# Network Design Foundation
# Fall 2011
# Graph Balancing : A special case of scheduling on unrelated machines

Manish Purohit, Avinash Das

December 6, 2011

## 1 Introduction

Here, we consider approximation algorithms for the Graph Balancing problem which is a special case of the job scheduling on unrelated parallel machines to minimize the makespan. We present an interpretation of the 2-approximation LP-rounding algorithm given by Lenstra et al. to the special case of graph balancing. We then show how the approximation guarantee can be improved by the technique of adding additional valid constraints to the linear program.

## 2 Generalized Problems

### 2.1 Scheduling on unrelated parallel machines

In this scheduling problem, we are given $n$ machines and $m$ jobs which need to be scheduled on these machines. Each job needs to be assigned to one of the machines. Job $j$ requires processing time $p_{ij}$ on machine $i$. The objective is to find a schedule that minimizes the makespan. Makespan is the time at which the last job finishes. It can also be defined as the load on the most heavily loaded machine. Lenstra *et al.* [2] gave an elegant 2-approximation for this problem and also proved that approximating it with an approximation ratio better than 1.5 is NP-hard.

### 2.2 Restricted Assignment

Restriced Assignment is a special case in which for a particular job $j$, the $p_{ij}$ entries are either $\infty$ or the same value $p_j$. Thus, a job has fixed processing time irrespective of the machine but can only be scheduled on a given subset of the machines.

## 3 Graph Balancing

Graph Balancing [1] is a special case of the Restricted Assignment problem in which every job can be scheduled on at most 2 machines. The problem is called *Graph Balancing* as one can consider the machines as vertices of a graph and the jobs as undirected edges between them. The problem would now be to simply direct the edges towards one of its end points. Further, in order to account for jobs that can be scheduled on one machine alone, we assign certain dedicated loads to each vertex.

Even this special case of graph balancing is NP-hard to approximate with a ratio better than 1.5.

### 3.1 Problem Formulation

The graph balancing problem can thus be formalized as shown. Given a multigraph $G = (V, E, p, q)$ where $V$ are vertices corresponding to machines and $E$ are the edges corresponding to the jobs respectively. Weights $p_e$ on edges denote the processing times while weights $q_v$ on vertices denote their dedicated loads. The desired output is an orientation of edges, which is defined as a mapping $\lambda : E \rightarrow V$ such that $\lambda(e)$ is incident with $e$ for each $e \in E$. The objective is to minimize the maximum load on any vertex where load on a vertex is defined as as the sum of its dedicated load ($q_v$) and the $p_e$ of edges oriented towards it (i.e. $\lambda(e) = v$).

### 3.2 Relaxed decision procedure

Using binary search for the optimal makespan $d$ and scaling all weights appropriately, we can convert the optimization problem to its decision version where we need to test if there exists an orientation with maximum load at most 1. It can be easily seen that if we have a $\rho$-relaxed decision procedure for the problem, then we obtain a $\rho$-approximation algorithm. A $\rho$-relaxed decision procedure is one that when asked if there exists any orientation with maximum load at most 1 returns "NO" if no such orientation exists and otherwise returns an orientation with maximum load at most $\rho$.

# 4 Approximation Algorithms

We present an interpretation of the 2-approximation of Lenstra et al. for the scheduling on unrelated machines to the special case of graph balancing. We then note a very simple deterministic rounding scheme to obtain 2-approximation. Finally, we show a 1.75 approximation for the graph balancing problem as obtained by Ebenlendr et al.

## 4.1 2-approximation - Lenstra et al

Consider a decision version of the problem where we are interested to know if there is a feasible solution with maximum load 1. A natural LP formulation of the same is given below.

**Linear Program LP1**

Find values $x_{ev} \geq 0$ for each $e \in E$ and $v \in e$, subject to:

For each edge $e \in E$, $u, v \in e$ :
$x_{eu} + x_{ev} = 1$
For each vertex $v \in V$ :
$q_v + \sum_{e:v \in e} x_{ev} p_e \leq 1$

Its a feasibility LP with no objective function. For every edge $e = (u, v)$, we have two variables, $x_{eu}$ and $x_{ev}$ denoting the orientation. The first constraint simply says that each edge needs to be oriented towards one end point. The second constraint ensures that the maximum load on any vertex is at most 1. Of course, its hard to find integral solutions for the variables satisfying these constraints. Hence, we solve the LP in polynomial time to obtain the fractional $x_{ev}$ values. In case the LP has no feasible solution, we surely do not have any integral solution satisfying the constraints and we can return "NO".

The interesting case occurs when the LP does return feasible fractional $x_{ev}$ values. Now, we need to obtain an integral solution so that the maximum load on any vertex does not exceed 2. This rounding is done in two steps, *Rotation* and *Tree assignment*.

*Rotation*: Arbitrarily choose a cycle C in the graph that has all edges fractionally oriented and orient it in an arbitrary direction and perform the rotation operation. Rotation modifies the solution $x$, so that it is still feasible and the number of integral values in $x$ increases. Repeat this procedure until

the fractionally oriented edges induce a forest, i.e. there are no fractional cycles. The **Rotate** procedure is as follows. Here, the $x_{eu}$ and $x_{ev}$ values are perturbed along the cycle so that one of the edges gets completely oriented and the LP still remains feasible.

**Algorithm 1:** Rotate Procedure

**Procedure Rotate** $(x, C)$

**foreach** *edge* $e$ *in* $C$*, where* $e$ *is directed from* $u$ *to* $v$ **do**
  $\delta_e = x_{eu} p_e$
**end**
$\delta = \min_{e \in C} \delta_e$
**foreach** *edge* $e$ *in* $C$*, where* $e$ *is directed from* $u$ *to* $v$ **do**
  $x_{eu} = x_{eu} - \frac{\delta}{p_e}$
  $x_{ev} = x_{ev} + \frac{\delta}{p_e}$
**end**
**return x**

*Tree Assignment*: Once the graph induced by the fractional edges forms a forest, root every tree at an arbitrary vertex and direct all edges away from the root. Since, after the rotation process the LP is still feasible, the load on every vertex is at most 1. Hence, as figure 1 indicates, after tree assignment, the load on any vertex can be at most 2. Further, all edges are now directed towards one of their end points. We thus obtain a feasible solution to the graph balancing problem with maximum load at most 2 and hence obtain a 2-approximation.

### 4.2 2-approximation - Deterministic LP-rounding

For the special case of graph balancing, the LP-rounding procedure described above is overkill. Since for every edge $e = (u, v)$, we have $x_{eu} + x_{ev} = 1$, it is guaranteed that either $x_{eu} \geq \frac{1}{2}$ or $x_{ev} \geq \frac{1}{2}$. Breaking ties arbitrarily, orient the edge towards the end-point with larger fractional orientation, i.e. direct the edge towards $u$ if $x_{eu} \geq \frac{1}{2}$ and vice versa. Let $X_{eu}$ and $X_{ev}$ be indicator variables to denote if the edge $e$ is directed towards $u$ or $v$.

This simple rounding scheme is also a 2-approximation. Consider,

$$\mathsf{Newload}(v) = q_v + \sum_{e : v \in e} X_{ev} p_e$$

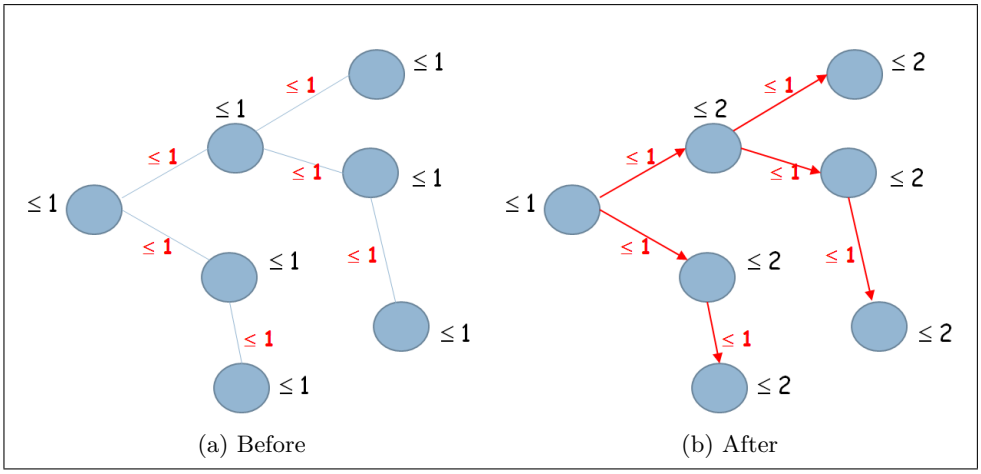


Figure 1: Tree Assignment

But, $X_{ev} \leq 2x_{ev}$ as per the rounding scheme

$$\begin{aligned}
\therefore \text{Newload}(v) &\leq q_v + 2\sum_{e:v\in e} x_{ev}p_e \\
&\leq 2(q_v + \sum_{e:v\in e} x_{ev}p_e) \\
&\leq 2\text{load}(v) \\
\therefore \text{Newload}(v) &\leq 2
\end{aligned}$$

However, the rounding scheme of Lenstra et al. is important as their approach can be extended to obtain a better approximation ratio of 1.75.

### 4.3 Integrality Gap of LP1

Unfortunately, the integrality gap of the LP1 considered above is 2. As shown in figure 2, consider a long path of edges each of weight $(1-\epsilon)$. Let both the end points of the path have dedicated loads of 1. It is easy to see that if the length of the path is more than $\frac{1}{\epsilon}$, LP1 is feasible but the best integral solution has maximum load $2-2\epsilon$. Hence it is not possible to obtain a better approximation for the graph balancing problem by simply strengthening the analysis.

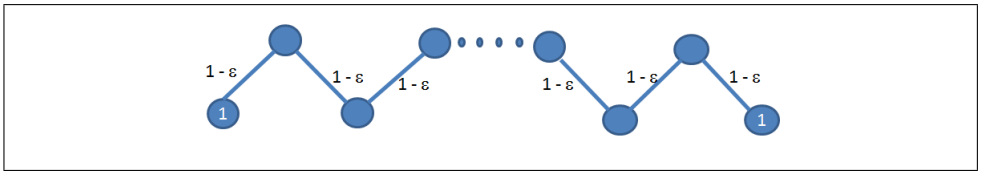

Figure 2: Integrality Gap of LP1

### 4.4 1.75 approximation

The essence of graph balancing lies in the critical observation of orientation of big edges. Big edges are defined as edges with $p_e > \frac{1}{2}$. If 2 big edges are oriented to a vertex, load of vertex will be more than 1. Therefore, the orientation of big edges and vertices will be one to one on a graph induced with big edges.

#### 4.4.1 Pre-processing

Let us discuss some notation that will be used for this document. Given a weighted graph $G = (V, E)$, $E^B = \{e \in E : p_e > 1/2\}$ be set of big edges and $G^B = (V, E^B)$, be the graph induced by the set of big edges. Further, given a fractional solution of a LP, $E_x = \{e \in E : 0 < x_{eu} < 1\}$ be set of edges for which LP gives a fractional solution and $G_x = (V, E_x)$, be the graph induced by these fractional edges. Composing, $G_x^B = (V, E^B \cap E_x)$ is a graph induced by the set of big edges with fractional assignments. Finally, given a tree $T = (V, E)$, a leaf pair is defined $L(T) = \{(v, e) \in V \times E : v \in V, deg(v) = 1, v \in e\}$.

A one to one orientation on a cycle in $E^B$ implies that if there is a vertex $v \in G^B$ which is not the part of the cycle but is connected to the cycle by a big edge $e$, then $e$ must be oriented away from the cycle in any integral feasible solution as shown in figure 3. Thus, we can preprocess the input instance to remove all such edges from the graph and assign their weights as dedicated load on the vertices. The graph induced by the big edges, $G^B$, would thus contain only disjoint cycles and trees.

#### 4.4.2 Tree constraint

Let us discuss the implication of big edges on some tree $T \subseteq G^B$. The number of edges in a tree is one less than number of the vertices. One to one mapping of edges to vertices implies that there would be exactly one vertex which would not be assigned any big edge. Therefore there can be at

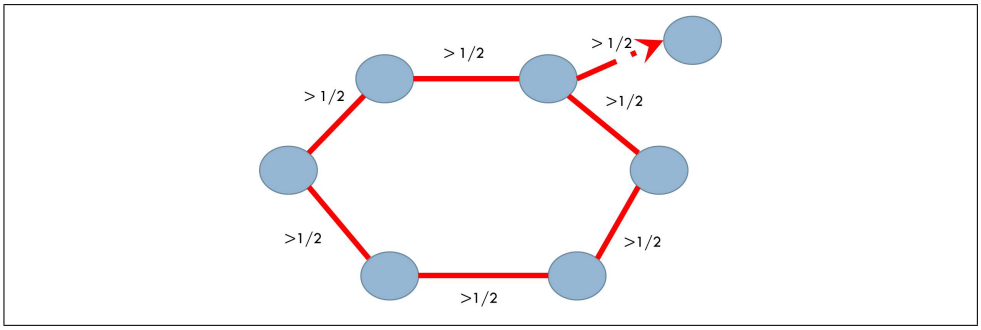

Figure 3: Illustration of a cycle in $G^B$

most one leaf in the tree that does not have a big edge oriented towards it. Therefore, any integral solution of the LP1 will follow:

$$\sum_{(v,e)\in L(T)} x_{ev} p_e \geq \left( \sum_{(v,e)\in L(T)} p_e \right) - 1$$

The LP1 can thus be reformulated to:

**Linear Program LP2**

Find values $x_{ev} \geq 0$ for each $e \in E$ and $v \in e$,
subject to:

For each edge $e \in E$, $u, v \in e$ :
$x_{eu} + x_{ev} = 1$ (Edge $e$)
For each vertex $v \in V$ :
$q_v + \sum_{e: v \in e} x_{ev} p_e \leq 1$ (Load at v)
For each tree $T \subseteq G^B$ :
$\sum_{(v,e)\in L(T)} x_{ev} p_e \geq \sum_{(v,e)\in L(T)} p_e - 1$ (Tree T)

The LP2 has exponential constraints for the tree condition. It can be solved using methods like the ellipsoid algorithm. Ebenlendr et al.[1] give a rounding scheme that gives a 1.75 approximation algorithm for graph balancing. The **Round** procedure given below describes their rounding scheme.

**Algorithm 2:** Round Procedure

**Procedure ROUND** $(G = (V, E, p, q), x)$
**while** $G_x$ *has an edge* **do** `// Main While`
  **if** $G_x$ *has a leaf pair* $(v, e)$ **then**
    Let $u$ be the vertex $u \in e$, $u \neq v$,
    **if** $x_{eu} p_e \leq 0.75$ **then** `// Leaf Assignment`
      $(x_{ev}, x_{eu}) := (1, 0)$
    **end**
    **else** `// Tree Assignment`
      Let $T = (V', E')$ be the component of $G_x^B$ containing $e$
      **foreach** $e' \in E', v', u' \in e'$ *such that* $v'$ *is on the path from* $v$ *to* $u'$ *in* $T$ **do**
        $(x_{e'v'}, x_{e'u'}) := (0, 1)$
      **end**
    **end**
  **end**
  **else** `// Rotation`
    Find a directed cycle C in the following way:
    Start a walk in an arbitrary vertex and **repeat**
      Append a new edge to the end of the walk; if possible, choose a big edge
    **until** *the walk contains a cycle, denote it* C.;
    $\mathtt{Rotate}(x, C)$
  **end**
**end**
Let $\gamma(e) := v$ for all pair $(e, v)$ with $x_{ev} = 1$
return $\gamma$

### 4.4.3 Analysis of round procedure

We will prove that the rounding procedure indeed gives an 1.75 approximation for the graph balancing problem by proving that following invariance is maintained for every iteration of the rounding procedure.

**Theorem 1** *Before and after each iteration of (Main While) during the round procedure for every vertex* $v \in V$ *:*

1. *The load of v is at most 1.75.*

2. *If $v$ is incident to any edge in $G_x$, it has load at most 1.25.*

3. *If $v$ is incident to a big edge in $G_x^B$, it has load at most 1.*

4. *For every tree $T$ that is a subgraph of $G_x^B$, the constraint (Tree $T$) is not violated.*

**Proof:** At the beginning all invariances are true, as LP gives a solution with load for each vertex is at most 1. Now, for each case we will verify if the invariance are all true:

- **(leaf assignment)**: As the edges are oriented toward the leaf, we need to check if the increase in load of leaf $v$ i.e. $(p_e - x_{ev}p_e)$ does not violate any invariance. We will consider 2 cases separately:

  1. $p_e > 1/2$: $p_e - x_{ev}p_e = x_{eu}p_e \leq 0.75$. As $v$ is connected to a big edge, $load(v) \leq 1$ before this iteration. Thus, after this iteration $load(v) \leq 1.75$. Further $v$ is no longer incident to any edge in $G_x$.
  2. $p_e \leq 1/2$: $p_e - x_{ev}p_e \leq pe \leq 0.5$. The load of $v$ can be at most 1.25 before the iteration. Thus, after this iteration $load(v) \leq 1.75$. Again, $v$ is no longer incident to any edge in $G_x$.

  Tree constraint is maintained during the leaf assignment. Therefore, all the invariances are maintained during the leaf assignment.

- **(tree assignment)**: $p_e > 1/2$. In the tree $T(V', E') \subseteq G_x^B : v \in L(T)$, consider any vertex $u' \in V'$. Let $e'$ be an edge on the path connecting $v$ to $u'$ in $T$ such that $u' \in e'$. As the edges are oriented away from the $v$ in the tree $T$, we need to check that the increase in $load(u')$ i.e. $p_{e'} - x_{e'u'}p_{e'}$ does not violate any invariance. Path joining $v$ to $u'$ in $T$ is also a subtree of $G_x^B$ and will hence follow the tree constraint $x_{ev}p_e + x_{e'u'}p_{e'} \geq p_e + p_{e'}$ - 1. Therefore:

$$\begin{aligned} p_{e'} - x_{e'u'}p_{e'} &\leq 1 - (p_e - x_{ev}p_e) \\ &\leq 1 - 0.75 = 0.25 \end{aligned}$$

  Therefore, increase in load of any vertex is at most 0.25. Since, before the iteration any vertex is connected to a big edge in $G_x^B$, its load will be at most 1. After the iteration it will be at most 1.25. Therefore, invariance is maintained during tree assignment.

- **(Rotation)**: As discussed in the previous section, the rotation does not change any of the relevant sum of loads hence first 3 invariances are maintained. Finally, it is easy to notice that even the tree constraint is not violated.

■

### 4.5 Integrality gap of LP2

The integrality gap of LP2 is indeed 1.75. This can be seen from the example given in figure 4 . There are three long disjoint paths between two terminal vertices each having odd number of edges. Every vertex has a dedicated load of 0.25 and the edge weights alternate between 1 and $0.5-\epsilon$ where the edges connected to the terminal nodes have weight 1. Again, for sufficiently long paths, the LP is feasible while any integral orientation will have to assign at least two edges to a vertex which leads to maximum load of $1.75-\epsilon$.

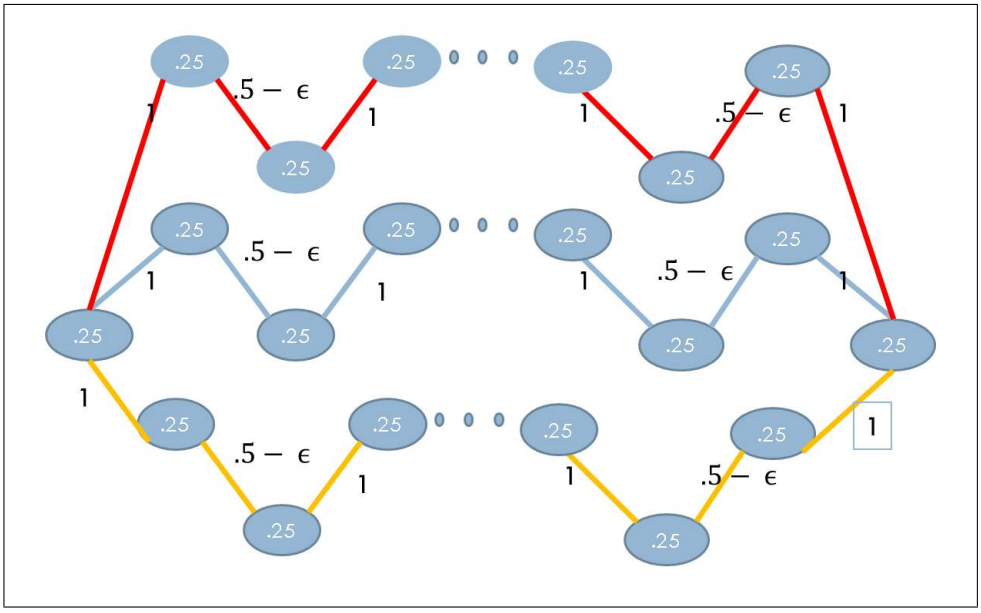


Figure 4: Integrality gap of LP2 is 1.75

### 4.6 Open Problems

Ebenlendr et al. show that it is NP-hard to approximate graph balancing with an approximation ratio better than 1.5 via a reduction from a variant of

3-SAT. It would be interesting to either obtain an approximation algorithm better than 1.75 or to improve the inapproximability bound. Since even LP2 has an integrality gap of 1.75, a more careful analysis of the same LP cannot be used to give a better approximation ratio. It may be possible to tighten the analysis by adding some more valid constraints to the linear program.

## References


[1] Tomáš Ebenlendr, Marek Křćal, and Jiří Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 483–490, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[2] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990. 10.1007/BF01585745.