LP Rounding

Design and Analysis of Algorithms
Andrei Bulatov

Linear Programming

Linear Programming

Instance

Objective function $z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

Constraints:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le b_2$$
$$\vdots$$
$$a_{m1}x1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m$$

Objective

Find values of the variables that satisfy all the constraints and maximize the objective function

Weighted Vertex Cover

Weighted vertex cover

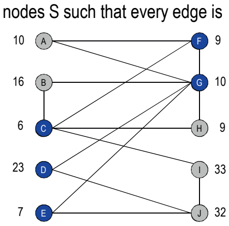
Instance

An undirected graph G = (V, E) with vertex weights $w_i \ge 0$

Objective

Find a minimum weight subset of nodes S such that every edge is incident to at least one vertex in S

total weight = 55



Weighted Vertex Cover: IP Formulation

Integer programming formulation.

– Model inclusion of each vertex i using a 0/1 variable x_i.

$$x_i = \begin{cases} 0 & \text{if vertex } i \text{ is not in vertex cover} \\ 1 & \text{if vertex } i \text{ is in vertex cover} \end{cases}$$

Vertex covers in 1-1 correspondence with 0/1 assignments:
$S = \{i \in V : x_i = 1\}$

– Objective function: minimize $\Sigma_i w_i x_i$.

– Must take either i or j: $x_i + x_j \ge 1$.

Weighted Vertex Cover: IP Formulation

$$\begin{array}{llcll} (ILP) \min & \sum_{i \in V} w_i x_i & & & \\ \text{such that} & x_i + x_j & \ge & 1 & (i,j) \in E \\ & x_i & \in & \{0,1\} & i \in V \end{array}$$

Observation.

If x^* is optimal solution to (ILP), then $S = \{i \in V : x^*_i = 1\}$ is a min weight vertex cover.

Integer Programming

Integer Programming

Given integers a_{ij} and b_i, find integers x_j that satisfy:

$$\begin{array}{lll} \max & c^t x & \\ \text{such that} & Ax & \ge b \\ & x & \text{integral} \end{array} \qquad \begin{array}{lcll} \sum_{j=1}^{n} a_{ij} x_j & \ge & b_i & 1 \le i \le m \\ x_j & \ge & 0 & 1 \le j \le n \\ x_j & & \text{integral} & 1 \le j \le n \end{array}$$

Observation.

Vertex cover formulation proves that integer programming is NP-hard search problem.

even if all coefficients are 0/1 and at most two variables per inequality

Compare to Linear Programming

Weighted Vertex Cover: LP Relaxation

Weighted vertex cover: Linear programming formulation.

$$
\begin{array}{llll}
(LP)\min & \sum_{i \in V} w_i x_i & & \\
\text{such that} & x_i + x_j & \geq 1 & (i,j) \in E \\
 & x_i & \geq 0 & i \in V
\end{array}
$$

Observation.

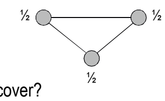
Optimal value of (LP) is less than or equal to the optimal value of (ILP).

Proof

LP has fewer constraints.

Weighted Vertex Cover: LP Relaxation

Note: LP is not equivalent to vertex cover.



How can solving LP help us find a small vertex cover?

Solve LP and round fractional values.

Weighted Vertex Cover

Theorem

If x^* is optimal solution to (LP), then $S = \{i \in V : x^*_i \geq ½\}$ is a vertex cover whose weight is at most twice the min possible weight.

Proof.

S is a vertex cover:

Consider an edge $(i, j) \in E$.

Since $x^*_i + x^*_j \geq 1$, either $x^*_i \geq ½$ or $x^*_j \geq ½$ implying (i, j) covered.

S has desired cost:

Let S* be optimal vertex cover. Then

LP is a relaxation

$x^*_i \geq ½$

$$\sum_{i \in S^*} w_i \geq \sum_{i \in S} w_i x_i^* \geq \tfrac{1}{2} \sum_{i \in S} w_i$$

Weighted Vertex Cover

Theorem

Linear Programming gives a 2-approximation algorithm for weighted vertex cover.

Theorem [Dinur-Safra, 2001]

If $P \neq NP$, then no ρ-approximation for $\rho < 1.3607$, even with unit weights.

$10\sqrt{5} - 21$

Open research problem.

Close the gap.

Generalized Load Balancing

Generalized Load Balancing

Instance

Set of m machines M; set of n jobs J.

Job j must run continuously on an authorized machine in $M_j \subseteq M$.

Job j has processing time t_j.

Each machine can process at most one job at a time.

Let J(i) be the subset of jobs assigned to machine i. The load of machine i is $L_i = \Sigma_{j \in J(i)} t_j$.

The makespan is the maximum load on any machine = $\max_i L_i$.

Objective

Assign each job to an authorized machine to minimize makespan.

GLB: Integer Linear Program

ILP formulation: x_{ij} denotes the time machine i spends processing job j.

$$
\begin{array}{lllll}
(IP)\min & L & & & \\
\text{such that} & \sum_i x_{ij} & = & t_j & \text{for all } j \in J \\
 & \sum_j x_{ij} & \leq & L & \text{for all } i \in M \\
 & x_{ij} & \in & \{0, t_j\} & \text{for all } j \in J \text{ and } i \in M_j \\
 & x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j
\end{array}
$$

GLB: Linear Program Relaxation

LP relaxation.

$$
\begin{array}{lllll}
(LP)\min & L & & & \\
\text{such that} & \sum_i x_{ij} & = & t_j & \text{for all } j \in J \\
 & \sum_j x_{ij} & \le & L & \text{for all } i \in M \\
 & x_{ij} & \ge & 0 & \text{for all } j \in J \text{ and } i \in M_j \\
 & x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j
\end{array}
$$

GLB: Lower Bounds

Lemma 1

Let L be the optimal value to the LP. Then, the optimal makespan $L^* \ge L$.

Proof.

LP has fewer constraints than IP formulation.

Lemma 2

The optimal makespan $L^* \ge \max_j t_j$.

Proof.

Some machine must process the most time-consuming job.

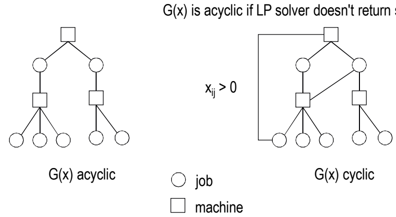
GLB: Structure of LP Solution

Lemma 3

Let x be solution to LP. Let G(x) be the graph with an edge from machine i to job j if $x_{ij} > 0$. Then G(x) is acyclic.

Proof. (deferred)

can transform x into another LP solution where G(x) is acyclic if LP solver doesn't return such an x



GLB: Rounding

Rounded solution: Find LP solution x where G(x) is a forest. Root forest G(x) at some arbitrary machine node r.

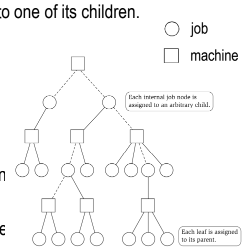
- If job j is a leaf node, assign j to its parent machine i.
- If job j is not a leaf node, assign j to one of its children.

Lemma 4.

Rounded solution only assigns jobs to authorized machines.

Proof.

If job j is assigned to machine i, then $x_{ij} > 0$. LP solution can only assign positive value to authorized machine



GLB: Lower Bounds

Lemma 5

If job j is a leaf node and machine i = parent(j), then $x_{ij} = t_j$.

Proof.

Since i is a leaf, $x_{ij} = 0$ for all $j \ne$ parent(i).

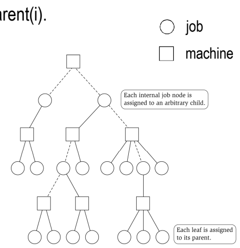
LP constraint guarantees $\Sigma_i x_{ij} = t_j$.

Lemma 6

At most one non-leaf job is assigned to a machine.

Proof.

The only possible non-leaf job assigned to machine i is parent(i).



GLB: Analysis

Theorem

Rounded solution is a 2-approximation algorithm

Proof.

Let J(i) be the jobs assigned to machine i.

By Lemma 6, the load L_i on machine i has two components:

- leaf nodes

$$\sum_{\substack{j \in J(i) \\ j \text{ is a leaf}}} t_j \overset{\text{Lemma 5}}{=} \sum_{\substack{j \in J(i) \\ j \text{ is a leaf}}} x_{ij} \overset{\text{LP}}{\le} \sum_{j \in J} x_{ij} \le \underbrace{L}_{\text{optimal value of LP}} \overset{\text{Lemma 1 (LP is a relaxation)}}{\le} L^*$$

- parent(i)

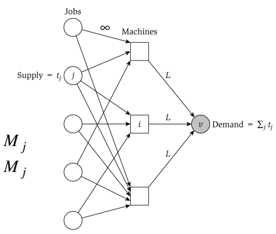
$$t_{\text{parent}(i)} \overset{\text{Lemma 2}}{\le} L^*$$

Thus, the overall load $L_i \le 2L^*$.

GLB: Flow Formulation

Flow formulation of LP.

$$\begin{array}{llll} \sum_i x_{ij} & = & t_j & \text{for all } j \in J \\ \sum_j x_{ij} & \le & L & \text{for all } i \in M \\ x_{ij} & \ge & 0 & \text{for all } j \in J \text{ and } i \in M_j \\ x_{ij} & = & 0 & \text{for all } j \in J \text{ and } i \notin M_j \end{array}$$



Observation.

Solution to feasible flow problem with value L are in one-to-one correspondence with LP solutions of value L.

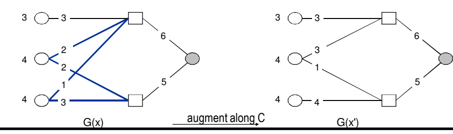
GLB: Structure of Solution

Lemma 3.

Let (x, L) be solution to LP. Let G(x) be the graph with an edge from machine i to job j if $x_{ij} > 0$. We can find another solution (x', L) such that G(x') is acyclic.

Proof. Let C be a cycle in G(x).

- Augment flow along the cycle C.
- At least one edge from C is removed (and none are added).
- Repeat until G(x') is acyclic.



Conclusions

Running time:

The bottleneck operation in our 2-approximation is solving one LP with mn + 1 variables.

Remark.

Can solve LP using flow techniques on a graph with m+n+1 nodes: given L, find feasible flow if it exists. Binary search to find L*.

Extensions:

unrelated parallel machines. [Lenstra-Shmoys-Tardos 1990]

- Job j takes t_{ij} time if processed on machine i.
- 2-approximation algorithm via LP rounding.
- No 3/2-approximation algorithm unless P = NP.