IIIS 2014 Spring: ATCS - Selected Topics in Optimization
Lecture date: Mar 13, 2014
Instructor: Jian Li TA: Lingxiao Huang Scribe: Yifei Jin
(Polishing not finished yet.)

1 Unrelated Parallel Machine Scheduling

1.1 Mathematical foundation

Lemma 1 *If an linear programming which satisfies*

$$Ax = b, \qquad A \in R^{m \times n}$$
$$x \geq 0$$

is feasible, any vertex solution has at most m non-zero variables.

Proof: If x is a vertex solution, it satisfies n tight constraints of LP. The $Ax = b$ consists of m constraints, Therefore at least $n - m$ of the tight constrains are from $x \geq 0$, i.e., at least $n - m$ of the variables are set to zero. Thus any vertex solution has no more than m non-zero variables. □

In figure 1, we can see an example of Lemma 1. The left contains one constraint. It represents a plane intersects the first quadrant. The right contains two constrains. It represents a line intersects the first quadrant.

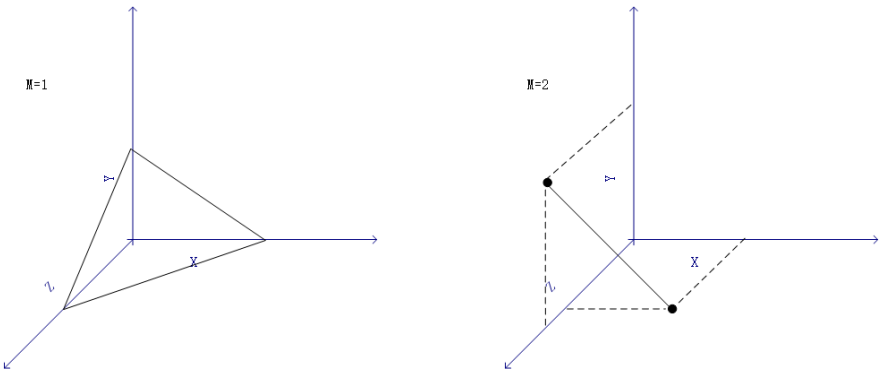

Figure 1: The example of lemma 1

1.2 Problem description

We have a set J of n jobs, and a set M of m machines. The processing time of job i on machine j is p_{ij}. We assign each job to exactly one machine. Define t_j as the total processing time of the jobs that are assigned to machine j. The makespan is the maximum time of all t_j. In the Scheduling on Unrelated Parallel Machines problem, the goal is to find an assignment of jobs to machines to minimize makespan. Figure 2 portrays the problem.

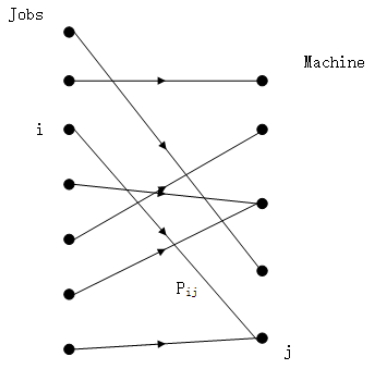

Figure 2: Unrelated Parallel Machine Scheduling Problem

1.3 Directly LP approximation algorithm

We can use Integer Programming to describe the problem as below:

$$\begin{aligned} \min \quad & t \\ s.t. \quad & \sum_{i=1}^{n} P_{ij} x_{ij} \leq t \\ & \sum_{j=0}^{m} x_{ij} = 1 \\ & x_{ij} = \{0, 1\} \end{aligned}$$

Here, t is makespan. x_{ij} represents whether job i is assigned to machine j or not. Other notation is the same as before. Our algorithm is simple. Just relax the Integer Programming to Linear Programming:

$$\begin{aligned} \min \quad & t \\ s.t. \quad & \sum_{i=1}^{n} P_{ij} x_{ij} \leq t \\ & \sum_{j=0}^{m} x_{ij} = 1 \\ & x_{ij} \geq 0 \end{aligned}$$

We define *Integrality Gap* as $\frac{LP}{IP}$. Then the gap can reach $\frac{1}{m}$. Figure 3 gives an example to show how to achieve the *Integrality Gap*. There is only one job and m machines. Each P_{ij} equals to one. Then the IP solution is 1, while LP solution is $\frac{1}{m}$.

`Question : The solution of LP problem may not be a feasible solution of origin problem. So we need to do random rounding after achieve an LP solution.`

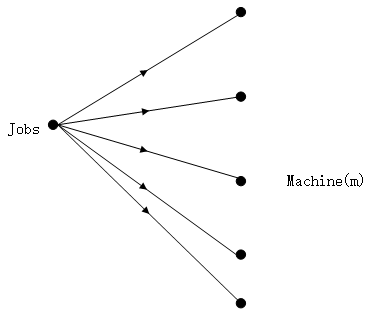

Figure 3: Worst Case of LP-approx

`Could we use some strategy, such as Chernoff bound, to improve the approximation ratio of the random rounding algorithm?`

It is unlikely to use random rounding to get a good approximation since the integrality gap is $\frac{1}{m}$. Think about the common strategy of random rounding. Suppose the solution of random rounding approximation is SOL. We use solution of LP which is denoted by LP to bound it. We have $SOL \leq k \cdot LP$.Since the optimal value of LP is no more than the optimal value of IP, we have $LP \leq OPT$. So we have $SOL \leq k \cdot LP \leq k \cdot OPT$. However if the integrality gap is as small as $\frac{1}{m}$, suppose our algorithm is d-approx algorithm, i.e., $SOL \leq d \cdot OPT$, but in the worst case, we have $SOL \leq d \cdot m \cdot LP$. It means through our analysis, we only get a $d \cdot m$-approx algorithm. Intuitively, $SOL \leq k \cdot LP$, the best SOL is OPT, $k \geq \frac{SOL}{LP} = \frac{IP}{LP} = m$. It means that our algorithm approximation would be no less than m.

1.4 Improved 2-approximation algorithm

1.4.1 Algorithm Description

The above LP is very natural. Unfortunately, it has unbounded integrality gap. To overcome this difficulty, we modify the LP slightly. Our strategy is to combine binary search and parametric pruning. Suppose we already know the optimal solution T_{opt}. We can ignore all P_{ij} which are larger than T_{opt}, i.e., we need to solve the following $LP(\lambda)$ problem:

$$\begin{aligned} \text{find} \quad & x \\ s.t. \quad & \sum_{i:(i,j)\in S_{T_{opt}}} P_{ij}x_{ij} \leq T_{OPT} \\ & \sum_{j:(i,j)\in S_{T_{opt}}} x_{ij} = 1 \\ & x_{ij} \geq 0 \end{aligned}$$

Here, $S_t = \{(i,j) |\ i \in J, j \in M, p_{ij} \leq t\}$.

Since we would not have the oracle of optimal value, we need to guess one for our algorithm and see whether it is feasible. We can use binary search to limit our guess t to the optima, since the interval of makespan is finite among $[\min_{i,j} P_{ij}, \sum_{ij} P_{ij}]$.

Formally, our algorithm is listed in algorithm 1

Algorithm 1: 2-approximation algorithm for Unrelated Parallel Machine Scheduling

Initially, let $t_1 = \min_{i,j} P_{ij} + \sum_{ij} P_{ij}$,$t_2 = 0$;
while $|t_1 - t_2| > \epsilon$ **do**
    solve the LP($\frac{t_1+t_2}{2}$) problem ;
    if *LP($\frac{t_1+t_2}{2}$) has solution* **then**
        set $t_2 = \frac{t_1+t_2}{2}$;
    else
        set $t_1 = \frac{t_1+t_2}{2}$;
Return t_1;

1.4.2 Algorithm Analysis

The $LP(\lambda)$ problem has at most $m + n$ constrains, but there has $n \times m$ variables. According to Lemma 1, we observe that in a vertex solution, the number of non-zero variables is no more than $m + n$. Then we ask a question: If we select a subset A in the job set J and a subset B in the machine set M. How many non-zero variables in the set $\{x_{i,j}|i \in A, j \in B\}$? A direct idea is no more than $|A| + |B|$.

Lemma 2 *In the Unrelated Parallel Machine Scheduling problem, we select a subset A in J, and a subset B in M. Then non-zero variables of set $\{x_{i,j}|i \in A, j \in B\}$ is no more than $|A| + |B|$*

Proof: It is easy to get that the solution of the problem

$$\begin{aligned} \min \quad & t \\ s.t. \quad & \sum_{i \in A} P_{ij} x_{ij} \leq t - \sum_{i \in \bar{A}} P_{ij} x_{ij} \\ & \sum_{j \in B} x_{ij} = 1 - \sum_{j \in \bar{B}} x_{ij} \\ & x_{ij} \geq 0 \end{aligned}$$

is the same as the original one. Suppose $\{x_{ij}\}$ is the optima of original LP. If we use the value of $\{x_{ij}\}$ to fix the right hand side of the inequality. The optimal solution would not change. And if we fix the right hand side variables, the problem only relates to set A and B. According to lemma 1, we get that non-zero variables of set $\{x_{i,j}, i \in A, j \in B\}$ is no more than $|A| + |B|$. □

Then we prove our algorithm is 2-approx. We first define the *almost tree*.

Definition 3 (almost tree) *A graph is an almost tree if every biconnected component has the property that there is at most 1 edge not in a spanning tree of this biconnected component.*

According to Lemma 2, it is easy to get the graph of solution $\{x_{ij}\}$ is an almost tree. Two inequality must hold based on our algorithm:

- $x_{ij} \neq 0$, $P_{ij} \leq t$

- for any machine $\sum P_{ij}x_{ij} \leq t$

At first we see the tree condition. Then we see the almost tree condition.

Tree See left part of figure 4. If the leaf node is machine, then in the optimal solution, the connected job need to be assigned to the machine. If the leaf node is job, it needs to be assigned to its father node(machine) in optimal solution. The machine's father node(a job) can be assigned to any machine. So the makespan is no more than $\sum P_{ij}x_{ij} + p'_{ij}$

$$makespan \leq (\sum P_{ij}x_{ij}) + p'_{ij} \leq \sum P_{ij}x_{ij} + t \leq 2t$$

Almost Tree Since we having process the tree condition, in almost tree, we only need to process the circle. Note that in this problem environment, any circle contains even nodes. If any node in the circle has been assigned to any node apart from the circle, then the circle degenerated into a path, the following proof is the same as a tree. If none of the node of the circle being assigned to other node, assign one job to one machine is optimal solution. So the optima is no more than 2t.

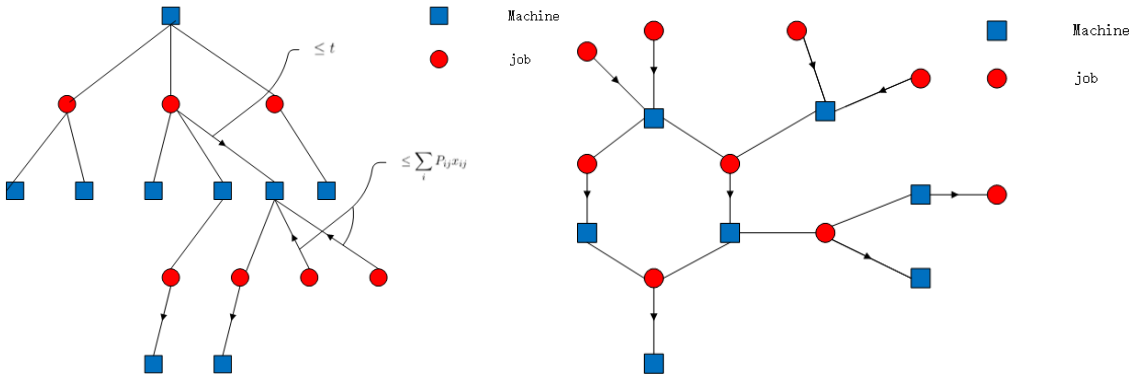

Figure 4: job assignment on almost tree

```
Note : This algorithm need to use binary search which is related to the length of
interval. But because the interval is from a positive number (denoted by a) to
a larger positive number (denoted by b). Without loss of generality, we can
search interval [0,log(b/a)]. So, although the algorithm is not strong polynomial,
it is weakly polynomial time.
```

2 Beck Fiala Theorem

2.1 Problem and Definition

Think about the *Discrepancy* problem we talk in lecture 2. We are given a set $\mathcal{U}$ of n elements and a family $\mathcal{F}$ of m subsets of $\mathcal{U}$. Each subset in $\mathcal{F}$ consists of several elements. We need an polynomial time algorithm to color each element in $\mathcal{U}$ into two colors, red(+1) and blue(-1), so that for any subset $S \in \mathcal{F}$, $|R(S) - B(S)|$ is as small as possible.

In the previous lecture, we use Chernoff bound to give an algorithm to make $|R(S) - B(S)| \leq O(\sqrt{n \log n})$. In this section, we prove $|R(S) - B(S)| \leq deg(\mathcal{H})$. Before our topic, we give some notations and definitions.

- $deg(\mathcal{H})$: We give a collections of subsets, denoted by $\mathcal{H}$. Denote t_i be the number of occurrence of element i. Then $deg(\mathcal{H})$ is the maximum of all t_i.
- Define $Disc(S)$ be $|R(S) - B(S)|$
- Define $Disc$ be $\max_S Disc(S)$

Then we have Beck Fiala Theorem below:

Theorem 4 (Beck Fiala Theorem) *Let $\mathcal{H}$ be any set system where no element is in more than $deg(\mathcal{H})$ sets. Then $Disc \leq 2deg(\mathcal{H})$*

2.2 Algorithm and Proof

As we using random rounding algorithm to prove the $O(\sqrt{n \log n})$ bound, we give a linear programming algorithm to prove the Beck Fiala Theorem. We define our LP problem below:

For any element $+1/-1$ integer x_i, we define a corresponding real number $y_i \in [-1, 1]$. Our feasibility linear programming problem (FP-disc) is :

$$\begin{aligned} \text{find} \quad & y \\ s.t. \quad & \sum_{x_i \in S_1} y_i = 0 \\ & \sum_{x_i \in S_2} y_i = 0 \\ & \vdots \end{aligned}$$

At first, we ignore set(edges) with size no more than $deg(\mathcal{H})$. In this condition, we have $m < n$. Because, considering the pairs (vertex,edge), any vertex appears no more than d times. So, $\#(vertex, edge) \leq n \cdot d$. And any edge has more than d vertices since we ignore smaller ones. Thus $\#(vertex, edge) > m \cdot d$. We solve the FP-disc problem in this condition. The feasible region is a k-dimension flat in R^n, just like figure 5. According to lemma 1, we have $n - k \geq n - m \geq 1$. So, when m is larger than zero, the problem FP-disc is feasible, and we can find a vertex solution. Based on the vertex solution, we can get at least one $y_i = -1/+1$. Fix those y_i, and ignore edges with active(not fixed) y size no more than d. Each time we decrease n by one and promise $n > m$. So at most through n iterations, the algorithm halts. The formal algorithm is 2.

Since the feasibility linear programming problem FP-disc always containing feasible solution, the solution y would not import error in the LP part. But each step we remove the edges with size d. The d vertex is arbitrary with respect to the optimal solution. So the maximum error is $2d$, i.e., $Disc \leq 2deg(\mathcal{H})$

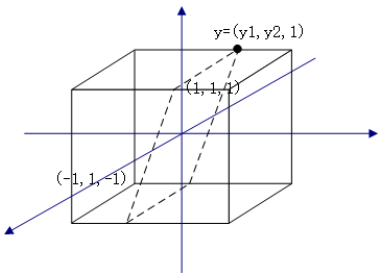

Figure 5: feasible region in R^n

Algorithm 2: LP for Discrepancy problem

```
Initially, set d = deg(H) ;
ignore edges with size ≤ d ;
count the edges number, denoted by m ;
count the vertex number, denoted by n ;
while m ≤ 0 do
    solve a vertex solution of the FP-disc problem ;
    fix value of one y_i = +1/ − 1 ;
    n ← n − 1 ;
    remove the edges with size ≤ d, update m ;
Return y ;
```

3 Matching Market

In economics, matching theory, also known as search and matching theory, is a mathematical framework attempting to describe the formation of mutually beneficial relationships over time. Matching theory has been especially influential in labor economics, where it has been used to describe the formation of new jobs such as kidney exchanges, as well as to describe other human relationships like marriage, roommate selection and so on.

In this section, we talk about a bipartite matching problem: n house sellers and n buyers. Our goal is to make each buyer take a house, meanwhile maximizing the total profit.

3.1 Valuations and Optimal Assignments

As Figure 6, we extend bipartite graph to introduce some additional features. First, we allow each individual to express how much they'd like each house in numerical form. Each buyer provides a numerical score for each houses. Define this score as the buyer's *valuations* for the respective house.

Using these valuations, we can evaluate the quality of an assignment of houses to buyers, just summing their valuation for what they get. We define *optimal assignment* as maximum possible quality. We can check that the assignment in right part of figure 6 is in fact the optimal assignment for the setting of valuations. Note that while the optimal assignment maximizes total happiness, it does not necessarily give everyone his favorite item.

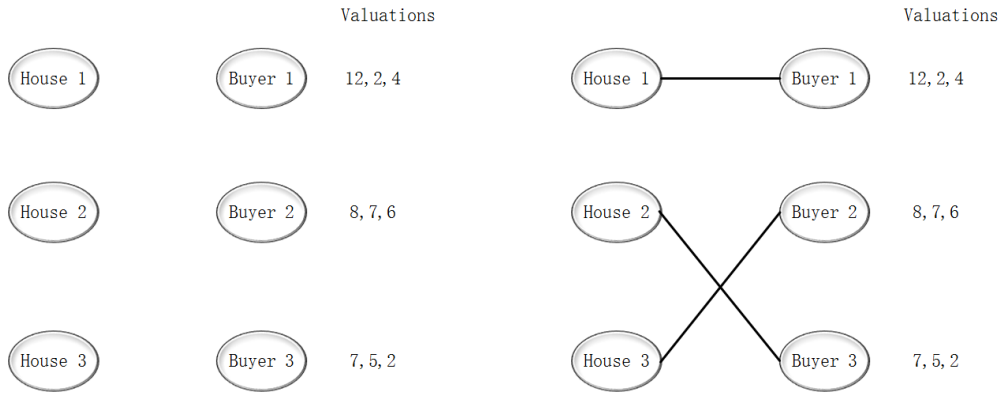

Figure 6: Matching Market. The left one is a setting of valuations. Each person's valuatios for the objects appears as a list. The right part is an optimal assignment with respect to these valuations

3.2 Prices and the Market-Clearing Property

Think of the real market, we do not have a central "administrator" who determines an optimal assignment. A more standard picture of a market involves much less central coordination, with individuals making decision based on price and their own valuations.

Suppose that each seller i puts his house up for sale, offering to sell it for a price $p_i \geq 0$. If a buyer j buys the house from seller i at this price, we will say that the buyer's payoff is her valuation for this house, minus the amount of money she had to pay: $v_{ij} - p_i$. So given a set of prices, if buyer j wants to maximize her payoff, she will buy from the seller i when the quantity $v_{ij} - p_i$ is maximized. We will see that if we replace the role of the central administrator by a particular scheme for pricing items, then allowing individuals to follow their own self-interest based on valuations and prices can still produce optimal assignments.

We call the seller or sellers that maximize the payoff for buyer j the *preferred sellers*. Given the possibility of ties, for a set of prices, we define the preferred-seller graph on buyers and sellers by simply constructing an edge between each buyer and her preferred seller or sellers.

If each buyer simply claims the house that she likes best, each buyer ends up with a different house, We call such a set of prices *market-clearing*. In Figure 7. We can see the three conditions about market-clearing prices.

3.3 Properties of Market-Clearing Prices

In a way, market-clearing prices feel a bit too good to be true: if sellers set prices by the right way, then self-interest runs its course and all the buyers get out of each others way and claim different houses. We've seen that such prices can be achieved in Figure 7. In fact, we have the following more general theorem:

Theorem 5 (Properties of Market-Clearing Prices)

- *Existence of Market-Clearing Prices: For any set of buyer valuations, there exists a set of market-clearing prices.*

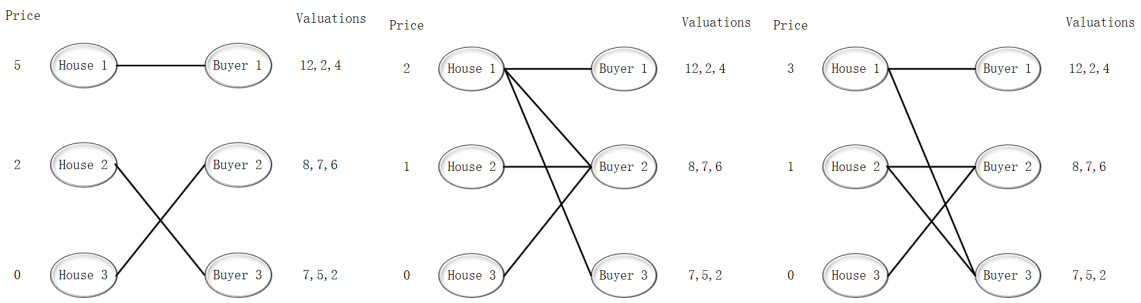

Figure 7: Matching Market: left: three sellers and three buyers, The result set of edges is the preferred-seller graph (PSG) for prices(5,2,0). It is also a perfect matching, i.e., the prices are market-clear prices. middle: The PSG for prices(2,1,0), there is no perfect matching, i.e., price don't clear the market. right: The PSG for prices(3,1,0). It is market-clear prices. The perfect matching is the same as the left figure.

- *For any set of market-clearing prices, a perfect matching in the resulting preferred-seller graph has the maximum total valuation of any assignment of sellers to buyers*

Proof: The Matching Market problem can be formalized as a LP problem:

$$\begin{aligned} \max \quad & \sum_{ij} v_{ij} x_{ij} \\ s.t. \quad & \sum_{i} x_{ij} \leq 1 \\ & \sum_{j} x_{ij} \leq 1 \\ & x_{ij} \geq 0 \end{aligned}$$

Here, x_{ij} represents whether house i is sold to buyer j. Its dual problem is:

$$\begin{aligned} \min \quad & \sum_{i} p_i + \sum_{j} q_j \\ s.t. \quad & p_i + q_j \geq v_{ij} \\ & p_i, q_j \geq 0 \end{aligned}$$

The primal problem has an optimal solution, since max-weight matching exists in complete bipartite graph. Suppose $\{x_{ij}\}$ is the optimal solution of primal problem. According to strong duality theorem, the dual problem has optimal solution. Then $\sum_{ij} v_{ij} x_{ij}$ is the max-weight matching. We have dual optima $\{p_i\}$ $\{q_j\}$. It means $\sum_i p_i + \sum_j q_j$ equals to max-weight matching.

We claim p_i is market clearing prices. Because:

$$\begin{aligned} q_j \geq v_{ij} - p_i, \forall i, \\ q_j \geq 0; \end{aligned}$$

So

$$q_j = \max(0, \max_i(v_{ij} - p_i))$$

It is the definition of payoffs exactly. According to complementary slackness,

$$x_{ij} = 1 \Longleftrightarrow q_j = v_{ij} - p_i$$

It means that i is the most preferred seller for buyer j and buyer j select house i. Thus (i, j) belongs to PSG. Above all, For any set of buyer valuations, there exists a set of market-clearing prices.

Since market clearing price is a solution of dual problem. $\sum_i p_i + \sum_j q_j = \sum_i (v_{ij} - p_j) + p_j = \sum_{ij} v_{ij} x_{ij}$. Moreover the max-weight is unique in primal problem. For any market clearing price, any perfect matching in PSG is a max-weight matching.

□

4 Primal-Dual for approximation algorithm

We have already used LP to get approximation algorithm for some problems, such as discrepancy. The main technique is random rounding. But the random-rounding algorithms need to solve the LP first. This is expensive and also not very insightful. Sometimes, the primal problem has no solution, but its dual has. So, we can use primal-dual scheme to solve a feasible solution of dual first. And relax some conditions to find a feasible solution of prime.

The primal-dual design scheme aims to address directly the combinatorial structure of the underlying problem by alternating between the primal and dual versions, connected by the complementary slackness conditions. If it is successful, the method leads to efficient and sophisticated algorithms.

4.1 Relaxed Complementary Slackness

Consider a primal-dual pair of LPs:

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \forall i \\ & x_j \geq 0 \end{aligned}$$

and

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} b_i y_i \\ s.t. \quad & \sum_{i=1}^{m} a_{ij} y_i \leq c_j, \forall j \\ & y_i \geq 0 \end{aligned}$$

Let $\alpha, \beta \geq 1$.

- relaxed primal complementary slackness:
  for each $1 \leq j \leq n, x_j \neq 0 \Longrightarrow \frac{c_j}{\alpha} \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$ (1)
- relaxed dual complementary slackness:
  for each $1 \leq i \leq m, y_i \neq 0 \Longrightarrow b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta b_i$ (2)

Theorem 6 (Relaxd Complementary Slackness) *Let $x = (x_1, ..., x_n)$ and $y = (y_1, ...y_m)$ be primal and dual feasible solutions satisfying conditions* (1) *and* (2) *above. Then*

$$\sum_{j=1}^{n} c_j x_j \leq \alpha \cdot \beta \cdot \sum_{i=1}^{m} b_i y_i.$$

Proof:

$$\begin{aligned} \sum_{j=1}^{n} c_j x_j &\leq \alpha \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j \\ &= \alpha \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \\ &\leq \alpha\beta \sum_{i=1}^{m} b_i y_i \end{aligned}$$

□

4.2 Hitting Set Problem

We are given a ground set $E = \{e_1, e_2, ..., e_n\}$ and a series of subsets $T_1, T_2, ...T_m \subseteq E$. Each element e_i has positive weight c_i. Our goal is to find the minimum weight set A, which satisfies $A \subseteq E$ and $A \cap T_i \neq \emptyset, \forall i$

```
Note that: set covering is equivalent to the hitting set problem. It is
easy to see this by observing that an instance of set covering can be
viewed as an arbitrary bipartite graph, with sets represented by vertices
on the left, the universe represented by vertices on the right, and edges
representing the inclusion of elements in sets. Then the task is to find
a minimum cardinality subset of left-vertices which covers all of the
right-vertices. In the Hitting set problem, the objective is to cover the
left-vertices using a minimum subset of the right vertices. Converting from
one problem to the other is therefore achieved by interchanging the two
sets of vertices.
```

It is easy to formulate hitting set problem to Integer programming as

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ s.t. \quad & \sum_{e \in T_i} x_e \geq 1 \\ & x_e \in \{0, 1\} \end{aligned}$$

Relax to LP:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ s.t. \quad & \sum_{e \in T_i} x_e \geq 1 \\ & x_e \geq 0 \end{aligned}$$

Its dual is:

$$\begin{aligned} \max \quad & \sum_{i} y_i \\ s.t. \quad & \sum_{i, e \in T_i} y_i \leq c_e \\ & y_i \geq 0 \end{aligned}$$

Instead solving the dual problem which would just give a lower bound on the optimal integral value OPT, we start with an initial primal-dual solution pair (x, y) and alternately improve both components, guided by the complementary slackness conditions.

Our strategy starts from a primal-dual solution pair (x, y), such as $(0, 0)$, which is primal infeasible and dual feasible but not optimal. We take care that the dual solution y stays feasible all the time and increase the x values towards making the primal solution feasible. When the primal solution becomes feasible, we stop and have a solution. The formal algorithm is listed in 3.

Algorithm 3: Primal-dual approximation for Hitting Set Problem

```
Initially E, c_e, T_i, (x, y) = (0, 0);
while x is not feasible for Primal do
    Select any T_i that is not hit ;
    Raise y_i until some constraint is tight ;
    Pick this element, i.e., set x_e = 1 ;
Return sum_{x_e=1} c_e;
```

Claim 7 *The solution is d-approximation, where* $d = \max_i |T_i|$.

Proof: Suppose the elements we have chosen are $\{A_i\}$, Then:

$$\begin{aligned}\sum_{e\in A} c_e &= \sum_{e\in A}\sum_{i,e\in T_i} y_i \\ &= \sum_{i}\sum_{e\in|A\cap T_i|} y_i \\ &= \sum_{i} y_i|A\cap T_i| \\ &\leq OPT\cdot\max_i|T_i| = d\cdot OPT\end{aligned}$$

□

Through relaxed complementary slackness, we can know the intuition of the proof. It is possible that subset $T_i \subseteq E$ for which $y_i > 0$ being hit multiple times, i.e. $\sum_{e\in T_i} x_e \geq 1$, up to $\sum_{e\in T_i} x_e = d$, where $d = \max_i |T_i|$. According to Theorem 6, $\alpha = 1, \beta = d$, we get $\sum_{e=1}^{n} c_e x_e \leq d\cdot\sum_i y_i \leq d\cdot OPT$

References

[1] Alina Ene, Lecture notes: Approximation Algorithms Scheduling on Unrelated Parallel Machines

[2] David Easley and Jon Kleinberg: Networks, Crowds, and Markets: Reasoning about a Highly Connected World.

[3] Thomas Rothvoß Discrepancy theory Or: How much balance is possible?

[4] Topic 1: Advanced Course in Algorithms of Aalto University

[5] Atila Abdulkadiroglu, Tayfun Sonmez: Matching Markets: Theory and Practice

[6] BeckCFiala theorem: http://en.wikipedia.org/wiki/Beck%E2%80%93Fiala_theorem