

Strongly Polynomial Algorithms for the Unsplittable Flow Problem [1]

Yossi Azar and Oded Regev

Presentation by Tomer Greenberg and Boris Kapchits

September 18, 2007

Introduction

Problem definition

Earlier results

Additional problem
variations

Algorithms for UFP

Classical UFP - Strongly
Polynomial

Additional results

Introduction

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

Input:

- a possibly directed graph $G = (V, E)$
- a capacity function on edges
- a set of requests which of them is a quadruple, consisting of two terminals, s_i and t_i , a demand d_i and profit r_i .

Output:

- sub-set of requests, and a unsplitted flow between the terminals of each request, such that the capacity constrain is fully met.

Optimal solution maximizes the sum of profits of the routed requests.

The problem is known to be NP-hard.

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- The private case of the problem is the problem of *Edge Disjoint Path*, EDP, where all the demands, profits and capacities are equal to 1. However, this problem is also NP-complete.
- Most of the results for UFP deal with the classical case, where the maximum demand of a request is not greater than the minimum edge capacity, $d_{max} \leq u_{min}$.
- The most popular approach is an LP rounding, which achieves a $O(\sqrt{m})$ approximation.
- There were numerous attempts to find a combinatorial algorithm, but for the best of authors knowledge, non prevailed. Some of the previous works have failed to achieve the $O(\sqrt{m})$ approximation ratio and some solved only the restricted version of the problem.

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- The private case of the problem is the problem of *Edge Disjoint Path*, EDP, where all the demands, profits and capacities are equal to 1. However, this problem is also NP-complete.
- Most of the results for UFP deal with the classical case, where the maximum demand of a request is not greater than the minimum edge capacity, $d_{max} \leq u_{min}$.
- The most popular approach is an LP rounding, which achieves a $O(\sqrt{m})$ approximation.
- There were numerous attempts to find a combinatorial algorithm, but for the best of authors knowledge, non prevailed. Some of the previous works have failed to achieve the $O(\sqrt{m})$ approximation ratio and some solved only the restricted version of the problem.

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- The private case of the problem is the problem of *Edge Disjoint Path*, EDP, where all the demands, profits and capacities are equal to 1. However, this problem is also NP-complete.
- Most of the results for UFP deal with the classical case, where the maximum demand of a request is not greater than the minimum edge capacity, $d_{max} \leq u_{min}$.
- The most popular approach is an LP rounding, which achieves a $O(\sqrt{m})$ approximation.
- There were numerous attempts to find a combinatorial algorithm, but for the best of authors knowledge, non prevailed. Some of the previous works have failed to achieve the $O(\sqrt{m})$ approximation ratio and some solved only the restricted version of the problem.

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- The private case of the problem is the problem of *Edge Disjoint Path*, EDP, where all the demands, profits and capacities are equal to 1. However, this problem is also NP-complete.
- Most of the results for UFP deal with the classical case, where the maximum demand of a request is not greater than the minimum edge capacity, $d_{max} \leq u_{min}$.
- The most popular approach is an LP rounding, which achieves a $O(\sqrt{m})$ approximation.
- There were numerous attempts to find a combinatorial algorithm, but for the best of authors knowledge, non prevailed. Some of the previous works have failed to achieve the $O(\sqrt{m})$ approximation ratio and some solved only the restricted version of the problem.

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- The private case of the problem is the problem of *Edge Disjoint Path*, EDP, where all the demands, profits and capacities are equal to 1. However, this problem is also NP-complete.
- Most of the results for UFP deal with the classical case, where the maximum demand of a request is not greater than the minimum edge capacity, $d_{max} \leq u_{min}$.
- The most popular approach is an LP rounding, which achieves a $O(\sqrt{m})$ approximation.
- There were numerous attempts to find a combinatorial algorithm, but for the best of authors knowledge, non prevailed. Some of the previous works have failed to achieve the $O(\sqrt{m})$ approximation ratio and some solved only the restricted version of the problem.

The paper is the first to present a combinatorial algorithm with approximation factor of $O(\sqrt{m})$ for the classical UFP problem, which is also the first strongly polynomial algorithm for this problem.

Additional problem variations

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- Extended UFP, where the capacities and demands can be arbitrary.

Additional problem variations

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- Extended UFP, where the capacities and demands can be arbitrary.
- Bounded UFP, where $d_{max} \leq \frac{1}{K} u_{min}$.

Additional problem variations

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- Extended UFP, where the capacities and demands can be arbitrary.
- Bounded UFP, where $d_{max} \leq \frac{1}{K} u_{min}$.
- Online algorithm for UFP , where the requests arrive one by one, while the graph is given in advance.

Additional problem variations

Introduction

Problem definition

Earlier results

Additional problem variations

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

- Extended UFP, where the capacities and demands can be arbitrary.
- Bounded UFP, where $d_{max} \leq \frac{1}{K} u_{min}$.
- Online algorithm for UFP , where the requests arrive one by one, while the graph is given in advance.

The authors propose algorithms to all the problem variations, however, in this presentation we will concentrate on the classical case.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Algorithms for UFP

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

■ $G = (V, E), |V| = n, |E| = m$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- $G = (V, E), |V| = n, |E| = m$
- Capacity function on edges is denoted by $u : E \rightarrow R^+$.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- $G = (V, E), |V| = n, |E| = m$
- Capacity function on edges is denoted by $u : E \rightarrow R^+$.
- Input request j is a quadruple (s_j, t_j, d_j, r_j) where s_j and t_j , are the pair of terminals, d_j is demand and r_j is profit.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- $G = (V, E)$, $|V| = n$, $|E| = m$
- Capacity function on edges is denoted by $u : E \rightarrow R^+$.
- Input request j is a quadruple (s_j, t_j, d_j, r_j) where s_j and t_j , are the pair of terminals, d_j is demand and r_j is profit.
- T is the set of requests, $|T| = l$.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- $G = (V, E)$, $|V| = n$, $|E| = m$
- Capacity function on edges is denoted by $u : E \rightarrow R^+$.
- Input request j is a quadruple (s_j, t_j, d_j, r_j) where s_j and t_j , are the pair of terminals, d_j is demand and r_j is profit.
- T is the set of requests, $|T| = l$.
- In those algorithms, where the requests are ordered, $L_j(e)$ denotes the relative load of an edge e after request j is routed, namely the flow through the edge divided by its capacity.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- $G = (V, E)$, $|V| = n$, $|E| = m$
- Capacity function on edges is denoted by $u : E \rightarrow R^+$.
- Input request j is a quadruple (s_j, t_j, d_j, r_j) where s_j and t_j , are the pair of terminals, d_j is demand and r_j is profit.
- T is the set of requests, $|T| = l$.
- In those algorithms, where the requests are ordered, $L_j(e)$ denotes the relative load of an edge e after request j is routed, namely the flow through the edge divided by its capacity.

The following lemma will be used further:

Lemma 1. *Given a sequence $\{a_1, a_2, \dots, a_n\}$, a non-increasing non-negative sequence $\{b_1, b_2, \dots, b_n\}$ and two sets $X, Y \subseteq \{1, \dots, n\}$, let $X^i = X \cap \{1, \dots, i\}$ and $Y^i = Y \cap \{1, \dots, i\}$. If for every $1 \leq i \leq n$ holds that $\sum_{j \in X^i} a_j > \alpha \sum_{j \in Y^i} a_j$ then $\sum_{j \in X} a_j b_j > \alpha \sum_{j \in Y} a_j b_j$*

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- For a request j , routed through path P the authors define

$$F(j, P) = \frac{r_j}{d_j \sum_{e \in P} \frac{1}{u(e)}}$$

- The lower bound on F is $\alpha_{min} = \frac{r_{min}}{n}$. This is a lower bound, since $u(e)$ is always greater than d and the path length is at most n .
- The upper bound on F is $\alpha_{max} = \frac{r_{max} u_{max}}{d_{min}}$.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- For a request j , routed through path P the authors define

$$F(j, P) = \frac{r_j}{d_j \sum_{e \in P} \frac{1}{u(e)}}$$

- The lower bound on F is $\alpha_{min} = \frac{r_{min}}{n}$. This is a lower bound, since $u(e)$ is always greater than d and the path length is at most n .
- The upper bound on F is $\alpha_{max} = \frac{r_{max} u_{max}}{d_{min}}$.
- The requests are divided into two sub-sets, according to their demand:
 - $T_1 = \{j | d_j \leq u_{min}/2\}$
 - $T_2 = \{j | d_j > u_{min}/2\}$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

PROUTE

run $Routine_2(T_1)$, run $Routine_2(T_2)$ and choose the better solution.

Routine₂(S):

for every $k = \lfloor \log \alpha_{min} \rfloor$ to $\lceil \log \alpha_{max} \rceil$

run $Routine_1(2^k, S)$ and remember the best of all runs.

Routine₁(α, S):

sort the requests in S according to non increasing order of r_j/d_j .

for all $j \in S$, according to above order

if \exists path P from s_j to t_j such that $F(j, P) > \alpha$ and the request can be routed on P without violating the capacity constrain

route the request j on P .

else

reject j .

Algorithm approximation and complexity

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:

$$|E_{heavy}| \geq \sqrt{m}$$

Second case:

$$|E_{heavy}| < \sqrt{m}$$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Theorem 1. *Algorithm PROUTE is an $O(\sqrt{m})$ approximation algorithm for classical UFP.*

Algorithm running time is weakly polynomial

- *Routine*₁ only sorts the requests and looks for an appropriate path, which can be done using a shortest path algorithm – all these operations are polynomial.
- But this routine is involved $\lceil \log \alpha_{max} \rceil - \lfloor \log \alpha_{min} \rfloor$ times, which is approximately $\log \frac{\alpha_{max}}{\alpha_{min}}$ which is polynomial, but not strongly polynomial.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Let us consider the set of requests Q , routed by the optimal solution. Let us consider the sub-set of Q that included into T_1 and the sub-set of Q that included into T_2

- Let us denote by Q' the sub-set that achieves greater profit among $Q \cap T_1$ and $Q \cap T_2$ and by i' the appropriate index, either 1 or 2.
- Let α' denote the greatest α such that $r(\{j \in Q' | F(j, Q_j) > \alpha'\}) \geq r(Q)/4$.
- Let $Q'_{high} = \{j \in Q' | F(j, Q_j) > \alpha'\}$
- Let $Q'_{low} = \{j \in Q' | F(j, Q_j) \leq 2\alpha'\}$

Note that both $r(Q'_{high}) \geq \frac{r(Q)}{4}$ and $r(Q'_{low}) \geq \frac{r(Q)}{4}$

We also can express $r(Q'_{low})$ using the definition of F :

$$\begin{aligned} r(Q'_{low}) &= \sum_{j \in Q'_{low}} r(j) = \sum_{j \in Q'_{low}} (F(j, Q_j) d_j \sum_{e \in Q_j} \frac{1}{u(e)}) \\ &\leq \sum_{j \in Q'_{low}} (2\alpha' \sum_{e \in Q_j} \frac{d_j}{u(e)}) \\ &= 2\alpha' \sum_{j \in Q'_{low}} \sum_{e \in Q_j} \frac{d_j}{u(e)} = 2\alpha' \sum_e \sum_{j \in Q | e \in Q_j} \frac{d_j}{u(e)} \\ &\leq 2\alpha' m \end{aligned}$$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

We also can express $r(Q'_{low})$ using the definition of F :

$$\begin{aligned} r(Q'_{low}) &= \sum_{j \in Q'_{low}} r(j) = \sum_{j \in Q'_{low}} (F(j, Q_j) d_j \sum_{e \in Q_j} \frac{1}{u(e)}) \\ &\leq \sum_{j \in Q'_{low}} (2\alpha' \sum_{e \in Q_j} \frac{d_j}{u(e)}) \\ &= 2\alpha' \sum_{j \in Q'_{low}} \sum_{e \in Q_j} \frac{d_j}{u(e)} = 2\alpha' \sum_e \sum_{j \in Q | e \in Q_j} \frac{d_j}{u(e)} \\ &\leq 2\alpha' m \end{aligned}$$

Since $r(Q'_{low}) \geq \frac{r(Q)}{4}$, it follows that

$$r(Q) \leq 8\alpha' m$$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Lower bound on the profit of approximated solution

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

From now on it is sufficient to consider the profit of a particular run:
 $Routine_1(\alpha', T_{i'})$.

- Let P denote all the requests routed in this run.
- For each $j \in P$, P_j denotes the path, chosen for it.
- E_{heavy} denotes edges that loaded by at least quarter of their capacity during $Routine_1(\alpha', T_{i'})$: $E_{heavy} = \{e \in E \mid L_l(e) \geq \frac{1}{4}\}$.

Lower bound on the profit of approximated solution

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(\mathcal{Q})$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

From now on it is sufficient to consider the profit of a particular run:
 $Routine_1(\alpha', T_{i'})$.

- Let P denote all the requests routed in this run.
- For each $j \in P$, P_j denotes the path, chosen for it.
- E_{heavy} denotes edges that loaded by at least quarter of their capacity during $Routine_1(\alpha', T_{i'})$: $E_{heavy} = \{e \in E \mid L_l(e) \geq \frac{1}{4}\}$.

There are two possibilities: the size of E_{heavy} is either greater than \sqrt{m} or smaller than that. We will consider the two cases separately.

First case: $|E_{heavy}| \geq \sqrt{m}$

First, we consider the case when $|E_{heavy}| \geq \sqrt{m}$:

$$\begin{aligned} r(P) &= \sum_{j \in P} (F(j, P_j) \sum_{e \in P_j} \frac{d_j}{u(e)}) \\ &\geq \alpha' \sum_{j \in P} \sum_{e \in P_j} \frac{d_j}{u(e)} \\ &\geq \alpha' \frac{1}{4} \sqrt{m} \end{aligned}$$

The last inequality follows from the assumption on the size of E_{heavy} , namely it is assumed that at least \sqrt{m} edges are loaded by at least quarter of their capacity.

Introduction

Algorithms for UFP

Notations

More notations

Approximation algorithm

Algorithm approximation and complexity

The optimal solution

Upper bound on $r(Q)$

Lower bound on the profit of approximated solution

First case: $|E_{heavy}| \geq \sqrt{m}$

Second case: $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of R

Proving the lower bound on the profit of P

Classical UFP - Strongly Polynomial

Unsplittable Flow Problem
Additional results

First case: $|E_{heavy}| \geq \sqrt{m}$

First, we consider the case when $|E_{heavy}| \geq \sqrt{m}$:

$$\begin{aligned} r(P) &= \sum_{j \in P} (F(j, P_j) \sum_{e \in P_j} \frac{d_j}{u(e)}) \\ &\geq \alpha' \sum_{j \in P} \sum_{e \in P_j} \frac{d_j}{u(e)} \\ &\geq \alpha' \frac{1}{4} \sqrt{m} \end{aligned}$$

The last inequality follows from the assumption on the size of E_{heavy} , namely it is assumed that at least \sqrt{m} edges are loaded by at least quarter of their capacity.

Considering upper bound on $r(Q)$ and lower bound on $r(P)$, it is clear that

$$\frac{r(Q)}{r(P)} \leq 32\sqrt{m} = O(\sqrt{m})$$

Introduction

Algorithms for UFP

Notations

More notations

Approximation algorithm

Algorithm approximation and complexity

The optimal solution

Upper bound on $r(Q)$

Lower bound on the profit of approximated solution

First case: $|E_{heavy}| \geq \sqrt{m}$

Second case: $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of R

Proving the lower bound on the profit of P

Classical UFP - Strongly Polynomial

Unsplittable Flow Problem
Additional results

Second case: $|E_{heavy}| < \sqrt{m}$

- Let R denote requests that are in Q'_{high} but not in the solution found by the algorithm, $R = Q'_{high} \setminus P$.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Second case: $|E_{heavy}| < \sqrt{m}$

- Let R denote requests that are in Q'_{high} but not in the solution found by the algorithm, $R = Q'_{high} \setminus P$.
- By inventing an upper bound on the profit of R , an upper bound on the approximation factor can be also found.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:

$$|E_{heavy}| \geq \sqrt{m}$$

Second case:

$$|E_{heavy}| < \sqrt{m}$$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Second case: $|E_{heavy}| < \sqrt{m}$

- Let R denote requests that are in Q'_{high} but not in the solution found by the algorithm, $R = Q'_{high} \setminus P$.
- By inventing an upper bound on the profit of R , an upper bound on the approximation factor can be also found.
- Each request j in R was not routed, because there was no feasible path in G , in particular the path G_j was not feasible. It means, that there was at least one edge on this path that its capacity constrain prevented the request from been routed. Let us denote this edge by e_j .

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:

$$|E_{heavy}| \geq \sqrt{m}$$

Second case:

$$|E_{heavy}| < \sqrt{m}$$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Second case: $|E_{heavy}| < \sqrt{m}$

- Let R denote requests that are in Q'_{high} but not in the solution found by the algorithm, $R = Q'_{high} \setminus P$.
- By inventing an upper bound on the profit of R , an upper bound on the approximation factor can be also found.
- Each request j in R was not routed, because there was no feasible path in G , in particular the path G_j was not feasible. It means, that there was at least one edge on this path that its capacity constrain prevented the request from been routed. Let us denote this edge by e_j .

Lemma 2. $e_j \in E_{heavy}$.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:

$$|E_{heavy}| \geq \sqrt{m}$$

Second case:

$$|E_{heavy}| < \sqrt{m}$$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- If $i' = 1$, it is immediate, since in this case $j \in R \subseteq Q' \subseteq T_1$ and hence $d_j \leq u_{min}/2$. If the request could not be routed, the edge e_j was loaded by at least half of its capacity and hence it certainly belongs to E_{heavy} .

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

- If $i' = 1$, it is immediate, since in this case $j \in R \subseteq Q' \subseteq T_1$ and hence $d_j \leq u_{min}/2$. If the request could not be routed, the edge e_j was loaded by at least half of its capacity and hence it certainly belongs to E_{heavy} .
- If $i' = 2$, two additional options should be considered:
 1. If $u(e) > 2u_{min}$, the only way to overflow it by j is to have e loaded by more than u_{min} , hence e is in E_{heavy} .
 2. if $u(e) \leq 2u_{min}$, note that only requests from T_2 are routed, and hence the edge has at least $u_{min}/2$ flow and therefore is in E_{heavy} .

We shall use Lemma 1 to bound the value of $r(R)$ as following:

- requests in $Routine_1$ are ordered according to r_j/d_j
- Let $R^k = R \cap \{1, \dots, i\}$
- Let $P^k = P \cap \{1, \dots, i\}$
- Let E^k denote the set of edges e_j from the Lemma 2, such that $E^k = \{e_j | j \in R^k\}$.
- Since each request in R^k is routed through an edge in E^k and the capacity constrain always holds, $d(R^k) \leq \sum_{e \in E^k} u(e)$.
- $d(P^k) \geq \frac{u(f)}{4}$, where f is the highest capacity edge of E^k .
- Since $|E^k| \leq |E_{heavy}|$ by Lemma 2 and $|E_{heavy}| < \sqrt{m}$ by assumption, it holds that: $d(R^k) < \sqrt{m}u(f) \leq 4\sqrt{m}d(P^k)$.

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

We shall use Lemma 1 to bound the value of $r(R)$ as following:

- requests in $Routine_1$ are ordered according to r_j/d_j
- Let $R^k = R \cap \{1, \dots, i\}$
- Let $P^k = P \cap \{1, \dots, i\}$
- Let E^k denote the set of edges e_j from the Lemma 2, such that $E^k = \{e_j | j \in R^k\}$.
- Since each request in R^k is routed through an edge in E^k and the capacity constrain always holds, $d(R^k) \leq \sum_{e \in E^k} u(e)$.
- $d(P^k) \geq \frac{u(f)}{4}$, where f is the highest capacity edge of E^k .
- Since $|E^k| \leq |E_{heavy}|$ by Lemma 2 and $|E_{heavy}| < \sqrt{m}$ by assumption, it holds that: $d(R^k) < \sqrt{m}u(f) \leq 4\sqrt{m}d(P^k)$.

Hence, by Lemma 1, taking α to be $4\sqrt{m}$ and $\{b_1, \dots, b_n\}$ to be $\frac{r_1}{d_1}, \dots, \frac{r_l}{d_l}$:

$$\sum_{j \in R} \frac{r_j}{d_j} d_j \leq 4\sqrt{m} \sum_{j \in P} \frac{r_j}{d_j} d_j$$

We shall use Lemma 1 to bound the value of $r(R)$ as following:

- requests in $Routine_1$ are ordered according to r_j/d_j
- Let $R^k = R \cap \{1, \dots, i\}$
- Let $P^k = P \cap \{1, \dots, i\}$
- Let E^k denote the set of edges e_j from the Lemma 2, such that $E^k = \{e_j | j \in R^k\}$.
- Since each request in R^k is routed through an edge in E^k and the capacity constrain always holds, $d(R^k) \leq \sum_{e \in E^k} u(e)$.
- $d(P^k) \geq \frac{u(f)}{4}$, where f is the highest capacity edge of E^k .
- Since $|E^k| \leq |E_{heavy}|$ by Lemma 2 and $|E_{heavy}| < \sqrt{m}$ by assumption, it holds that: $d(R^k) < \sqrt{m}u(f) \leq 4\sqrt{m}d(P^k)$.

Hence, by Lemma 1, taking α to be $4\sqrt{m}$ and $\{b_1, \dots, b_n\}$ to be $\frac{r_1}{d_1}, \dots, \frac{r_l}{d_l}$:

$$\sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j$$

Proving the lower bound on the profit of P

$$\sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j$$

Hence

$$r(R) = \sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j = 4\sqrt{m}r(P)$$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Proving the lower bound on the profit of P

$$\sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j$$

Hence

$$r(R) = \sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j = 4\sqrt{m}r(P)$$

But since $Q'_{high} \subseteq R \cup P$

$$r(Q'_{high}) \leq r(R) + r(P) \leq (1 + 4\sqrt{m})r(P)$$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Proving the lower bound on the profit of P

$$\sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j$$

Hence

$$r(R) = \sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j = 4\sqrt{m}r(P)$$

But since $Q'_{high} \subseteq R \cup P$

$$r(Q'_{high}) \leq r(R) + r(P) \leq (1 + 4\sqrt{m})r(P)$$

Recall that $r(Q'_{high}) \geq r(Q)/4$ and hence

$$r(Q) \leq 4r(Q'_{high}) \leq 4(1 + 4\sqrt{m})r(P)$$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Proving the lower bound on the profit of P

$$\sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j$$

Hence

$$r(R) = \sum_{j \in R} r_j \leq 4\sqrt{m} \sum_{j \in P} r_j = 4\sqrt{m}r(P)$$

But since $Q'_{high} \subseteq R \cup P$

$$r(Q'_{high}) \leq r(R) + r(P) \leq (1 + 4\sqrt{m})r(P)$$

Recall that $r(Q'_{high}) \geq r(Q)/4$ and hence

$$r(Q) \leq 4r(Q'_{high}) \leq 4(1 + 4\sqrt{m})r(P)$$

Therefore

$$\frac{r(Q)}{r(P)} = O(\sqrt{m})$$

Introduction

Algorithms for UFP

Notations

More notations

Approximation
algorithm

Algorithm
approximation and
complexity

The optimal solution

Upper bound on
 $r(Q)$

Lower bound on the
profit of approximated
solution

First case:
 $|E_{heavy}| \geq \sqrt{m}$

Second case:
 $|E_{heavy}| < \sqrt{m}$

Proof of Lemma 2

Bounding the profit of
 R

Proving the lower
bound on the profit of
 P

Classical UFP - Strongly
Polynomial

Unsplittable Flow Problem
Additional results

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

In order to make the algorithm polynomial, the ratio $\alpha_{max}/\alpha_{min} = n \frac{r_{max}}{r_{min}} \frac{u_{max}}{d_{min}}$ should be bounded. This can be done by following steps:

Making the algorithm strongly polynomial

In order to make the algorithm polynomial, the ratio $\alpha_{max}/\alpha_{min} = n \frac{r_{max}}{r_{min}} \frac{u_{max}}{d_{min}}$ should be bounded. This can be done by following steps:

- For all the edges, whose capacity is greater than $l \cdot d_{max}$, the capacity should be changed to $l \cdot d_{max}$. This is the maximum flow that can be needed to serve all the requests and hence the action does not change the optimal solution.

Making the algorithm strongly polynomial

In order to make the algorithm polynomial, the ratio $\alpha_{max}/\alpha_{min} = n \frac{r_{max}}{r_{min}} \frac{u_{max}}{d_{min}}$ should be bounded. This can be done by following steps:

- For all the edges, whose capacity is greater than $l \cdot d_{max}$, the capacity should be changed to $l \cdot d_{max}$. This is the maximum flow that can be needed to serve all the requests and hence the action does not change the optimal solution.
- Each request, whose profit is below r_{max}/l should be thrown away. The profit of all this requests together is at most r_{max} . Hence, the profit of the optimal solution is no less than half the original optimal solution, and the approximation factor of the algorithm is still the same.

Making the algorithm strongly polynomial

In order to make the algorithm polynomial, the ratio $\alpha_{max}/\alpha_{min} = n \frac{r_{max}}{r_{min}} \frac{u_{max}}{d_{min}}$ should be bounded. This can be done by following steps:

- For all the edges, whose capacity is greater than $l \cdot d_{max}$, the capacity should be changed to $l \cdot d_{max}$. This is the maximum flow that can be needed to serve all the requests and hence the action does not change the optimal solution.
- Each request, whose profit is below r_{max}/l should be thrown away. The profit of all this requests together is at most r_{max} . Hence, the profit of the optimal solution is no less than half the original optimal solution, and the approximation factor of the algorithm is still the same.
- Take the requests whose demand is not greater than u_{min}/l and route them through any simple path. This requests can not interfere each other. The rest of requests are treated as before. One of this solutions will supply the required approximation.

Strongly polynomial - proof

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

- The additional steps themselves are strongly polynomial.

Strongly polynomial - proof

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

- The additional steps themselves are strongly polynomial.
- The first preprocessing phase limits u_{max} to $l \cdot d_{max}$

Strongly polynomial - proof

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

- The additional steps themselves are strongly polynomial.
- The first preprocessing phase limits u_{max} to $l \cdot d_{max}$
- $\frac{r_{max}}{r_{min}}$ is now at most l by the second preprocessing phase.

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

- The additional steps themselves are strongly polynomial.
- The first preprocessing phase limits u_{max} to $l \cdot d_{max}$
- $\frac{r_{max}}{r_{min}}$ is now at most l by the second preprocessing phase.
- After these two phases the number of iterations is at most $\log\left(nl^2 \frac{d_{max}}{d_{min}}\right)$.

Strongly polynomial - proof

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

- The additional steps themselves are strongly polynomial.
- The first preprocessing phase limits u_{max} to $l \cdot d_{max}$
- $\frac{r_{max}}{r_{min}}$ is now at most l by the second preprocessing phase.
- After these two phases the number of iterations is at most $\log(nl^2 \frac{d_{max}}{d_{min}})$.
- If $S = T_1$, $d_{max} \leq \frac{u_{min}}{2}$ and $d_{min} \geq \frac{u_{min}}{l}$.

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

- The additional steps themselves are strongly polynomial.
- The first preprocessing phase limits u_{max} to $l \cdot d_{max}$
- $\frac{r_{max}}{r_{min}}$ is now at most l by the second preprocessing phase.
- After these two phases the number of iterations is at most $\log(nl^2 \frac{d_{max}}{d_{min}})$.
- If $S = T_1$, $d_{max} \leq \frac{u_{min}}{2}$ and $d_{min} \geq \frac{u_{min}}{l}$.
- For $S = T_2$, $d_{max} \leq u_{min}$ and $d_{min} \geq \frac{u_{min}}{2}$.

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Making the algorithm strongly polynomial

Strongly polynomial - proof

Additional results

- The additional steps themselves are strongly polynomial.
- The first preprocessing phase limits u_{max} to $l \cdot d_{max}$
- $\frac{r_{max}}{r_{min}}$ is now at most l by the second preprocessing phase.
- After these two phases the number of iterations is at most $\log(nl^2 \frac{d_{max}}{d_{min}})$.
- If $S = T_1$, $d_{max} \leq \frac{u_{min}}{2}$ and $d_{min} \geq \frac{u_{min}}{l}$.
- For $S = T_2$, $d_{max} \leq u_{min}$ and $d_{min} \geq \frac{u_{min}}{2}$.
- Hence, there are at most $O(\log n + \log l)$ iterations.

Introduction

Algorithms for UFP

Classical UFP - Strongly
Polynomial

Additional results

Additional results

Bibliography

Additional results

Introduction

Algorithms for UFP

Classical UFP - Strongly
Polynomial

Additional results

Additional results

Bibliography

- The authors also present a strongly polynomial approximation algorithm for Extended UFP problem, which is a rather simple extension of algorithms presented so far. In addition an algorithm for K-bounded UFP is presented.

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

Additional results

Bibliography

- The authors also present a strongly polynomial approximation algorithm for Extended UFP problem, which is a rather simple extension of algorithms presented so far. In addition an algorithm for K-bounded UFP is presented.
- Separate analysis shows that in general, an approximation for extended case is more difficult than approximation for the classical UFP. For the directed graphs, the authors present a lower bound on the approximation factor.

Introduction

Algorithms for UFP

Classical UFP - Strongly Polynomial

Additional results

Additional results

Bibliography

- The authors also present a strongly polynomial approximation algorithm for Extended UFP problem, which is a rather simple extension of algorithms presented so far. In addition an algorithm for K-bounded UFP is presented.
- Separate analysis shows that in general, an approximation for extended case is more difficult than approximation for the classical UFP. For the directed graphs, the authors present a lower bound on the approximation factor.
- For a special case of the problem, where $r_j = d_j$ for any request, and hence the requests do not need to be sorted, the algorithm for K-bounded UFP can be turned to an online algorithm in a very simple way, achieving an approximation factor of $O(K \cdot D^{\frac{1}{K}})$, which meets the lower bound on the approximation factor.

Introduction

Algorithms for UFP

Classical UFP - Strongly
Polynomial

Additional results

Additional results

Bibliography

- [1] Yossi Azar and Oded Regev. Strongly polynomial algorithms for the unsplittable flow problem. *Lecture Notes in Computer Science*, 2081:15+, 2001.