An Improved Approximation Algorithm for Multiway Cut
Gruia Calinescu Howard Karlo Yuval Rabani
Final report for CS 6150

Gurupragaash Annasamy Mani   Praveen Thiraviya Rathinam

December 18, 2015

1   Multiway Cut

Consider an undirected graph G = (V, E) with vertices V = $\{1, 2,\ldots, n\}$ and edges E with weights w: E $\rightarrow R^+$. Let T = $\{1, 2, ..., k\} \subseteq$ V be a set of terminals. Multiway Cut is the problem of finding a minimum cost cut C $\subseteq$ E such that no connected component of G(V, E - C) contains two terminals from T. In case of just two terminals, minimum multiway cut is just a min s-t cut problem and hence can be solved in polynomial time by running a maximum flow algorithm from one terminal to the other. But when the number of terminals becomes ≥ 3, then the problem of computing the minimum weight multiway cut is NP-hard and max SNP-hard even for fixed $k \geq 3$ by Dahlhaus, Johnson, Papadimitriou, Seymour and Yannakakis. In other words, there is a constant $\delta > 1$ such that it is NP-Hard to even approximate the solution to within a ratio of less than δ. Unless P = NP, there is no polynomial-time approximation scheme for Multiway Cut.

2   APX-Hard/SNP-Hard

APX(Approximable) class is a set of NP optimization problems in which algorithms can be used to find an answer within some fixed multiplicative factor of the optimal answer. APX class is also known as MAX-SNP. A problem is said to be SNP-hard if it does not have a Polynomial Time Approximation Scheme(PTAS).

3   Related Work

3.1   Isolation Heuristic

Dahlhaus et al.[6] initiated the study of multiway cut problems. They proposed a simple combinatorial isolation heuristic to approximate the solution to these kinds of problems.In isolation heuristic, given a undirected weighted graph G = (V,E) and K terminals, in each iteration we attach one terminal to the source and all the other terminals to the sink and then run a max-flow algorithm and find the min-cut for each turn. Let the edge set after each iteration be E_i. Now the lowest k-1 cuts are taken and the union of them gives the multi-way cut. This algorithm gives an approximation of $2(1 - \frac{1}{k})$.

Proof:

- Run the isolation heuristic scheme and get the set of edges. Let they be A.

- Let E^* be the optimal edge set for the multiway cut with K terminals. Then it means if E^* is removed then there will be K disjoint connected graphs $(V_1, V_2, \cdots, V_k)$, each having one terminal respectively.

- $E^* = \sum_{i=1}^{k} E_i^*$ and each E_i^* represents the edges removed to disconnect V_i from rest.

- Let $\delta(V_i)$ denote the set of all outgoing edges from V_i.

- Now we can say that, $w(E_i) \leq w(\delta(V_i))$ because both isolate the terminal i from the rest and we know E_i is the mincut. So $w(E_i)$ cannot be greater than $w(\delta(V_i))$

- Lets says there are m edges between V_i and V_{i+1}. Then both $\delta(V_i)$ and $\delta(V_{i+1})$ will include those m edges.

- $2w(E^*) = \sum_{i=1}^{k} \delta(V_i)$. It is 2 times since each edge is double counted in the process.

- We know $w(A) \leq (1 - \frac{1}{k})(\sum_{i=1}^{k} w(E_i^*))$. This is because, A is union of first K-1 smallest set. In this expression, we are adding up all the K values. Since only $k - 1$ values are taken, we are multiplying it with $1 - \frac{1}{k}$ and since it the union of all these set, it will be equal to or less than the summation of indiviual sets.

$$w(A) \leq (1 - \frac{1}{k})(\sum_{i=1}^{k} w(E_i^*))$$
$$\leq (1 - \frac{1}{k})(\sum_{i=1}^{k} w\delta(V_i))$$
$$\leq 2(1 - \frac{1}{k})w(E^*)$$
$$w(A) \leq 2(1 - \frac{1}{k})OPT$$

3.1.1 Alon's Improvement

Noga Alon [5] observed that for the special cases of k = 4 and k = 8 improvements can be obtained using a variant of the isolation heuristic. For k = 4, the Isolation Heuristic provides a guarantee of 3/2. An improved guarantee of 4/3 can be obtained as follows: For each partition of the terminals into sets S_1, S_2 of size two, use max flow techniques to compute the minimum cut that separates the terminals in S_1 from those in $S2$. Output the union of the two best such cuts. The reader can readily verify that this union is a 4-way cut whose weight is at most 4/3 optimal. This approach requires only three max flow computations versus the four needed by the Isolation Heuristic, so it is faster as well.For k = 8, the guarantee of our theorem can be improved from 7/4 to 12/7. Unfortunately, the above approach did not yield improvements over the Isolation Heuristic for any values of k other than 4 and 8.

3.2 Greedy Split Algorithm

There is another greedy algorithm technique that can be used to solve the multiway cut problem. This algorithm also has an approximation ratio of $2 - \frac{2}{k}$. The algorithm is as follows:

- Find the cheapest cut that splits G into 2 components such that each contains atleast a terminal
- Find the cheapest cut dividing the 2 components such that each of the 3 components contains atleast a terminal
- Find the cheapest cut dividing the 3 components such that each of the 4 components contains atleast a terminal
- The steps are repeated until each of the k components contains a terminal. At each step, the algorithm chooses the cheapest cut among all components.

3.3 Polyhedral Approach

Chopra et al.[2, 3] and Cunningham[4] investigated on solving multiway cut problem using a polyhedral approach.But these two approaches too had an approximation ratio of $2(1-\frac{1}{k})$.

Chopra et al. proposed an integer formulation and studied the associated polyhedron. They further proposed an extended formulation which was tighter than all known solution at that time and also observed that when the underlying graph is a tree, the multiway cut problem can be solved in linear time by a straightforward dynamic programming algorithm.

Cunningham showed that for one particular formulation of the problem, the value of the minimum multiway cut is almost twice of its linear relaxation.

3.4 Non-linear Formulation

Bertsimas et al.[1] proposed a non-linear formulation of the multiway cut. Here the optimal solution formulated was an integral one. They suggested several polynomial time-solvable relaxations and gave a simple randomized rounding argument yielding the same approximation ratio of $2(1 - \frac{1}{k})$

4 Preliminaries of Proposed Solution

4.1 Basic Notations

4.1.1 Simplex Solution

A simplex is a generalization of the notion of a convex polyhedron to arbitrary dimensions. In our case, we solve the linear programs using the simplex formulation. A k dimension simplex has $k+1$ vertices. Linear programs can be solved using the simplex approach.Linear program operates on simplicial cones, and these become proper simplices with an additional constraint. The simplicial cones in question are the corners (i.e., the neighborhoods of the vertices) of a geometric object called a polytope. The shape of this polytope is defined by the constraints applied to the objective function. For solving the multiway cut, we keep the k terminals as vertices and develop a $k - 1$ dimensional convex polytope given by $\{x \in \mathbb{R}^k |\ (x \geq 0) \wedge (\sum_i x_i = 1)\}$

4.1.2 L1 Norm

L1-norm is also known as Mean-Absolute Error(MAE) or Sum of Absolute Difference(SAD). It is basically used to find the sum of the all the value of a variable. L1 norm of x is basically denoted by $||x||$. It can also be used to find the difference between two variables

$$\text{SAD}(x_1, x_2) = ||x_1 - x_2|| = \sum |x_1 - x_2|$$

4.1.3 Unit Vector

Unit vectors are used for denoting the spatial direction and generally represent the axes of a cartesian co-ordinate system. In our case, the unit vectors are used to represent the position of the vertices of the graph on the $k-1$ dimensional simplex. For j = 1, 2,...,k, $e^j \in \mathbb{R}$ denotes the unit vector given by $(e^j)_j = 1$ and $(e^j)_i = 0$ for all i $\notin$ j.

4.1.4 Semimetric

Semimetric is a pair (V, d), where V is a set and d is a function which operates on V such that $d : V \times V \to \mathbb{R}$, $\forall u, v \in V$.

$$d(u, v) = d(v, u) \geq 0$$
$$d(u, u) = 0$$
$$d(u, v) \leq d(u, w) + d(w, v)$$

5 Linear Programmning Relaxations

5.1 LP1 and LP2

Minimize $\quad \sum_{uv \in E} c(u, v) \, d(u, v)$ is the objective function

such that $\quad (V, d)$ is a semimetric $\qquad\qquad (1)$

$\qquad\qquad d(t_1, t_2) = 1 \quad \forall t_1, t_2 \in T, t_1 \neq t_2 \qquad (2)$

$\qquad\qquad d(u, v) \in (0, 1) \forall u, v \in V \qquad\qquad (3)$
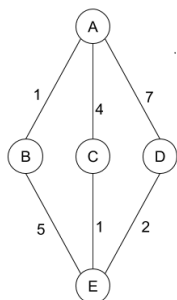
Consider the graph in Figure 1. Let nodes A and E be terminals and we want to find a cut. In the LP, all the $d(u, v)$ where uv is an valid edges is the decision variable and the value of it can be 0 or 1. From equation 3, we can say that $d(A, E) = 1$ and we don't know the other values. But based on equation 7, we can write $d(A, E) \leq d(A, B) + d(B, E)$, $d(A, E) \leq d(A, C) + d(C, E)$ and $d(A, E) \leq d(A, D) + d(B, D)$. These contribute to the additional constraints which are derived

Figure 1: Sample Graph

from the constraint (V, d) is a semimetric.Now the LP can should assign atleast value 1 to one of the pairs in $(d(A, B), d(B, E)), d(A, C), d(C, E)$ and $d(A, D) + d(B, D)$. Assign the d(u, v) with a value 1 means, selecting the edge for the cut. Since our objective function is to minimize it, the LP will find which edge costs minimum and that will be the optimal solution. The authors want to come up with much more stricter constraints to reduce the number of valid solution. Lets look into the constraint $\sum_{t \in T} d(u, t) = k - 1 \quad \forall u \in V$ using the example from the Figure.In this example, the nodes A, B and C are terminals. If we run our LP without the new constraints, there are four valid solutions, which represent the cuts (AD, DB), (AD, DC), (DB, DC) and (AD, DB, DC). In this set of values, (AD, DB, DC) is not a optimal one but still is a valid solution. In a large graph, there could be many such solution, which could increase the number of valid solution. But this one can be removed easily with the new constraint, $\sum_{t \in T} d(u, t) = k - 1 \quad \forall u \in V$. The new constraint says, if there is a node u, as long as its not connected to more than one terminal it is fine. So if there are k nodes, then $\sum_{t \in T} d(u, t) = k - 1$, which means only $K - 1$ cuts are required and not K cuts

5.2 Successive Linear Programming

- As discussed earlier, any valid solution will split the Graph $G(V, E)$ into K graphs $(C_1, C_2, \cdots, C_k)$, with each C_i having a terminal S_i
- Let $\delta(C_i)$ represent the set of edges which goes out of C_i
- Each vertex $v \in V$ is represented as x_i^j and this value is set to 1 if the ith vertex is part of the component C_i, else its set to 0. Since a vertex can be a part of only one component, we can say that $\sum_{j=1}^{k} x^j = 1, \forall x \in V$,
- We define another notation z_e^i which operates on all the edges in E and is set to 1 if the edge e is in $\delta(C_i)$ else its set to 0
- If e is in the $\delta(C_i)$, then it connect (u, v) where $u \in C_i, v \notin C_i$. Thus we can write z_e^i as $z_e^i = x_u^i - x_v^i$, since x_u^i will be 1 and x_v^i will be 0, as per the definition of x. If e is not in the $\delta(C_i)$, then both u and v belong to a same component, then both x_u^i and x_v^i will be 1 if both belong to component C_i else both will be 0.

- Now for the objective function is $\frac{1}{2} \sum_{e \in E} c_e * \sum_{i=1}^{k} z_e^i$, where c_e is the cost/weight of the edge. We are multiplying it with $\frac{1}{2}$ because of the double counting which we discussed in the previous explanation.

$$\text{Minimize} \quad \frac{1}{2} \sum_{e \in E} c_e * \sum_{i=1}^{k} z_e^i$$

$$\text{such that} \quad z_e^i = |x_u^i - x_v^i|, \quad \forall e \in E$$

$$\sum_{i=1}^{k} x_v^i = 1, \quad \forall v \in V$$

$$x_{s_i}^i = 1, \quad \forall s_i \in T \text{ (Set of Terminals)}$$

$$x_v^j \geq 0, \quad \forall v \in V \quad \text{(Got by adding LP relaxation to } x_v^j \in (0, 1), \quad \forall v \in V)$$

6 Rounding Algorithm

The authors propose a randomized rounding algorithm to find the multiway cut with cost within a factor of $1.5 - \frac{1}{k}$ of the optimal solution. Take an optimal solution to the relaxation with edges whose endpoints differ in at most two coordinates, and let OPT denote its cost. Once we find the multiway cut of expected cost, we can use lemma 2 and proposition 3 and extend it to general cases.

Let $E_i = (u, v) \in E \mid x_i^u \neq x_j^v$. This is possible since two vertices of the edges lie in two different sets. Let $W_i = \sum_{e \in \epsilon_i} c(e).d(e)$. Without loss of generality, assume that W_k is the greatest of $W_1, \ldots, W_k$. The algorithm is used to get an integral solution for the simplex in which all the terminals are present as vertices. We define B(i,ρ) to be $\{u \in V \mid x_i^u > 1 - \rho\}$ and it contains the set of nodes which are closer to a terminal i in the simplex. Value of ρ lies in thr range (0,1). Here B represents a ball around each terminal i having a radius ρ and our intention is to find how the various nodes in the graph lie in the simplex using the algorithm.

The algorithm operates as follows. First, pick ρ at random in (0, 1) and an ordering σ from $(1, 2, \ldots, k-1, k)$ and (k - 1, k - 2, $\ldots$, 1, k). Then partition V into $V_1, \ldots, V_k$ as follows. Proceed in the order given by σ. Each V_i should contain all vertices in B(i, ρ) that have not already been assigned to a previous V_i. At the end, assign all unused vertices to V_k. The sets $V_1, \ldots, V_k$ are the components after removing the cut, and edges between vertices in two different sets are in the cut.More formally, the algorithm is:

- Compute an optimal solution to relaxation

- Renumber the terminals so that Wk is largest among $W_1, \ldots, W_k$.

- Pick uniformly at random $\rho \in (0, 1)$ and $\sigma \in (1, 2, \ldots, k-1, k)$ or $(k-1, k-2, \ldots, 1, k)$. Here k is used as the overflow bin and is used only if you are not able to assign a node to any of the k-1 terminals in the simplex. Initially all the nodes are placed in the overflow bin and then the below steps are run.

- For j = 1 to k 1: $V_{\sigma_j} \leftarrow B(j, \rho)$ - $\bigcup_{i:i<j} V_{\sigma_i}$. Here basically we assign a node to a terminal only if they are not assigned to any of the previously traversed terminal regions.

- $V_k \leftarrow V - \bigcup_{i<k} V_i$. In this step, we assign all the remaining unassigned vertices to the overflow bin V_k once the above loop is complete.

- Let C be the set of edges that run between sets in the partition $V_1, \ldots, V_k$.

7 Subdivisions

Lets say we have a graph $G(V, E)$ with K terminals and $c(u, v)$ representing the cost of the edge connecting nodes(u,v). We can create a graph G' from G, by spliting the edges across (u, v), into two edges by introducing a node in the edge. The new edges are (u, w) and (w, v) and w is a new node which is a normal node and not a terminal. Now we

assign $c(u,v)$ to both $c(u,w), c(w,v)$ and the edge (u,v) is removed. We do this process some finite number of times and now we have a graph $G'(V', E')$ which is constructed from $G(V, E)$. The authors claim that if G' has a multiway cut C' of cost Z, then we can construct a cut C for G which has a cost atmost Z. We can do this by, looking into the edge in C'. $\forall e in C'$ check if $e in E$, then add it to C. If $e \notin E$ find the edge $e' \in E$ which was split to get $e \in E'$ and add e' to C. This set C is the multiway cut for G.

8 Why subdivision

Like we described earlier, for the algorithm to work, the nodes in the graph should have the property(the property where they differ by atmost 2). But the graphs normally do not have this property. We induce this property into the graph with the help of subdivision method which we discussed previously. We pick an edge, check if the vectors of nodes(u,v) from the edge differ by more than two. If so we introduce a new node w. Now the distance between (u,v) is split into two (u,w) and (w,v). These two edges will further be split if they do not follow the property. This process is continued till all the edges in the graph satisfy the property. Worst case it will be k such divisions for each edge and thereby introducing a max of $k|E|$ nodes in the graph. If we run the algorithm in the system we will can get a solution which is $(1.5 - \frac{1}{k})OPT$. This is what is discussed in Proposition 2 and Lemma 3 in the paper

9 Why the algorithm works

The edges in the graph can be split into two groups E_0 and E_{ij}. E_0 equals set of edges where $x^u = x^v$(u,v are the nodes connected by the edge). The other edges fall under E_{ij} where $(x_i^u \neq x_i^v x_j^u \neq x_j^v)$ which means, there are two terminals (i,j), associated with nodes of the edge. There are 4 values associated with this edge $x_i^u, x_j^u, x_i^v, x_j^v$. Lets find the max of the 4 and keep that as x_i^u and rearrange others accordingly. The authors claim that $\forall uv \in E_{ij}$, either $x_i^u \geq x_i^v \geq x_j^v \geq x_j^u$ or $x_i^u \geq x_j^u \geq x_i^v \geq x_j^u$. This is because we know $\sum_{i=1}^{k} x_i^u = 1 = \sum_{i=1}^{k} x_i^v$ and since they differ by at max two, we can write $x_i^u + x_j^u = x_i^v + x_j^v$. Since x_i^u is the max value, the min value will be x_j^u and the other two will lie in the the middle and thus the inequality.
Lets analyse the probability of an edge in the CUT from our algorithm.

- Edges $uv \in E \setminus \bigcup_{t:t<k} E_{tk}$. The edges which have $x_u = x_v$ will always be together no matter the value of σ, ρ(Assigned to the same terminal). So $\Pr(uv \in C) = 0, \quad \forall uv$ if$(x_u = x_v)$. Now for the edges in E_{ij}, $for all l \in 1, 2, 3 \cdots, k$ $i, j x_l^u = x_l^v$ and hence they will either be assigned to the same terminal or they wont be assigned to any of the terminal in l. So a cut is possible only when either of (u,v) is assigned to (i,j). This depends on the value of ρ. When $i \prec j$, from the inequality explained in the previous paragraph, we can say that if ρ is close to $1 - x_j^u$, then both the nodes u, v will be assigned to i since the circle is big enough to contain both, so $\Pr(C_{uv}) = 0$. If the value is in range $(1 - x_i^u, 1 - x_i^v]$, then the circle is small and hence only u will be assigned to i. When $j \prec i$, we dont have any restriction on the value of ρ. It can

taken any value in range $(1 - x_i^u, 1 - x_j^u]$ and still only v will be assigned to j. The value of ρ should be more than $1 - x_j^u$ for it to include u. Let $I_L = (1 - x_i^u, 1 - x_i^v]$ and $I_R = (1 - x_j^v, 1 - x_j^u]$.

$$\Pr[uv \in C] \leq \Pr[(j \prec i)(\rho \in I_L \cup I_R)] + \Pr[(i \prec j)(\rho \in I_L)] \tag{4}$$

$$\leq \Pr[(j \prec i)(\rho \in I_L)] + \Pr[(j \prec i)(\rho \in I_R)] + \Pr[(i \prec j)(\rho \in I_L)] \tag{5}$$

$$\leq \frac{1}{2}d(u,v) + \frac{1}{2}d(u,v) + \frac{1}{2}d(u,v) \quad \frac{1}{2} \text{ is the probability of } i \prec j \text{ and viceversa} \tag{6}$$

$$\leq 1.5d(u,v) \tag{7}$$

- Edges $uv \in \bigcup\limits_{t:t<k} E_{tk}$. Same explanation as about but here i will always preced k and hence one of them nodes must be assigned with i is processed.

$$\Pr[uv \in C] \leq \Pr[\rho \in I_L] \tag{8}$$

$$\leq d(u,v) \tag{9}$$

10 Proof

- Let Z_i be the weight of the edges present in C_i which is the set of edges leaving the connected component i.

$$Z_i = \sum_{uv \in \bigcup\limits_{t:t \neq i} E_{ti}} c(u,v)d(u,v) \tag{10}$$

- Find the Z_i which is greatest. Let it be Z_k. There we can write the above equation as

$$\sum_{uv \in \bigcup\limits_{t:t \neq i} E_{ti}} c(u,v)d(u,v) \leq k * Z_k \tag{11}$$

- From the LP's we know that

$$\sum_{i=1}^{k} Z_i = 2 * OPT \tag{12}$$

- From the above two equations we can say that, $Z_k \leq \frac{2}{k}OPT$

- From the analyis we can calculate the expected value of the cost of the edges as,

$$Expected[c(C)] = \sum_{uv \in E} c(u,v) \Pr(uv \in C)$$

$$\leq \sum_{uv \in E \setminus \bigcup_{t:t<k} E_{tk}} c(u,v) \Pr(uv \in C) + \sum_{uv \in \bigcup_{t:t<k} E_{tk}} c(u,v) \Pr(uv \in C)$$

$$\leq 1.5(OPT) + 0.5(Z_K)$$

$$\leq (1.5 + 0.5\frac{2}{k})OPT$$

$$\leq (1.5 + \frac{1}{k})OPT$$

References

[1] BERTSIMAS, D., TEO, C., AND VOHRA, R. V. Nonlinear formulations and improved randomized approximation algorithms for multiway and multicut problems. Working papers 3838-95., Massachusetts Institute of Technology (MIT), Sloan School of Management.

[2] CHOPRA, S., AND OWEN, J. Extended formulations for the a-cut problem. *Mathematical Programming 73*, 1 (1996), 7–30.

[3] CHOPRA, S., AND RAO, M. R. On the multiway cut polyhedron. *Networks 21*, 1 (1991), 51–89.

[4] CUNNINGHAM, W. The optimal multiterminal cut problem.

[5] DAHLHAUS, E., JOHNSON, D. S., PAPADIMITRIOU, C. H., SEYMOUR, P. D., AND YAN-NAKAKIS, M. The complexity of multiway cuts (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1992), STOC '92, ACM, pp. 241–251.

[6] DAHLHAUS, E., JOHNSON, D. S., PAPADIMITRIOU, C. H., SEYMOUR, P. D., AND YAN-NAKAKIS, M. The complexity of multiterminal cuts. *SIAM J. Comput. 23*, 4 (Aug. 1994), 864–894.