

Lecture Notes 7: Scheduling

Professor: Yossi Azar

Scribe: Tomer Kazaz

Introduction

This lecture will cover the following topics:

- Multi commodity flow
- Scheduling problem
 - Identical machines
 - Related machines

Integer routing via Multi commodity flow

(s_i, t_i) - Requests

$G = (V, E)$ $c : E \rightarrow R^+$ is capacity function.

The width of each request is 1.

v_i - profit if we serve the path from s_i to t_i , else the profit is 0.

We set P_i to be the path from s_i to t_i if served (otherwise $P_i = \emptyset$).

$l_e = \{i \mid e \in P_i\} \leq c_e$ - The load of an edge equals the number of paths using it.

Goal: Maximize the overall profit.

Phase 1 - Find relaxed solution

- Flow instead of path
- Requests can be served fractionally and receive fractional profit

$Max\{\sum v_i x_i\}$

x_e^i - Total flow of type i going through edge e (in the relaxed solution)

x_i - Total flow of type i .

$\forall e : \sum_i x_e^i \leq c_e$ (make sure each edge doesn't exceed its capacity)

Preservation constraints:

$\sum_{e \in (s_i, v)} x_e^i - \sum_{e \in (v, s_i)} x_e^i = x_i \quad \forall i$ (What leaves the source minus what enters it)

$\sum_{e \in (u, v)} x_e^i - \sum_{v \in (v, u)} x_e^i = 0 \quad \forall u \in V - \{s_i, t_i\}$ (Incoming flow equals to outgoing flow)

We change the constraint $x_e^i, x_i \in \{0, 1\}$ into the following:

$$0 \leq x_e^i \leq 1$$

$$0 \leq x_i \leq 1$$

Phase 2 - Rounding

Claim1: Flow from s to t may be split to $\leq |E|$ paths.

Proof:

Iteratively remove paths.

DAG - in each iteration at least one edge is removed and we keep legal flow.

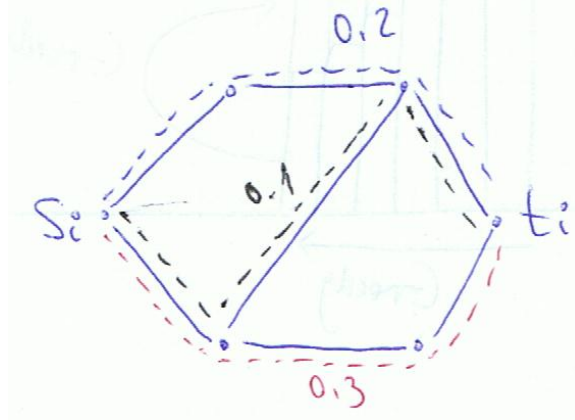
According to *Claim1* the flow may be split to maximum m paths:

For flow of type i :

$P_1^i, P_2^i, \dots, P_m^i$ - paths of type i .

$f_1^i, f_2^i, \dots, f_m^i$ - Amount of flow on the above paths.

Figure 1: Example of a feasible fractional solution



Denote by x_i^* the optimal solution of the LP for the total flow of type i .

Clearly, $x_e^i = \sum_{j|e \in P_j^i} f_j^i$ and $x_i^* \equiv \sum_j f_j^i$.

In the rounding stage we would like to choose one path and serve the **whole** request.

Rounding Algorithm:

Choose path P_j^i with probability f_j^i as disjoint events and we choose \emptyset with probability $1 - \sum_j f_j^i = 1 - x_i^*$

Note: The above algorithm will not always work.

When will it work?

If the capacities are much larger than the unit width of the request.

Therefore - we assume that $c_e = \Omega(\frac{\log m}{\epsilon^2})$ for $\epsilon > 0$ (m - number of edges)

Then the algorithm gets $1 + \epsilon$ approximation.

Proof:

X_j^i - Indicator variable that flow of path P_j^i has been chosen.

$X_i = \sum_j X_j^i$ - Indicator variable that flow i (x_i) has been chosen.

$$E(X_j^i) = f_j^i$$

$$E(X_i) = E(\sum_j X_j^i) = \sum_j E(X_j^i) = \sum_j f_j^i \equiv x_i^*$$

$$E(\text{Goal function}) = E(\sum_i v_i X_i) = \sum_i v_i E(X_i) = \sum_i v_i x_i^* = OPT_{relaxed} \geq OPT_{integer}$$

We would like to know what is the expectation of the flow of each edge.

We would like to show that the flow is less than the capacity.

$$f_e = \sum_{i,j|e \in P_j^i} X_j^i$$

$$E(f_e) = \sum_i \sum_{j|e \in P_j^i} E(X_j^i) = \sum_i \sum_{j|e \in P_j^i} f_j^i = \sum_i x_e^i \leq c_e.$$

So we do not have overflow in terms of expectation but it is still not enough.

We would like to show that with high probability the load of an edge is not greater than its capacity.

Chernoff-Hoffding Bound:

If we sum a lot of independent variables, we would like to see what is the chance that the summation is different from the expectation.

Let X_1, \dots, X_n independent variables. $0 \leq X_i \leq 1$.

$$X = \sum_{i=1}^n X_i. \quad E(X_i) = \mu_i$$

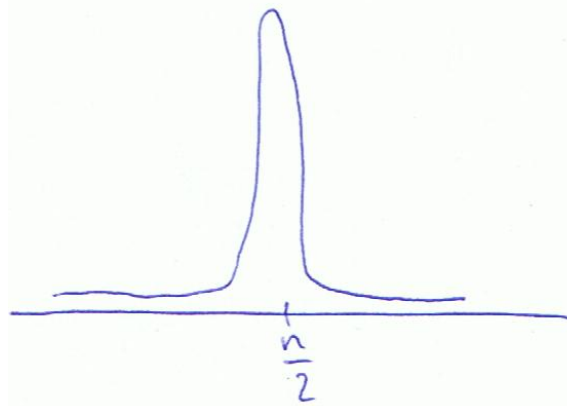
$$E(X) = \sum_i \mu_i = \mu$$

Theorem: $\Pr(X > (1 + \delta) \cdot \mu) \leq e^{-\frac{\delta^2 \cdot \mu}{(2 + \frac{2\delta}{3})}}$

For $\delta < 1$ it is at most $e^{-\frac{\delta^2 \mu}{3}}$.

The Chernoff bound is good when the expectation μ is large.

For instance, given $n = 1000$ drops of a coin with probability of $\frac{1}{2}$



Note: The Chernoff-Hoffding theorem is standalone. The variables do not need to be identical distributed.

Using the Chernoff bound:

For each specific edge (assuming $c(e) \geq \frac{6 \cdot \log m}{\varepsilon^2}$) we get:

$$\Pr(f_e \geq (1 + \varepsilon) \cdot c(e)) \leq e^{-\varepsilon^2 \cdot \frac{6 \cdot \log m}{\varepsilon^2} \cdot \frac{1}{3}} = e^{-2 \log m} = \frac{1}{m^2}$$

Actually we would like to understand what is the probability when some edge exceeds its capacity:

$$\Pr(\exists e \mid f_e \geq (1 + \varepsilon) \cdot c(e)) \leq \sum_{e \in E} \Pr(f_e \geq (1 + \varepsilon) \cdot c(e)) \leq m \cdot \frac{1}{m^2} = \frac{1}{m}$$

So far we proved $(1 + \varepsilon) \cdot c_e$, and we still needs to fix this and get rid of the $(1 + \varepsilon)$ factor.

Note: If the expectation of the load is smaller than the capacity we can add dummy random variables with probability of 1, so the expectation of the load would be exactly as the capacity.

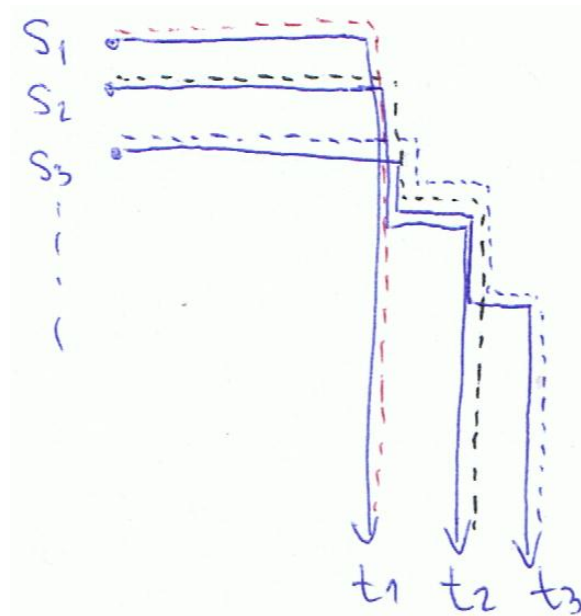
Now the chance to exceed is smaller, especially when we remove the dummy random variables.

Two solutions for the deviation of $(1 + \varepsilon)$:

1. In phase I we solve the LP according to the original capacities and then we reduce the solutions in a factor of $(1 + \varepsilon)$.
2. In phase I we reduce all the capacities in a factor of $(1 + \varepsilon)$ [before running the LP]. Value of the optimal fractional solution is decreased by at most a factor of $(1 + \varepsilon)$. We just reduce all flows in factor of $(1 + \varepsilon)$ and then get a feasible solution to the new system with loss of $(1 + \varepsilon)$

Remark: Byproduct of our result is that there is a small gap between integer vs fractional optimal solution (called integrality gap) for networks with large capacity.

Example: gap between integer vs fraction solution for unit capacity network



In the above example, every 2 paths are intersected. Therefore, the best solution is a single path

The fractional solution can get all of the paths (since it's fractional) - it would pass half a flow in each path.

$$OPT = 1$$

$$OPT_{relaxed} = \frac{n}{2}$$

Scheduling

m **identical** machines indexed by $1 \leq j \leq m$

n jobs indexed by $1 \leq i \leq n$

job i has weight w_i .

Feasible solution - assign the n jobs to the m machines.

$A(i)$ - The machine on which job i has been assigned.

Load over machine j is defined as $l_j = \sum_{i|A(i)=j} w_i$.

The cost of a solution is the load on the most loaded machine - $\max_j l_j$.

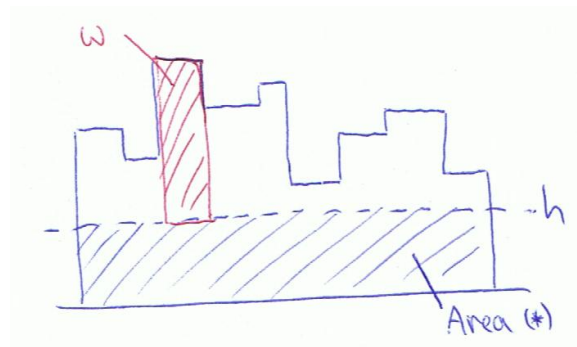
Goal - Find an assignment which minimizes the completion time of all jobs (i.e. minimize the maximum loaded machine).

Greedy Algorithm:

1. Arrange the jobs in some order.
2. Assign each job on the current least loaded machine.

Claim - Greedy algorithm is 2-approximation.

Proof by "drawing":



w is the job that made the most loaded machine be the most loaded.

Area (*) is full for sure, otherwise we would have chosen an other machine.

(1) $OPT \geq h$ due to volume constraints.

(2) $OPT \geq w$ since w is the height of some job which should also be assigned by OPT algorithm to some machine.

$$ALG(i) = h + w \leq OPT + OPT \leq 2 \cdot OPT$$

Equation (1) may be replaced with $OPT \geq \frac{m \cdot h + w}{m} = h + \frac{w}{m}$. That would give us the following analysis:

$$ALG(i) = h + w \leq \underbrace{h + \frac{1}{m} \cdot w}_{\leq OPT} + \underbrace{\left(1 - \frac{1}{m}\right) \cdot w}_{\leq OPT} \leq \left(2 - \frac{1}{m}\right) \cdot OPT$$

Claim: The 2-approximation is tight.

Proof: by example

Figure 2: **Greedy** behavior for $m \times (m - 1)$ jobs of size 1 and one job of size m

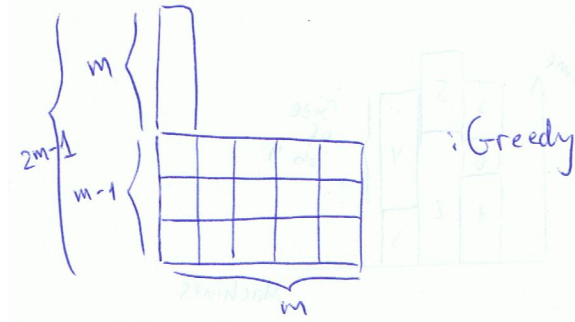
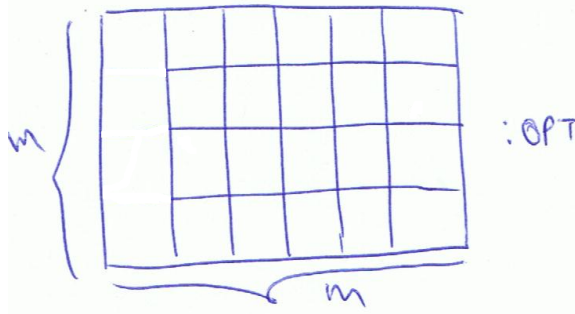


Figure 3: **OPT** behavior for $m \times (m - 1)$ jobs of size 1 and one job of size m



$$\frac{ALG(I)}{OPT(I)} = \frac{2m-1}{m} = 2 - \frac{1}{m}$$

”Sorted Greedy” Algorithm

1. Sort the jobs from large to small.
2. Put job over the less loaded machine at the moment.

Claim - ”Sorted Greedy” algorithm is $\frac{4}{3}$ -approximation.

Note: The exact approximation is $\frac{4}{3} - \frac{1}{3m}$

Idea of the proof - Consider two cases:

1. The job which made the most loaded machine to reach its load $\leq \frac{OPT}{3}$
 In this case we get $ALG(I) = h + w \leq OPT + \frac{OPT}{3} = \frac{4}{3} \cdot OPT$
2. The job which made the most loaded machine to reach its load $> \frac{OPT}{3}$
 Assume it is the last $w > \frac{1}{3}OPT$ job by deleting later jobs.
 There are at most $2m$ jobs.
 This is the order that "Sorted Greedy" would assign the jobs:
 It can be proved that OPT and "Sorted Greedy" assign the jobs in the same order,



and therefore we get 1-approximation.

Notes:

- The mentioned problem is NP-hard.
- The decision problem is NP-complete.
- For $m = 2$ the hardness of the problem comes from a reduction to the *Partition* problem - Given a group of numbers we would like to split it into 2 sets such that the summations of the sets are equal.
- The first algorithm ("Greedy") is an online algorithm.
- The second algorithm ("Sorted Greedy") is not online since it sorts the input at the first stage.

PTAS

PTAS - Polynomial time approximation scheme.

We talk about it later in the course.

$\forall \varepsilon > 0$ we can produce an $(1 + \varepsilon)$ -approximation algorithm.

Running time is polynomial (The dependence on ε may be exponential).

Related Machines

m **related** machines indexed by $1 \leq j \leq m$

v_j - speed of machine j .

n jobs indexed by $1 \leq i \leq n$

job i has weight w_i .

Feasible solution - assign the n jobs to the m machines.

$A(i)$ - The machine on which job i has been assigned.

Load over machine j is defined as $l_j = \frac{1}{v_j} \cdot \sum_{i|A(i)=j} w_i$.

The cost of the solution is the load of the most loaded machine which is $\max_j l_j$.

Goal - Find an assignment which minimizes the completion time of all jobs (i.e. minimum maximum loaded machine).

Greedy Algorithm:

1. Arrange the jobs in some order.
2. Assign each job on a machine such that after the assignment would be less loaded.
Formally - Put on machine j such that $l'_j + \frac{w_i}{v_j}$ is minimal.

Claim: (with no proof) - The above greedy algorithm is $O(\log m)$ -approximation

Claim: (with no proof) - Even if we sort the jobs from small to large, we still get $\Theta(\log m)$ -approximation.

Claim: (with no proof) - Sorting the jobs from large to small would get a constant approximation.

General scheme (online):

- We arrange the jobs in a some order.
- We guess the value of the optimal solution - λ (this can be made online - details are omitted).
- We show an algorithm that gives a 2λ -approximation.

"Socially weak" Algorithm:

Put the job on the slowest machine, such that after the assignment the load of the machine $\leq 2\lambda$

More formally:

- Sort the machines from slowest to fastest.
- Put on a machine with minimal index j such that $l_j + \frac{w_i}{v_j} \leq 2\lambda$
- If there is no such index j then the algorithm fails.

Claim: If $OPT \leq \lambda$ then the algorithm never fails (i.e. we would find an assignment to all jobs) and it is 2-approximation.

Note: how do we choose λ ?

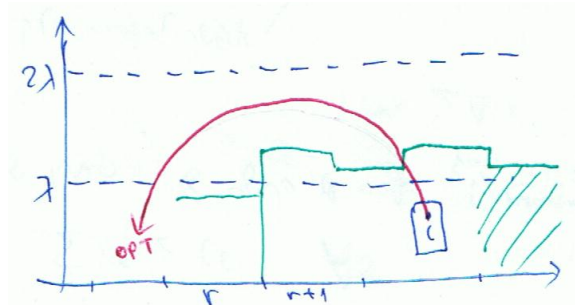
Binary search of the solution.

We run the algorithm. If it succeeds then we divide λ by 2 and run again to see if the algorithm still succeeds. If it fails then we multiply λ by 2 etc.

The boundaries of the binary search: $\sum \frac{w_i}{v_j} \leq OPT \leq \sum \frac{w_i}{v_{max}}$. The ratio between the lower and upper bound is maximum m .

The binary search can be replaced for an online algorithm with additional loss of factor 4.

Proof:



Assume in the contrary that the algorithm fails. We look at the time it fails:

For each job $\frac{w_i}{v_m} \leq \lambda$ and hence $l_m > \lambda$.

It can't be true that all loads of the machines are greater than λ , because if it was true, then according to volume preservation law we would get that $OPT > \lambda$.

Therefore there must be a machine with index r with load $\leq \lambda$.

We denote with r the maximum index of such a machine with load $\leq \lambda$.

According to volume preservation law there is a job i that the algorithm assigned to machine with index larger than r and OPT has assigned to a machine with index $\leq r$.

Since the machines are sorted by speed $\frac{w_i}{v_r} \leq OPT$ and hence job i may be assigned on machine r . It is surely "gets into it" since there is a "gap" $\leq \lambda$.

We got a **contradiction** since OPT could assign to a machine with index r or smaller but it has assigned it to a machine with index $> r$.

Note:

It is possible to get a PTAS in this case too (later in the course...)