Algorithmic Methods 06/12/2010

Lecture Notes 8: Approximations for Scheduling

Professor: Yossi Azar *Scribe:Ety Haitsin*

Introduction

In first section, we will show a reduction from optimality to feasibility for LP with non-polynomial number of constraints. Other sections will deal with scheduling models: we will show a 2 approximation algorithms for Restricted assignment and Unrelated machines models.

1 Reduction from optimal to feasible solution

In third lecture we showed how to make a reduction from finding an optimal solution of LP to finding a feasible one by using the duality form of LP (feasible solution for a new LP that combines the primal and the dual form together and a constraint $c^t x = y^t b$, will be the optimal solution for the primal LP). But if the primal LP has exponential number of constrains then it will be impossible to use this technique because the dual form will have exponential number of variables. In this case we will have to do a binary search on the optimal solution:

Make binary search on t:

$$\left.\begin{array}{l} Min\, c^t x \\ Ax \geq b \end{array}\right\} \rightarrow \left.\begin{array}{l} Ax \geq b \\ c^t x \leq t \end{array}\right\}$$

Lemma 1.1. *If x, x' are vertices of a polytope and if $2^{-2L} > |c^t x - c^t x'|$ then $c^t x = c^t x'$*

Proof. As we defined vertices in forth lecture:

$$c^t x = \frac{z_1}{d_1},\ c^t x' = \frac{z_2}{d_2},\ |d_1|, |d_2| < 2^L$$

We get $|c^t x - c^t x'| = |\frac{z_1}{d_1} - \frac{z_2}{d_2}| = |\frac{z_1 d_2 - z_2 d_1}{d_1 d_2}|$
$|z_1 d_2 - z_2 d_1| \geq 1$ and it is an integer number, then $|\frac{z_1 d_2 - z_2 d_1}{d_1 d_2}| > \frac{1}{2^{2L}}$ which is a contradiction to the assumption. □

Note 1.2. Ellipsoid algorithm does not necessary finds a vertex, but by Theorem B, which is constructive, we always can find a feasible vertex.

2 Scheduling - Restricted Assignment

The model: m machines, n tasks, each task i has a set $M(i) \in M$ which are the machines that this task can be assigned to, and size w_i. Load on a machine j defined as $l_j = \sum_{i|A(i)=j} w_i$

2.1 Simple case

Model: $\forall_i w_i = 1$ Is there a solution with resulting load $\leq k$?
$k = 1$: This is exactly Maximal Matching problem, which can be solved in polynomial time.
$k \geq 1$: We duplicate each machine k times and run the Maximal Matching algorithm.
The optimal k value can be found by binary search, the solution will be optimal because all the values are integer.

2.2 General case

In this case there are a few steps:

1. Find fractional solution
2. Open cycles
3. Matching in Forest graph

2.3 Find fractional solution

We build a directed bipartite graph $G = (U, V, E)$ where vertices U are tasks and vertices V are machines, and add edges by following rule: $j \in M(i) \iff (i, j) \in E$. Capacities of all edges in E is ∞. Then we add two vertices: sink s and target t, to the graph. s is connected to all vertices U (tasks) and capacity of those edges is $C(s, u_i) = w_i$. Every vertex in V (machines) has an edge to t, with capacity $C(v_j, t) = T$.
Lets solve the decision problem on T:
We find the maximal flow in this graph from s to t. If flow value is $\sum_{i \in Tasks} w_i$ then the result is feasible (for the original problem), otherwise there is no feasible solution on T. Note, that a feasible flow solution gives us a feasible scheduling solution such that maximal load is up to T.
We will find the optimal T by binary search.

2.4 Open cycles

Lets take the graph and the flow we got from previous step (we take only the bipartite part of the graph). Every edge from U to V (from task to machine), that current flow doesn't use at all, will be deleted. Then, we get a graph that may still contain undirected cycles.

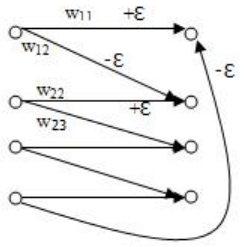

Figure 1: Example of an undirected cycle, and flow changes step.

For every task in this cycle: to the first edge we will add ϵ flow and from the second we will subtract the same flow. In this way the flow that enters the machines doesn't change at all. This way we can get a perturbation of the solution (many different solutions with the same value for different ϵ), as long as ϵ is small enough so no edges have negative flow.

$$\begin{cases} w_{i,i} = w_{i,i} + \epsilon \\ w_{i,i+1} = w_{i,i+1} - \epsilon \\ \epsilon = Min_i(w_{i,i+1}) \end{cases}$$

Now we just opened one cycle. Note that in the end of this step one edge has no more flow on it and then it is deleted from the support set of edges. So we iteratively can open cycles one by one, each time one edge will be deleted so number of this steps is polynomial.

2.5 Matching in Forest graph

Now we have a graph witch is a forest. We will focus on the leafs of the graph:
While there are tasks leafs ($u_i \in U$ is a leaf), we assign this task to the only machine it is assigned to.
Now we have forest graph with only machines leafs, for each such a leaf v_j, choose any edge (i, j) and assign task i to machine j and delete the task and the machine from the graph. Because both those vertices are removed no tasks can become leafs (in the new forest), so we can repeat these steps as long as there would be no tasks on the graph.

Now lets check the load on the machines after the algorithm:

$$\forall_j \, l_j \leq l_j^{frac} + w_i \leq OPT + OPT \leq 2 * OPT$$

We get that this algorithm yields 2 approximation.

3 Scheduling - Unrelated Machines

Model: m machines, n tasks. Every task has different weight on every machine, which can also be ∞, which means that task cannot be assigned to this machine.

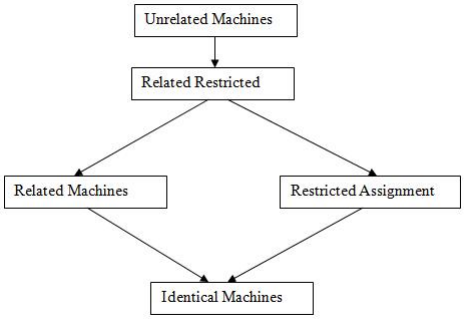

Figure 2: Relations between different scheduling models. In Related Restricted model the load of task i on machine j is one of $\{\frac{w_i}{v_j}, \infty\}$

3.1 General case

Here again we have the same steps like in previous model:

1. Find fractional solution
2. Open cycles
3. Matching in Forest graph

3.2 Find fractional solution

This time we cannot use the flow algorithm to find the fractional solution, but we need an LP to find it:

$$\begin{cases} Min\, 0 & \\ \forall_i \sum_j x_{ij} = 1 & (1) \\ \forall_j \sum_i w_{ij} x_{ij} \leq T & (2) \\ \forall_{i,j}\, If\ w_{ij} > T\ then\ x_{ij} = 0 & (3) \\ \forall_{i,j}\, x_{ij} \geq 0 & \end{cases}$$

It seems intuitive to make the target function to be $Min\, T$, but then the constrains of type (3) would be illegal (Note that those constrains are not linear, they are possible only in case when T is known in advance).
We got an LP that solves feasibility on given T. We can find the optimal T by binary search.

3.3 Open cycles

Now we build a bipartite graph and a flow from the given LP solution (in Restricted Assignment model we get the assignment from flow, know we doing the reverse reduction), take only the edges that have flow on them, and search for undirected cycles in this graph. Note, that we cannot open those cycles as in previous model because $w_{i,j} \neq w_{i,j+1}$ (then, if we use the same technique the load on machines will change).
But the idea is the same:
Let $f_{ij} = w_{ij} x_{ij}$

$$\begin{cases} f_{11} = f_{11} + \epsilon \\ f_{12} = f_{12} - \epsilon * \frac{w_{12}}{w_{11}} \\ f_{22} = f_{22} + \epsilon * \frac{w_{12}}{w_{11}} \\ f_{23} = f_{23} - \epsilon * \frac{w_{12}}{w_{11}} * \frac{w_{23}}{w_{22}} \\ \vdots \\ f_{k,k+1} = f_{k,k+1} - \epsilon * \frac{w_{12}}{w_{11}} * \frac{w_{23}}{w_{22}} \cdots * \frac{w_{k,k+1}}{w_{kk}} = f_{k,k+1} - \epsilon * \Pi \end{cases}$$

In this flow change, all the machines except first one, have the same load. The load of first machine will change; if $\Pi \geq 1$ the load may only decrease and the solution is still feasible. Otherwise, we will reverse the cycle and then load on first machine will certainly become smaller ($\Pi' = \frac{1}{\Pi} \geq 1$).

Now we just opened one cycle. Note that in the end of this step one edge has no more flow on it and the it is deleted from the support set of edges. So we iteratively can open cycle one by one, each time one edge will be deleted so number of this steps is polynomial.

3.4 Matching in Forest graph

This part is identical to the Restricted Assignment model, so we won't repeat it.

Now lets check the load on the machines after the algorithm:

$$\forall_j \, l_j \leq l_j^{frac} + w_{ij} \leq T + T \leq 2 * T$$

Note, that we need the constraint of (3) type to claim that every w_{ij} that assigned to machine is smaller than T.

We get that his algorithm yields 2 approximation.

Now, we prove the analysis is tight (we show an example where this algorithm yields 2 approximation), even for restricted assignment:

m machines, $m(m-1)$ tasks of size 1, one task of size m.

The fractional solution will assigned uniformly all small task, and the big one would be assigned fractionally to all machines.

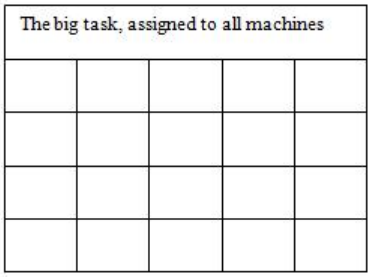

Figure 3: Fractional solution

In this case, one of the machines will get the whole task in addition to all the rest $m-1$ small tasks that it already got. This machine's load will be $(m-1)+m = 2m-1$, while the optimal solution gives value m.

$\frac{Alg}{OPT} = \frac{2m-1}{m} = 2 - \frac{1}{m}$.

4 PTAS - Identical machines

A PTAS is an algorithm which takes an instance of an optimization problem and a parameter $\epsilon > 0$ and, in polynomial time, produces a solution that is within a factor $1+\epsilon$ of being optimal.

Decision problem on given T:

Find a solution with value up to $(1+\epsilon)T$

Or return "No solution for given T"

Lemma 4.1. *If we can solve the decision problem, then we can get a solution with $1+\epsilon'$ approximation ratio (when $\epsilon' = 2\epsilon$).*

Proof. The solution can be found by binary search (the limits can be found by a simple approximation algorithm).
The searching continues until $\frac{T_2(1+\epsilon)}{T_1} < 1 + 2\epsilon$ (where T_1 and T_2 are adjacent decision problem parameters, when T_1 is not feasible, and $T_2(1+\epsilon)$ is feasible). □