Multicut and Integer Multicommodity Flow in Trees

CS 602 Course Presentation

Yash Sadhwan(23M0818) and Hritiz Gogoi (22D1142)

Indian Institute of Technology Bombay

November 8, 2023

Presentation Outline

Introduction to Multicut problem

Approximation Algorithm

Introduction to Multicut problem

Introduction

- The multicut problem is a generalization of s-t mincut problem.

Introduction

- The multicut problem is a generalization of s-t mincut problem.
- In this class of problems we are given a Graph (V,E), capacities/weights on those graphs (c_j) and k pairs of source of terminals $(s_1, t_1), (s_2, t_2)...(s_k, t_k)$ where each pair is distinct.

Introduction

- The multicut problem is a generalization of s-t mincut problem.
- In this class of problems we are given a Graph (V,E), capacities/weights on those graphs (c_j) and k pairs of source of terminals $(s_1, t_1), (s_2, t_2)...(s_k, t_k)$ where each pair is distinct.
- The goal is to find a set of edges of minimum capacity such that removal of those edges result in separation of each s_i from corresponding t_i

Introduction

- The multicut problem is a generalization of s-t mincut problem.
- In this class of problems we are given a Graph (V,E), capacities/weights on those graphs (c_j) and k pairs of source of terminals $(s_1, t_1), (s_2, t_2)...(s_k, t_k)$ where each pair is distinct.
- The goal is to find a set of edges of minimum capacity such that removal of those edges result in separation of each s_i from corresponding t_i
- This problem is quite complex for arbitrary graphs so we will restrict it to trees.

Introduction

- The multicut problem is a generalization of s-t mincut problem.
- In this class of problems we are given a Graph (V,E), capacities/weights on those graphs (c_j) and k pairs of source of terminals $(s_1, t_1), (s_2, t_2)...(s_k, t_k)$ where each pair is distinct.
- The goal is to find a set of edges of minimum capacity such that removal of those edges result in separation of each s_i from corresponding t_i
- This problem is quite complex for arbitrary graphs so we will restrict it to trees.
- Even in trees the decision version of the problem is **NP-Hard**

Introduction

- The multicut problem is a generalization of s-t mincut problem.
- In this class of problems we are given a Graph (V,E), capacities/weights on those graphs (c_j) and k pairs of source of terminals $(s_1, t_1), (s_2, t_2)...(s_k, t_k)$ where each pair is distinct.
- The goal is to find a set of edges of minimum capacity such that removal of those edges result in separation of each s_i from corresponding t_i
- This problem is quite complex for arbitrary graphs so we will restrict it to trees.
- Even in trees the decision version of the problem is **NP-Hard**
- Multicut problem is used in network routing and network design including transportation and communication networks.

Formulation

- Denote by p_i unique path from s_i to t_i (unique by property of trees)

Formulation

- Denote by p_i unique path from s_i to t_i (unique by property of trees)
- We notice that for s_i to be disconnected from t_i we must choose to remove atleast one edge in path p_i.

Formulation

- Denote by p_i unique path from s_i to t_i (unique by property of trees)
- We notice that for s_i to be disconnected from t_i we must choose to remove atleast one edge in path p_i.
- So we formulate the problem as an integer program (IP):

Formulation

- Denote by p_i unique path from s_i to t_i (unique by property of trees)
- We notice that for s_i to be disconnected from t_i we must choose to remove atleast one edge in path p_i.
- So we formulate the problem as an integer program (IP):
- $$\begin{aligned} \min \Sigma_{e \in E} c_e x_e \\ \text{s.t.} \Sigma_{e \in p_i} x_e \geq 1 \ i = 1, 2 \ldots k \\ x_e \in \{0, 1\} \end{aligned} \tag{1}$$

Formulation

- We can relax the IP to a Linear Program (LP) as

Formulation

- We can relax the IP to a Linear Program (LP) as
-

$$
\begin{aligned}
\min \Sigma_{e \in E} c_e x_e \\
\text{s.t.} \Sigma_{e \in p_i} x_e \geq 1 \; i = 1, 2 \ldots k \\
x_e \geq 0 \; e \in E
\end{aligned} \tag{2}
$$

Formulation

- We can relax the IP to a Linear Program (LP) as
- $$\begin{aligned} \min \Sigma_{e \in E} c_e x_e \\ \text{s.t.} \Sigma_{e \in p_i} x_e \geq 1 \; i = 1, 2 \ldots k \\ x_e \geq 0 \; e \in E \end{aligned} \tag{2}$$
- Let us formulate the dual of this Linear Program

$$\begin{aligned} \max \Sigma_{i=1}^{k} f_i \\ \text{s.t.} \Sigma_{e \in p_i} f_i \leq c_e \; e \in E \\ f_i \geq 0 \; i = 1, 2 \ldots k \end{aligned} \tag{3}$$

Formulation

- This dual can be interpreted as sending a flow through each path such that sum of total flow passing through an edge does not exceed its capacity.

Formulation

- This dual can be interpreted as sending a flow through each path such that sum of total flow passing through an edge does not exceed its capacity.
- This is precisely the LP relaxation version of multicommodity flow problem.

Formulation

- This dual can be interpreted as sending a flow through each path such that sum of total flow passing through an edge does not exceed its capacity.
- This is precisely the LP relaxation version of multicommodity flow problem.
- To put in other words, if each (s_i, t_i) pair is assigned its own commodity then the goal is to maximize the total flow of the system

Approximation Algorithm

Primal Dual Schema

- We will construct a primal-dual schema based algorithm that will find a multicut and multicommodity flow that are within a factor 2 of each other.

Primal Dual Schema

- We will construct a primal-dual schema based algorithm that will find a multicut and multicommodity flow that are within a factor 2 of each other.
- We will treat multicut as the primal problem and ensure primal complementary slackness, $\alpha = 1$, while relaxing dual complementary slackness, $\beta = 2$.

Primal Dual Schema

- We will construct a primal-dual schema based algorithm that will find a multicut and multicommodity flow that are within a factor 2 of each other.
- We will treat multicut as the primal problem and ensure primal complementary slackness, $\alpha = 1$, while relaxing dual complementary slackness, $\beta = 2$.
- Primal complementary slackness condition is: $x_e(\Sigma_{e \in p_i} f_i - c_e) = 0 \forall e \in E$

Primal Dual Schema

- We will construct a primal-dual schema based algorithm that will find a multicut and multicommodity flow that are within a factor 2 of each other.
- We will treat multicut as the primal problem and ensure primal complementary slackness, $\alpha = 1$, while relaxing dual complementary slackness, $\beta = 2$.
- Primal complementary slackness condition is: $x_e(\Sigma_{e \in p_i} f_i - c_e) = 0 \forall e \in E$
- Either an edge is assigned zero or its capacity is tight.

Primal Dual Schema

- We will construct a primal-dual schema based algorithm that will find a multicut and multicommodity flow that are within a factor 2 of each other.
- We will treat multicut as the primal problem and ensure primal complementary slackness, $\alpha = 1$, while relaxing dual complementary slackness, $\beta = 2$.
- Primal complementary slackness condition is:
  $x_e(\Sigma_{e \in p_i} f_i - c_e) = 0 \forall e \in E$
- Either an edge is assigned zero or its capacity is tight.
- Dual complementary slackness condition is:
  $f_i(\Sigma_{e \in p_i} x_e - 1) = 0$ which is $f_i \neq 0 \implies \Sigma_{e \in p_i} x_e = 0$
  We relax this condition to be
  $f_i \neq 0 \implies \Sigma_{e \in p_i} x_e \leq 2$

Primal Dual Schema

- This schema ensures that relaxed optimal multicut is atmost twice of the minimum multicut and relaxed dual flow is atleast half of maximum flow.

Primal Dual Schema

- This schema ensures that relaxed optimal multicut is atmost twice of the minimum multicut and relaxed dual flow is atleast half of maximum flow.
- We will now construct an algorithm to implement the primal dual schema.

The algorithm

- **Initialization:** $\mathsf{f} \longleftarrow 0$; $\mathsf{D} \longleftarrow \phi$

The algorithm

- **Initialization:** f $\longleftarrow$ 0; D $\longleftarrow \phi$
- **Flow Routing:** For each vertex v, in non increasing order of depth, do: For each pair (s_i, t_i) such that $\mathrm{lca}(s_i, t_i) =$ v, greedily route integral flow from s_i to t_i Add to D all edges that were saturated in the current iteration in arbitrary order.

The algorithm

- **Initialization:** f $\longleftarrow$ 0; D $\longleftarrow \phi$
- **Flow Routing:** For each vertex v, in non increasing order of depth, do: For each pair (s_i, t_i) such that lca$(s_i, t_i) =$ v, greedily route integral flow from s_i to t_i Add to D all edges that were saturated in the current iteration in arbitrary order.
- Let $e_1, e_2, ...e_l$ be ordered list of edges in D.

The algorithm

- **Initialization:** f $\longleftarrow$ 0; D $\longleftarrow$ ϕ
- **Flow Routing:** For each vertex v, in non increasing order of depth, do: For each pair (s_i, t_i) such that lca(s_i, t_i) = v, greedily route integral flow from s_i to t_i Add to D all edges that were saturated in the current iteration in arbitrary order.
- Let $e_1, e_2, ...e_l$ be ordered list of edges in D.
- **Reverse Delete:** For j = l downto 1 do:
  If D - { e_j } is a multicut in G, then D $\longleftarrow$ D - { e_j }

The algorithm

- **Initialization:** f $\longleftarrow$ 0; D $\longleftarrow$ ϕ
- **Flow Routing:** For each vertex v, in non increasing order of depth, do: For each pair (s_i, t_i) such that lca(s_i, t_i) = v, greedily route integral flow from s_i to t_i Add to D all edges that were saturated in the current iteration in arbitrary order.
- Let $e_1, e_2, ...e_l$ be ordered list of edges in D.
- **Reverse Delete:** For j = l downto 1 do:
  If D - { e_j } is a multicut in G, then D $\longleftarrow$ D - { e_j }
- Output the flow and multicut D

The algorithm

- **Lemma:** Let (s_i, t_i) be a pair with nonzero flow, and let $\text{lca}(s_i, t_i) = \text{v}$. Atmost one edge is picked in the multicut from each of the two paths, s_i to v and t_i to v

The algorithm

- **Lemma:** Let (s_i, t_i) be a pair with nonzero flow, and let $\text{lca}(s_i, t_i) = \text{v}$. Atmost one edge is picked in the multicut from each of the two paths, s_i to v and t_i to v
- **Proof idea:** Let for contradiction, there exists two edges e_1 and e_2 from s_i to v where e_2 is deeper edge. We argue that there must be another path through e_2 say s_j to t_j for which e_2 is the only edge of D. Argue that D must contain an edge from s_j to t_j path which cannot be e_2 arriving at a contradiction.

The algorithm

- **Lemma:** Let (s_i, t_i) be a pair with nonzero flow, and let $\text{lca}(s_i, t_i) = \text{v}$. Atmost one edge is picked in the multicut from each of the two paths, s_i to v and t_i to v
- **Proof idea:** Let for contradiction, there exists two edges e_1 and e_2 from s_i to v where e_2 is deeper edge. We argue that there must be another path through e_2 say s_j to t_j for which e_2 is the only edge of D. Argue that D must contain an edge from s_j to t_j path which cannot be e_2 arriving at a contradiction.
- **Theorem:** The algorithm achieves approximation guarantees of factor 2 for minimum multicut and factor $\frac{1}{2}$ for maximum integer multicommodity flow problem on trees

The algorithm

- **Lemma:** Let (s_i, t_i) be a pair with nonzero flow, and let $\text{lca}(s_i, t_i) = \text{v}$. Atmost one edge is picked in the multicut from each of the two paths, s_i to v and t_i to v
- **Proof idea:** Let for contradiction, there exists two edges e_1 and e_2 from s_i to v where e_2 is deeper edge. We argue that there must be another path through e_2 say s_j to t_j for which e_2 is the only edge of D. Argue that D must contain an edge from s_j to t_j path which cannot be e_2 arriving at a contradiction.
- **Theorem:** The algorithm achieves approximation guarantees of factor 2 for minimum multicut and factor $\frac{1}{2}$ for maximum integer multicommodity flow problem on trees
- **Proof idea:** D is a multicut since it contains atleast one saturated edge for the path from any s_i to t_i. From Lemma atmost two edges are picked in the multicut from each path carrying non zero flow satisfying relaxed dual complementary slackness. Argue from previous lemma that capacity of multicut is atmost twice flow. Claim follows from feasible flow $\leq$ max flow $\leq$ min cut $\leq$ feasible cut.

Thank You!