Optimization (2MMD10/2DME20), lecture 5

Gerhard Woeginger

Technische Universiteit Eindhoven

Fall 2015, Q1

Program for this week

Dealing with NP-hard problems: Approximation

- Basic definitions
- Ad-hoc approaches
- LP-based approaches
- Limits of approximability

Basic definitions

First comment:

We leave decision problems, and return to optimization problems

Definition

Let X be a minimization problem.

- The optimal objective value of instance I is denoted $\mathrm{opt}(I)$.
- The objective value returned by algorithm A is denoted $A(I)$.

The *worst case guarantee* of algorithm A is $\sup_I A(I)/\mathrm{opt}(I)$.

- worst case guarantee always ≥ 1
- small worst case guarantee = good

For maximization problems

- worst case guarantee is $\inf_I A(I)/\mathrm{opt}(I)$
- always ≤ 1; large guarantee = good

Ad-hoc approaches

Vertex cover (1)

Vertex cover (VC)

Instance: a graph $G = (V, E)$
Goal: find a vertex cover of smallest possible size
(vertex cover = subset of vertices that touches every edge)

Approximation algorithm

1. Determine a maximal matching M
2. Output: the set S that contains all endpoints of edges in M

Theorem

This poly-time approximation algorithm has worst case guarantee 2.

- time complexity; feasibility; guarantee
- lower bound: $\text{opt}(I) \geq |M|$

Makespan minimization (1)

Makespan minimization

Instance: m machines; n jobs with processing times $p_1, \dots, p_n$
Goal: assign jobs to machines so that the maximum workload
(= makespan) is minimized

Lower bounds:

- $\text{opt}(I) \;\geq\; \max p_i$
- $\text{opt}(I) \;\geq\; \frac{1}{m} \sum_{i=1}^{n} p_i$

List scheduling algorithm

Work through the job list one by one,
and always assign current job to machine with currently smallest workload

Example

$m = 3$ machines, and jobs with processing times $1, 1, 1, 1, 1, 1, 3$

Makespan minimization (2)

Theorem

List scheduling has worst case guarantee $2 - 1/m$.

Proof:

- Consider machine i that determines the makespan
- Consider last job j assigned to machine i
- Consider moment when j was assigned to i

Worst case example

m machines;
$(m-1)m$ jobs with processing time 1; one job with processing time m

Travelling Salesman Problem (1)

Travelling Salesman Problem (TSP)

Instance: cities $1, \ldots, n$; distances $d(i,j)$
Goal: find roundtrip of smallest possible length

Assumption (!!!)

We now assume that the distances satisfy the triangle inequality
$d(x,y) + d(y,z) \;\geq\; d(x,z)$ for all cities x, y, z

Lower bounds:

- $\mathrm{opt}(I) \;\geq\;$ length of minimum spanning tree MST
- $\mathrm{opt}(I) \;\geq\;$ twice the length of minimum weight perfect matching for any (even size) subset of cities

Travelling Salesman Problem (2)

Double-tree algorithm

1. Compute a minimum spanning tree MST
2. Double every edge in MST to get a Eulerian graph
3. Compute a Euler tour in the doubled MST
4. Shortcut the Euler tour to a TSP tour

Theorem

Double-tree algorithm has worst case guarantee 2.

Travelling Salesman Problem (3)

Christofides-Serdyukov algorithm

1. Compute a minimum spanning tree MST
2. Compute a minimum perfect matching M for odd-degree cities in MST
3. Construct the union of MST and M to get a Eulerian graph
4. Compute a Euler tour in MST union M
5. Shortcut the Euler tour to a TSP tour

Theorem

Christofides-Serdyukov algorithm has worst case guarantee $3/2$.

LP-based approaches

1. Find an exact IP formulation
2. Relax integrality constraints (IP $\rightarrow$ LP)
3. Solve the LP relaxation
4. Round the optimal LP solution to approximate IP solution

Weighted vertex cover (1)

Weighted vertex cover (VC)

Instance: a graph $G = (V, E)$; weights $w : V \to \mathbb{R}^+$
Goal: find a vertex cover of smallest possible weight

IP formulation

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V} w(v) \cdot x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for every edge } [u, v] \in E \\ & x_v \in \{0, 1\} & \text{for every vertex } v \in V \end{array}$$

LP relaxation

$$\begin{array}{lll} \text{minimize} & \sum_{v \in V} w(v) \cdot x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for every edge } [u, v] \in E \\ & 0 \leq x_v \leq 1 & (\text{or simply } 0 \leq x_v) \quad \text{for } v \in V \end{array}$$

Weighted vertex cover (2)

Lower bound: $\text{opt}(I) = \text{IP-opt} \geq \text{LP-opt}$

Approximation algorithm

1. Compute the optimal LP solution x_v^*
2. Round the LP solution x_v^* to a feasible IP-solution $\tilde{x}_v$:
   If $x_v^* < 1/2$ then $\tilde{x}_v = 0$
   If $x_v^* \geq 1/2$ then $\tilde{x}_v = 1$

Theorem

This poly-time approximation algorithm has worst case guarantee 2.

- time complexity; feasibility; guarantee

Weighted vertex cover (3): Gaps

Note that the approach is centered around three values:

- the optimal value of the IP: opt_{IP}
- the optimal value of the LP: opt_{LP}
- the result of the rounding: app

Observation

$\text{opt}_{LP} \;\leq\; \text{opt}_{IP} \;\leq\; \text{app} \;\leq\; 2\text{opt}_{LP}$

Two examples (with unit weights)

- Odd cycle C_{2k+1} yields $\text{opt}_{LP} = k + \frac{1}{2}$, $\text{opt}_{IP} = k + 1$, $\text{app} = 2k + 1$.
- Complete graph K_{2k} yields $\text{opt}_{LP} = k$, $\text{opt}_{IP} = 2k - 1$, $\text{app} = 2k$.

Therefore the integrality gap of the LP relaxation is $\text{opt}_{IP}/\text{opt}_{LP} = 2$

Communication delay scheduling (1)

Communication delay scheduling (COMM-DELAY)

Instance: unit time jobs $J_1, \ldots, J_n$;
precedence constraints between some jobs
Goal: find a feasible schedule on n machines
that obeys unit communication delays and minimizes makespan

- unit time jobs: job J_a runs from $S(J_a)$ to $C(J_a) := S(J_a) + 1$

- precedence constraints = partial order " $\rightarrow$ " on the jobs
- if $J_a \rightarrow J_b$ then $C(J_a) \leq S(J_b)$
  $\iff$ J_a must be completed before J_b is started

- unit communication delay for $J_a \rightarrow J_b$
  if J_a and J_b run on same machine then $C(J_a) \leq S(J_b)$
  if J_a and J_b run on different machines then $C(J_a) + 1 \leq S(J_b)$

- number n of machines is not a bottleneck

Communication delay scheduling (2)

Example

- Four jobs J_1, J_2, J_3, J_4
- Precedence constraints:
  $J_1 \to J_2;\;\; J_1 \to J_3;\;\; J_2 \to J_4;\;\; J_3 \to J_4;$

- Simple schedule:
  If all four jobs are run on different machines then makespan=5

- Better schedule:
  If all four jobs are run on same machine then makespan=4

Communication delay scheduling (3a)

Notation:

- $\mathrm{Pred}(J_a)$ denotes the set of all predecessors J_b of J_a (with $J_b \to J_a$)
- $\mathrm{Succ}(J_a)$ denotes the set of all successors J_b of J_a (with $J_a \to J_b$)

Observation

At most one predecessor of J_a can complete at $C(J_a) - 1$.
At most one successor of J_a can start at $C(J_a)$.

Modelling idea:

Introduce 0-1-variable x_{ab} that indicates the delay of $J_a \to J_b$

- $x_{ab} = 0$ means that J_b starts directly after J_a on same machine
- $x_{ab} = 1$ means that J_b starts at time $C(J_a) + 1$ or later

Corresponding inequality: $C(J_b) \geq C(J_a) + 1 + x_{ab}$

Observation

$$C(J_b) \;=\; \max\{C(J_a) + 1 + x_{ab} : \; J_a \to J_b\}$$

Communication delay scheduling (3b)

IP formulation

$$
\begin{array}{llcll}
\min & C & & & \\
\text{s.t.} & \sum_{i \in \mathrm{Pred}(j)} x_{ij} & \geq & |\mathrm{Pred}(j)| - 1 & \text{for } j = 1, \ldots, n \\
 & \sum_{i \in \mathrm{Succ}(j)} x_{ji} & \geq & |\mathrm{Succ}(j)| - 1 & \text{for } j = 1, \ldots, n \\
 & C_i + 1 + x_{ij} & \leq & C_j & \text{for } J_i \to J_j \\
 & 1 \ \leq \ C_j \ \leq \ C & & & \text{for } j = 1, \ldots, n \\
 & x_{ij} \in \{0, 1\} & & & \text{for } J_i \to J_j
\end{array}
$$

Variables:

- C_j: real variable encodes completion time of J_i
- x_{ij}: 0-1-variable encodes delay of $J_i \to J_j$
- C: real variable encodes makespan of schedule

Communication delay scheduling (3c)

LP relaxation

$$\begin{array}{lll}
\min & C & \\
\text{s.t.} & \sum_{i\in \mathrm{Pred}(j)} x_{ij} \;\ge\; |\mathrm{Pred}(j)|-1 & \text{for } j=1,\ldots,n \\
& \sum_{i\in \mathrm{Succ}(j)} x_{ji} \;\ge\; |\mathrm{Succ}(j)|-1 & \text{for } j=1,\ldots,n \\
& C_i+1+x_{ij} \;\le\; C_j & \text{for } J_i \to J_j \\
& 1 \;\le\; C_j \;\le\; C & \text{for } j=1,\ldots,n \\
& 0 \le x_{ij} \le 1 & \text{for } J_i \to J_j
\end{array}$$

Variables:

- C_j: real variable encodes completion time of J_i
- x_{ij}: real variable encodes relaxed delay of $J_i \to J_j$
- C: real variable encodes makespan of schedule

Communication delay scheduling (4)

Approximation algorithm

1. Compute the optimal LP solution x^*_{ij}, C^*_j, C^*.
2. Round the LP solution to a feasible IP-solution $\tilde{x}_{ij}$, $\tilde{C}_j$, $\tilde{C}$.

How to round the LP solution

For every precedence constraint $J_i \to J_j$ do:
If $x^*_{ij} < 1/2$ then $\tilde{x}_{ij} = 0$
If $x^*_{ij} \geq 1/2$ then $\tilde{x}_{ij} = 1$

For every job J_j do:

$$\tilde{C}_j \;=\; \max\left\{\tilde{C}_i + 1 + \tilde{x}_{ij} : \; J_i \to J_j\right\}$$

For the makespan do:

$$\tilde{C} \;=\; \max\{\tilde{C}_i\}$$

Communication delay scheduling (5)

Lemma (feasibility)

The rounded solution $\tilde{x}_{ij}$, $\tilde{C}_j$, $\tilde{C}$ is feasible for the IP.

$$\sum_{i \in \mathrm{Pred}(j)} \tilde{x}_{ij} \;\geq\; |\mathrm{Pred}(j)| - 1 \qquad \text{and} \qquad \sum_{i \in \mathrm{Succ}(j)} \tilde{x}_{ij} \;\geq\; |\mathrm{Succ}(j)| - 1$$

Communication delay scheduling (6)

Lemma (guarantee, part 1)

For every constraint $J_i \to J_j$, we have $1 + \tilde{x}_{ij} \;\le\; \frac{4}{3}(1 + x^*_{ij})$.

Proof: trivial if $\tilde{x}_{ij} = 0$; easy if $\tilde{x}_{ij} = 1$

Lemma (guarantee, part 2)

For every job J_i, we have $\tilde{C}_i \;\le\; \frac{4}{3} C^*_i$.

Proof: Induction plus $\tilde{C}_j = \max \left\{ \tilde{C}_i + 1 + \tilde{x}_{ij} : \; J_i \to J_j \right\}$

Lemma (guarantee, part 3)

The makespan satisfies $\tilde{C} \le \frac{4}{3} C^*$.

Communication delay scheduling (7)

Lower bound: $\text{opt}(I) = \text{IP-opt} \geq \text{LP-opt}$

Theorem

This poly-time approximation algorithm has worst case guarantee $4/3$*.*

- time complexity; feasibility; guarantee

Communication delay scheduling (8a): Gaps

Example

- $3k+1$ jobs $A_1, \ldots, A_{k+1}$; $B_1, \ldots, B_k$; $C_1, \ldots, C_k$
- Precedence constraints:
  $A_i \to B_i$ and $A_i \to C_i$ for $i = 1, \ldots, k$
  $B_i \to A_{i+1}$ and $C_i \to A_{i+1}$ for $i = 1, \ldots, k$

- $\text{opt}_{IP} \le 3k+1$
  $(A_1, B_1, C_1, \; A_2, B_2, C_2, \; A_3, B_3, C_3, \; \ldots, A_k, B_k, C_k, \; A_{k+1})$

- $\text{app} \ge 4k+1$
  ($x^*_{ij} = 1/2$ for all constraints $J_i \to J_j$; and hence $\tilde{x}_{ij} \equiv 1$)

Observation

For large numbers of jobs, app may come arbitrarily close to $\frac{4}{3}\text{opt}_{IP}$.

Communication delay scheduling (8b): Gaps

Example

- Job are partitioned into $k+1$ levels $0, 1, \ldots, k$, with 2^i jobs at level i
- Every job at level i has two successors at level $i+1$
  Every job at level i has one predecessor at level $i-1$

- $\text{opt}_{IP} \geq 2k+1$
- $\text{opt}_{LP} \leq \frac{3}{2}k+1$ $\qquad$ ($x^*_{ij} = 1/2$ for all constraints $J_i \to J_j$)

Observation

For large numbers of jobs, opt_{IP} may come arbitrarily close to $\frac{4}{3}\text{opt}_{LP}$.

Therefore the integrality gap of our LP relaxation is $4/3$.

Limits of approximability

Approximation Schemes

Definition (for minimization problem)

A Polynomial Time Approximation Scheme (PTAS) is
a family of approximation algorithms A_ε for $\varepsilon > 0$
with approximation guarantee $1+\varepsilon$, and
for every fixed ε running time polynomially bounded in instance size

Typical running times for PTAS:
$n^{1/\varepsilon}$, n^{2/ε^3}, $(1/\varepsilon)^{1/\varepsilon} n^4$, n^2/ε^5, $3^{1/\varepsilon} n^3$, $(4/\varepsilon)!\, n^{2/\varepsilon}$

For maximization problems
approximation guarantee of A_ε is $1-\varepsilon$

Makespan minimization (1)

Makespan minimization on $m = 2$ machines

Instance: n jobs with processing times $p_1, \ldots, p_n$
Goal: assign jobs to two machines so that the makespan is minimized

- Let $L := \max\left\{\max p_i,\ \ \frac{1}{m}\sum_{i=1}^{n} p_i\right\}$, and recall $L \le \mathrm{opt}(I)$
- Let $\varepsilon > 0$ be desired precision (for worst case ratio $1+\varepsilon$)
- Classify processing times into big ($p_j > \varepsilon L$) and small ($p_j \le \varepsilon L$)

- There are at most $2/\varepsilon$ big jobs
- There are at most $2^{2/\varepsilon}$ assignments of big jobs to machines
- If the value ε is fixed, then the values $2/\varepsilon$ and $2^{2/\varepsilon}$ are constants

Makespan minimization (2)

Approximation algorithm

1. Compute all $2^{2/\varepsilon}$ assignments of big jobs to machines
2. For each such assignment,
   add the small jobs greedily to the schedule for big jobs
3. Output the best schedule found

- One of the $2^{2/\varepsilon}$ assignments agrees with the assignment of big jobs in optimal schedule
- Let B denote the makespan (of big jobs) in that assignment
- If Greedy does not increase B: optimal schedule found
  If Greedy increases B: workload difference $\leq \varepsilon L$

Theorem

Makespan minimization on $m = 2$ machines has a PTAS.

In-approximability (1)

Chromatic number $\chi(G)$ = minimum number of colors in proper coloring

Chromatic number (COLORING)

Instance: an undirected graph $G = (V, E)$
Goal: find proper coloring of V with smallest possible number of colors
(colors $1, 2, \ldots, k$; adjacent vertices receive different colors)

Fact (from Exercise 81)

There exists polynomial time transformation from 3-SAT to COLORING such that
satisfiable 3-SAT instances translate into graphs with $\chi(G) \leq 3$
unsatisfiable 3-SAT instances translate into graphs with $\chi(G) \geq 4$

Theorem

If COLORING has poly-time approximation algorithm with ratio $r < 4/3$, then P=NP.

In-approximability (2)

Communication delay scheduling (COMM-DELAY)

Instance: unit time jobs $J_1, \ldots, J_n$;
precedence constraints between some jobs
Goal: find a feasible schedule on n machines
that obeys unit communication delays and minimizes makespan

Fact (Hoogeveen, Lenstra & Veltman, 1994)

There exists poly-time transformation from 3-SAT to COMM-DELAY such that
satisfiable 3-SAT instances translate into Is with $\text{opt}(I) \leq 6$
unsatisfiable 3-SAT instances translate into graphs with $\text{opt}(I) \geq 7$

Theorem

If COMM-DELAY has poly-time approximation algo with ratio $r < 7/6$, then P=NP.

In-approximability (3)

The Gap Technique is a method for establishing in-approximability of a minimization problem X with integral objective values:

1. Take an NP-hard problem Y
2. Construct a poly-time transformation from Y to X such that
   YES-instances of Y translate into X-instances with value $\leq A$
   NO-instances of Y translate into X-instances with value $\geq B$
3. Conclude:
   If X has poly-time approximation algorithm with ratio $r < B/A$ then P=NP

Homework 5

- Read chapters 1, 2, and 5 in the lecture notes of David Williamson
- Recommended exercises:
  93, 94, 95, 97, 98, 99, 101, 104, 106, 108

Collection of exercises can be downloaded from:
`http://www.win.tue.nl/~gwoegi/optimization/`

Attention!
Weeks 6-7 (Oct 6; Oct 9; Oct 13; Oct 16):

- 2MMD10: lecture tue 1+2,3+4; instructions fri 5+6
- 2DME20: lecture tue 3+4, fri 5+6; instructions tue 1+2

- 2MMD10: same lecture rooms as in weeks 1-5
- 2DME20: all lectures in flux 1.06