{LP}$. To see this note that if we choose the ball of good radius r^* at each stage, we will have that

$$\begin{aligned} w(F) &\leq \rho \sum_{i=1}^{k} \tilde{V}_i(r_i^*) \\ &\leq \rho(\text{Total volume}) + \rho k\alpha OPT_{LP} \\ &= \rho((1 + k\alpha)OPT_{LP}) \end{aligned}$$

We now derive values for α and ρ such that we can indeed guarantee $r^* < \frac{1}{2}$.

3.1 Approximation algorithm and analysis

Here is the aproximation algorithm more formally.

1. **Input**: Graph $G(V, E)$ with edge weights w_e. Pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
2. Solve LP-relaxation and get a feasible solution.
3. $F \leftarrow \emptyset$, $H \leftarrow G$.
4. **while** there exists an i such that s_i is connected to t_i in $(V, E - F)$
5. Pick such i.
6. Pick $r < \frac{1}{2}$ such that $weight(C(r)) = \sum_{e \in C(r)} w_e \leq \rho \tilde{V}_H(r)$.
7. Add edges in $C_H(r)$ to F.
8. Remove from H, all edges in $Int(B_H(r))$ and $C_H(r)$.
9. **end while**
10. **Output**: F.

The following lemmas are immediate.

Lemma 1. *The algorithm terminates.*

Proof. We separate at least one pair of vertices per iteration of the while loop, and there are a finite number of these pairs, so the loop terminates. □

Lemma 2. *The algorithm returns a multicut.*

Proof. The only difficulty in seeing this is that in some iteration, when we separated s_i and t_i, we ended up with both s_j and t_j in S for some j. Then the algorithm would remove S from the graph and s_j and t_j might not be separated in the solution returned by the algorithm.

Suppose this happens. Then by construction of the algorithm, it follows that $d(s_i, s_j) \leq r$ and $d(s_i, t_j) \leq r$. This means that there is a path from s_j through s_i to t_j of length at most $2r$. Since $r < 1/2$, this path is of length less than 1 which can't happen since we start with a feasible solution to the LP-relaxation. □

Assume for the time being, that $\rho = 2\ln\left(\frac{\alpha+1}{\alpha}\right)$. We will prove later that this works, i.e., for this value of ρ, we can find an $r < 1/2$ such that $weight(C(r)) \leq \rho\tilde{V}(r)$. Also at this point, we choose that α, which had previously been unspecified, should be set to $\alpha = 1/k$.

Theorem 1. *The algorithm gives a factor* $4\ln(k+1)$ *approximation.*

Proof. We start with the inequality we established earlier:

$$\begin{aligned} w(F) &\leq \rho((1+k\alpha) + OPT_{LP}) \\ &= 2\rho OPT_{LP} \\ &= 4\ln(k+1)OPT_{LP} \\ &\leq 4\ln(k+1)OPT \end{aligned}$$

□

The following lemma completes the proof.

Lemma 3. *For* $\rho = 2\ln\left(\frac{\alpha+1}{\alpha}\right)$*, there is an* $r < 1/2$ *such that* $weight(C(r)) \leq \rho\tilde{V}(r)$.

Proof. Suppose no such r exists. Then, except for finitely many points, we have

$$\frac{1}{\tilde{V}}\frac{d\tilde{V}}{dr} > \rho \tag{1}$$

Integrating (1) between points of discontinuity (r, r'), we have

$$\int_{r+}^{r'-} \frac{1}{\tilde{V}}\frac{d\tilde{V}}{dr} > \rho(r'-r) \tag{2}$$

$$\text{i.e.} \qquad \ln\left(\frac{\tilde{V}(r'-)}{\tilde{V}(r+)}\right) > \rho(r'-r) \tag{3}$$

Here $\tilde{V}(r'-)$ stands for the limit of $\tilde{V}(\cdot)$ at r' from below and $\tilde{V}(r+)$ stands for the limit of $\tilde{V}(\cdot)$ at r from above. As $\tilde{V}(\cdot)$ is a non-decreasing function, we can replace the limits in the inequality above by the values at r and r'.

$$\ln\left(\frac{\tilde{V}(r')}{\tilde{V}(r)}\right) > \rho(r'-r)$$

Summing this over all intervals from 0 through 1/2, we get

$$\ln\left(\frac{\tilde{V}(1/2)}{\tilde{V}(0)}\right) > \frac{1}{2}\rho$$

So,

$$\tilde{V}(1/2) > \tilde{V}(0)e^{\rho/2} \tag{4}$$

$$= \alpha OPT_{LP} \cdot e^{\rho/2} \tag{5}$$

Since $\tilde{V}(1/2)$ is only a part of the total volume (plus αOPT_{LP}), this implies that:

$$\begin{aligned}(\alpha+1)OPT_{LP} &\geq \tilde{V}(1/2) \\ &> \alpha OPT_{LP} \cdot e^{\rho/2}\end{aligned}$$

and thus,

$$\rho < 2\ln\left(\frac{\alpha+1}{\alpha}\right)$$

which is a contradiction. Thus, there must be some $r < 1/2$ that suits our needs at each iteration. □

4 Sparsest Cut

We now use some of the ideas underlying the approximation algorithm for Minimum Multi-Cut to develop an $O(\log k)$ approximation algorithm for Sparsest Cut.

We first point out that is suffices to find a multi-cut of small sparsity, i.e., a subset $F \subseteq E$. Such a subset may induce multiple partitions (cuts). Let F induce the partition ρ of V, where $\rho = \{S_1, S_2, \ldots, S_m\}$. We define

$$\sigma(\rho) = \frac{\sum_{e\in F} w_e}{\sum_{i\in I(\rho)} d_i},$$

where $I(\rho) = \{i \mid s_i \text{ and } t_i \text{ belong to different partitions in } \rho\}$.

Claim 1.

$$\min_{1\leq i\leq m} \sigma(S_i) \leq \sigma(\rho)$$

Proof. This follow from a well known inequality which states that if $a_i \geq 0$ and $b_i \geq 0$ then,

$$\min_i \left(\frac{a_i}{b_i}\right) \leq \frac{\sum_i a_i}{\sum_i b_i}$$

The proof of this inequality is left as an exercise. □

As we can efficiently find the partition ρ from F, compute the sparsities $\sigma(S_i)$, and find the minimum, the Claim shows that if suffices to find a multi-cut F with low sparsity.

4.1 Integer program and LP relaxation

We'll now formulate an Integer Program for finding such a subset F. This is very similar to the Minimum Cut formulation. There is a variable for every edge and a variable for every commodity.

$$\begin{aligned}&\min \frac{\sum_e w_e x_e}{\sum_i d_i y_i} \\ &y_i \leq \sum_{e\in P} x_e \quad (\forall i)(\forall\ path\ P\ (s_i, t_i)) \\ &x_e \in \{0,1\} \quad (\forall e \in E) \\ &y_i \in \{0,1\} \quad (\forall i \in [k])\end{aligned}$$

Here x_e is an indicator that edge $e \in F$, and y_i indicates whether $i \in I(F)$. The above formulation models the problem exactly, but is not linear due to the objective being a ratio of linear functions rather than merely a linear function. We can relax the integrality constraints and make the objective linear by introducing another constraints which forces the denominator to be 1. The LP relaxation is given below:

$$\begin{aligned}
&\min \sum_e w_e x_e \\
&y_i \leq \sum_{e \in P} x_e \quad (\forall i)(\forall\ path\ P\ (s_i, t_i)) \\
&\sum_i d_i y_i = 1 \\
&x_e, y_i \geq 0
\end{aligned}$$

A couple notes are in order.

- The x_e's can be viewed as lengths connecting the two vertices. This interpretation is similar to that given for the LP of the Minimum Multicut problem. In the optimal solution, y_i's for $d_i > 0$ are the shortest path distance between s_i and t_i where x_e's are the edge lengths.
- In spite of an exponential number of constraints, the above Linear program can be solved efficiently because of the existence of an efficient separation oracle (as in the min multicut problem). We can check if a constraint is violated by finding the shortest path with x_e's as edge lengths for every (s_i, t_i) and then check if it is $\geq y_i$.

4.2 Approximation algorithm and analysis

Like in the approximation algorithm for Minimum Multi-Cut, we use the optimal solution x_e^* of the LP relaxation, and consider the shortest-path metric d that is induces when the x_e's are interpreted as edge lengths. The above observations imply that

$$\sigma(d) \doteq \frac{\sum_{e=(u,v) \in E} w_e d(u,v)}{\sum_i d_i d(s_i, t_i)}$$

satisfies

$$\sigma(d) \leq OPT_{LP} \leq OPT. \tag{6}$$

A particularly interesting type of metric in this context is a *cut metric* μ_S for some $S \subseteq V$, where the distance is 0 for two points on the same side of the cut $(S, \overline{S})$, and 1 otherwise. If the metric d happens to be a cut metric $d = \mu_S$, then we are home free. This is because then $\sigma(d)$ equals the sparsity of S, so (6) we have found a cut S of minimum sparseness. This corresponds to the case where the optimum of the LP relaxation happens to be integral, and cannot happen in general unless $\mathsf{P} = \mathsf{NP}$.

A more general case that we can handle similarly are metrics d in the cone spanned by the cut metrics, i.e., nonnegative linear combinations of cut metrics:

$$d = \sum_S \lambda_S \mu_S, \tag{7}$$

where $\lambda_S \geq 0$. This is because for such a metric we can write

$$\begin{aligned}\sigma(d) &= \frac{\sum_e w_e d(e)}{\sum_i d_i d(s_i, t_i)} \\ &= \frac{\sum_e w_e \sum_S \lambda_S \mu_S(e)}{\sum_i d_i \sum_S \lambda_S \mu_S(s_i, t_i)} \\ &= \frac{\sum_S \lambda_s \sum_e w_e \mu_s(e)}{\sum_S \lambda_s \sum_i d_i \mu_s(s_i, t_i)}\end{aligned}$$

Because the numerator is just a sum of the numerators of each of $\sigma(S)$, and the denominator is a sum of denominators of the same, the ratio is in fact a mediant of all the $\sigma(S)$. By the property we used before, we have that

$$\min_{\lambda_S > 0} \sigma(S) \leq \sigma(d). \tag{8}$$

Thus, if we can efficiently decompose d as in (7) then we're again home free and can efficiently find a cut of minimum sparsity. Again, unless $\mathsf{P} = \mathsf{NP}$, this cannot be the general case, but the following characterization of this case is critical: the cone of cut metrics consists exactly of those metrics that embed isometrically into real space with the L_1-metric. Recall that a metric *isometrically* embeds into another metric if there exists a mapping from the first to the second that preserves distances, and that

$$L_p : (x, y) \mapsto \| x - y \|_p = \left(\sum_i |x_i - y_i|^{\frac{1}{p}} \right)$$

defines a metric on $\mathbb{R}^\ell$ for every real $p \geq 1$ and every positive integer ℓ.

Lemma 4 (Decomposability Lemma)**.** *A metric $d' : V \times V \to [0, \infty)$ isometrically embeds in $(\mathbb{R}^\ell, L_1)$ for some ℓ iff $d' = \sum \lambda_S \mu_S$ with $\lambda_S > 0$. Moreover, given d' we can compute a decomposition of cut metrics with less than $|V| \cdot \ell$ terms in polynomial time.*

Proof. $\boxed{\Longleftarrow}$ Each μ_S is isometrically embeddable in $(\mathbb{R}^1, l_1)$ - simply assign every element on one side of the cut to 0 and every element on the other side of the cut to 1. $\lambda_s \mu_S$ is just a scaling and doesn't change anything to the embeddability of the metric. Now summing over all possible cuts we get that $\sum_S \lambda_S \mu_S$ as the embedded metric.

$\boxed{\Longrightarrow}$

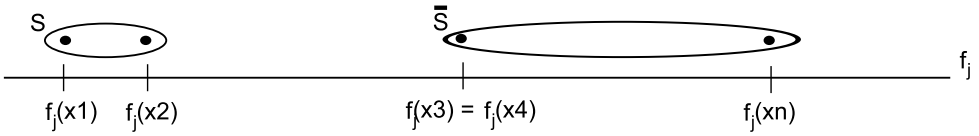


Figure 1: Illustration of the cut metric induced by a set S of vertices

Say f is an isometric embedding of $V \to \mathbb{R}^l$. Consider the j'th component of the embedding and call it f_j. We order the elements of V as $x_1, x_2, \ldots, x_n$ such that $f_j(x_m) \leq f_j(x_{m+1})$ for $1 \leq m < n$. Define $S_m = \{x_1, x_2, \ldots, x_m\}$ and consider a particular $x \leq y$. $\mu_{S_m}(x, y) = 1$ iff $x \in S_m$ and $y \notin S_m$. In the figure this corresponds to x being in the left set and y in the right set. Therefore

$$|f_j(x) - f_j(y)| = \sum_{m=1}^{n-1} (f_j(x_{m+1}) - f_j(x_m)) \mu_{S_m}(x, y).$$

This equation may need some explanation. This works, because the total distance between x and y is the sum of the distances between all the consecutive x_r's in the interval $[x, y]$. The distance between any two consecutive x_r and x_{r+1} in $[x, y]$ is counted exactly once, namely when $m = r$. The distances between consecutive x_r's outside the interval $[x, y]$ are not counted, because they are multiplied by $\mu_{S_m}(x, y)$, which is 0 for m such that $x_m \notin [x, y]$.

Note that the above is clearly a sum of cut metrics with $\lambda_S = f_j(x_{m+1}) - f_j(x_m) \geq 0$. Summing over all possible dimension j we get

$$\begin{aligned} ||f(x) - f(y)||_1 &= \sum_{j=1}^{l} |f_j(x) - f_j(y)| \\ &= \sum_{j=1}^{l} \left(\sum_{m=1}^{n-1} (f_j(x_{m+1}) - f_j(x_m)) \mu_{S_m}(x, y) \right) \end{aligned}$$

Note that $||f(x) - f(y)||_1 = d'(x, y)$. So this argument also shows that the number of components is upper bounded by the number of dimensions ℓ times the number of points $|V|$. Finding the decomposition can be done in polynomial time. □

Now, in order to obtain our approximation algorithm for the general case, we make use of the fact that any metric d can be efficiently embedded into real space under the L_1-metric with small distortion. More specifically, the following result is key.

Theorem 2 (Small Distortion Embeddings into L_1)**.** *There exists a randomized polynomial time algorithm that takes (V, d) as input and produces an embedding $f : V \to \mathbb{R}^\ell$ such that with probability at least 1/2:*

1. $(\forall e = (u, v) \in E) d'(u, v) \leq O(\log k) \cdot d(u, v)$,
2. $(\forall i) d'(s_i, t_i) \geq d(s_i, t_i)$,

where $d'(x, y) \doteq ||f(x) - f(y)||_1 = \sum_{j=1}^{\ell} |f_j(x) - f_j(y)|$.

We will prove Theorem 2 in the next section.

Note that $\sigma(d') \leq O(\log k)\sigma(d)$. Since d' isometrically embeds into L_1, we can apply Lemma 4 and finds a cut S with sparsity $\sigma(S) \leq \sigma(d') \leq O(\log k)\sigma(d) \leq O(\log k)OPT$. Putting everything together, this gives us an $O(\log k)$ approximation algorithm for Sparsest Cut

Theorem 3. *There exists a randomized approximation algorithms for Sparsest Cut with approximation factor $O(\log k)$.*

Proof. Here is the algorithm:

1. We solve the LP relaxation and get optimal solutions x_e^*, y_i^*. We consider the shortest path metric d induced by the x_e^*'s.
2. Approximate d by d' by isometrically embedding it in $(\mathbb{R}^\ell, L_1)$.
3. Decompose d' as $d' = \sum \lambda_S \mu_S$ with $\lambda_S > 0$ and output the S that minimizes $\sigma(S)$.

The analysis follows from the discussion above. □

5 Small distortion embeddings into L_1

We prove the following slightly stronger version of Theorem 2.

Theorem 4 (Key Theorem)**.** *There exists a randomized polynomial time algorithm that takes* (V, d) *as input and produces an embedding* $f : V \to \mathbb{R}^\ell$ *with* $\ell = O(\log^2(k))$ *such that*

1. $(\forall e = (u, v) \in E) d'(u, v) \leq \ell \cdot d(u, v)$,
2. $\Pr[(\forall i) d'(s_i, t_i) \geq c \log(k) d(s_i, t_i)] \geq \frac{1}{2}$,

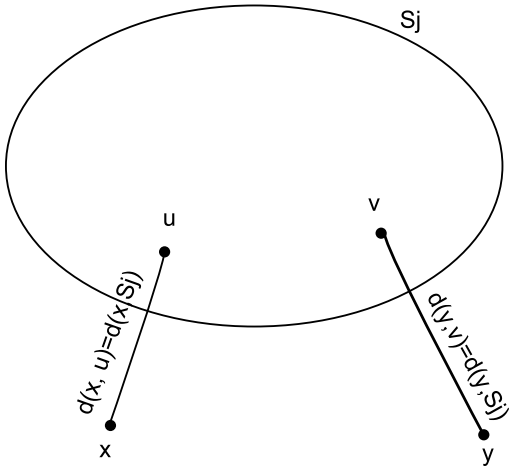
for some constant c *where* $d'(x, y) = ||f(x) - f(y)||_1 = \sum_{j=1}^{\ell} |f_j(x) - f_j(y)|$.

We leave it as an exercise to show that Theorem 2 follows.

Note that the theorem tries to prove some sort of datacompression. The original metric assigns a number for every pair of points or $O(n^2)$ numbers, where $n = |V|$. In our embedding, we will aggregate distance information and set $f_j(x) = d(x, S_j)$ for some $S_j \subseteq V$, or, the metric assigns a distance between a point and a subset. We choose the new metric to be $d(x, S_j) = min_{u \in S_j} d(x, u)$.

5.1 Proof of statement (1)

FACT 1. $\forall x, y \in V : |d(x, S_j) - d(y, S_j)| \leq d(x, y)$.



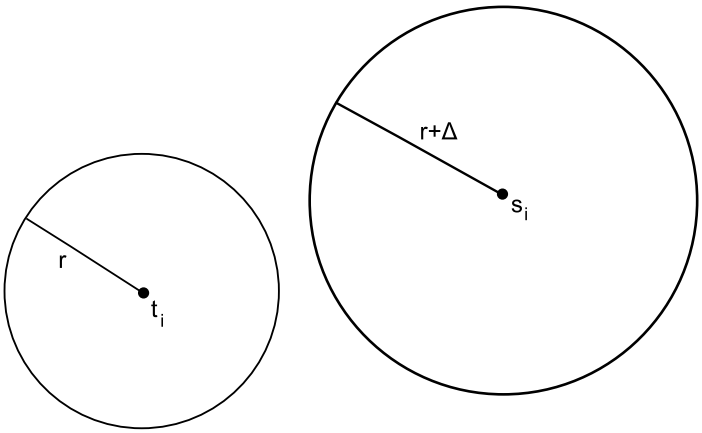
Suppose u is the closest element in S_j to x and v is the closest element in S_j to y, then $d(x, S_j) = d(x, u) \leq d(x, v)$. By the triangle inequality we know that $d(x, v) \leq d(x, y) + d(y, S_j)$, therefore $d(x, S_j) - d(y, S_j) \leq d(x, y)$. Together with the same derivation started from y this gives us the above statement.

We are now ready to prove statement (1) from the key theorem. We know that if each f_j is of the form $f_j(x) = d(x, S_j)$ for some $S_j \subseteq V$ then $d'(x, y) = ||f(x) - f(y)||_1 = \sum_{j=1}^{l} |f_j(x) - f_j(y)| \leq l \cdot d(x, y)$.

5.2 Proof of statement (2)

In this section we will show how to pick the S_j appropriately. Note first that the S_j only depend on the sources and the sinks. Therefore we will only consider those in the next part: we thus define $V' = \{s_i\} \cup \{t_i\}$. Note that $|V'| \leq 2k$.

FACT 2. Let us fix commodity i.



We will define two balls centered at s_i and t_i, with parameres r, Δ specified later and choose the sets S_j such that the following holds:

$$B(s_i, r + \Delta) \cap S_j = \emptyset \qquad (*)$$
$$B(t_i, r) \cap S_j \neq \emptyset.$$

Then we know that $f_j(s_i) - f_j(t_i) \geq \Delta$. We now introduce randomness in order for the sets S_j to be independent from the (s_i, t_i) pair chosen.

Suppose both balls ($B(s_i, r + \Delta)$ and $B(t_i, r)$) contain about p points in V'. If we put each $v \in V'$ in S_j with probability about $\frac{1}{p}$, independently, then each of the balls satisfies

$$\Pr[B \cap S_j = \emptyset] \in (\epsilon, 1 - \epsilon)$$

for some constant $\epsilon > 0$. If both balls are disjoint then

$$\Pr[(*)] \geq \epsilon^2$$

and if $(*)$ holds then $f_j(s_i) - f_j(t_i) \geq \Delta$. Thus,

$$E\left[|f_j(s_i) - f_j(t_i)|\right] \geq \epsilon^2 \cdot \Delta$$

5.2.1 Construction of S_j

We want to make sure that the number of elements of $B \cap V'$ in S_j is approximately 1, but we can't use an exact count (like p above) as the sets S_j would be dependent on (s_i, t_i). Therefore we choose independently, s.t.

$$(\forall v \in V') \Pr[v \in S_j] = \frac{1}{2^{j+1}}.$$

5.2.2 Construction of the balls

Let

$$r_j^{(x)} = \min\{r : |B(x,r) \cap V'| \geq 2^j\}$$

for $0 \leq j \leq \lfloor \log |V'| \rfloor$. Remember we have fixed i. For this specific i, we define

$$r_j = \max(r_j^{(s_i)}, r_j^{(t_i)})$$

Assume for the moment that $r_j^{(s_i)} \geq r_j^{(t_i)}$; the other case is symmetric. Consider $\circ B(s_i, r_j)$, the open ball around s_i with radius r_j, and $B(t_i, r_{j-1})$, the closed ball around t_i with radius r_{j-1}. Then

$$\begin{aligned} |\circ B(s_i, r_j) \cap V'| &\leq 2^j \\ |B(t_i, r_{j-1}) \cap V'| &\geq 2^{j-1} \end{aligned}$$

So the probability that the bigger of the two balls is empty is given by

$$\begin{aligned} \Pr[\circ B(s_i, r_j) \cap S_j = \emptyset] &= \left(1 - \frac{1}{2^{j+1}}\right)^{(|\circ B \cap V'|)} \\ &\geq \left(1 - \frac{1}{2^{j+1}}\right)^{2^j} \\ &\geq \frac{1}{2} \end{aligned}$$

and the probability that the smaller of the two balls is not empty is given by

$$\begin{aligned} \Pr[B(t_i, r_{j-1}) \cap S_j \neq \emptyset] &= 1 - \left(1 - \frac{1}{2^{j+1}}\right)^{(|B \cap V'|)} \\ &\geq 1 - \left(1 - \frac{1}{2^{j+1}}\right)^{2^{j-1}} \\ &\geq 1 - e^{-\frac{1}{4}} \end{aligned}$$

If $r_j \leq \frac{d(s_i,t_i)}{2}$, both events denoted by $(*)$ are independent, so if we then choose $\epsilon = min(e^{-\frac{1}{4}}, \frac{1}{2})$, then

$$\Pr[(***)] \geq \epsilon^2$$

where $(***)$ represents

$$|f_j(s_i) - f_j(t_i)| \geq r_j - r_{j-1} \quad (***)$$

In particular,

$$E\left[|f_j(s_i) - f_j(t_i)|\right] \geq \epsilon^2 \cdot (r_j - r_{j-1}) \quad (**)$$

which is obtained by applying Markov's inequality.

Let $t = max\{j : r_j \leq \frac{d(s_i,t_i)}{2}\}$. Summing $(**)$ over all $j = 1, 2, \ldots, t$ gives

$$E\left[\sum_{j=1}^{t} |f_j(s_i) - f_j(t_i)|\right] \geq \epsilon^2 \cdot r_t$$

because of the telescoping sum and the fact that $r_0 = 0$. We also have

$$E\left[|f_{t+1}(s_i) - f_{t+1}(t_i)|\right] \geq \epsilon^2(r_{t+1} - r_t)$$

if we redefine $r_{t+1} = \frac{d(s_i,t_i)}{2}$. This is because r_{t+1} would be larger than $\frac{d(s_i,t_i)}{2}$ and by redefining it as a smaller quantity the inequality is certainly true. Adding the last two equations we obtain

$$E\left[\sum_{j=1}^{t+1} |f_j(s_i) - f_j(t_i)|\right] \geq \epsilon^2 \cdot \frac{d(s_i,t_i)}{2}$$

We now have the result (2) in expectation for each individual commodity. We apply a concentration result to arrive at (2) for all commodities simultaneously. Say we run the process m times independently, where m remains to be determined. Let d'_q be the result of the q'th run. Then for some fixed i and j

$$\Pr[(***) \text{ holds for} \geq \frac{\epsilon^2}{2}m \text{ of the runs}] \geq 1 - e^{-\frac{\epsilon^2}{8}m},$$

by applying Chernoff's inequality:

$\Pr\left[\sum_{j=1}^{m} X_j \leq (1-\eta)\mu\right] < e^{-\frac{\eta^2}{2}}m$, where we let X_i be the indicator variable for $(***)$ on the i'th run, and $\mu = \sum_{q=1}^{m} E[X_q]$ for which we choose $\eta = \frac{1}{2}$. Note that $\mu \geq \epsilon^2 \cdot m$.

Let $A = \#i\text{'s} * \#j\text{'s}$. Then $A \leq k(\log k + O(1))$. So if

$$e^{-\frac{\epsilon^2}{8}m} \leq \frac{1}{2A}$$

then

$$\Pr[(\forall i, j)(***) \text{ holds for} \geq \frac{\epsilon^2}{2}m \text{ of the runs}] \geq \frac{1}{2}$$

In that case consider $d' = \sum_{q=1}^{m} d'_q$.

$$\begin{aligned} d'(s_i,t_i) &= \sum_{q=1}^{m} d'_q(s_i,t_i) \\ &\geq \frac{\epsilon^2}{2} \cdot m \cdot \frac{d(s_i,t_i)}{2} \end{aligned}$$

By picking $m = O(\log k)$ we have that (2) holds and our algorithm produces an embedding in $\mathbb{R}^l$ with $l = O(\log^2(k))$. The embedding thus looks like: $f(x) = [d(x, S_1), d(x, S_2), \ldots, d(x, S_{O(\log^2(k))})]$.

5.3 Algorithm

To summarize, here is the resulting embedding algorithm:

Input: A metric space (V, d).
Output: An embedding to a new metric space $f : (V, d) \to (\mathbb{R}^{O(\log^2(k))}, L_1)$.
EMBEDDING$((V, d))$
For $i = 1$ to $O(\log^2(k))$
  For all $a \in V$
    Put a in S_i with probability $\frac{1}{2^{i+1}}$.
  For all $x \in V$
    Set $f(x) = [d(x, S_1), d(x, S_2), \ldots, d(x, S_{O(\log^2(k))})]$.
return f

6 Balanced Cut

We now use our $O(\log k)$ approximation algorithm for Sparsest Cut to efficiently obtain balanced cuts of small capacity.

Let us first formalize what we mean by a balanced cut and the Balanced Cut problem.

Definition 1 (Balanced Cut)**.** *Given a graph $G = (V, E)$, a cut $(S, \overline{S})$ is α-balanced iff $|S| \in [\alpha \cdot n, (1 - \alpha) \cdot n]$, where $n = |V|$. That is, a cut is α-balanced if both S and $\overline{S}$ contain at least a fraction α of the vertices.*

Definition 2 (Balanced Cut Problem)**.** *Given a graph $G = (V, E)$ with edge weights $w_e \geq 0$ and some $\alpha \leq 1/2$, the goal is to find an α-balanced cut $(S, \overline{S})$ of minimum cost.*

$$Cost(S) = \sum_{e \in \delta(S)} w_e$$

Additionally, let $OPT_\alpha(G) = \min\big(\{Cost(S) \mid (S, \overline{S}) \text{ is } \alpha\text{-balanced}\}\big)$.

We will use uniform sparsest cuts to solve the balanced cut problem. Recall the sparsest cut problem is defined as follows: We are given a graph $G = (V, E)$ with edge weights $w_e \geq 0$, k commodity pairs $(s_i, t_i) \in V \times V$, and a demand d_i for each pair. The goal is to find $S \subseteq V$ that minimizes

$$\sigma(S) = \frac{\sum_{e \in \delta(S)} w_e}{\sum_{i \in I(S)} d_i},$$

where $I(S) = \{i \mid s_i \text{ and } t_i \text{ are on different sides of } S\}$. The *uniform* sparsest cut problem has a commodity between every pair of vertices (so $k = \binom{n}{2}$), and the demand for each pair, $d_i = 1$. Let $OPT_{SC}(G)$ denote the cost $\sigma(S)$ of an optimal solution S to the uniform sparsest cut problem.

The idea for the balanced cut approximation algorithm is similar to that of the greedy set cover approximation algorithm, which at each step picks a set with minimal marginal cost. Here we will at each step choose some set $\tilde{S} \in \overline{S}$, keeping the cost $\sigma_H(\tilde{S})$ low (H is the graph with vertices in S remove). The actual algorithm is as follows.

Algorithm 1: Balanced Cut Approximation Algorithm

(1) $H \leftarrow G$
(2) $S \leftarrow \emptyset$
(3) **while** $|S| < \alpha n$
(4) $\quad \tilde{S} \leftarrow$ subset of $V(H)$ s.t. $|\tilde{S}| \leq \frac{|V(H)|}{2}$ and $\sigma_H(\tilde{S}) \leq \rho_H \cdot OPT_{SC}(H)$
(5) $\quad S \leftarrow S \cup \tilde{S}$
(6) $\quad H \leftarrow H \setminus \tilde{S}$
(7) Return S

We can use our sparsest cut approximation algorithm to select $\tilde{S}$ each iteration. That algorithm returns a set of vertices S^*, but $\overline{S^*}$ creates a cut with the same sparseness, so we can choose whichever one leads to the size $\tilde{S} \leq \frac{|V(H)|}{2}$ as desired. Choosing $\tilde{S}$ in this manner, what approximation factor ρ_H is achieved? Because the sparsest cut algorithm gives factor $O(\log k)$ result, $\rho_H = O(\log(\#\text{ pairs in H})) = O(\log(|V(H)|^2)) = O(\log n)$ is the approximation factor of the sparseness of the cut for each iteration. Since we will only care about this asymptotic notion, we will just take $\rho = O(\log n)$ and use that everywhere instead of ρ_H as it is an upper bound.

Claim 2. *The algorithm returns an α-balanced cut, provided $\alpha \leq 1/3$.*

Proof. Since the while loop guarantees that $|S| \geq \alpha \cdot |V(G)|$ when the algorithm terminates, we only need to argue that we do not overshoot. That is, we only need to check that the solution S returned has size $|S| \leq (1-\alpha)n$.

Consider one iteration of the while loop. At its entry, we have $|S| < \alpha n$, and we know $|\tilde{S}| \leq \frac{|V(H)|}{2} = \frac{n-|S|}{2}$.

$$\begin{aligned} |S \cup \tilde{S}| &\leq |S| + \frac{n-|S|}{2} \\ &= \frac{n+|S|}{2} \\ &< \frac{n+\alpha n}{2} \\ &= \frac{1+\alpha}{2} n \end{aligned}$$

With $\alpha \leq 1/3$, we have $1 + \alpha < 2(1-\alpha)$, and making that substitution above, get $|S \cup \tilde{S}| < (1-\alpha)n$, which was desired. □

Claim 3. $(\forall \alpha < \beta \leq 1/2)$, $Cost(S) \leq \frac{3(1-\alpha)}{\beta-\alpha} \rho \cdot OPT_\beta(G)$.

Proof. Let S^* be an optimal β-balanced cut for G; that is, $Cost(S^*) = OPT_\beta(G)$. Let H_i be H at iteration i, $V_i = V(H_i)$, $\tilde{S}_i$ is $\tilde{S}$ at iteration i, and $S_i^* = S^* \cap V_i$.

$$\begin{aligned} Cost(\tilde{S}_i) &= \sigma_{H_i}(\tilde{S}_i) \cdot |\tilde{S}_i| \cdot |V_i \setminus \tilde{S}_i| \text{ (uses definition of } \sigma\text{, and uniform setting)} \\ &\leq \rho \cdot \sigma_{H_i}(S_i^*) \cdot |\tilde{S}_i| \cdot |V_i \setminus \tilde{S}_i| \end{aligned}$$

Now we will focus on the term

$$\sigma_{H_i}(S_i^*) = \frac{\sum_{e \in \delta(S_i^*)} w_e}{|S_i^*||V_i \setminus S_i^*|}$$

First we know that the numerator is certainly $\leq OPT_\beta(G) = \sum_{e \in \delta(S^*)} w_e$, because $S_i^* \subseteq S^*$ (note also that $V_i \subseteq V$). In other words, the edges that cross the cut S^* in a subset of the graph definitely cross the same cut in the full graph, and thus their cost is no more than the total cost of the cut.

Next, we want to get a lower bound on the denominator. Both $|S^*| \geq \beta n$ and $|S^*| \geq (1-\beta)n$, due to the fact that S^* is a β-balanced cut in G. At iteration i, $|\overline{V_i}| = |S| < \alpha n$, so

$$\begin{aligned}
|S_i^*| &= |S^* \cap V_i| \\
&= |S^* \cap (V \setminus \overline{V_i})| \\
&= |S^* \setminus (S^* \cap \overline{V_i})| \\
&= |S^*| - |S^* \cap \overline{V_i}| \text{ (b/c second term is subset of first)} \\
&> \beta n - \alpha n \\
&= (\beta - \alpha)n
\end{aligned}$$

... and ...

$$\begin{aligned}
|V_i \setminus S_i^*| &= |V_i| - |S_i^*| \text{ (again b/c second term subset of first)} \\
&\geq (1-\alpha)n - (1-\beta)n \\
&= (\beta - \alpha)n
\end{aligned}$$

So both $|S_i^*|$ and $|V_i \setminus S_i^*|$ are bounded below by $(\beta - \alpha)n$. Because $|V_i| \geq \frac{2}{3}n$, we also know that one of those two sets is in fact $\geq \frac{1}{3}n$ on any given iteration. Then we can craft this bound on their product:

$$|S_i^*||V_i \setminus S_i^*| \geq \frac{1}{3}(\beta - \alpha)n^2$$

Together with the bound on the numerator, we can now give the following bound:

$$\sigma_{H_i}(S_i^*) \leq \frac{3OPT_\beta(G)}{(\beta - \alpha)n^2}$$

Inserting this in our expression for $Cost(\tilde{S}_i)$ above, and using the fact that $|V_i \setminus \tilde{S}_i| \leq n$ (because $|V_i| \leq n$):

$$Cost(\tilde{S}_i) \leq \rho \cdot \frac{3OPT_\beta(G)}{(\beta - \alpha)n} \cdot |\tilde{S}_i|$$

Now we sum over the iterations to obtain the following:

$$\begin{aligned}
Cost(S) &\leq \sum_i Cost(\tilde{S}_i) \quad \text{(see following paragraph)} \\
&= \rho \cdot \frac{3OPT_\beta(G)}{(\beta - \alpha)n} \cdot \sum_i |\tilde{S}_i| \\
&\leq \rho \cdot \frac{3OPT_\beta(G)}{(\beta - \alpha)n} \cdot (1-\alpha)n \quad (\bigcup \tilde{S}_i = S \text{ and } \tilde{S}_i \cap \tilde{S}_j = \emptyset) \\
&= \frac{3(1-\alpha)}{\beta - \alpha} \rho \cdot OPT_\beta(G)
\end{aligned}$$

The inequality $Cost(S) \leq \sum_i Cost(\tilde{S}_i)$ holds because each edge $e = (u, v)$ which crosses cut S had vertex u inserted into S on some specific iteration, i. On that iteration, the weight of edge e was counted because u and v were in H but u was in $\tilde{S}_i$ while v was not. Over all iterations, all edges cutting S were counted, so the cost is no greater than the sum of the iterative costs.

Note also that an edge (u, v) that crosses the cut S at some intermediate point in the algorithm will not cross the final cut if v is later added to S. This means the first inequality may be a very large overestimate of $Cost(S)$. See Figure 2. □

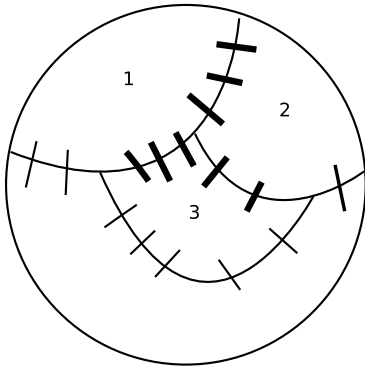


Figure 2: Edges crossing S at some step of algorithm. Note that edges in bold do not cross final cut.

Theorem 5. *For any $\alpha \leq 1/3$ and $\alpha < \beta \leq 1/2$, we can in polynomial time find an α-balanced cut of cost at most $O(\log n) \cdot OPT_\beta$. ($\alpha$ and β are treated as constants.)*

Proof. Follows from Claim 1 and Claim 2. □

Note that OPT_β could be much larger than OPT_α, and therefore the algorithm is only a *pseudo*-approximation algorithm.

7 Minimum Cut Linear Arrangement Problem

Finally, we give an example of an NP-hard optimization problem on graphs to which we can apply our DP-approach based on our pseudo-approximation algorithm for Balanced Cut.

We first introduce the problem.

Definition 3 (Minimum Cut Linear Arrangement Problem)**.** *Given a graph $G = (V, E)$ with edge weights $w_e \geq 0$, the goal is to find an ordering $\pi : V \to \{1, 2, ..., n\}$ such that*

$$C(G) = \max_{1 \leq i \leq n} w(E \cap (I_i \times \overline{I_i}))$$

is minimized, where $I_i = \{v \in V \mid \pi(v) \leq i\}$. See Figure 3 for an example.

The approximation algorithm for the minimum cut linear arrangement (MCLA) problem is as follows. We first call the balanced cut approximation algorithm to partition V into a 1/3-balanced cut $(S, \overline{S})$. Then we recursively solve the two subproblems, $G|_S$ and $G|_{\overline{S}}$, induced by S and $\overline{S}$.

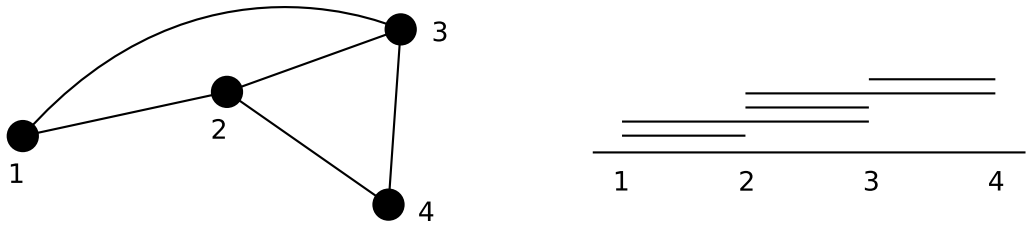


Figure 3: Left: An arrangement π for some graph G. Right: The edges of G under the arrangement π. Note that $C(G)$ for this arrangement is 3 (assume all edge-weights are 1), for $i = 2$.

Finally, the solution is derived by simply concatenating the solutions to the two subproblems. Note that neither $|S|$ nor $|\bar{S}|$ is greater than $(1-\alpha)n$, and therefore each subproblem is at least a factor $(1-\alpha)$ smaller than the original problem. The level of recursion in this approach is therefore at most $\log_{\frac{1}{1-\alpha}} n$.

Theorem 6. *The above algorithm is a factor $O(\log^2 n)$ approximation algorithm for the minimum cut linear arrangement problem.*

Proof.

$$\begin{aligned} C(G) &\leq \max(C(G|_S), C(G|_{\overline{S}})) + Cost(S) \\ &= \max(C(G|_S), C(G|_{\overline{S}})) + O(\log n) OPT_{\beta}(G) \\ &\leq \max(C(G|_S), C(G|_{\overline{S}})) + O(\log n) OPT_{1/2}(G) \\ &\leq \max(C(G|_S), C(G|_{\overline{S}})) + O(\log n) OPT_{MCLA}(G) \end{aligned}$$

To see why $OPT_{\beta=1/2}(G) \leq OPT_{MCLA}(G)$, consider an arrangement π which realizes OPT_{MCLA}. We then have a trivial 1/2-balanced cut by taking $S = \{v \in V \mid \pi(v) \leq |V|/2\}$. The cost of this cut cannot be more than OPT_{MCLA}, because OPT_{MCLA} is an upper bound on the cost of any i-cut in this arrangement. $OPT_{\beta=1/2}(G)$ can be no more than the cost of this cut, because this is itself a 1/2-balanced cut, whose value is possibly greater than, but not less than the optimal 1/2-balanced cut.

Recall that the level of recursion is at most $\log_{\frac{1}{1-\alpha}} n$, so that expanding the recurrence relation, we get

$$\begin{aligned} C(G) &\leq O(\log_{\frac{1}{1-\alpha}} n) \cdot O(\log n) \cdot OPT_{MCLA} \\ &= O(\log^2 n) OPT_{MCLA} \end{aligned}$$

□

One final note is that the sparsest cut problem can be approximated with factor $O(\sqrt{\log n})$ by using semi-definite programming. Thus, the approximation factor of the minimum cut linear arrangement problem can be reduced to $O(\log^{3/2} n)$.