Duality, Warm Starting and Sensitivity Analysis In Mixed-Integer Programming

Menal Guzelsoy

Committee Members:

Sanjeeb Dash
Garth Isaak
Jeff Linderoth
Ted Ralphs

Department of Industrial and Systems Engineering
Lehigh University

Proposal Presentation, April 2006

"Mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true."

Bernard Russell (1872-1970) –Mysticism and Logic, ch 4, 1917

Outline

General mathematical duality

Duality is a tool to transform a *mathematical object* to another to introduce a new approach to solve a problem.

Given a class of objects $\{O_1, O_2, ...\}$, a dual mapping $\Psi : O_i \rightarrow O_j$ has the property $\Psi^2 = I$.

- Set Theory and Logic (De Morgan Laws)
- Geomety (Pascal's Theorem & Brianchon's Theorem)
- Combinatorial Math. (Graph Coloring)

Why do we need duality in *mathematical programming* and how can we characterize it?

Mathematical Programming Duality

- Depending on its structure, it can be hard to solve a mathematical program.
- In most cases, we must first develop procedures to get either a locally optimal solution or, at least, a bound to estimate the solution.
- Later, these estimates can be embedded within larger algorithms to find an optimal solution.
- If these results are also valid for a neighborhood of the problem, then they can be used for sensitivity analysis and re-solving of a modified problem.

This methodology is the basis for what is called *duality theory* for mathematical programs.

Characterization

Consider a mathematical programming problem,

$$\begin{aligned} P(\zeta, X) = \min_{x} \quad & \zeta(x) \\ \text{s.t} \quad & x \in X \end{aligned}$$

$\zeta : \Phi \rightarrow \mathbb{R}$, $X \subseteq \Phi$.

Construct $\eta : \Theta \rightarrow \mathbb{R}$, and $U \subseteq \Theta$, such that for any $x \in X$, $u \in U$, we have $\eta(u) \leq \zeta(x)$. Then, the problem to get the best lower bound is

$$\begin{aligned} D(\eta, U) = \max_{u} \quad & \eta(u) \\ \text{s.t} \quad & u \in U \end{aligned}$$

Characterization

For a domain Ξ, let $\mathscr{F}(\Xi) = \{f \mid f : \Xi \rightarrow \mathbb{R}\}$ and $\mathscr{X}(\Xi) = \{X \mid X \subseteq \Xi\}$. Then,

Dual Mapping

$$\psi : (\mathscr{F}(\Phi) \times \mathscr{X}(\Phi)) \rightarrow (\mathscr{F}(\Theta) \times \mathscr{X}(\Theta))$$

Primal Problem

$$P(\zeta, X) = \min_{x} \quad \zeta(x) \quad \text{s.t} \quad x \in X$$

$\xrightarrow{\psi}$

Dual Problem

$$D(\eta, U) = \max_{u} \quad \eta(u) \quad \text{s.t} \quad u \in U$$

such that $D(\eta, U) \leq P(\zeta, X)$ for a given ζ and X.

Characterization

- D is called a *dual function*.
- If in addition, $D(\eta, U) = P(\zeta, X)$ with $P(\zeta, X) > -\infty$, then it is called a *strong dual* function.
- If $P(\zeta, X) \rightarrow -\infty$, then dual problem is infeasible.
- if $D(\eta, U) \rightarrow \infty$, then primal problem is infeasible.
- η and ζ are not necessarily in the same function "class".
- X and U may have different properties (i.e., X may be nonconvex and U convex).
- Therefore, ψ is usually not invertible.

Linear Programming (LP) duality

For $\Phi = \mathbb{R}^n$, LP in standard form is a mathematical program where ζ is linear and X is a polyhedron.

LP Primal Problem

$$P_{LP}(A, b, c) = P(\zeta, X)$$

$$\zeta(x) = cx,\ c \in \mathbb{R}^n,$$
$$X = \{Ax = b, x \in \mathbb{R}^n_+\}.$$

$\xrightarrow{\psi_{LP}}$

LP Dual Problem

$$D_{LP}(A, b, c) = D(\eta, U)$$

$$\eta(u) = ub,\ u \in \mathbb{R}^m,$$
$$U = \{uA \leq c,\ u \in \mathbb{R}^m\}.$$

LP duality properties

With an optimal solution pair x^*, u^*,

- Proof of optimality: $u^* b = cx^*$.
- Complementary slackness:

$$u_i^*(a^i x^* - b_i) = 0 \quad \forall i$$
$$(c_j - u^{*\prime} a_j) x_j^* = 0 \quad \forall j$$

- u_i^* is the *shadow price* of resource i.
- $c_j - u^* a_j$ is the *reduced cost* of process j.

By these properties, it is possible to develop

- direct solution algorithms,
- sensitivity analysis tools,
- re-solving procedures,
- decomposition algorithms to solve large-scale instances.

Mixed-Integer Linear Programming (MILP) Duality

A MILP is a mathematical program ζ is linear and X is restricted to the form $(\mathbb{Z}^r \times \mathbb{R}^{n-r} \cap \mathscr{P})$, $r \leq n$ and $\mathscr{P} \subseteq \mathbb{R}^n$ is a polyhedron.

MILP Primal Problem

$$P_{MILP}(A, b, c, r) = P(\zeta, X)$$

$\zeta(x) = cx,\ c \in \mathbb{R}^n,$
$X = \{Ax = b, x \in \mathbb{Z}^r \times \mathbb{R}^{n-r}_+\}.$

$\xrightarrow{\psi_{MILP}}$

MILP Dual Problem

$$D_{LP}(A, b, c, r) = D(\eta, U)$$

$\eta =??,$
$U =??.$

- For *mixed-integer linear programs* (MILPs), a theory well integrated with practice has not yet been introduced.
- The dual problems for MILPs are either not effective or not computationally tractable.
- We don't know a nice pair of (η, U) which will give a strong dual function.
- However, we can construct a dual problem to find a strong dual function.

Mixed-Integer Programming Duality

That is,

Dual Problem to get best dual function

$$\begin{aligned} \max_{\eta, U} \quad & D(\eta, U) \\ \text{s.t} \quad & D(\eta, U) \leq P(\zeta, X) \end{aligned}$$

- Note that, even if this problem could be solved, it would be useless for postoptimal analysis. Why?
- What we need is to derive a dual function that is also bounding the *value function* in a neighborhood.

The *value function* is

$$z(d) = P_{MIP}(\bullet, d, \bullet, \bullet)$$

We improve the dual mapping as

$\psi_{MIP} : (\mathcal{L}^n \times \cup_k \mathscr{X}(\mathbb{Z}^k \times \mathbb{R}^{n-k})) \rightarrow (\mathscr{F}(\Theta \times \Omega) \times \mathscr{X}(\Theta \times \Omega))$ such that
$\psi_{MIP}(A, b, c, r) := (f, U)$ with

$$f(u, d) \leq cx, \text{ for all } u \in U(d) \subseteq \Theta,\ x \in \mathcal{S}(d), \text{and } d \in \Omega.$$

$\mathcal{L}^n = \{f : \mathbb{R}^n \rightarrow \mathbb{R} \mid f \text{ is linear}\}$,
$\mathcal{S}(d) = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ : Ax = d\}$,
$\Omega = \{d \in \mathbb{R}^m : \mathcal{S}(d) \neq \emptyset\}$.

Setting $F(d) = D(f(u, d), U(d))$, we obtain

$$F(d) \leq z(d) \text{ for all } d \in \Omega$$

Then, the problem to get a dual function - best for fixed right-hand-side (RHS) b, but also valid for any RHS - is

Dual Problem to get best dual function

$$w(b) = \max\{F(b) \mid F(d) \leq z(d),\ d \in \mathbb{R}^m,\ F \in \Upsilon^m \subseteq \Lambda^m\}$$

or,

$$w(b) = \max\{F(b) \mid F(Ax) \leq cx,\ x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+,\ F \in \Upsilon^m \subseteq \Lambda^m\}.$$

$\Lambda^m = \{f : \mathbb{R}^m \rightarrow \mathbb{R}\}$.

- This problem is guaranteed to give a strong dual function since $F^*(d) = z(d)$ when $\mathcal{S}(d) \neq \emptyset$, and $F^*(d) = 0$ elsewhere, is a feasible solution. Therefore, $F^*(b) = z(b)$.
- Can we further restrict Υ^m and still guarantee to get a strong dual function for any RHS?
  - $\mathcal{L}^m$: Linear functions? ✖
  - C^m: Convex functions? ✖
  - $\mathcal{F}^m$: Subadditive functions? ✔

Example

$$\begin{aligned} z(d) = \min \quad & x_1 + x_2 + x_3 + x_4 \\ \text{s.t} \quad & 2x_1 - 2x_2 + x_3 - x_4 = d \\ & x_1, x_2 \in \mathbb{Z}_+, x_3, x_4 \in \mathbb{R}_+ \end{aligned}$$

Furthermore, restricting Υ^m to linear functions, we get,

$$\begin{aligned} w(d) = \max \quad & ud \\ \text{s.t} \quad & -1/2 \leq u \leq 1/2 \\ & u \in \mathbb{R} \end{aligned}$$

What is $w(d)$?

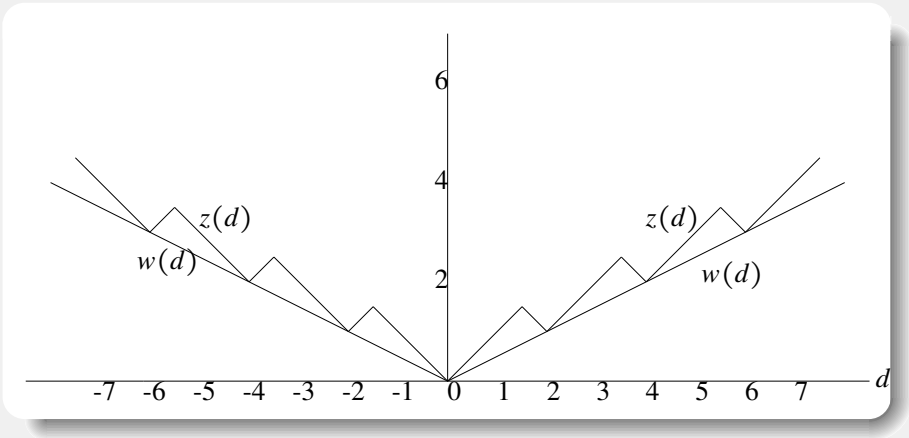


Figure: Value functions of the problem and its LP relaxation.

Formulation

Denoting $\Gamma^m = \{F \in \mathcal{F}^m \mid F(0) = 0\}$,

Subadditive Dual Problem

$$\begin{aligned} w(d) = \max \quad & F(d) \\ & F(a_j) \leq c_j \quad j \in I \\ & \bar{F}(a_j) \leq c_j \quad j \in N \setminus I \\ & F \in \Gamma^m \end{aligned}$$

where a_j is the j^{th} column of A and

$$\bar{F}(d) = \limsup_{\delta \to 0^+} \frac{F(\delta d)}{\delta} .$$

$\bar{F}$

- $\bar{F}$ is the *upper d-directional derivative* of F at 0.
- It is defined if $F(\delta d) < \infty,\ \forall\ \delta \geq 0$.
- If it is defined,
  - Regular limit can be substituted with limsup.
  - It is positively homogeneous as for any $\lambda > 0$, $\bar{F}(\lambda d) = \lambda \bar{F}(d)$.
  - $F(\lambda d) \leq \bar{F}(d)\lambda,\ \lambda \geq 0$.
  - F is always dominated by $\bar{F}$

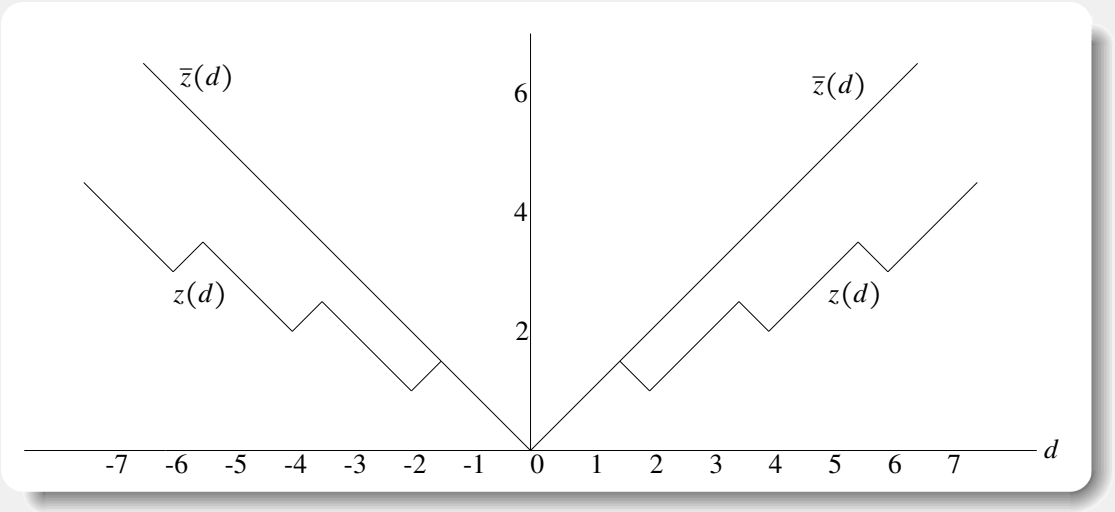


Figure: Directional derivative of the value function of the example.

Subadditive Dual

- (Weak duality) For a given RHS b and a feasible solution pair (x, F), $F(b) \leq cx$.
- (Strong Duality) For a given RHS b, $w(b) = z(b)$. Furthermore, if the primal (or dual) is infeasible, then the dual (or the primal) is infeasible.

Example:

$$
\begin{aligned}
w(d) = \max \quad & F(d) & \\
& F(2) & \leq 1 \\
& F(-2) & \leq 1 \\
& \bar{F}(1) & \leq 1 \\
& \bar{F}(-1) & \leq 1 \\
& F \in \Gamma^m &
\end{aligned}
$$

Consider the feasible functions:
$F_1(d) =$
$\min\{\lceil d/2 \rceil, d + \lceil -d/2 \rceil\}$
$F_2(d) =$
$\min\{-\lceil d/2 \rceil, -d - \lceil -d/2 \rceil\}$

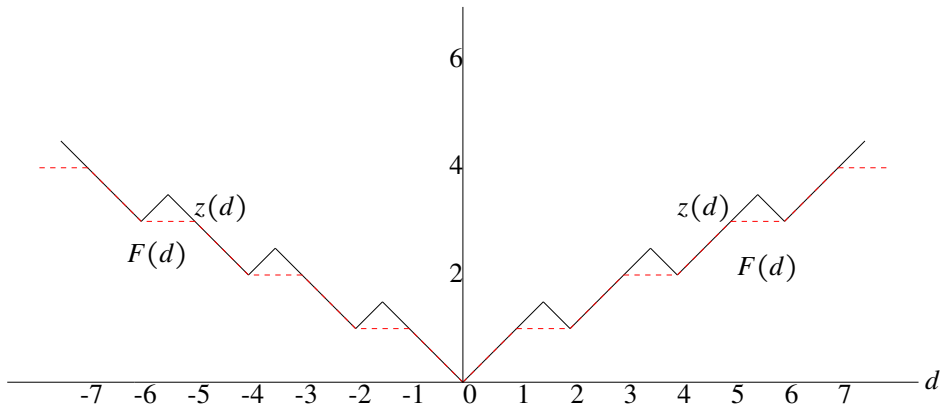


Figure: Observe that $F(d) = \max\{F_1(d), F_2(d)\}$ is an optimal dual function on some line segments and feasible otherwise.

Properties

- If F is an optimal dual solution, we can define the reduced costs.

$$c_j - F(a_j),\ j \in I,\ \ c_j - \bar{F}(a_j),\ j \in N \backslash I$$

- Furthermore, if x is an optimal primal solution, we can define the complementary slackness conditions.

$$\begin{aligned} x_j(c_j - F(a_j)) &= 0,\ \forall j \in I \\ x_j(c_j - \bar{F}(a_j)) &= 0,\ \forall j \in N \backslash I \end{aligned}$$

- If $F \in \Gamma^m$, the inequality

$$\sum_{j \in I} F(a_j)x_j + \sum_{j \in N \backslash I} \bar{F}(a_j)x_j \ \geq\ F(d)$$

  is a valid inequality for MILP with RHS d.
- Any valid inequality is either equivalent or dominated by an inequality in the form above (*subadditive representation*).
- For and d,

$$conv(\mathcal{S}(d)) = \{x : \sum_{j \in I} F(a_j)x_j + \sum_{j \in N \backslash I} \bar{F}(a_j)x_j \ \geq\ F(d),\ F \in \Gamma^m,\ x \geq 0\}$$

Methods

- Neither the general dual problem nor the subadditive dual problem is a manageable program and can be solved with a direct approach.
- There are a number of ways we can utilze to get a feasible (and optimal in some cases) dual functions.
  - Relaxation algorithms
  - Transformation algorithms
  - Primal solution algorithms

Methods

- The Value Function (Jeroslow & Blair -77/79/82/84, Blair -95, Williams -96)
- Cutting Plane/Gomory's Procedure (Gomory -69, Chvátal -73, Wolsey -81)
- Group Relaxation (Gomory -69, Johnson -73, Klabjan -02)
- Lagrangian Relaxation (Fisher -81, Jeroslow & Blair -79, wolsey -81)
- Linear Representation (Jeroslow & Blair -78, Wolsey -81, Lasserre -04)
- Generating Functions (Lasserre -04)
- Branch-and-Bound (Wolsey -81)
- Branch-and-Cut -

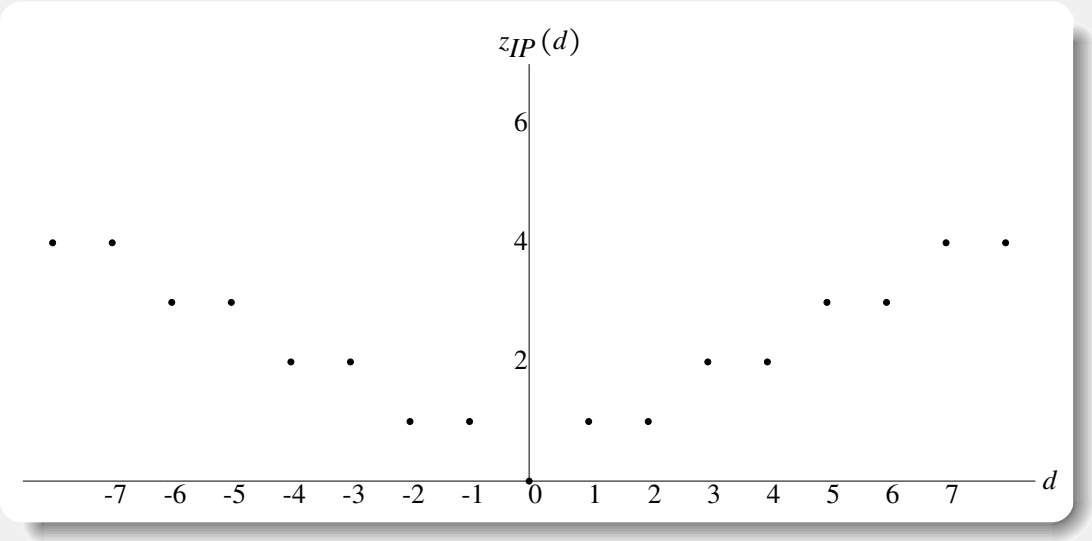
What do we know about the value function?

- It is subadditive over Ω.
- It is piecewise polyhedral.
- For an ILP problem, it can be obtained by a finite number of limited operations on the elements of RHS:

(i) multiplication with rationals
(ii) nonnegative rational combinations
(iii) rounding up
(iv) taking the maximum

} *Chvátal forms.* } *Gomory forms.*

Example: The value function of the example with all variables assumed to be integer is $z_{IP}(d) = \max\{\lceil d/2 \rceil, \lceil -d/2 \rceil\}$ and is clearly a Gomory function.



Jeroslow Formula

- There is one-to-one correspondence between ILP instances and Gomory functions.
- For MILP case, the value function z still can be represented by a Gomory function if $c_j = 0,\ j \in N \backslash I$ or otherwise, can be written as a minimum of finitely many Gomory functions.
- *Jeroslow formula* is a closed-form equivalent of the value function consisting of a Gomory function and a correction term.

Example: The value function of the example is

$$z(d) = \min \left\{ \begin{array}{l} \max \left\{ \left\lceil \frac{\lceil d \rceil}{2} \right\rceil, \left\lceil \frac{-\lceil d \rceil}{2} \right\rceil \right\} + \lceil d \rceil - d, \\ \max \left\{ \left\lceil \frac{-\lceil -d \rceil}{2} \right\rceil, \left\lceil \frac{\lceil -d \rceil}{2} \right\rceil \right\} + \lceil d \rceil + d \end{array} \right\}$$

- There is a Chvátal function that is optimal to the subadditive dual of an ILP with RHS $b \in \Omega_{IP}$ and $z_{IP}(b) > -\infty$.
- The procedure:
  In iteration k, we solve the following LP

$$\begin{aligned} z_{IP}(b)^{k-1} = \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & \textstyle\sum_{j=1}^{n} f^i(a_j)x_j \geq f^i(b) \qquad i = 1, \ldots, k-1 \\ & x \geq 0 \end{aligned}$$

- The k^{th} cut, $k > 1$, is dependent on the RHS and written as:

$$f^k(d) = \left\lceil \sum_{i=1}^{m} \lambda_i^{k-1} d_i + \sum_{i=1}^{k-1} \lambda_{m+i}^{k-1} f^i(d) \right\rceil \quad \text{where} \quad \lambda^{k-1} = (\lambda_1^{k-1}, \ldots, \lambda_{m+k-1}^{k-1}) \geq 0$$

- Assume that $b \in \Omega_{IP}$, $z_{IP}(b) > -\infty$ and the algorithm terminates after $k+1$ iterations.
- If u^k is the optimal dual solution to the LP in the final iteration, then

$$F^k(d) = \sum_{i=1}^{m} u_i^k d_i + \sum_{i=1}^{k} u_{m+i}^k f^i(d),$$

is a Chvátal function with $F^k(b) = z_{IP}(b)$ and furthermore, it is optimal to the subadditive ILP dual problem.

Example: Let $b = 3$. At first iteration, we add the constraint

$$\lceil 2/2 \rceil x_1 + \lceil -2/2 \rceil x_2 + \lceil 1/2 \rceil x_3 + \lceil -1/2 \rceil x_4 \geq \lceil 3/2 \rceil$$

from the weight $\lambda_1 = 1/2$, i.e., the cut $x_1 - x_2 + x_3 \geq 2$. After resolving, we get an integer primal solution with the dual solution $u = (0, 1)$. Then the corresponding optimal dual function is:

$$F^1(d) = 0d + 1\lceil d/2 \rceil = \lceil d/2 \rceil$$

- What does this mean?

The algorithm

In a typical branch-and-bound,

- The original feasible region $\mathcal{P}$ is divided into a *partition* $\mathcal{P}_1, ..., \mathcal{P}_k$ such that $\mathcal{P} \equiv \cup_k \mathcal{P}_k$.
- At each tree node, that corresponds to a member $\mathcal{P}_t$, we basically solve an LP relaxation to get a lower bound.
- According to the optimal solutions of the LP relaxations of analyzed nodes, we keep an upper bound obtained from the optimal solution values of those nodes that yield a feasible solution to original problem.
- We prune node t
  - *feasibly* if it has either an optimal solution value greater than the current upper bound or an optimal solution which is a feasible solution to the original problem,
  - *in-feasibly* if it is just infeasible.
- Algorithm exits when all partition members are analyzed it is guaranteed that iterating further would not decrease the upper bound.

In a typical branch-and-cut, we also generate cuts at each node.

- It is easy to get a feasible dual function from branch-and-bound.
- For branch-and-cut, we have to take care of the cuts.
  - Case 1: Do we know the subadditive representation of each cut?
  - Case 2: Do we know the RHS dependency of each cut?
  - Case 3: Otherwise, we can use some proximity results or the variable bounds.

Case 1

If we know the subadditive representation of each cut:
At a node t, we solve the LP relaxation of the following problem

$$\begin{aligned} z^t(b) = \min \quad & cx & & \\ \text{s.t} \quad & Ax & \geq & \ b \\ & x & \geq & \ l^t \\ & -x & \geq & -\mathrm{g}^t \\ & H^t x & \geq & \ h^t \\ & x & \in & \ \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \end{aligned}$$

where $\mathrm{g}^t,\ l^t \in \mathbb{R}^r$ are the branching bounds applied to the integer variables and $H^t x \geq h^t$ is the set of added cuts in the form

$$\sum_{j \in I} F^t_k(a^k_j) x_j + \sum_{j \in N \setminus I} \bar{F}^t_k(a^k_j) x_j \ \geq \ F^t_k(\sigma_k(b)) \qquad k = 1, \ldots, \nu(t),$$

$\nu(t)$: the number of cuts generated so far,
$a^k_j, j = 1, \ldots, n$: the columns of the problem that the k^{th} cut is constructed from,
$\sigma_k(b)$: is the mapping of b to the RHS of the corresponding problem.

Case 1

Let T be the set of leaf nodes, $u^t, \underline{u}^t, \overline{u}^t$ and w^t be the dual feasible solution used to prune $t \in T$. Then,

$$F(d) = \min_{t \in T} \{u^t d + \underline{u}^t l^t - \overline{u}^t g^t + \sum_{k=1}^{\nu(t)} w^t{}_k F_k^t(\sigma_k(d))\}$$

is an optimal dual function, that is, $z(b) = F(b)$.

- We can restore the subadditivity if the variables are bounded and the branching bounds are implied to these bounds.
- However, it is very hard to know the subadditivity characterization of each cut.

Case 2

If we know the RHS dependencies of each cut:
We know for

- Gomory fractional cuts.
- Knapsack cuts
- Mixed-integer Gomory cuts
- ?

Then, we do the same analysis as before.

Case 3

In the absence of subadditive representations or RHS dependencies:
For each node $t \in T$, let $\hat{h}^t$ be such that

$$\hat{h}^t_k = \sum_{j=1}^{n} h^t_{kj} \hat{x}_j \qquad \text{with} \qquad \hat{x}_j = \begin{cases} l^t_j & \text{if } h^t_{kj} \geq 0 \\ g^t_j & \text{otherwise} \end{cases}, \quad k = 1, \ldots, \nu(t)$$

where h^t_{kj} is the k^{th} entry of column h^t_j. Furthermore, define

$$\tilde{h}^t = \sum_{j=1}^{n} h^t_j \tilde{x}_j \qquad \text{with} \qquad \tilde{x}_j = \begin{cases} l^t_j & \text{if } w^t h^t_j \geq 0 \\ g^t_j & \text{otherwise} \end{cases}.$$

Then the function

$$F(d) = \min_{t \in T} \{u^t d + \underline{u}^t l^t - \overline{u}^t g^t + \max\{w^t \tilde{h}^t,\ w^t \hat{h}^t\}\}$$

is a feasible dual solution and hence $F(b)$ yields a lower bound.

- This approach is the easiest way and can be used for bounded MILPs (binary programs), however it is unlikely to get an effective dual feasible solution for general MILPs.

Post-optimal analysis

- One may want to study the effect of small data variation on the outcome in order to determine how sensitive the decision made is to perturbation.
- For instance, it may be desirable to see how the optimum behaves when available resources change unexpectedly.
- Various modeling approaches such as *robust* or *stochastic* optimization have been developed to decrease the impact of uncertainty.
- However, the possible scenarios still may not be known a priori or it may be required to optimize again when new problem data become certain

Post-optimal analysis

We discuss the following questions:

- *Local sensitivity analysis:* For what ranges of problem data does the current solution remain optimal?
- *Global parametric analysis* How can we efficiently get all the optimal solutions in a given range of problem data?
- *Warm Starting:* If the current solution is not optimal, can we make use of the information collected through the solution process to get or estimate the optimal solution of the modified problem?

In the LP case, these questions are well-studied, however the techniques introduced for MILPs are very limited and depend on the *sufficient optimality conditions* associated with the solution procedure.

Basic Observations:

Let (x^*, F^*) (with F^* subadditive) be the optimal primal-dual solution pair for MILP with (A, b, c).

$(A, b, c) \rightarrow (A, \tilde{b}, c)$:

1. F^* remains dual feasible.
2. Let $Y^* = \{y \mid F^*(Ay) = cy\}$. If F^* is still optimal, then the new optimal solution is in Y^*.

Basic Observations:

$(A, b, c) \rightarrow (A, b, \tilde{c})$:

1. x^* remains primal feasible.
2. If $F^*(a_j) \leq \tilde{c}_j$ for all j, then F^* remains dual feasible.
3. If $F^*(a_j) \leq \tilde{c}_j$ when $x^*_j = 0$ and $\tilde{c}_j = c_j$ otherwise, then x^* remains optimal.
4. If $c_j \leq \tilde{c}_j$ when $x^*_j = 0$ and $\tilde{c}_j = c_j$ otherwise, then x^* remains optimal.
5. Let the vector g denote the upper bounds on the variables. If $\tilde{c}_j \leq c_j$ when $x^*_j = \mathrm{g}^*_j$ and $\tilde{c}_j = c_j$ otherwise, then x^* remains optimal.

Note that if one drops the subadditivity requirement of F^*, then (2) and (3) are no longer valid.

Basic Observations:

$(A, b, c) \rightarrow (\tilde{A}, \tilde{b}, \tilde{c})$:

1. If the original problem is a relaxation of the new problem, then x^* remains optimal.
2. When a new activity $(\tilde{c}_{n+1}, \tilde{a}_{n+1})$ is introduced, $(x^*, 0)$ remains feasible.
3. Furthermore, if $F^*(\tilde{a}_{n+1}) \leq \tilde{c}_{n+1}$, then $(x^*, 0)$ remains optimal since F^* remains dual feasible and $cx* = F^(b)$.
4. When a new constraint $(\tilde{a}^{m+1})$ with RHS $\tilde{b}_{m+1}$ is introduced, if x^* is still feasible, then it remains optimal.
5. Furthermore, let the function $\tilde{F} : \mathbb{R}^{m+1} \rightarrow \mathbb{R}^m$ be defined by $\tilde{F}(d, d_{m+1}) = F^*(d)$. Then $\tilde{F}$ is dual feasible for the new problem since $\tilde{F}$ is subadditive and $\tilde{F}(a_j, \tilde{a}_j^{m+1}) = F^*(a_j) \leq c_j,\ j \in I$.

Note that if one drops the subadditivity requirement of F^*, then (3) is no longer valid

Cutting Plane Algorithm

- For an ILP with (A, b, c), assume in iteration $(k-1)$ algorithm terminates and we know either the subadditive representations or the RHS dependencies.
- Let B^k be the final optimal basis and define the function $\varphi^k(d) = (d, f^1(d), ..., f^{k-1}(d))$.
- Note that when we change the RHS to $\varphi^k(\tilde{b})$, existing cuts are modified to be valid for $\mathcal{S}_{IP}(\tilde{b})$ and B^k still remains dual feasible.
- If $\tilde{x} = B^{k-1}\varphi^k(\tilde{b}) \geq 0$ and integer, then it is an optimal solution to the modified ILP problem with $(A, c, \tilde{b})$.

Branch-and-Cut Algorithm

- Let B^t be the optimal basis for the LP relaxation of leaf node t and define the function
$$\varphi^t(d) = (d, F_1^t(\sigma_1(d)), ..., F_{\nu(t)}^t(\sigma_{\nu(t)}(d)))$$
to shift the RHS of node t when b is modified. Furthermore, let
$$\omega^t(d) = u^t d + \underline{u}^t l^t - \overline{u}^t g^t + \sum_{k=1}^{\nu(t)} w^t{}_k F_k^t(\sigma_k(d))$$
with current feasible solution $(u^t, \underline{u}^t, \overline{u}^t, w^t{}_k)$.
- Let F be the dual function derived as we discussed before and let the upper bound function be
$$V(d) = \min_t \{\omega^t(d) \mid B^{t\,-1}\varphi^t(d) \geq 0 \text{ and } \left(B^{t\,-1}\varphi^t(d)\right)_j \text{ integer for all j } \in I\}.$$
- For a MILP with $(A, \tilde{b}, c)$, if $V(\tilde{b}) = F(\tilde{b})$, then $z(\tilde{b}) = V(\tilde{b})$. Otherwise, $F(\tilde{b}) \leq z(\tilde{b}) \leq V(\tilde{b})$.

Branch-and-Cut Algorithm

- The sufficient conditions above are unlikely to be satisfied, even if the current solution remains optimal
- Therefore, instead of testing sufficiency for a problem with $\tilde{b}$, it might be better to determine the ranges over which the RHS can be shifted while the current solution remains optimal.
- For each feasibly pruned leaf node t, let the range $\Delta^t(b)$ be such that $B^{t-1}(\varphi^t(b) + \varphi^t(\Delta^t(b))) \geq 0$. Then, the current solution remains optimal for $d \in \cap_t [b + \Delta^t(b)]$.

Branch-and-Cut Algorithm

- Even if the above tests fail, one can still get useful bounds on the optimal solution value.
- For instance, Wolsey and Schrage[85] show that data collected from the branch-and-bound tree can be used to get a lower bound for modified problem instance.
  - Utilizing the same tree with the new RHS, they get bounds for each modified subproblem from LP relaxation dual solutions.
  - These bounds are are then recursively added to yield the best possible lower bound for the modified problem
- There are some other tools for sensitivity analysis that are independent of the solution algorithm
  - Assume in ILP, $\mathcal{S}_{IP}(d) = \{x \in \mathbb{Z}^n_+ \ | Ax \geq d\}$. Let $\Delta(A)$ be the maximum absolute values of the determinants of the square submatrices of A. Then,

$$|z_I(b) - z_I(\tilde{b})| \leq n\Delta(A)(\|b - \tilde{b}\|_\infty + 2)\|c\|_1 .$$

Cutting Plane Algorithm

- Let A^k be the constraint matrix of the last subproblem.
- If $\tilde{c}_{B^k} {B^k}^{-1} A^k - \tilde{c} \geq 0$, then the current solution is also optimal to the problem with $(A, b, \tilde{c})$.

Branch-and-Cut Algorithm

- Let

$$W(q) = \min_t \{qx^t \mid x^t = B^{t-1}\varphi^t(b) \geq 0 \text{ and } \left(B^{t-1}\varphi^t(b)\right)_j \text{ integer for all j } \in I\}.$$

  be the upper bound function
- Furthermore, let

$$G(q) = \min_t \{\omega^t(b) \mid q_j \geq u^t a_j + \underline{u}_j^t - \overline{u}_j^t + w^t h_j^t, \quad j = 1, \ldots, n\}.$$

- If $W(\tilde{c}) = G(\tilde{c})$, then $Z(\tilde{c}) = W(\tilde{c})$. Otherwise, $G(\tilde{c}) \leq Z(\tilde{c}) \leq W(\tilde{c})$.

- *Warm starting* is a technique to collect information while the solution procedure of one instance is in progress and to make use of this information to initiate the solution procedure for a related problem.
- In the LP case, warm starting information consists of the optimal basis of the previous problem.
- Similarly, we define the warm starting information for a MILP with the components of the solution algorithm.
- For instance, in branch-and-cut algorithm, it refers to a given subtree (starting partition), final basis of each node and enough information to shift each cut to be valid for a given modification.

Note that any subtree of the previous solution algorithm can be used as a starting partition.

Warm Starting Procedure

Let T^w is given as an input to resolve a modified problem.

- Check if T^w is a partition of the modified problem.
- Check the feasibility of current LP basis of each leaf node for the LP relaxation of modified node.
- Check the sufficient conditions to see if a resolve is needed.
- Get upper and lower bounds. Form a candidate list with these nodes and continue branch-and-cut algorithm.

SYMPHONY

- *SYMPHONY* software framework which is a customizable open-source parallel MILP solver maintained by Prof. Ralphs.
- It was first introduced as a flexible, parallel solver for hard combinatorial problems.
- The default solver behaviors can be altered with hundreds of parameters and over 50 *user callback* functions.
- The user can have complete control over branching rules, cutting plane generation, management of the cut pool, processing of several tree nodes, diving strategies, and limited column generation.
- SYMPHONY includes modules for solving the
  - Traveling Salesman Problem,
  - Vehicle Routing Problem,
  - Set Partitioning Problem,
  - Mixed Postman Problem,
  - Matching Problem,
  - Capacitated Network Routing Problem and
  - Multi-Criteria Knapsack Problem.

SYMPHONY

Recently, SYMPHONY has been fully integrated with the libraries of the Computational Infrastructure for Operations Research (COIN) repository, including

- The Cut Generation Library (CGL) for generating cutting planes,
- The Open Solver Interface (OSI), a C++ class that provides a standard API for accessing a variety of solvers.
- SYMPHONY is now implemented as a callable library and can be used as a stand-alone generic MILP solver accessible either through calls to the native C subroutines of the API or through a C++ class derived from the COIN-OR OSI.

Callable Library

- Through callable library, SYMPHONY can be easily embedded to any other code to be used with any other algorithms.
- The main functions are:
  - sym_open_environment()
  - sym_parse_command_line()
  - sym_load_problem()
  - sym_explicit_load_problem()
  - sym_solve()
  - sym_resolve()
  - sym_mc_solve()
  - sym_close_environment()

Callable Library

As an example of the use of the library functions, Figure shows the code for implementing a generic MILP solver with default parameter settings.

```
int main(int argc, char **argv)
{
  sym_environment *p = sym_open_environment();
  sym_parse_command_line(p, argc, argv);
  sym_load_problem(p);
  sym_solve(p);
  sym_close_environment(p);
}
```

It is possible to read an MPS or GMPL/AMPL file.

OSI Interface

```
int main(int argc, char **argv)
{
  OsiSymSolverInterface si;
  si.parseCommandLine(argc, argv);
  si.loadProblem();
  si.initialSolve();
}
```

Implementation

- We have added to SYMPHONY warm starting capability and basic sensitivity analysis tools.
- For now, because of the complications we have described before, these methods can be invoked only for rim vectors modifications and when cut generation is disabled (only for modifying the right-hand-side).
- We intend to extend these techniques to other possible modification and to be equipped with advanced sensitivity analysis tools.

Warm Starting

When user asks SYMPHONY to keep warm starting information,

- The warm start structure starts to keep the description of the tree as well as the other auxiliary data needed to restart the computation. This description contains
  - complete information about the subproblem corresponding to each node,
  - the branching decisions,
  - and warm start information for the LP relaxation of the subproblem.
- All information is stored compactly using SYMPHONY's native data structures by storing only the differences between a child and its parent.
- In addition to the tree itself, other relevant information such as the current bounds and the best feasible solution found so far.

The user

- can save a warm start to disk,
- read one from disk,
- or restart the computation from any warm start after modifications.

Modifying Parameters

- The solution process can be interrupted with a satisfied condition and some regulators of the algorithm are changed.
- The tree manager simply traverses the tree and loads in leaf nodes marked as candidates for processing.

```
int main(int argc, char **argv)
{
  OsiSymSolverInterface si;
  si.parseCommandLine(argc, argv);
  si.loadProblem();
  si.setSymParam(OsiSymFindFirstFeasible, true);
  si.setSymParam(OsiSymSearchStrategy, DEPTH_FIRST_SEARCH);
  si.setSymParam(OsiSymKeepWarmStart, true);
  si.initialSolve();
  si.setSymParam(OsiSymFindFirstFeasible, false);
  si.setSymParam(OsiSymSearchStrategy, BEST_FIRST_SEARCH);
  si.resolve();
}
```

Modifying Problem Data

- In this case, SYMPHONY makes corresponding modifications to the leaves of the current search tree to allow execution of the algorithm to continue.
- To initialize the algorithm, we first check which partition of final tree should be used. We have four options:
  - Take the first k nodes.
  - Take all nodes above level k in the tree.
  - Take the first $p\%$ of the nodes.
  - Take all nodes above the level $p\%$ of the tree depth.
- If any of these are chosen, all nodes but those in chosen subtree are discarded and the computation is warm started from this given subtree.
- Then, each leaf node, regardless of its status after termination of the previous solve call, are inserted into the queue of candidate nodes and reprocessed with the changed rim vectors.

Modifying Problem Data

Use of SYMPHONY's warm start capability.

```
 int main(int argc, char **argv)
{
   OsiSymSolverInterface si;
   si.parseCommandLine(argc, argv);
   si.setSymParam(OsiSymKeepWarmStart, true);
   si.initialSolve();
   for(int i=100; i<110; i++){
     si.setObjCoeff(i, 400);
   }
   si.setSymParam(OsiSymWarmStartTreeLevel, 6);
   si.resolve();
}
```

Modifying Problem Data

Use of SYMPHONY's warm start capability.

```
 int main(int argc, char **argv)
{
   OsiSymSolverInterface si;
   CoinWarmStart * ws;
   si.parseCommandLine(argc, argv);
   si.setSymParam(OsiSymNodeLimit, 100);
   si.setSymParam(OsiSymKeepWarmStart, true);
   si.initialSolve();
   ws = si.getWarmStart();
   si.setSymParam(OsiSymNodeLimit, -1);
   si.resolve();
   for(int i=100; i<110; i++){
     si.setObjCoeff(i, 400);
   }
   si.setWarmStart(ws);
   si.resolve();
   si.initialSolve();
```

Sensitivity Analysis

- Our warm start structure is also equipped to collect more information, namely,
  - the feasible solutions found so far and
  - the dual solutions of each node.
- In order to save memory, these solutions are kept in sparse form.
- For any modification, upper and lower bounds are obtained by sufficient optimality conditions.
- For modifications to the right-hand-side, a better lower bound is obtained by Wolsey and Schrage's algorithm.

Sensitivity Analysis

Performing sensitivity analysis with SYMPHONY.

```
  int main(int argc, char **argv)
{
   OsiSymSolverInterface si;
   si.parseCommandLine(argc, argv);
   si.loadProblem();
   si.setSymParam(OsiSymSensitivityAnalysis, true);
   si.initialSolve;
   int ind[2];
   double val[2];
   ind[0] = 4;    val[0] = 7000;
   ind[1] = 7;    val[1] = 6000;
   double lb = si.getLbForNewRhs(ind, ind, val);
}
```

Introduction

As a small demonstration of warm starting we tested the code above on the MIPLIB file **p0201** where the original corresponding coefficients vary between 300 and 500.

	CPU Time	Search Tree Nodes
Generate warm start	16	100
Solve original problem (from warm start)	1	106
Solve modified problem (from scratch)	27	166
Solve modified problem (from warm start)	4	195

Table: Warm starting a computation with **p0201**

Using Warm Starting: Change in the Objective Function

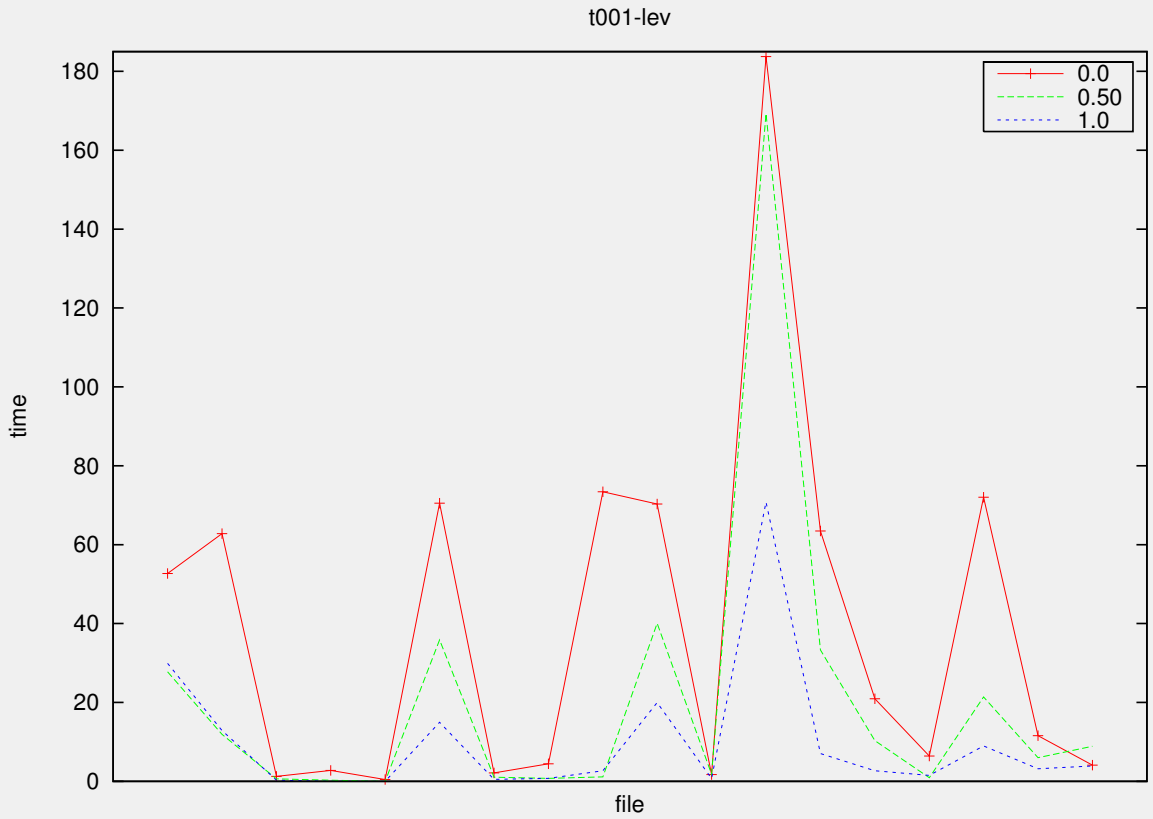


Table: Warm start after 1% modification on a random subset of objective coefficients of random size. Warm start consists of nodes above the r% level of the tree, $r \in \{0, 50, 100\}$

Using Warm Starting: Change in the Objective Function

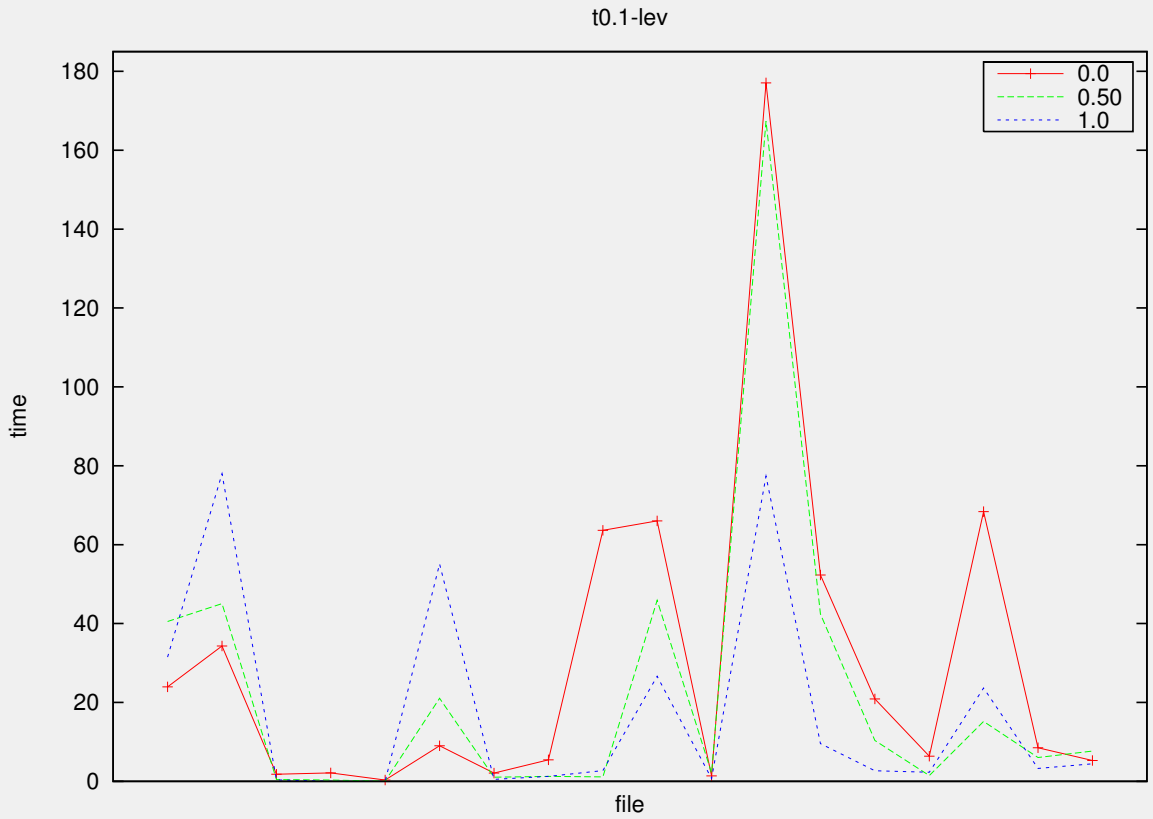


Table: Warm start after 10% modification on a random subset of objective coefficients of random size. Warm start consists of nodes above the r% level of the tree, $r \in \{0, 50, 100\}$

Using Warm Starting: Change in the Objective Function

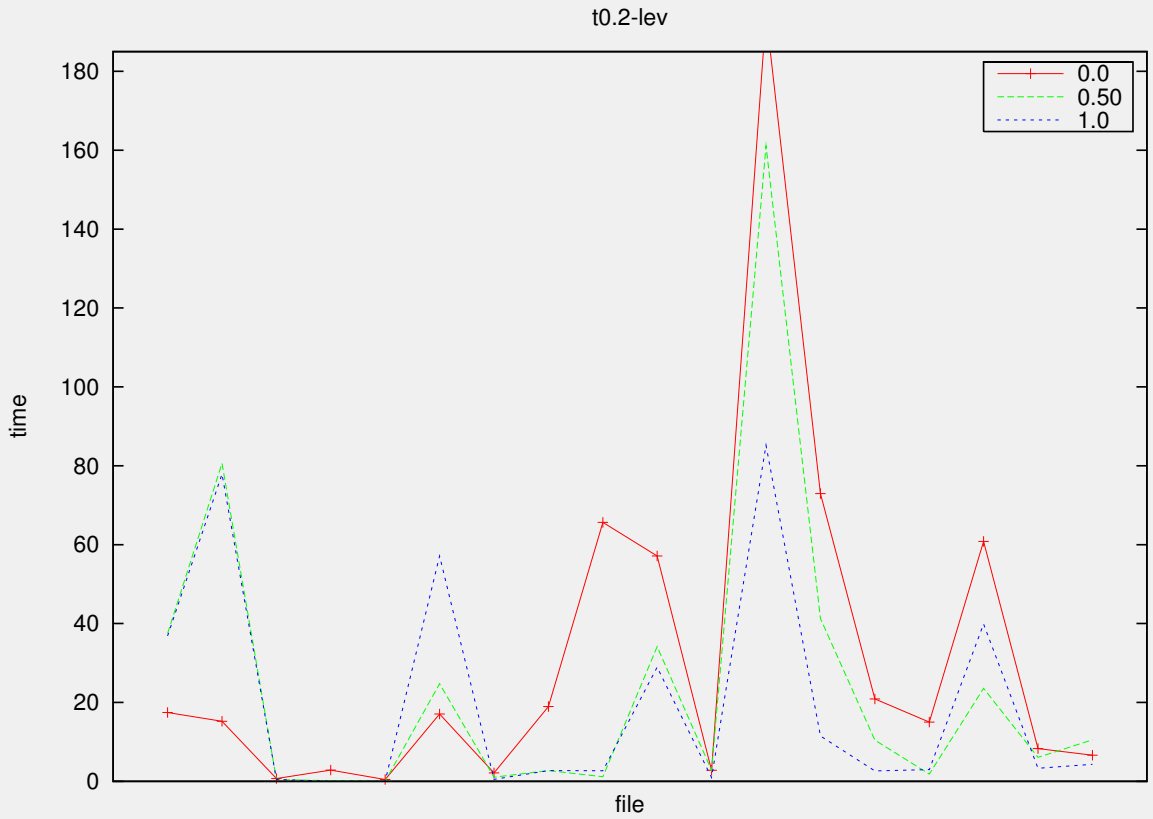
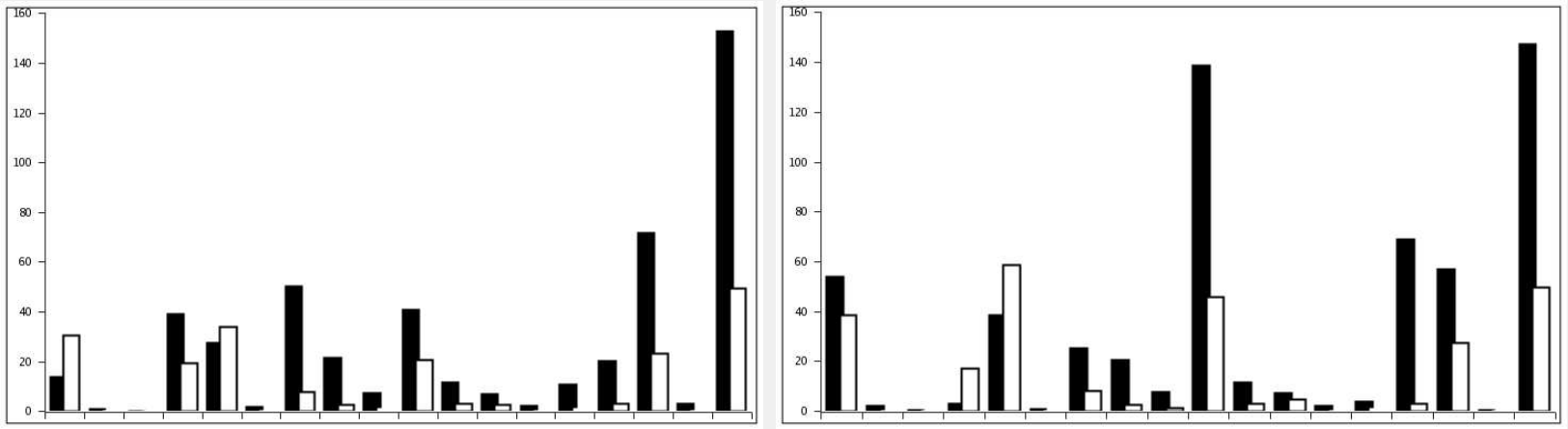


Table: Warm start after 20% modification on a random subset of objective coefficients of random size and use the nodes above the r% level of the tree, $r \in \{0, 50, 100\}$

Using Warm Starting: Change in the Objective Function



Black: without warm starting
White: with warm starting

Table: Warm start after random perturbation of $+/-10\%$ on a random subset of objective coefficients of size $0.1n$ (left) and of size $0.2n$ (right)

Example: Warm-Start on MIPLIB problems/RHS Case

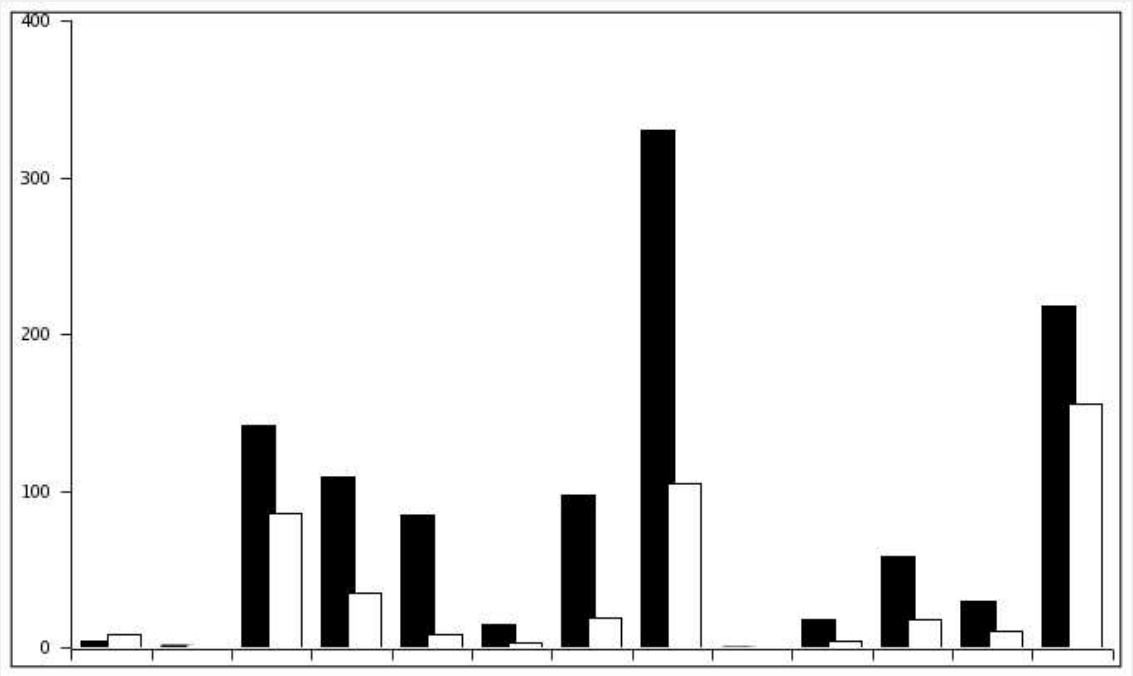


Table: Warm start after random perturbation of $+/-20\%$ on a random subset of rhs of size $0.1m$.

Using Warm Starting: Change in the Right-hand Side

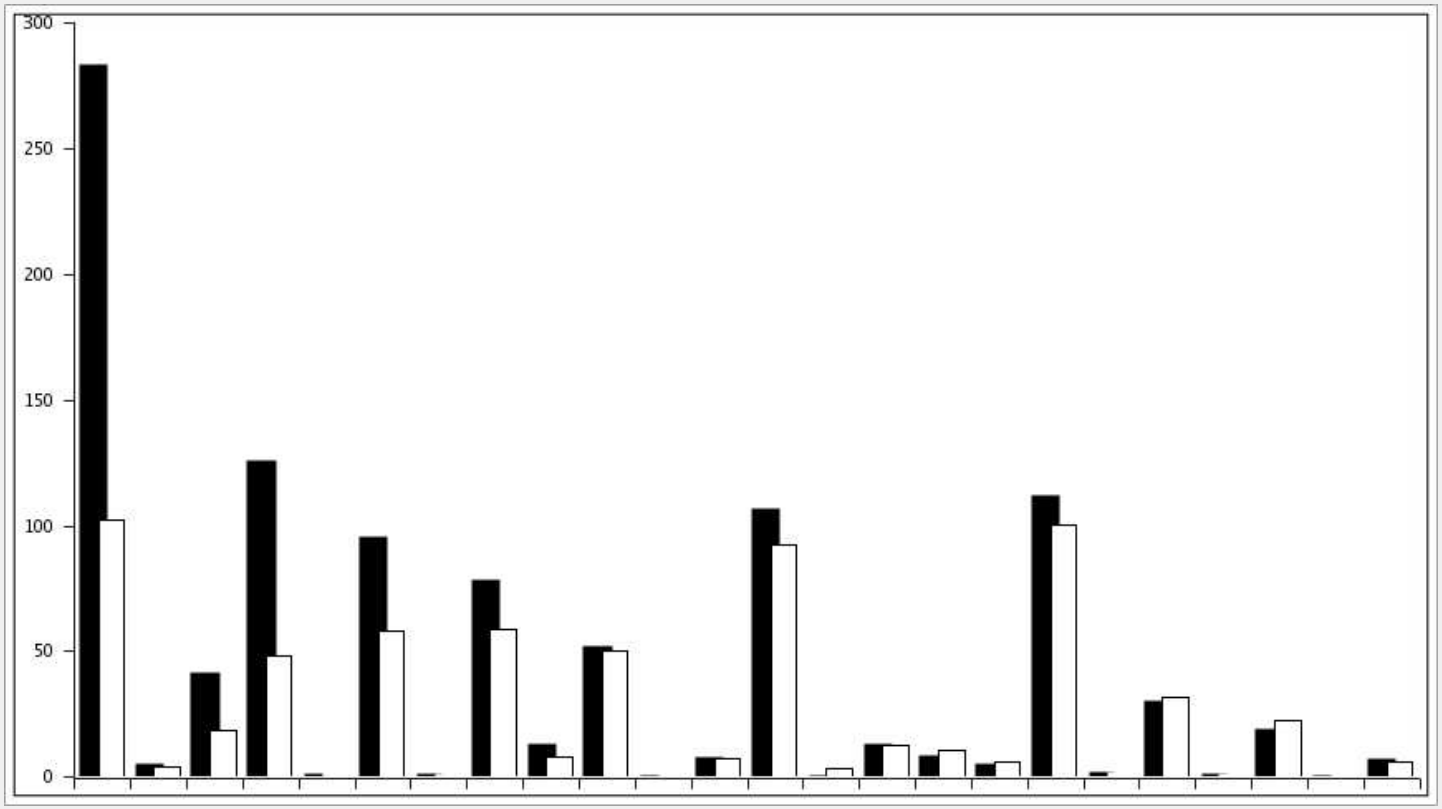


Table: Change rhs b of a knapsack problem between $b/2$ and $3b/2$ and warm start using the nodes above the 25% level of the tree.

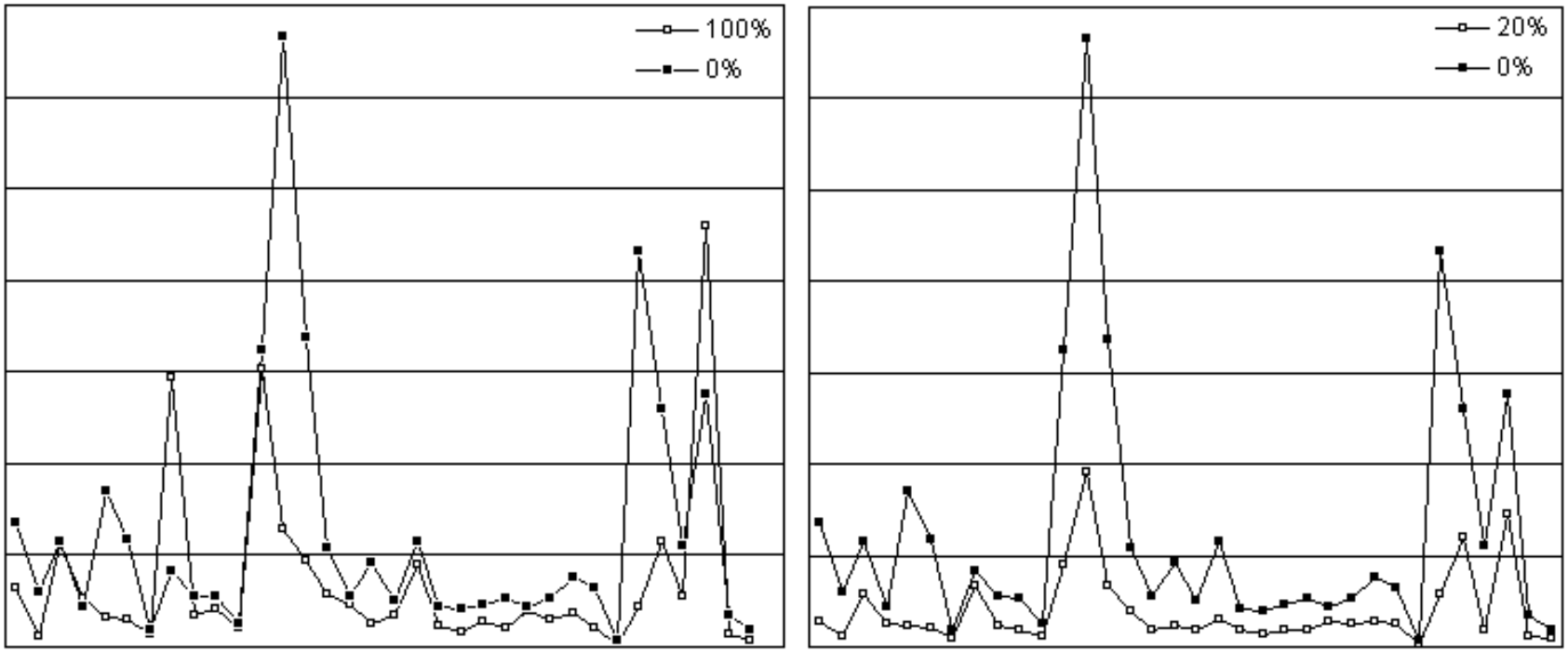
- The ability to resolve after modifying problem data is not only useful for sensitivity analysis but also has a wide range of applications in practice.
  - Decomposition methods
  - Stochastic MILP problems
  - Parametric MILP problems
  - Multicriteria MILP problems
  - Determining irreducibe infeasible subsystems

  depend on solving a family of integer programs.
- To illustrate the warm starting capability, we have implemented algorithms for solving 2-stage stochastic integer programs and bicriteria integer programs.

Stochastic Programming

Problem	Tree Size Without WS	Tree Size With WS	% Gap Without WS	% Gap With WS	CPU Without WS	CPU With WS
storm8	1	1	-	-	14.75	8.71
storm27	5	5	-	-	69.48	48.99
storm125	3	3	-	-	322.58	176.88
LandS27	71	69	-	-	6.50	4.99
LandS125	37	29	-	-	15.72	12.72
LandS216	39	35	-	-	30.59	24.80
dcap233_200	39	61	-	-	256.19	120.86
dcap233_300	111	89	0.387	-	1672.48	498.14
dcap233_500	21	36	24.701	14.831	1003	1004
dcap243_200	37	53	0.622	0.485	1244.17	1202.75
dcap243_300	64	220	0.0691	0.0461	1140.12	1150.35
dcap243_500	29	113	0.357	0.186	1219.17	1200.57
sizes3	225	165	-	-	789.71	219.92
sizes5	345	241	-	-	964.60	691.98
sizes10	241	429	0.104	0.0436	1671.25	1666.75

Bicriteria Programming



Sensitivity Analysis We report the results of testing the code above on the MIPLIB3 file **flugpl**. See Table 4.11 in the paper.

Sensitivity Analysis

Biciteria solver can also be used for complete parametric analysis of a single objective coefficient. As an illustration of this type of sensitivity analysis, we applied the followings:

```
int main(int argc, char **argv)
{
   OsiSymSolverInterface si;
   si.parseCommandLine(argc, argv);
   si.loadProblem();
   si.setObj2Coeff(0, 1);
   si.multiCriteriaBranchAndBound();
}
```

Sensitivity Analysis

$$
\begin{array}{lrcl}
\min & -8x_1 + \varphi x_2 & & \\
\text{s.t.} & 7x_1 + x_2 & \leq & 56 \\
& 28x_1 + 9x_2 & \leq & 252 \\
& 3x_1 + 7x_2 & \leq & 105 \\
& x_1, x_2 & \geq & 0, \text{integral}
\end{array}
$$

φ range	$Z_2(\varphi)$	x_1^*	x_2^*
$(-\infty, -16.000)$	15φ	0	15
$(-16.000, -8.000)$	$-32 + 13\varphi$	4	13
$(-8.000, -2.667)$	$-40 + 12\varphi$	5	12
$(-2.667, -1.333)$	$-56 + 6\varphi$	7	6
$(-1.333, \infty)$	-64	-8	0

"If paradise on earth exists anywhere in the world, it cannot lie very far from here!"
Stefan Zweig quoting Amerigo Vespucci – A Land of The Future

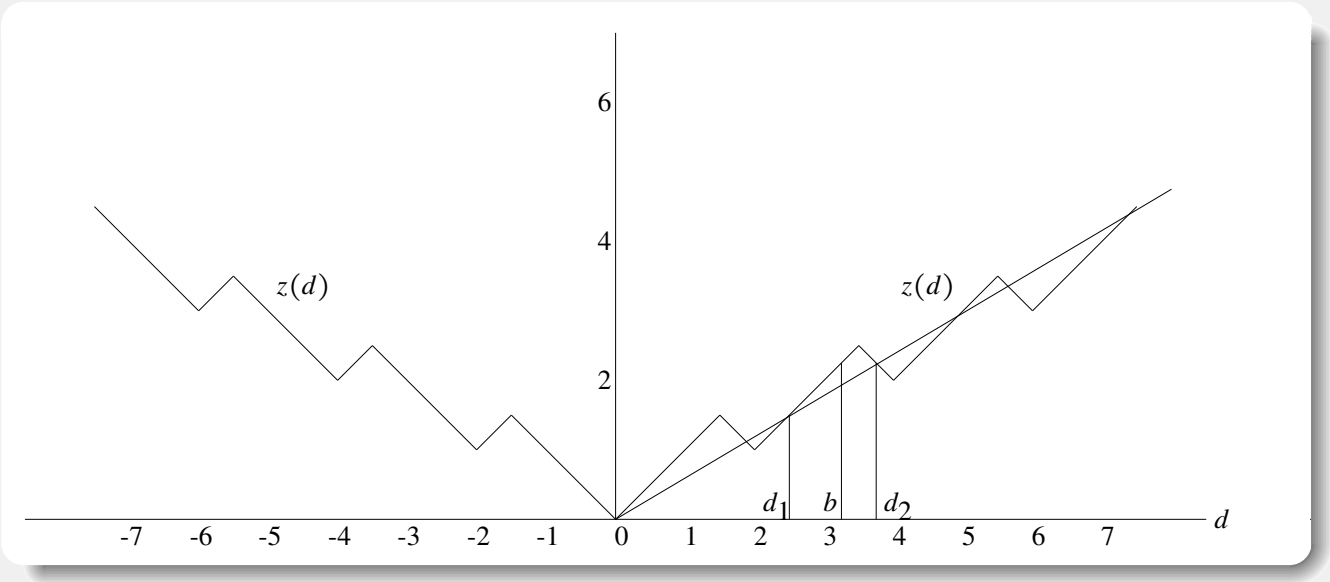
or

"Even hallucinations have laws!"
Angels in America

- Duality theory for integer programming models is still far behind the expectations and advances achieved for the linear programming case.
- We have only a little information about the properties of dual functions,
- It is not easy to get a dual function even from the primal solution algorithm.
  - For Gomory's cutting plane algorithm, the rank of the cuts may be so large and therefore, the formulas may be so nested
  - Consequently the difficulties in managing implementation issues for Lagrangian relaxation quadratic counterpart the chances are good that convergence problems would also exist
  - Klabjan's formulation and the suggested algorithm is a considerable step however, the row generation stage, which depends on enumeration of feasible vectors, makes it a less preferable choice.
  - Currently, there is no easy way to extract an effective dual function from the information produced by a branch-and-cut algorithm.
- In a similar fashion, Lasserre's natural dual formulation is useful in theory, but it lacks the flexibility to be applied in practice.
- The value function of a MILP remains as a closed box.
- These difficulties complicate the sensitivity analysis and warm-starting.

Main Goals

- Can we further improve the subadditive dual theory to be more practical by somehow restricting Γ^m or by other means?
- Efficiency? Effectiveness? Consider the graphs:



A linear dual function feasible for all $d \in [d_1, d_2]$.

Main Goals

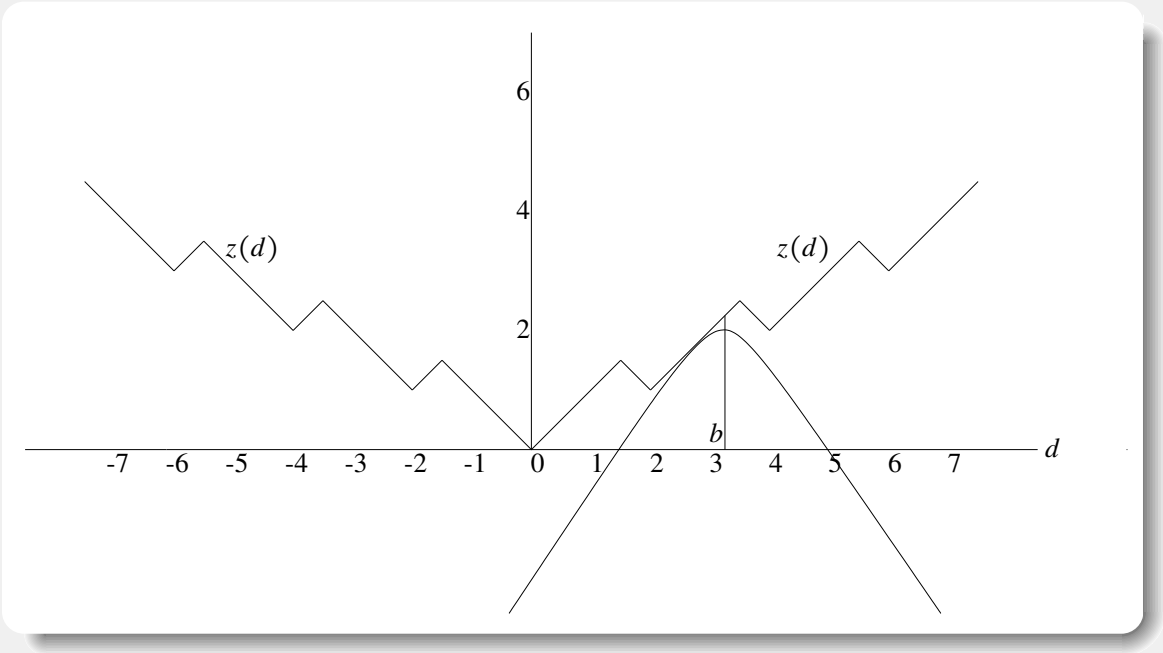


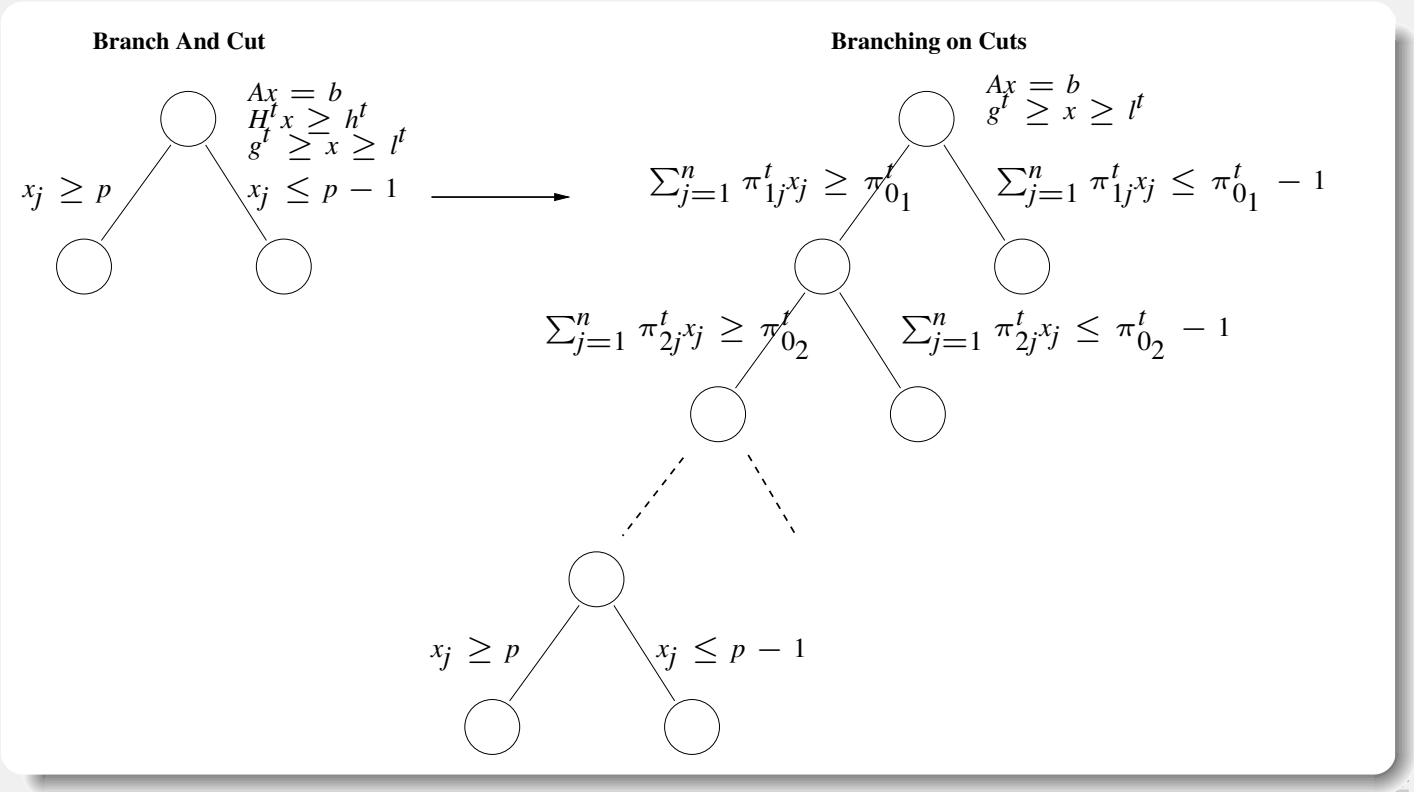
Figure: A concave dual feasible function.

- How can we combine them?

Main Goals

- Using the branch-and-cut algorithm, is there a way to derive a strong subadditive dual function?
- Can we restructure the branch-and-cut tree for this purpose?

Restructuring branch-and-cut tree to branching on cuts tree:



Main Goals

- What is the best strategy to choose T^w to warm start?
- What is the most efficient way to deal with the additional difficulties associated with a full-featured branch-and-cut algorithm, such as preprocessing and reduced cost fixing.
- How can we make further use of the information gathered from the branch-and-cut tree for warm starting? For instance,
  - Assume that a feasible subadditive dual function F is obtained from the solution procedure.
  - The RHS is modified $b \rightarrow \tilde{b}$, and an upper bound $UB(\tilde{b})$ is known for $z_I(\tilde{b})$.
  - If $c_j - F(a_j) > 0$ and

$$v = \left\lceil \frac{UB(\tilde{b}) - F(\tilde{b})}{c_j - F(a_j)} \right\rceil > 0$$

    for a column j, then there is an optimal solution x^* with $x_j^* \leq v - 1$ for the modified problem.
  - So, using reduced cost fixing over this function will preprocess/tighten the variable bounds before warm starting.

Other Questions

- How can we make use of all the information gathered during the solution procedure in order to extract the best sensitivity information, possibly at the cost of losing the feasibility of dual function for all RHS.
- Can we somehow derive a formula similar to Jeroslow's that will extend the subadditive dual solutions of ILP to MILP?
- How about the relation between these dual formulations?
- Are there specific dual formulations or dual functions optimal to some structured problems?
- Can we make use of MILP duality further to fill the gaps and to develop methods analogous to those in LP duality theory?

Other Queestions

- How can we use all the gathered information to come up with a dual feasible function that will yield the best bound for a modified RHS?
- Observe that the function obtained from Wolsey' and Schrage's algorithm is nothing but another and probably better dual feasible function than the one introduced before, since their algorithm also includes dual information coming from the other nodes.
- This formulation reveals the fact dual feasible function that can be obtained from a branch-and-cut tree is not unique.

Other Questions

- For upper bound analysis, we believe that there is much more information that can be extracted from the most common solution algorithm.
- Note the feasible solutions $x_k, k \in K$.
  - Clearly, when the objective function is modified, $x = \sum_{i \in K} \lambda_i x_i$ is another feasible solution to the modified problem if $x \geq 0$ and $\sum_{i \in K} \lambda_i = 1$ for all $\lambda_i \in Z$.
- Note that bicriteria optimization could be used for objective case parametric programs. Can we extend the same idea to RHS and constraint matrix parametric programs?