MORE WAYS TO USE DUAL INFORMATION IN MILP

ISMP2015

Philipp M. Christophel
Menal Güzelsoy
Imre Pólik

AGENDA

1. Reusing Dual Information
2. Dual Tightening
3. Branching Variable Selection
4. Computational Results

DUAL INFORMATION IN BRANCH AND BOUND

- Dual information:
  - Dual solutions from feasible node LPs
  - Dual rays from infeasible node LPs
- Tree search generates a large amount of dual information
- Globally valid and can be reused elsewhere in tree
- How?

REUSING DUAL INFORMATION

DUAL INFORMATION IN BRANCH AND BOUND

- Traditional usage
  - Reduced cost fixing (local and root)
  - Conflict analysis
- New ways
  - Keep dual rays and solutions
  - Prune nodes
  - Subtree bounds, simulate subtrees (lookahead branching)
  - Symmetry
  - Dual ray fixing (no incumbent or bound needed)
  - Dual tightening (reduced cost fixing)
  - Variable selection

- Node LP problem

$$\begin{aligned} z^* = \min\ & cx \\ \text{s.t. } & L \leq Ax \leq U \\ & \ell \leq x \leq u \end{aligned}$$

- For a dual solution y, dual objective value is

$$z(y) = \sum_{i:y_i>0} L_i y_i + \sum_{i:y_i<0} U_i y_i + \sum_{j:r_j(y)>0} \ell_j r_j(y) + \sum_{j:r_j(y)<0} u_j r_j(y)$$

  where $r_j(y) = c_j - A_j^T y$ is the reduced cost of variable j.
- From LP duality: $z^* \geq z(y)$.

- We can tighten bounds by LP duality
  - Let $\bar{z}$ be the current cutoff/upper bound and $r_j(y) > 0$
  - If we fix $l_j = u_j$, then the resulting dual objective value is

$$z'(y) = z(y) + (u_j - l_j) r_j(y)$$

  - We can tighten u_j, if $z'(y) > \bar{z}$
- Tightening function for variable j

$$\pi_j(y) = \frac{\bar{z} - z(y)}{r_j(y)}$$

- We can tighten u_j if

$$\min_{y: r_j(y) > 0} \pi_j(y) < u_j - l_j$$

- Same applies to l_j

TIGHTENING FUNCTION - PROPERTIES

$$\pi_j(y) = \frac{\bar{z} - z(y)}{r_j(y)}$$

- Numerator is piecewise linear, convex
- Denominator is linear
- The fraction is piecewise pseudolinear, overall quasiconvex
- Due to pseudolinearity the minimum is taken at the boundaries of the regions in the numerator
  - That is where the reduced costs change sign
- Can be minimized with a modified simplex algorithm, but expensive

TIGHTENING FUNCTION

$$\pi_j(y) = \frac{\bar{z} - z(y)}{r_j(y)}$$

Classical approach:

- Minimize numerator
  - Solve node LP and use the optimal dual solution y^*
- Maximize denominator (keeping LP optimal): find true shadow prices
  - Sensitivity analysis, expensive

New approaches:

1. Use dual solutions collected from other node LPs
2. Use dual solutions collected from dual simplex iterations
3. Search neighborhood of y^*
4. Search an improving direction for y^*

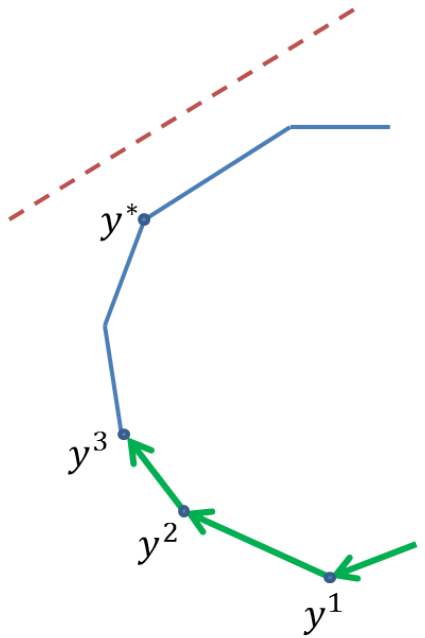
1. TREE/SIMPLEX DUAL SOLUTIONS

Dual solutions from tree search

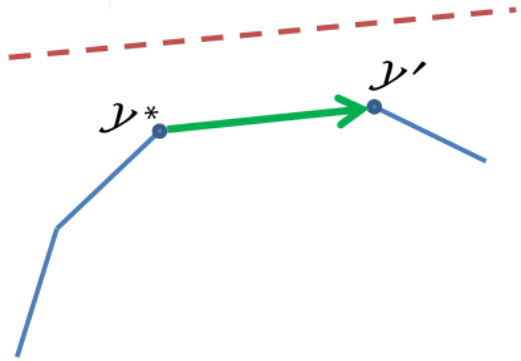
- A dynamic list of dual solutions as they are found in tree
- Based on effectiveness and length, priority to most recent ones

Dual solutions from simplex iterations

- We can generate dual solutions at each dual simplex iteration
- Periodically check the $z(y), r_j(y)$ pair
- Keep the best pair that minimizes $\pi_j(y)$
- Mostly done anyway to check the objective cutoff



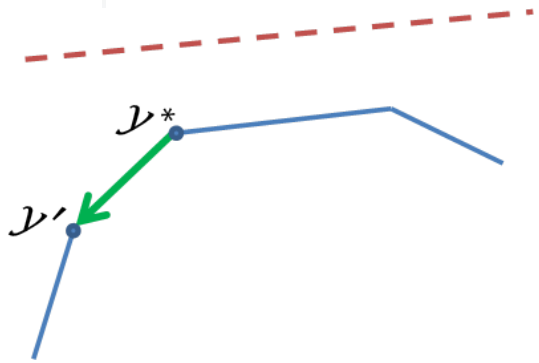
2. NEIGHBORHOOD SEARCH: OPTIMAL FACE



1. Set a basic, primal degenerate variable as leaving
2. Compute pivot row
3. Perform a simple dual ratio test to see how far the reduced cost can be pushed

- Objective value stays constant
- The new reduced cost is the dual steplength
- Nonbasic variables in the pivot row may also get an improved reduced cost
- Just a step on the dual optimal face

3. NEIGHBORHOOD SEARCH: NONOPTIMAL FACE

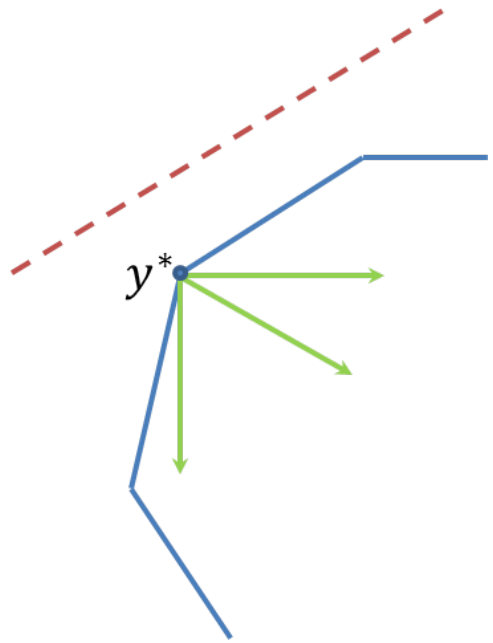


Extend to primal nondegenerate variables (incl. fractionals):

1. Set a basic, primal nondegenerate variable as leaving
   - The objective value will decrease
2. Compute pivot row
3. Perform a simple dual ratio test to see how far the reduced cost can be pushed
   - Pseudolinear, can check the derivative at the beginning
   - Hopefully the steplength is large enough, so that overall the tightening improves

This is still on the surface of the dual polyhedron, not inside.

DUAL TIGHTENING | 4. IMPROVING DIRECTION



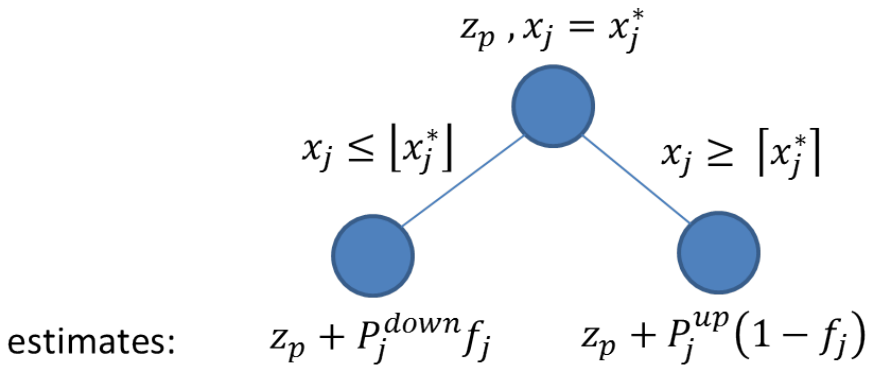
- Coordinate/gradient descent
- Move in the direction of a dual ray
- Move in the direction towards another dual solution

BRANCHING VARIABLE SELECTION

- Store dual solutions/rays and use them for variable selection
- However, we lose information as the dual information list changes dynamically
- Our solution:
  - Add information from dual solutions to pseudocosts
  - Add information from dual rays to conflict scores

TRADITIONAL PSEUDOCOSTS

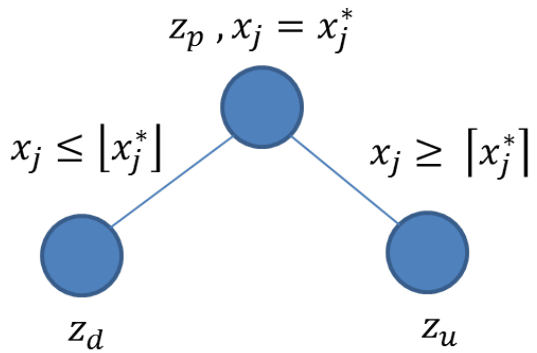
- The pseudocost of a variable is an estimate of change in the objective function if the variable is tightened by one unit
- P_j^{down} and P_j^{up}
- At a node t, let $f_j = x_j^* - \lfloor x_j^* \rfloor$.



- How do we maintain pseudocosts?

TRADITIONAL PSEUDOCOSTS

- Pseudocosts are updated after branching



- Objective change per unit at node t:

$$\delta_j^{\text{down}}(t) = \frac{z_d - z_p}{f_j} \quad , \quad \delta_j^{\text{up}}(t) = \frac{z_u - z_p}{1 - f_j}$$

- Pseudocost of variable j is the average:

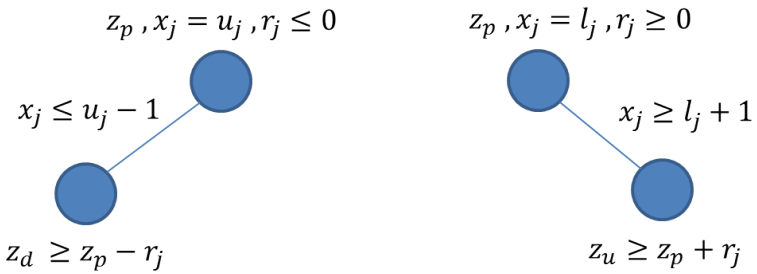
$$P_j^{\text{down}} = \frac{\sum_{t \in T_j} \delta_j^{\text{down}}(t)}{|T_j|} \quad , \quad P_j^{\text{up}} = \frac{\sum_{t \in T_j} \delta_j^{\text{up}}(t)}{|T_j|}$$

- They can be initialized/updated with strong branching

IMPROVED PSEUDOCOSTS

- Traditionally, pseudocosts are updated only by primal information
  - Only for the fractional/basic variable that is branched on
  - One update at a time
  - Only after the child is evaluated
- Improve pseudocosts using dual information
  - Through reduced costs of nonbasic variables
  - Whenever a new dual solution is found
  - Multiple updates

IMPROVED PSEUDOCOSTS



- Objective change (lower bound) per unit at node t:

$$\gamma_j^{\text{down}}(t) = -r_j \quad \text{if } r_j \leq 0 \quad , \quad \gamma_j^{\text{up}}(t) = r_j \quad \text{if } r_j \geq 0$$

- Improved pseudocost of variable j:

$$P_j^{\text{down}} = \frac{\sum_{t \in T_j} \delta_j^{\text{down}}(t) + \sum_{t \in S_j} \gamma_j^{\text{down}}(t)}{|T_j| + |S_j|}$$

$$P_j^{\text{up}} = \frac{\sum_{t \in T_j} \delta_j^{\text{up}}(t) + \sum_{t \in S_j} \gamma_j^{\text{up}}(t)}{|T_j| + |S_j|}$$

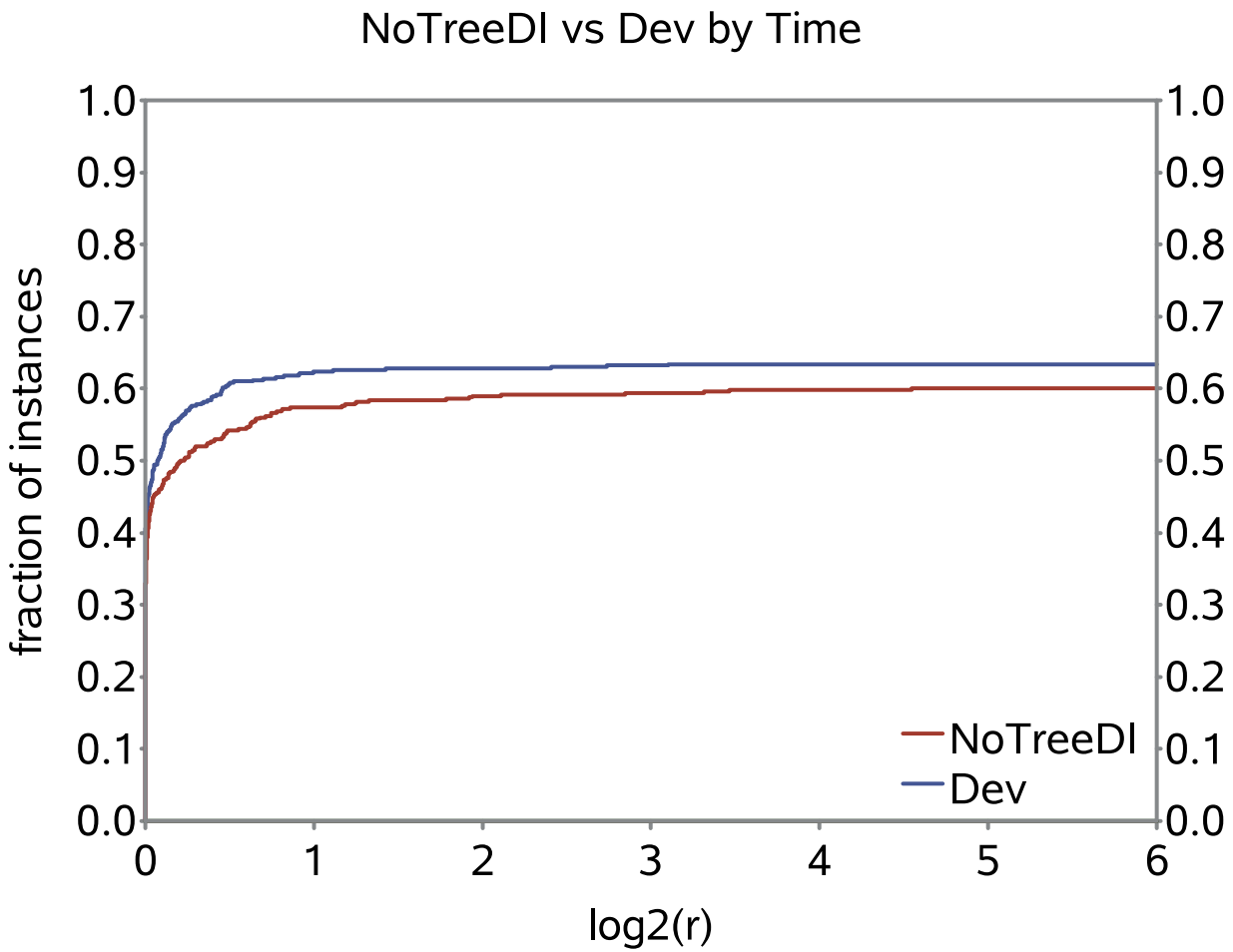
COMPUTATIONAL RESULTS

SAS/OR

- 500 test instances (internal + public)
- 1h time limit

COMPUTATIONAL RESULTS

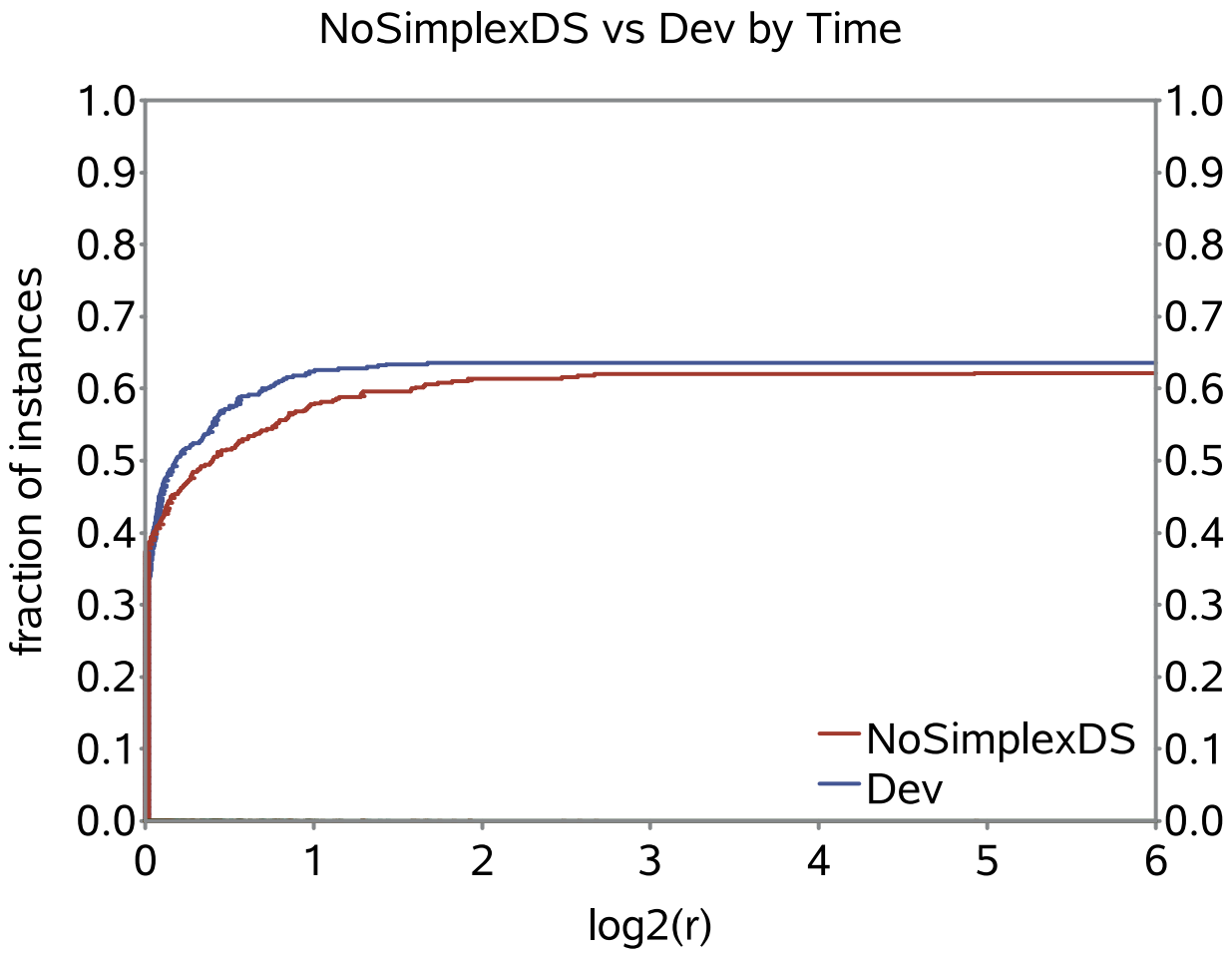
SAS/OR - DUAL INFORMATION FROM TREE



- 6% speedup + 17 instances

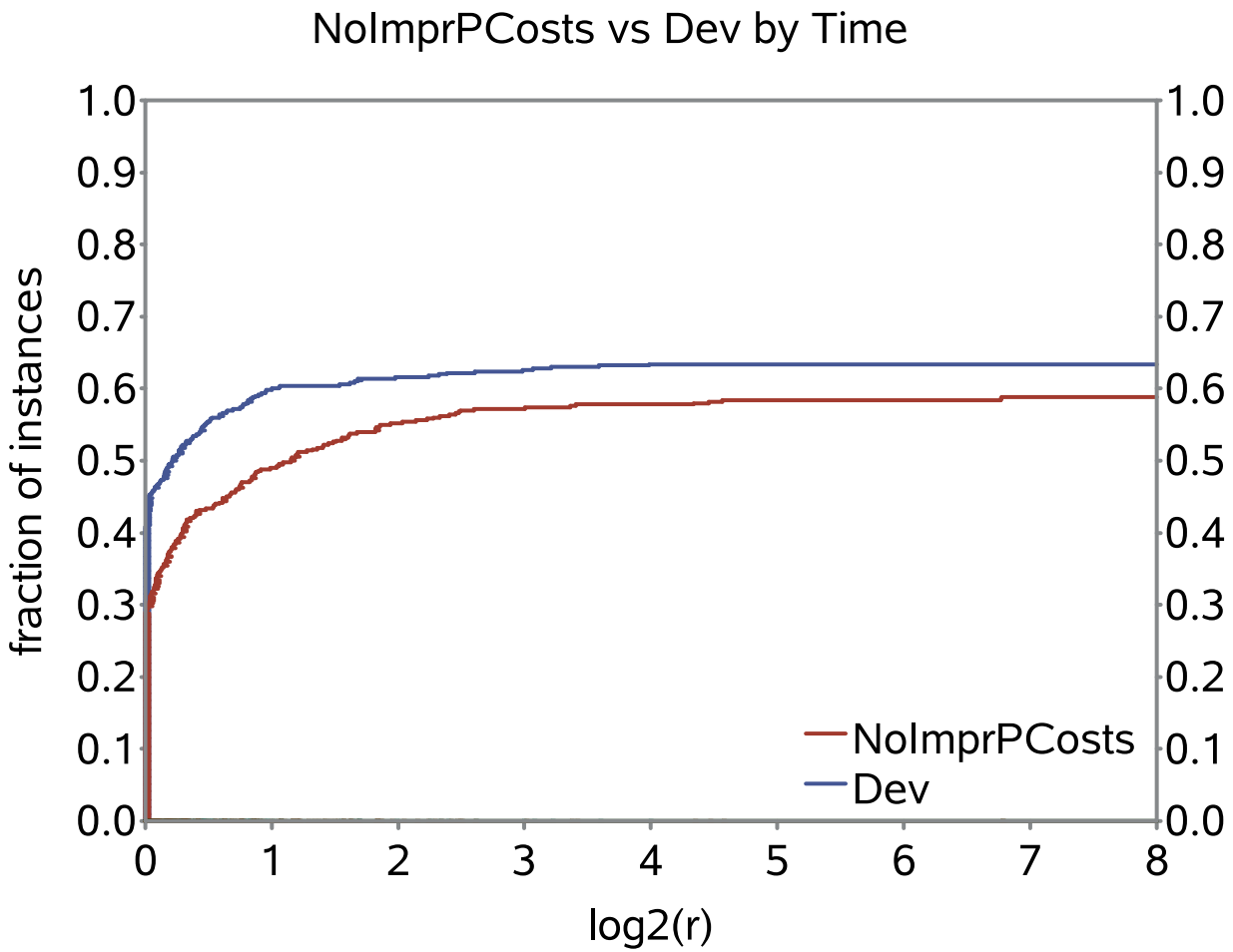
COMPUTATIONAL RESULTS

SAS/OR - DUAL SOLUTIONS FROM SIMPLEX



- 8% speedup + 7 instances

SAS/OR - IMPROVED PSEUDOCOSTS



- 18 % speedup + 23 instances

http://support.sas.com/or

More ways to use dual information in MILP