A New Approximation Technique for Resource-Allocation Problems

Barna Saha & Aravind Srinivasan

Department of Computer Science
University of Maryland, College Park, MD 20742

1st ICS, 2010

Relax and Round Paradigm

- **Relaxation**: Given an instance of an optimization problem, enlarge the set of feasible solutions I to some $I' \supset I$.
  - Example: Linear-programming (LP) relaxation of an integer program
- **Rounding**: Efficiently compute an optimum solution $x \in I'$, map x to nearby $y \in I$ and prove that y is near optimum in I

Relax and Round Paradigm

- **Relaxation**: Given an instance of an optimization problem, enlarge the set of feasible solutions I to some $I' \supset I$.
  - Example: Linear-programming (LP) relaxation of an integer program
- **Rounding**: Efficiently compute an optimum solution $x \in I'$, map x to nearby $y \in I$ and prove that y is near optimum in I

Relax and Round Paradigm

- **Relaxation**: Given an instance of an optimization problem, enlarge the set of feasible solutions I to some $I' \supset I$.
  - Example: Linear-programming (LP) relaxation of an integer program
- **Rounding**: Efficiently compute an optimum solution $x \in I'$, map x to nearby $y \in I$ and prove that y is near optimum in I

Relax and Round Paradigm

- **Relaxation**: Given an instance of an optimization problem, enlarge the set of feasible solutions I to some $I' \supset I$.
  - Example: Linear-programming (LP) relaxation of an integer program
- **Rounding**: Efficiently compute an optimum solution $x \in I'$, map x to nearby $y \in I$ and prove that y is near optimum in I

We will focus on LP-relaxation of 0-1-integer program and LP rounding methods.

LP-Rounding

Various LP-rounding approaches exist.

- Deterministic rounding
- Randomized rounding
    - Independent randomized rounding
    - Dependent randomized rounding

LP-Rounding

Various LP-rounding approaches exist.

- Deterministic rounding
- Randomized rounding
  - Independent randomized rounding
  - Dependent randomized rounding

LP-Rounding

Various LP-rounding approaches exist.

- Deterministic rounding
- Randomized rounding
  - Independent randomized rounding
  - Dependent randomized rounding

LP-Rounding

Various LP-rounding approaches exist.

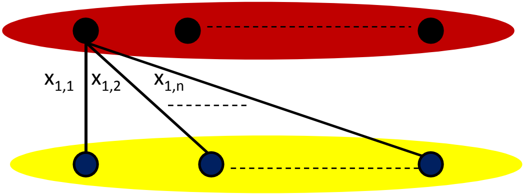
- Deterministic rounding
- Randomized rounding
  - Independent randomized rounding
  - Dependent randomized rounding

LP-Rounding

Various LP-rounding approaches exist.

- Deterministic rounding
- Randomized rounding
    - Independent randomized rounding
    - Dependent randomized rounding

Assignment Problem with Extra Linear Constraints



$$\sum_{j:(i,j)\in E} x_{i,j} = b_i \quad \forall i \in V \qquad \text{(Assign Constraint)}$$

$$\sum_{j:(i,j)\in E} a_{i,j} x_{i,j} \leq T_i \quad \forall i \in V \qquad \text{(Extra Linear Constraints)}$$

Limitation of Independent Rounding

Assignment Problem with Extra Linear Constraints

$$\sum_{j:(i,j)\in E} x_{i,j} = b_i \quad \forall i \in V \qquad \text{(Assign Constraint)}$$

$$\sum_{j:(i,j)\in E} a_{i,j}x_{i,j} \leq T_i \quad \forall i \in V \qquad \text{(Extra Linear Constraints)}$$

- Similar kind of problems studied by:
  - General linear constraints: Arora, Frieze & Kaplan [FOCS 96]
  - Constant number of constraints: Papadimitriou & Yannakakis [FOCS 00], Grandoni, Ravi & Singh [ESA 09]

Limitation of Independent Rounding

Assignment Problem with Extra Linear Constraints

$$\sum_{j:(i,j)\in E} x_{i,j} = b_i \quad \forall i \in V \qquad \text{(Assign Constraint)}$$

$$\sum_{j:(i,j)\in E} a_{i,j}x_{i,j} \leq T_i \quad \forall i \in V \qquad \text{(Extra Linear Constraints)}$$

- Independent rounding by Raghavan & Thompson.
- By Chernoff-Hoeffding bound, with high probability extra linear constraints are violated by $\pm\tilde{O}(\sqrt{|V|}\max_{i,j} a_{i,j})$.
- Vertices violate matching constraints.

Dependent Rounding

Assignment Problem with Extra Linear Constraints

$$\sum_{j:(i,j)\in E} x_{i,j} = b_i \quad \forall i \in V \qquad \text{(Assign Constraint)}$$

$$\sum_{j:(i,j)\in E} a_{i,j} x_{i,j} \leq T_i \quad \forall i \in V \qquad \text{(Extra Linear Constraints)}$$

- Several works on Dependent Rounding:
  - Ageev & Sviridenko [Journal of Combinatorial Optimization 04]
  - Srinivasan [FOCS 01]
  - Gandhi, Khuller, Parthasarathy, Srinivasan [FOCS 02]
  - Kumar, Marathe, Parthasarathy, Srinivasan[FOCS 05]

Dependent Rounding

Assignment Problem with Extra Linear Constraints

$$\sum_{j:(i,j)\in E} x_{i,j} = b_i \quad \forall i \in V \qquad \text{(Assign Constraint)}$$

$$\sum_{j:(i,j)\in E} a_{i,j} x_{i,j} \leq T_i \quad \forall i \in V \qquad \text{(Extra Linear Constraints)}$$

- Work of Gandhi et al. achieves
  - All matching constraints are satisfied with probability 1.
  - Variables incident on a vertex are negatively correlated.
  - Can still apply Chernoff-Hoeffding bound to get, $\pm\tilde{O}(\sqrt{|V|}\max_{i,j} a_{i,j})$ violation.

Our Rounding Method

Assignment Problem with Extra Linear Constraints

$$\begin{aligned} \sum_{j:(i,j)\in E} x_{i,j} &= b_i \quad \forall i \in V && \text{(Assign Constraint)} \\ \sum_{j:(i,j)\in E} a_{i,j}x_{i,j} &\leq T_i \quad \forall i \in V && \text{(Extra Linear Constraints)} \end{aligned}$$

- Using our rounding method we get,
  - All matching constraints are satisfied with probability 1.
  - Extra linear constraints are violated only by $\pm max_{i,j} a_{i,j}$.
  - Can also handle additional cost constraint: $\sum_{(i,j)\in E} c_{i,j}x_{i,j} \leq C$.

New Rounding Method

Our Rounding Method

$$
\begin{aligned}
\text{minimize} \quad & cx \\
s.t \quad & \\
& Ax \leq b \\
& x \in [0,1]^n
\end{aligned}
$$

- We have an n dimensional system of linear constraints, $Ax \leq b$ with additional constraints, $x \in [0,1]^n$.
- We are given some $x^* \in [0,1]^n$
- We want to round x^* to integer solution.

Our Rounding Method: Main Idea

- Linear constraints define a polytope.

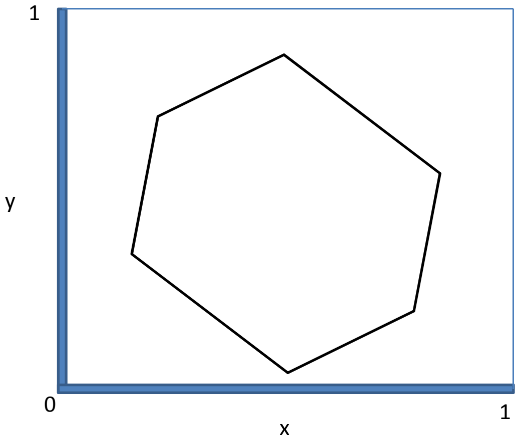
Our Rounding Method: Main Idea

- Not at a vertex of the polytope:
  - Randomly move on the current facet.
    - The expected value of each variable does not change.
  - Either round a new variable or make some other constraint tight.

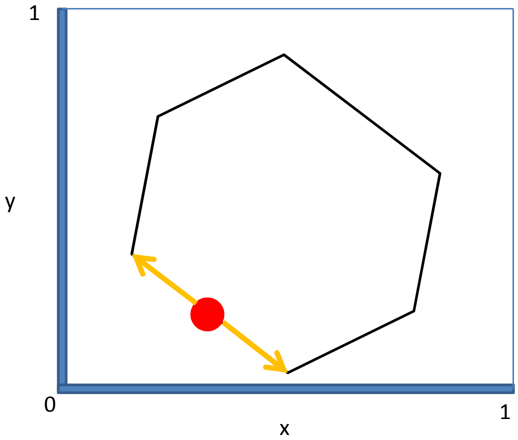
Our Rounding Method: Main Idea

- At a vertex:
  - Relax the polytope by reducing the number of tight constraints.
    - Drop or combine constraints.

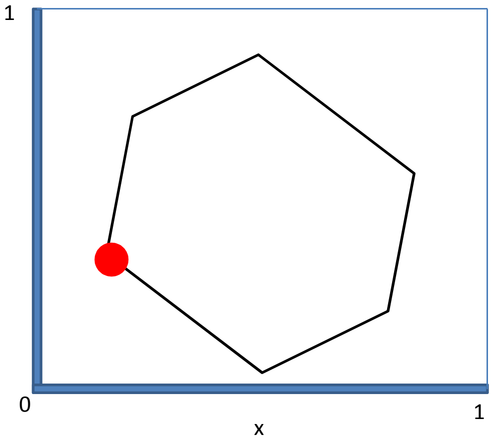
Our Rounding Method: Example in 2-dimension



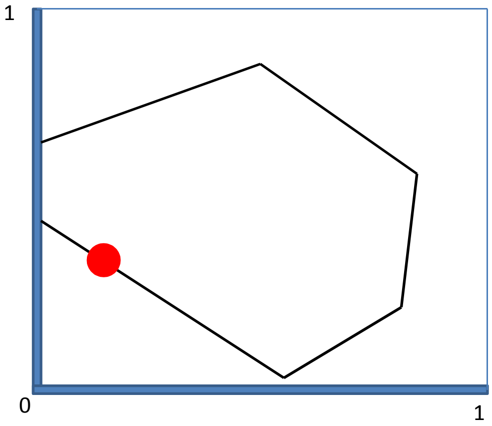
Our Rounding Method: Example in 2-dimension



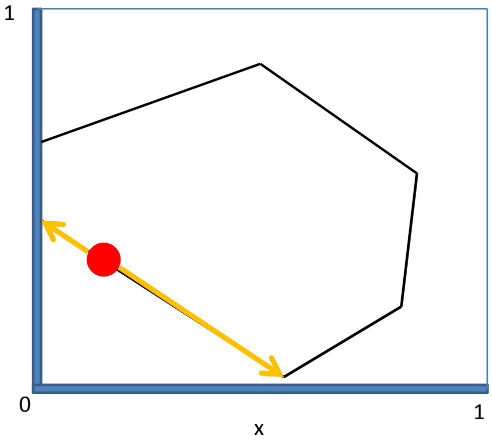
Our Rounding Method: Example in 2-dimension



Our Rounding Method: Example in 2-dimension



Our Rounding Method: Example in 2-dimension



Our Rounding Method

Properties

1. If y^* is the final rounded solution, then $E[y^*] = x^*$
2. A variable rounded to $0, 1$ is never changed.
3. A constraint dropped or combined is never reinstated.
4. At each rounding step, either a new variable gets rounded or a new constraint becomes tight.

Our Rounding Method

Properties

1. If y^* is the final rounded solution, then $E[y^*] = x^*$
2. A variable rounded to $0, 1$ is never changed.
3. A constraint dropped or combined is never reinstated.
4. At each rounding step, either a new variable gets rounded or a new constraint becomes tight.

Property (2), (3) and (4) ensure that the process terminates after $O(n + m)$ steps, where n is the total number of variables and m is the total number of constraints.

Our Rounding Method

Properties

1. If y^* is the final rounded solution, then $E[y^*] = x^*$
2. A variable rounded to $0, 1$ is never changed.
3. A constraint dropped or combined is never reinstated.
4. At each rounding step, either a new variable gets rounded or a new constraint becomes tight.

If no constraint is dropped or combined, then the integer solution obtained is optimum.

Our Rounding Method

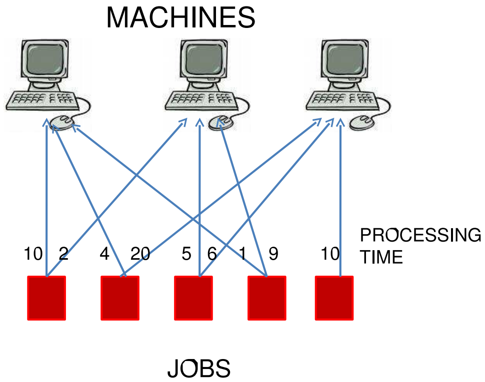
Properties

1. If y^* is the final rounded solution, then $E[y^*] = x^*$
2. A variable rounded to $0, 1$ is never changed.
3. A constraint dropped or combined is never reinstated.
4. At each rounding step, either a new variable gets rounded or a new constraint becomes tight.

Choice of constraints to drop or combine is problem specific and are chosen in a way to minimize the violation of the original constraints.

Applications

Unrelated Parallel Machine Scheduling & GAP [8, 9, 10, 1, 5, 7, 6, 2, 3, 4]

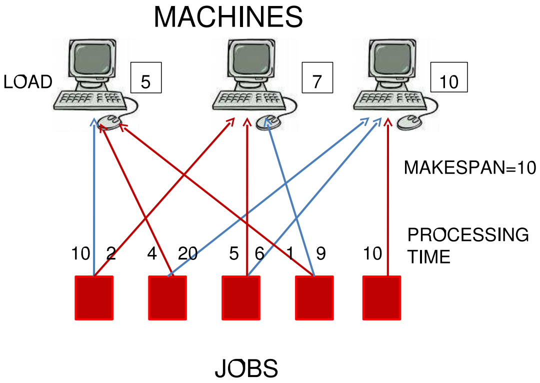


- $p_{i,j}$: processing time of job j on machine i. They are unrelated.

Unrelated Parallel Machine Scheduling & GAP [8, 9, 10, 1, 5, 7, 6, 2, 3, 4]

- **Makespan Minimization**: Minimize the maximum total load (sum of processing time of the allocated jobs) on any machine.

Unrelated Parallel Machine Scheduling & GAP [8, 9, 10, 1, 5, 7, 6, 2, 3, 4]



Unrelated Parallel Machine Scheduling & GAP [8, 9, 10, 1, 5, 7, 6, 2, 3, 4]

- **Makespan Minimization**: Minimize the maximum total load (sum of processing time of the allocated jobs) on any machine.
- 2 approximation for makepspan minimization in UPM by Lenstra, Shmoys & Tardos.

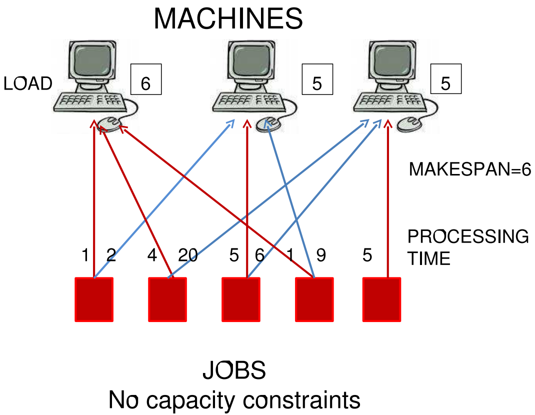
Unrelated Parallel Machine Scheduling & GAP [8, 9, 10, 1, 5, 7, 6, 2, 3, 4]

- **Generalized Assignment Problem (GAP)**: We incur a cost of $c_{i,j}$ if we schedule job j on machine i. Minimize makespan within a cost C.
- (2,1) approximation for makespan and cost for GAP by Shmoys & Tardos.

Applications

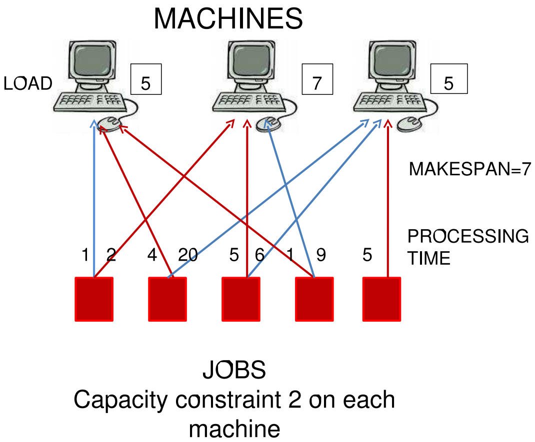
- Extension of unrelated parallel machine scheduling and generalized assignment problem with
  - Hard capacity constraints on machines.
  - Hard profit constraints with outliers.

Applications



- GAP with hard capacity constraint on machines

Applications



- GAP with hard capacity constraint on machines

Applications

- GAP with hard capacity constraint on machines
  - Handling hard capacity constraints is often tricky.
    - Capacitated covering problems, capacitated facility location problem

Applications

- GAP with hard capacity constraint on machines
  - Servers often have limits on the number of jobs they can process.
  - Studied by References.

Applications

- GAP with hard capacity constraint on machines
  - Previously known: 3 approximation to makespan for *identical machines* without any cost constraint.
  - Our result: $(2, 1)$ approximation for GAP with hard capacities.

Applications

- GAP with outliers and hard profit constraints
  - Some jobs may be dropped.
  - Profit associated for scheduling a job.
    - Total profit of the scheduled jobs must be at least Π.

Applications

- GAP with outliers and hard profit constraints
  - Studied by Gupta et al. in APPROX 09.
  - If the optimum makespan is T with profit Π and cost C, the best known approximation bound was $(\Pi, 3T, (1+\epsilon)C)$.
  - We improve it to $(\Pi, (2+\epsilon)T, (1+\epsilon)C)$, for any constant $\epsilon > 0$.

Applications:Others

- Max-min fair allocation problem
  - Closing the integrality gap for a configuration LP considered by Bansal & Sviridenko, Asadpour & Saberi.
  - Extension to equitable partitioning of items.

Applications:Others

- Random bipartite $(b-)$matching with sharp tail bounds for given linear functions.
  - Better approximation factor for special kind of linear functions.

Applications:Others

- Overlay network design.
  - Studied by Andreev, Maggs, Meyerson & Sitaraman.
  - Better approximation factor.

GAP with hard capacity constraints on machines

- Guess the optimum makespan T.

$$\sum_{i,j} c_{i,j} x_{i,j} \leq C \qquad \text{(Cost)} \qquad \sum_{i,j} x_{i,j} = 1 \ \forall j \qquad \text{(Assign)}$$

$$\sum_{j} p_{i,j} x_{i,j} \leq T \ \forall i \qquad \text{(Load)} \qquad \sum_{j} x_{i,j} \leq b_i \ \forall i \qquad \text{(Capacity)}$$

$$x_{i,j} \in \{0,1\} \ \forall i,j$$

$$x_{i,j} = 0 \quad \text{if } p_{i,j} > T$$

- Relax the constraint "$x_{i,j} \in \{0,1\} \ \forall (i,j)$" to "$x_{i,j} \in [0,1] \ \forall (i,j)$" to obtain the LP relaxation.
- Solve the LP to obtain the optimum LP solution x^*

GAP with hard capacity constraints on machines

$$\sum_{i,j} c_{i,j} x_{i,j} \leq C \qquad \text{(Cost)} \qquad \sum_{i,j} x_{i,j} = 1 \ \forall j \qquad \text{(Assign)}$$

$$\sum_{j} p_{i,j} x_{i,j} \leq T \ \forall i \qquad \text{(Load)} \qquad \sum_{j} x_{i,j} \leq b_i \ \forall i \qquad \text{(Capacity)}$$

$$x_{i,j} \in \{0,1\} \ \forall i,j$$

$$x_{i,j} = 0 \quad \text{if } p_{i,j} > T$$

- Ignore (Cost) constraint.
- The expected value of the cost remains same.

GAP with hard capacity constraints on machines

- M_k: Set of all machines with k jobs fractionally assigned to it.

(D1) for each $i \in M_1$, we drop its load and capacity constraints.

(D2) for each $i \in M_2$, we drop its load constraint and rewrite its capacity constraint as $x_{i,j_1} + x_{i,j_2} \leq \lceil x_{i,j_1} + x_{i,j_2} \rceil$, where j_1, j_2 are the two jobs fractionally assigned to i.

(D3) for each $i \in M_3$ for which *both* its load and capacity constraints are tight, drop its load constraint.

GAP with hard capacity constraints on machines

Proof Steps

- Show that the algorithm never reaches a vertex of the polytope. Thus we always make progress.
- Show that dropping constraints **(D1)**, **(D2)** and **(D3)** does not affect capacity and violates makespan only by a factor of 2. ▸ proof
- If the final rounded vector is y^*, then $E[y^*] = x^*$. Thus the expected cost remains C.
  - Can be derandomized directly by the method of conditional expectation to get a cost bound of C.

▸ Skip

GAP with hard capacity constraints on machines

Proof Steps

- Show that the algorithm never reaches a vertex of the polytope. Thus we always make progress.
- Show that dropping constraints **(D1)**, **(D2)** and **(D3)** does not affect capacity and violates makespan only by a factor of 2. ▸ proof
- If the final rounded vector is y^*, then $E[y^*] = x^*$. Thus the expected cost remains C.
  - Can be derandomized directly by the method of conditional expectation to get a cost bound of C.

▸ Skip

GAP with hard capacity constraints on machines

Proof Steps

- Show that the algorithm never reaches a vertex of the polytope. Thus we always make progress.
- Show that dropping constraints **(D1)**, **(D2)** and **(D3)** does not affect capacity and violates makespan only by a factor of 2. proof
- If the final rounded vector is y^*, then $E[y^*] = x^*$. Thus the expected cost remains C.
  - Can be derandomized directly by the method of conditional expectation to get a cost bound of C.

Skip

GAP with hard capacity constraints on machines

In no iteration a vertex of the current polytope is reached.

Skip

GAP with hard capacity constraints on machines

In no iteration a vertex of the current polytope is reached.

Notations

- $m_k = |M_k|$, the number of machines with k jobs fractionally scheduled on it.
- n'= the remaining number of jobs that are yet to be assigned permanently to a machine.
- v = the number of variables $x_{i,j} \in (0, 1)$.
- t= the number of linearly independent tight constraints in the current polytope

Skip

GAP with hard capacity constraints on machines

In no iteration a vertex of the current polytope is reached.

Lower and upper bound on v

- $v = m_1 + 2m_2 + 3m_3 + 4m_4 + \ldots$
- $v \geq 2n'$

Hence averaging,

$$v \geq n' + \frac{m_1}{2} + m_2 + \frac{3}{2}m_3 + 2m_4 + \ldots$$

Skip

GAP with hard capacity constraints on machines

In no iteration a vertex of the current polytope is reached.

Number of constraints

- (Assign) constraints: n'
- Tight (Load) and (Capacity) constraints: by our "dropping constraints" steps (**D1**), **D2**) and (**D3**), the number of tight constraints ("Load" and/or "Capacity") contributed by the machines is at most $m_2 + m_3 + \sum_{k \geq 4} 2m_k$.

Hence the total number of constraints
$t \leq n' + m_2 + m_3 + \sum_{k \geq 4} 2m_k$.

Skip

GAP with hard capacity constraints on machines

In no iteration a vertex of the current polytope is reached.

Compare number of variables and constraints

- $v \geq n' + \frac{m_1}{2} + m_2 + \frac{3}{2}m_3 + 2m_4 + \frac{5}{2}m_5 + \ldots$
- $t \leq n' + m_2 + m_3 + 2m_4 + 2m_5 + \ldots$

For the system to be determined, $v \leq t$.

$$m_1 = 0, m_3 = 0, m_5 = m_6 = m_7 = \ldots = 0$$

$$t = v$$

Skip

GAP with hard capacity constraints on machines

In no iteration a vertex of the current polytope is reached.

- Remaining machines have all degree 2 or 4.
- All tight (Assign) and (Capacity) constraints are counted in t.
- But they are not all linearly independent.

Skip

GAP with hard capacity constraints on machines

Capacity constraints are not violated.

- Capacities are integers.
- Capacity constraint is dropped only for machines in M_1.
- Capacity of those machines must be ≥ 1.

Skip

GAP with hard capacity constraints on machines

Makespan is violated at most by a factor of 2

Let X denote the final rounded variable. Then

$$\forall i, \ \sum_{j \in J} X_{i,j} p_{i,j} < \sum_{j} x^*_{i,j} p_{i,j} + \max_{j \in J : x^*_{i,j} \in (0,1)} p_{i,j} \leq 2T$$

- Load constraint is dropped for machines in M_1. But only one extra job (already fractionally assigned to it) can get permanently scheduled on it.

Skip

GAP with hard capacity constraints on machines

Makespan is violated at most by a factor of 2

Let X denote the final rounded variable. Then

$$\forall i, \quad \sum_{j \in J} X_{i,j} p_{i,j} < \sum_{j} x^*_{i,j} p_{i,j} + \max_{j \in J : x^*_{i,j} \in (0,1)} p_{i,j} \leq 2T$$

- Load constraint is dropped for machines in M_2.
  - Capacity$\in (0,1]$: at most one job can be assigned.
  - Capacity$(1,2]$:to start with total fractional assignment is more than 1 and finally all 2 jobs can get permanently assigned to it.

Skip

GAP with hard capacity constraints on machines

Makespan is violated at most by a factor of 2

Let X denote the final rounded variable. Then

$$\forall i, \quad \sum_{j \in J} X_{i,j} p_{i,j} < \sum_{j} x^*_{i,j} p_{i,j} + \max_{j \in J : x^*_{i,j} \in (0,1)} p_{i,j} \leq 2T$$

- Load constraint is dropped for machines in M_3, when they have tight capacity constraints.
  - Capacity of any such machine i must be 1 or 2.
  - Capacity$=$ 1: argued as above.
  - Capacity$=$ 2: to start with total fractional assignment of the jobs was 2. Finally all 3 jobs can get permanently assigned to it.

Skip

Future Direction

- Possible connection with iterative rounding and extensions
- Connection to discrepancy theory ▸ Details.

Thank You!

1. L. Tsai, “Asymptotic analysis of an algorithm for balanced parallel processor scheduling”,SIAM J. Comput., 1992.
2. Z. Chi and G. Wang and X. Liu and J. Liu, “Approximating Scheduling Machines with Capacity Constraints”, FAW 2009.
3. G. Woeginger,“A comment on scheduling two parallel machines with capacity constraints”, Discrete Optimization 2005.
4. H. Yang and Y. Ye and J. Zhang, “An approximation algorithm for scheduling two parallel machines with capacity constraints”, Discrete Appl. Math. 2003.
5. J. Zhang and Y. Ye, “On the Budgeted MAX-CUT problem and its Application to the Capacitated Two-Parallel Machine Scheduling”, 2001

BACK.

- Lattice approximation problem: given $A \in \{0,1\}^{m \times n}$ and $p \in [0,1]^n$, obtain a $q \in \{0,1\}^n$ such that $||A.(q-p)||_\infty$ is small.
- $\text{lindisc}(A) = \max_{p \in [0,1]^n} \min_{q \in \{0,1\}^n} ||A(q-p)||_\infty$
- Several results for bounding $\text{lindisc}(A)$ for different matrices.
- Our approach can be potentially useful in bounding the lindisc of random column-sparse matrices.

END.

Y. Azar and A. Epstein.
Convex programming for scheduling unrelated parallel machines.
In *Proc. of the ACM Symposium on Theory of Computing*, pages 331–337. ACM, 2005.

N. Bansal and M. Sviridenko.
The Santa Claus problem.
In *STOC '06: Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 31–40, 2006.

M. Bateni, M. Charikar, and V. Guruswami.
Maxmin allocation via degree lower-bounded arborescences.
In *STOC '09: Proceedings of the 41st annual ACM Symposium on Theory of computing*, pages 543–552, 2009.

D. Chakrabarty, J. Chuzhoy, and S. Khanna.

On allocating goods to maximize fairness.
In *FOCS '09: 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009.

T. Ebenlendr, M. Křćal, and J. Sgall.
Graph balancing: a special case of scheduling unrelated parallel machines.
In *SODA '08: Proceedings of the Nineteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 483–490, 2008.

A. Gupta, R. Krishnaswamy, A. Kumar, and D. Segev.
Scheduling with outliers.
In *Proc. APPROX*, 2009.
Full version available as arXiv:0906.2020.

V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan.

A unified approach to scheduling on unrelated parallel machines.
Journal of the ACM, 56(5), 2009.

J. K. Lenstra, D. B. Shmoys, and É. Tardos.
Approximation algorithms for scheduling unrelated parallel machines.
Mathematical Programming, 46:259–271, 1990.

D. B. Shmoys and É. Tardos.
An approximation algorithm for the generalized assignment problem.
Mathematical Programming, 62:461–474, 1993.

M. Skutella.
Convex quadratic and semidefinite relaxations in scheduling.
Journal of the ACM, 46(2):206–242, 2001.