An Approximation Algorithm for Cover Inequality Separation

Wasu Glankwamdee

References

- An Fast Algorithm for Cover Inequality Separation by O. Oguz, Department of Industrial Engineering, Bilkent University, Turkey.
- Benchmarking Optimization Software with Performance Profiles by E. D. Dolan and J. J. Moré, Mathematical Programming, 91, 201-213, 2002.
- Performance Profiles Python Script by M. Friedlander, www.cs.ubc.ca/∼mpf/pprof.html

Performance Profiles

- We have n_s solvers and n_p problems.
- $t_{p,s}$ = computing time required to solve problem p by solver s.
- Performance ratio,

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}.$$

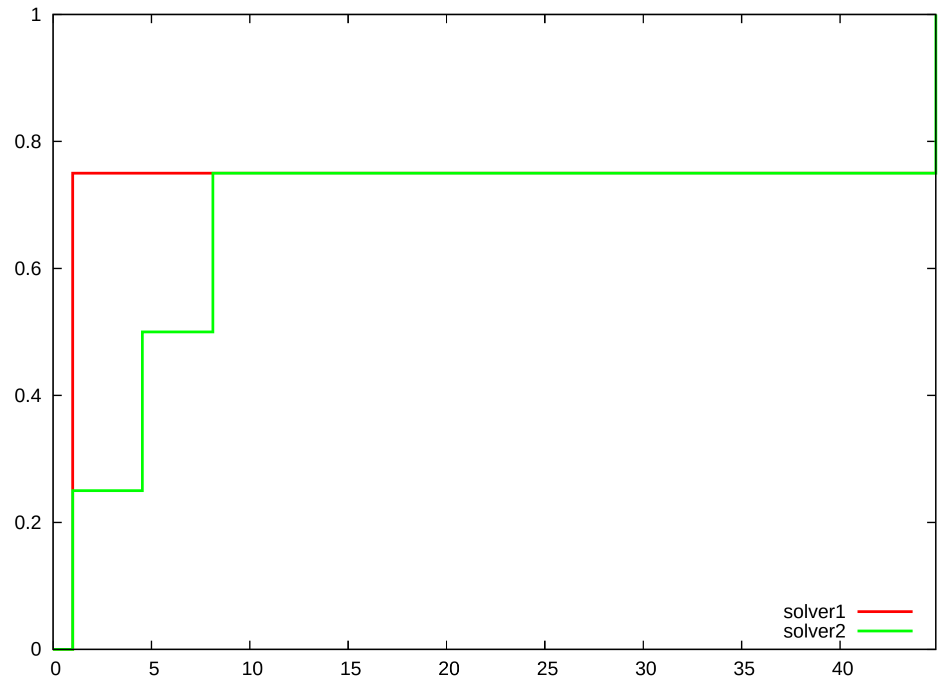
- Probability for solver $s \in \mathcal{S}$ that $r_{p,s}$ is within a factor of $\tau \in R$ of the best possible ratio,

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}.$$

Example

	solver1	solver2	$r_{p,s1}$	$r_{p,s2}$
p1	-1	4685.71	X	1
p2	2.91	23.66	1	8.13
p3	296.94	113323.79	1	44.87
p4	23.71	107.62	1	4.54

Performance Profile

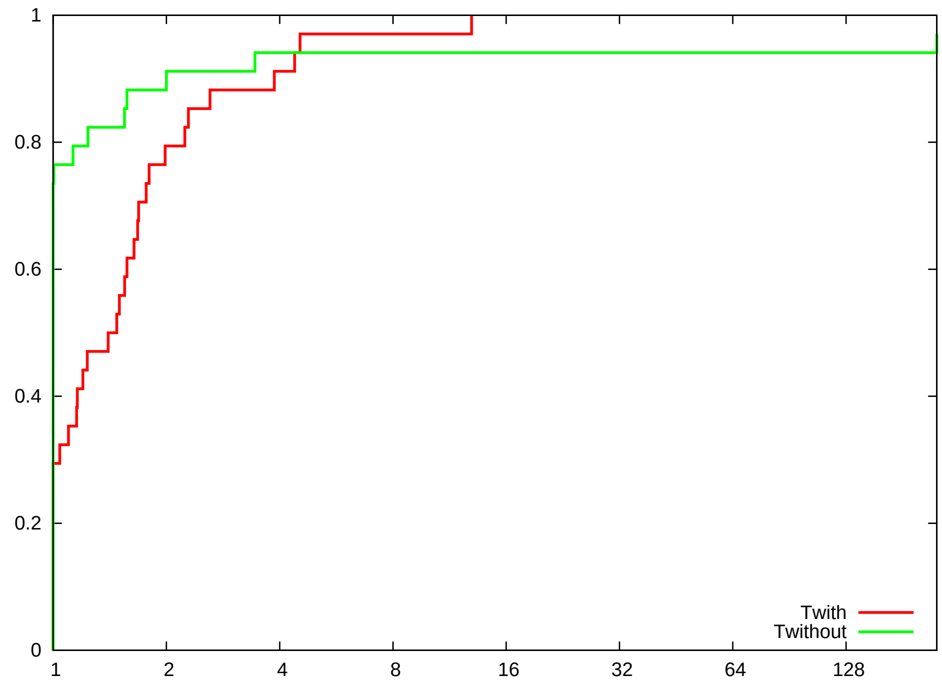


Cover Inequalities for $0-1$ Knapsack

- $X = \{x \in B^n : \sum_{j=1}^{n} a_j x_j \leq b\}$.
- A set $C \subseteq N$ is a cover if $\sum_{j \in C} a_j > b$.
- A cover is minimal if $C \setminus \{j\}$ is not a cover for any $j \in C$.
- If $C \subseteq N$ is a cover for X, the cover inequality $\sum_{j \in C} x_j \leq |C| - 1$ or $\sum_{j \in C} (1 - x_j) \geq 1$ is valid for X.

Performance Profile

- 55 mixed $0-1$ instances from MIPLIB, 8 hours.



Separation Problem

- Given a nonintegral point x^* with $0 \le x_j^* \le 1$ for all $j \in N$.
- We wish to know whether x^* satisfies all the cover inequalities.
- Does there exists a set $C \subseteq N$ with $\sum_{j \in C} a_j > b$ for which $\sum_{j \in C}(1 - x_j^*) < 1$?

$$\zeta = \min\{\sum_{j \in N}(1 - x_j^*) z_j : \sum_{j \in N} a_j z_j \ge b + 1, z \in B^n\} < 1?$$

where $z_j = 1$ if $j \in C$ and $z_j = 0$ otherwise.

- If $\zeta < 1$, there exists a violated cover that cuts off x^*.
- Separation problem is NP-Complete.

Solving $0-1$ Knapsack by DP

- $\max\{Cx : Ax \leq b, x \in B^n\}$.
- Solve recursion,

$$f_r(\lambda) = \max\{f_{r-1}(\lambda), f_{r-1}(\lambda - a_i) + c_r\}$$

where $1 \leq r \leq n$ and $0 \leq \lambda \leq b$.

- Record solution pairs $(\lambda, f_r(\lambda))$.
- Optimal solution $= f_n(b)$.
- Complexity: $O(nb)$ in time and $O(n \log b)$ in space.

Scaling Up

- Original

$$\min \sum_{j=1}^{n} (1 - x_j) z_j$$

- Scaled Up

$$\min \sum_{j=1}^{n} d_j z_j$$

where $d_j = \lfloor 10^k (1 - x_j) \rfloor$

- Subject to

$$\sum_{j=1}^{n} a_j z_j \geq b + 1$$

$$z_j \in \{0, 1\}$$

Facts About Scaling Up

- Maximum error $R(k)$,

$$R(k) = \sum_{j=1}^{n} \left((1 - x_j) - \frac{d_j}{10^k} \right).$$

- $R(k) > 0$ and $\lim_{k \to \infty} R(k) = 0$.

Important Fact

- There exists a violated cut only when the objective function value is less than 10^k.
- Consider the objective function as a constraint and the new objective is to maximize the knapsack size.

Inversion

- max

$$\sum_{j=1}^{n} a_j z_j$$

subject to

$$\sum_{j=1}^{n} d_j z_j \leq 10^k$$

$$z_j \in \{0, 1\}$$

- Solve recursion,

$$f_j(y) = \max\{f_{j-1}(y), f_{j-1}(y - d_j) + a_j\}$$

where $1 \leq j \leq n$ and $0 \leq y \leq 10^k$.

Facts About Inversion

- Find $f_n(y) > b$, then stop.
- RHS of the constraint is bounded by 10^k for every problem instance.
- Complexity is $10^k O(n)$.

Expected Error

- Error depends on the number of $z_j = 1$ in the inversion problem.
- How many $z_j = 1$ in each violated cut?
- Rewrite the constraint in the inversion as,

$$\sum_{j=1}^{n} \frac{\lfloor 10^k (1 - x_j^*) \rfloor}{10^k} z_j \leq 1$$

- Let $S = \{j | z_j = 1\}$. $|S|$ is a random variable.
- Assume $\frac{\lfloor 10^k (1 - x_j^*) \rfloor}{10^k} z_j$ is a random variable with uniform distribution on $[0, 1]$ and iid.

Proof by Counting Process

- Count the number of $z_j = 1$.

$$N(t) = \max\{n | \sum_{j=1}^{n} x_j \leq t\}$$

where $x_j = \frac{\lfloor 10^k(1-x_j^*) \rfloor}{10^k} z_j$.

- $E[N(1)] = e = 2.71$.
- For $k = 2$, the truncated part has uniform distribution in $[0, 0.01]$ with mean $= 0.005$.
- Expected error $= 2.71 \times 0.005 = 0.01355$.

Computational Results

- Solve a set of $0-1$ IP problems range between 10 and 1000 constraints, 20 and 2000 variables. $k = 2$ or 3.
- The algorithm generates 15% more cuts than the greedy algorithm.