Algorithmic Methods	03/01/2011

Lecture Notes 12: Approximating Generalized Steiner Forest and PTAS for Scheduling

Professor: Yossi Azar *Scribe: Yigal Lazarev*

Introduction

In the first part of the lecture we will see how to get a factor of 2 approximation for the Generalized Steiner Forest problem. In the second part, we will see a PTAS (Polynomial Time Approximation Scheme) for the Identical Machines Scheduling problem.

1 2-Approximation for Generalized Steiner Forest

1.1 Problem definition

Input An undirected graph $G = (V, E)$, a weight function $c : E \to \mathbb{R}^+$ and a set of pairs $S = \left\{(s_i, t_i)_{i=1}^k\right\}$.
Feasible solution A forest $G' = (V', E')$, in which $\forall (s_i, t_i) \in S$ there is a path from s_i to t_i.
Goal A feasible solution that minimizes $\sum_{e \in E'} c(e)$.

1.2 Alternative definition

The GSF problem can be defined in an alternative way.
Input An undirected graph $G = (V, E)$, a weight function $c : E \to \mathbb{R}^+$ and k disjoint sets of vertices $S_1, S_2, \ldots, S_k,\ S_i \subseteq V$.
Feasible solution A forest $G' = (V', E')$, in which $\forall u, v \in S_i, 1 \leq i \leq k$, u is connected to v.
Goal A feasible solution that minimizes $\sum_{e \in E'} c(e)$.

1.3 Algorithm

We use Local Ratio to solve the problem and get 2-approximation. Let us define $A = \{u|(u = s_i \vee u = t_i) \wedge s_i \neq t_i\}$: the set of vertices that have not yet been contracted with their pair. The algorithm:

1. If $A = \emptyset$ return $\left(V = \bigcup_{i=1}^{k} s_i, E = \emptyset\right)$.
2. Generate new weights: $c'(u,v) = \begin{cases} \epsilon & u \in A \oplus v \in A \\ 2\epsilon & u \in A \wedge v \in A \end{cases}$ and 0 otherwise
   where $\epsilon = min\{\epsilon_1, \epsilon_2\}$,
   $\epsilon_1 = min\{d(u,v)|(u \in A) \oplus (v \in A)\}$ and $\epsilon_2 = \frac{1}{2} min\{d(u,v)|u \in A \wedge v \in A\}$.
   Subtract c' from c.
3. If $w(e) = 0$ contract e and redefine A.
4. Solve recursively.
5. Expand edges of zero weight (that have been contracted).
6. Fix the recursive solution: drop all excessive edges. Excessive edges are:
   - Edges that close a cycle.
   - Edges that do not participate in a shortest path from s_i to t_i for all $1 \leq i \leq k$.

   Therefore the leafs of our forest are a subset of $\bigcup_{i=1}^{k} \{s_i, t_i\}$.

1.4 Correctness

Trivial case: If $A = \emptyset$ then $E = \emptyset$. Hence, OPT= 0 and ANYFIXED= 0.

Theorem 1.4.1. *In the graph for which the weight of an edge is according to c', OPT $\geq |A| \cdot \epsilon$ for any A for which $|A| \geq 2$.*

Proof. Let us assume that OPT connects sets $A_1, A_2, \ldots, A_r$. We have seen in the proof for Steiner Tree approximation algorithm that $\text{OPT}|_{A_i} \geq |A_i| \cdot \epsilon$ for $|A_i| \geq 2$. We can easily deduce that $\text{OPT} \geq \sum_{i=1}^{r} |A_i| \cdot \epsilon = |A| \cdot \epsilon$. □

Theorem 1.4.2. *In the graph for which the weight of an edge is according to c', ANYFIXED $\leq 2(|A| - 1) \cdot \epsilon$ for any A for which $|A| \geq 2$.*

Proof. Again as seen in the proof for Steiner Tree problem, a tree that contains leafs only from A will have a weight of at most $2(|A| - 1) \cdot \epsilon$. A forest that connects some of A has an even smaller weight. □

By combining theorems 1.4.1 and 1.4.2, we get that the algorithm returns a factor of 2 approximation for the Generalized Steiner Forest problem.

2 Scheduling

In this section we will see how to get a PTAS for the Identical Machines Scheduling problem. First let us recall the problem definition:
Input m machines and n tasks, task i is of size $p_i, 1 \leq i \leq n$.
Feasible solution An assignment $a_i \in \{1, \ldots, m\},\ 1 \leq i \leq n$ of all the tasks to the machines.
Goal Minimize $max\{l_j\}$ where $l_j = \sum_{i|a_i=j} p_i$.
We saw a 2-approximation greedy algorithm, and a $\frac{4}{3}$-approximation sorting greedy algorithm. We will now introduce a PTAS for scheduling.

2.1 PTAS for Scheduling

PTAS definition $\forall \epsilon > 0$ there exists an algorithm A_ϵ that approximates the problem to a factor of $1 + \epsilon$ and runs in polynomial time with some dependency on $\frac{1}{\epsilon}$.

2.2 The Approximated Decision Problem

Input $m, n, \{p_i\}, T$
Return

1. Negative if no assignment with maximal load $\leq T$ exists.
2. If there exists an assignment with maximal load $\leq T(1+\epsilon)$, return such an assignment.

Note In case both of the options are true - there is no assignment with maximal load $\leq T$ *and* there exists an assignment with maximal load $\leq T(1 + \epsilon)$ - the algorithm may return 1 or 2.

Theorem 2.2.1. *If one can solve the approximated decision problem for $\frac{\epsilon}{2}$ in polynomial time, then the optimization problem can be approximated to a factor of $1 + \epsilon$ in polynomial time.*

Proof. Perform a binary search on T. Initialization: $\left[\frac{1}{2}T_2, T_2\right]$ where T_2 is the factor 2 approximation solution using the greedy algorithm. At each point, $[T_1, T_2]$ is our search range, and we know there is no solution with maximal load $\leq T_1$ but there is a solution with maximal load $\leq T_2$. Assign $T = \sqrt{T_1 \cdot T_2}$ (binary search on a logarithmic scale) and solve the approximated decision problem for T. If the result was negative, the new range is $(T, T_2]$, otherwise the new range is $[T_1, T]$. We continue the search until $\frac{T_2}{T_1} \approx 1 + \frac{\epsilon}{3} \Rightarrow \frac{T_2\left(1+\frac{\epsilon}{2}\right)}{T_1} < 1 + \epsilon$. We will now find out how many iterations are needed until the search is stopped. The ratio between T_2 and T_1 at every step is a square-root of the ratio in the previous step. Note that $\sqrt{\sqrt{\sqrt{\ldots\sqrt{2}}}} = 2^{\frac{1}{2^k}}$. At first $\frac{T_2}{T_1} \leq 2$, so we need to find k for which $2^{\frac{1}{2^k}} \leq 1 + \frac{\epsilon}{3} \Rightarrow 2 \approx \left(1 + \frac{\epsilon}{3}\right)^{2^k}$. Take log of both sides to get $1 \leq 2^k \log\left(1 + \frac{\epsilon}{3}\right) \approx 2^k \cdot \frac{\epsilon}{3}$ therefore $k = O\left(\log \frac{1}{\epsilon}\right)$. $\square$

2.3 Algorithm for Solving the Approximated Decision Problem

The algorithm for solving the approximated decision problem is as follows:

1. Ignore tasks with load $p_i < \epsilon \cdot T$.
2. Solve the approximated decision problem for the remaining tasks (algorithm will be given later on).
3. If the result is (A) negative, return negative. If the result is positive, order arbitrarily the tasks ignored in stage 1, and assign iteratively to the least loaded machine. If the resulting load is $\leq T(1+\epsilon)$ then (B) return positive answer. Otherwise (C) return negative.

Theorem 2.3.1. *The algorithm is correct*

Proof. Let us inspect the 3 return cases of the algorithm:

- *case (A)* There is no solution for a subset of the tasks, therefore there is no solution for the entire set of tasks.
- *case (B)* There is a solution and we found it.
- *case (C)* The maximal load $> T(1+\epsilon)$ and since every 'small' task has a load of at most $\epsilon \cdot T$ then ALL the machines' load is at least T, therefore $\sum_{i=1}^{m} p_i > m \cdot T \Rightarrow$ OPT $> T$ and a negative answer is a correct one.

□

Next we need to show how to perform step 2 of the algorithm. Our inputs are tasks with loads $\epsilon \cdot T \leq p_i \leq T$. Replace p_i with $q_i \leq p_i$ such that $q_i = \epsilon \cdot T + k \cdot \epsilon^2 \cdot T = T\left(\epsilon + \epsilon^2 \cdot k\right), k \in \mathbb{N}$ is maximal. Algorithm:

1. Solve the *exact* decision problem for $\{q_i\}, T$ (algorithm will be given, again, later on).
2. If the result was negative (A) return negative answer. Otherwise (B) return a positive answer with the same assignment as the solution to the exact decision problem.

Theorem 2.3.2. *The algorithm for solving step 2 of the algorithm for solving the approximated decision problem is correct.*

Proof. In case (A), there is no solution for loads rounded downwards, therefore there is no solution for the original loads. In case (B), the load of the machines is multiplied by at most $\frac{p_i}{q_i} \leq \frac{q_i + \epsilon^2 \cdot T}{q_i} \leq 1 + \frac{\epsilon^2 \cdot T}{q_i} \leq 1 + \frac{\epsilon^2 \cdot T}{\epsilon \cdot T} = 1 + \epsilon$ therefore we get a $1+\epsilon$ approximation. □

In the next lecture, we will see how to solve step 1, and our PTAS for Identical Machines Scheduling will be complete.