LP Rounding

Design and Analysis of Algorithms
Andrei Bulatov

30/11/2016

Linear Programming

Instance
Objective function: \( z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \)
Constraints:
- \( a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n \leq b_1 \)
- \( a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n \leq b_2 \)
- \( a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n \leq b_m \)

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

Weighted Vertex Cover

Instance
An undirected graph \( G = (V, E) \) with vertex weights \( w_i \geq 0 \)
Objective
Find a minimum weight subset of nodes \( S \) such that every edge is incident to at least one vertex in \( S \)

Weighted Vertex Cover: IP Formulation

Integer programming formulation.
- Model inclusion of each vertex \( i \) using a 0/1 variable \( x_i \).
- Vertex covers in 1-1 correspondence with 0/1 assignments: \( S = \{ i \in V : x_i = 1 \} \)
- Objective function: minimize \( \sum_i w_i x_i \).
- Must take either \( i \) or \( j \): \( x_i + x_j \geq 1 \).

Integer Programming

Given integers \( a_j \) and \( b_j \), find integers \( x_j \) that satisfy:
\[
\max \sum_{j=1}^{m} x_j b_j \quad \text{subject to} \quad \sum_{i=1}^{n} a_{ij} x_j \geq b_j, \quad 1 \leq j \leq m
\]
\[
x_j \geq 0, \quad 1 \leq j \leq m
\]

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

Compare to Linear Programming.
Weighted Vertex Cover: LP Relaxation

Weighted vertex cover: Linear programming formulation.

\[
(LP) \min \sum_{i \in V} w_i x_i \\
\text{such that} \quad x_i + x_j \geq 1 \quad (i, j) \in E \\
x_i \geq 0 \quad i \in V
\]

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

**Proof**
LP has fewer constraints.

Weighted Vertex Cover

**Theorem**
If \(x^*\) is optimal solution to \((LP)\), then \(S = \{i \in V : x^*_i \geq \frac{1}{2}\}\) 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 \(x^*_i \geq \frac{1}{2}\) or \(x^*_j \geq \frac{1}{2}\) implying \((i, j)\) covered.

\(S\) has desired cost:
Let \(S^*\) be optimal vertex cover. Then
\[
\sum_{i \in S^*} w_i \geq \frac{1}{2} \sum_{i \in S} w_i
\]

Open research problem.
Close the gap.

**Theorem** [Dinur-Safra, 2001]
If \(P \neq NP\), then no \(\rho\)-approximation for \(\rho < 1.3607\), even with unit weights.

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 \(L_i = \sum_{j \in J(i)} t_j\).
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: \(x_i\) denotes the time machine \(i\) spends processing job \(j\).

\[
(LP) \min \sum_{i \in M} L_i \\
\text{such that} \quad \sum_{j \in J} x_{ij} = t_j \quad \text{for all} \quad j \in J \\
\sum_{j \in J} x_{ij} \leq L_i \quad \text{for all} \quad i \in M \\
x_{ij} \in (0, t_j) \quad \text{for all} \quad j \in J \text{ and } i \in M_j \\
x_{ij} = 0 \quad \text{for all} \quad j \in J \text{ and } i \in M_j
\]
GLB: Linear Program Relaxation

LP relaxation.

\[
(LP) \begin{align*}
\min & \quad L \\
\text{such that} & \quad \sum_j x_{ij} = t_j \quad \forall j \in J \\
& \quad \sum_i x_{ij} \leq L \quad \forall i \in M \\
& \quad x_{ij} \geq 0 \quad \forall j \in J \text{ and } i \in M_j \\
& \quad x_{ij} = 0 \quad \forall j \in J \text{ and } i \in M_j
\end{align*}
\]

GLB: Lower Bounds

Lemma 1

Let \( L \) be the optimal value to the LP. Then, the optimal makespan \( L^* \geq L \).

Proof.

LP has fewer constraints than IP formulation.

Lemma 2

The optimal makespan \( L^* \geq \max_j t_j \).

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 machines.

GLB: Analysis

Proof.

Let \( J(i) \) be the jobs assigned to machine \( i \). By Lemma 5, the load \( L \) on machine \( i \) has two components:

- leaf nodes
  \[
  \sum_{j : \text{leaf}} t_j = \sum_{j : \text{leaf}} x_{ij} \leq L \leq L^* \quad \text{Lemma 5 (LP is a relaxation)}
  \]
  \[
  \text{optimal value of LP}
  \]

- parent(i)
  \[
  f_{\text{parent}(i)} \leq L^* \quad \text{Lemma 6}
  \]

Thus, the overall load \( L \leq 2L^* \).
GLB: Flow Formulation

Flow formulation of LP.
\[
\begin{align*}
\sum_{j} x_{ij} &= t_j \quad \text{for all } j \in J \\
\sum_{i} x_{ij} &\leq L \quad \text{for all } i \in M \\
x_{ij} &\geq 0 \quad \text{for all } j \in J \text{ and } i \in M_j \\
x_{ij} &= 0 \quad \text{for all } j \in J \text{ and } i \notin M_j
\end{align*}
\]

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 \( t_j \) time if processed on machine \( i \).
- 2-approximation algorithm via LP rounding.
- No 3/2-approximation algorithm unless \( P = NP \).