Duality, Warm Starting, and Sensitivity Analysis for MILP

Ted Ralphs and Menal Guzelsoy
Industrial and Systems Engineering
Lehigh University

SAS Institute, Cary, NC, Friday, August 19, 2005

Outline of Talk

- A little bit of theory
  - Duality
  - Sensitivity analysis
  - Warm starting
- A little bit of computation
  - SYMPHONY 5.1
  - Examples

Introduction to Duality

- For an optimization problem

$$z = \min\{f(x) \mid x \in X\},$$

called the *primal problem*, an optimization problem

$$w = \max\{g(u) \mid u \in U\}$$

such that $w \leq z$ is called a *dual problem*.

- It is a *strong dual* if $w = z$.
- Uses for the dual problem
  - Bounding
  - Deriving optimality conditions
  - Sensitivity analysis
  - Warm starting

Some Previous Work

- R. Gomory (and W. Baumol) ('60–'73)
- G. Roodman ('72)
- E. Johnson (and Burdet) ('72–'81)
- R. Jeroslow (and C. Blair) ('77-'85)
- A. Geoffrion and R. Nauss ('77)
- D. Klein and S. Holm ('79–'84)
- L. Wolsey (and L. Schrage) ('81–'84)
- ...
- D. Klabjan ('02)

Duals for ILP

- Let $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$ nonempty for $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$.
- We consider the (bounded) pure integer linear program $\min_{x \in \mathcal{P} \cap \mathbb{Z}^n} c^\top x$ for $c \in \mathbb{R}^n$.
- The most common dual for this ILP is the well-known *Lagrangian dual*.
  - The Lagrangian dual is not generally strong.
  - Blair and Jeroslow discussed how to make the Lagrangian dual strong by for ILP by introducing a quadratic penalty term.
- How do we derive a strong dual? Consider the following more formal notion of dual (Wolsey).

$$
\begin{aligned}
w_{IP}^g &= \max_{g:\mathbb{R}^m \to \mathbb{R}} \{g(b) \mid g(Ax) \leq c^\top x, x \geq 0\} && (1) \\
&= \max_{g:\mathbb{R}^m \to \mathbb{R}} \{g(b) \mid g(d) \leq z_{IP}(d), d \in \mathbb{R}^m\}, && (2)
\end{aligned}
$$

where $z_{IP}(d) = \min_{x \in \mathcal{P}^I(d)} c^\top x$ is the *value function* and $\mathcal{P}^I(d) = \{x \in \mathbb{Z}^n \mid Ax = d, x \geq 0\}$

Subadditive Duality

- Solutions to the dual (2) bound the value function from below.
- Any function that agrees with the value function at b, including the value function itself, is optimal.
- This shows the dual (2) is strong.
- Question: Under what restrictions on the function g does this remain a strong dual?
  - g linear results in the dual of the LP relaxation $\Rightarrow$ not strong.
  - g convex also results in the dual of the LP relaxation $\Rightarrow$ not strong. (Jeroslow)
  - g subadditive $\Rightarrow$ strong (Gomory; Johnson; Jeroslow).
  - In this case, the dual simplifies to

$$w^s_{IP} = \max\{f(b) \mid f(a^i) \leq c_i, f \text{ subadditive}, f(0) = 0\},$$

  - This is called the *subadditive dual*

Optimality Conditions

- Weak Duality
  - $c^\top x \geq f(b)$ for all $x \in \mathcal{P} \cap \mathbb{Z}^n$ and dual feasible f.
  - The primal problem is infeasible if $w_{IP}^s = \infty$.
  - The dual problem is infeasible if $z_{IP} = \infty$.
- Strong Duality
  - If either the primal problem or the dual problem has a finite optimal value, then
    * There exists $x^* \in \mathcal{P} \cap \mathbb{Z}^n$ and f^* dual feasible such that $c^\top x^* = f^*(b)$.
    * $c_j - f^*(a_j) \geq 0 \; \forall j \in 1, \ldots, n$.
    * If $x_j^* > 0$, then $c_j - f^*(a_j) = 0$.
  - If the primal problem is infeasible, then either the dual is infeasible or $w_{IP}^s = \infty$.
  - If the dual problem is infeasible, then either the primal is infeasible or $z_{IP} = -\infty$.

Subadditivity and Valid Inequalities

- These results on valid inequalities follow directly from strong duality.

  Proposition 1. $\pi^\top x \leq \pi^0 \; \forall x \in \mathcal{P} \cap \mathbb{Z}^n$ *if and only if there exists* f *subadditive with* $f(a_j) \geq \pi_j \; \forall j \in 1, \ldots, n$ *and* $f(b) \leq \pi_0$.

  Proposition 2. conv$(\mathcal{P} \cap \mathbb{Z}^n) = \{x \in \mathbb{R}^n_+ \mid \sum_{i=1}^n f(a_j)x_j \leq f(b) \; \forall \; f$ *subadditive*, $f(0) = 0\}$.

- These results are important for sensitivity analysis and warm starting.
- We can similarly derive theorems of the alternative and other generalizations of standard results from LP.

Optimal Solutions to the Subadditive Dual

- The subadditive dual has most of the nice properties of the LP dual.
- With an optimal solution, we can calculate reduced costs, perform local sensitivity analysis, etc.
- How do we find optimal solutions to the subadditive dual?
- Ideally, these would be a by-product of a practical algorithm.
- If the primal function has a finite optimum, then the value function is an optimal solution.
- The value function has a closed form and is a member of the class $\mathcal{C}$ of *Gomory functions* defined by:

1. $f \in \mathcal{C}$ if $f(v) = \lambda v$ for $\lambda \in \mathbb{Q}^r, r \in \mathbb{N}$.
2. If $f \in \mathcal{C}$, then $\lfloor f \rfloor \in \mathcal{C}$.
3. If $f, g \in \mathcal{C}$ and $\alpha, \beta \geq 0$, then $\alpha f + \beta g \in \mathcal{C}$.
4. If $f, g \in \mathcal{C}$, then $\max\{f, g\} \in \mathcal{C}$.

Optimal Solutions from Gomory's Cutting Plane Procedure

- An optimal dual solution f^* that is a member of the class of *Chvátal functions* (Gomory functions without requirement 4) can be obtaind as a by-product of Gomory's cutting plane procedure.

$$f^*(d) = \sum_{i=1}^{m} \lambda_i^r d_i + \sum_{i=1}^{r} \lambda_{m+i}^r f^i(d)$$

for some finite r, where $f^j(d)$ is defined recursively by

$$f^0(d) = \lambda^{0\top} d, \lambda^0 \in \mathbb{Q}^m$$

$$f^j(d) = \lfloor \sum_{i=1}^{m} \lambda_i^{j-1} d_i + \sum_{i=1}^{j-1} \lambda_{m+i}^{j-1} f^i(d) \rfloor$$

and

$$\lambda^{j+i} = (\lambda_1^{j-1}, ..., \lambda_{m+j-1}^{j-1}) \geq 0$$

.

Another Approach to Generating Optimal Dual Functions

- Klabjan recently suggested another approach for the case when A and b are both nonnegative.
- Given a vector $\alpha \in \mathbb{R}^m$, we define a generator subadditive function $f_\alpha : \mathbb{R}^m_+ \to \mathbb{R}$ by

$$
\begin{aligned}
f_\alpha(d) = \alpha d - max \quad & \textstyle\sum_{i \in E} (\alpha a_i - c_i) x_i \\
& A^E x \leq d \\
& x \in \mathbb{Z}^E_+
\end{aligned}
$$

  where $E = \{i \in N : \alpha a_i > c_i\}$ and A^E is the submatrix of A consisting of the columns with indices in E.
- This can be seen as a generalization of the LP dual.
- Klabjan showed that there always exists an $\alpha \in \mathbb{R}^m$ such that f_α is an optimal dual function.

Example

$$b = \begin{bmatrix} 1 & 2 & 2 & 1 \end{bmatrix}$$

$$c = \begin{bmatrix} 10 & 1 & 3 & 5 & 0 & 2 & -0.5 & -2 & 3 & 4 & -5 & -3 & -1 & 3 & -1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Optimality Conditions from Branch and Bound

- Consider the implicit optimality conditions associated with branch and bound.
- Let $\mathcal{P}_1, \ldots, \mathcal{P}_s$ be a partition of $\mathcal{P}$ into (nonempty) subpolyhedra.
- Let LP_i be the linear program $\min_{x^i \in \mathcal{P}_i} c^\top x^i$ associated with the subpolyhedron $\mathcal{P}_i$.
- Let B^i be an optimal basis for LP_i.
- Then the following is a valid lower bound

$$L = \min\{c_{B^i}(B^i)^{-1}b + \gamma_i \mid 1 \leq i \leq s\},$$

  where γ_i is the constant factor associated with the nonbasic variables fixed at nonzero bounds.
- A similar function yields an upper bound.
- We call a partition that yields lower and upper bounds equal is called an *optimal partition*.

Optimal Dual Functions from Branch and Bound

- The function

$$L(d) = \min\{c_{B^i}(B^i)^{-1}d + \gamma_i \mid 1 \leq i \leq s\},$$

is not subadditive, but provides an optimal solution to (2).

- The corresponding upper bounding function is

$$U(c) = \min\{c_{B^i}(B^i)^{-1}b + \beta_i \mid 1 \leq i \leq s, \hat{x}^i \in \mathcal{P}^I\}$$

- These functions can be used for local sensitivity analysis, just as one would do in linear programming.
  - For changes in the right-hand side, the lower bound remains valid.
  - For changes in the objective function, the upper bound remains valid.
  - It is possible to use a given partition to compute new bounds, but this requires additional computation.

Some Additional Details

- The method presented only applies to branch and bound.
  - Cut generation complicates matters because cuts may not be valid after changes to rim vectors.
  - Bounds tightened by reduced cost may also not be valid.
- We have to deal with infeasibility of subproblems.
- One can compute an "allowable range," as in LP, but it may be very small or empty.
- The bounds produced may not be useful.
- Question: What else can we do?
- Answers:
  - Continue solving from a "warm start" to improve bounds.
  - Perform a parametric analysis.

Warm Starting

- Question: What is "warm starting"?
- Question: Why are we interested in it?
- There are many examples of algorithms that solve a sequence of related ILPs.
  - Decomposition algorithms
  - Stochastic ILP
  - Parametric/Multicriteria ILP
  - Determining irreducible inconsistent subsystem
- For such problems, warm starting can potentially yield big improvements.
- Warm starting is also important for performing sensitivity analysis outside of the allowable range.

Warm Starting Information

- Question: What is "warm starting information"?
- Many optimization algorithms can be viewed as iterative procedures for satisfying a set of optimality conditions, often based on duality.
- These conditions provide a measure of "distance from optimality."
- Warm starting information can be seen as additional input data that allows an algorithm to quickly get "close to optimality."
- In integer linear programming, the *duality gap* is the usual measure.
- A good *starting partition* may reduce the initial duality gap.
- It is not at all obvious what makes a good starting partition.
- The most obvious choice is to use the optimal partition from a previous computation.

Parametric Analysis

- For global sensitivity analysis, we need to solve parametric programs.
- Along with Saltzman and Wiecek, we have developed an algorithm for determining all Pareto outcomes for a bicriteria MILP.
- The algorithm consists of solving a sequence of related ILPs and is asymptotically optimal.
- Such an algorithm can be used to perform global sensitivity analysis by constructing a "slice" of the value function.
- Warm starting can be used to improve efficiency.

Bicriteria MILPs

- The general form of a bicriteria (pure) ILP is

$$
\begin{aligned}
\text{vmax}\ & [cx, dx], \\
\text{s.t.}\quad & Ax \leq b, \\
& x \in \mathbb{Z}^n.
\end{aligned}
$$

- Solutions don't have single objective function values, but pairs of values called *outcomes*.
- A feasible $\hat{x}$ is called *efficient* if there is no feasible $\bar{x}$ such that $c\bar{x} \geq c\hat{x}$ and $d\bar{x} \geq d\hat{x}$, with at least one inequality strict.
- The outcome corresponding to an efficient solution is called *Pareto*.
- The goal of a bicriteria ILP is to enumerate Pareto outcomes.

A Really Brief Overview of SYMPHONY

- SYMPHONY is an open-source software package for solving and analyzing mixed-integer linear programs (MILPs).
- SYMPHONY can be used in three distinct modes.
  - Black box solver: Solve generic MILPs (command line or shell).
  - Callable library: Call SYMPHONY from a C/C++ code.
  - Framework: Develop a customized black box solver or callable library.
- Fully integrated with the Computational Infrastructure for Operations Research (COIN-OR) libraries.
- The interface and features of SYMPHONY 5.1 give it the look and feel an LP solver.
- SYMPHONY Solvers
  - Generic MILP
  - Multicriteria MILP
  - Traveling Salesman Problem
  - Vehicle Routing Problem
  - Mixed Postman Problem
  - Set Partitioning Problem
  - Matching Problem
  - Network Routing

Basic Sensitivity Analysis

- SYMPHONY will calculate bounds after changing the objective or right-hand side vectors.

```
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;
   lb = si.getLbForNewRhs(2, ind, val);
}
```

Warm Starts for MILP

- To allow resolving from a warm start, we have defined a SYMPHONY warm start class, which is derived from `CoinWarmStart`.
- The class stores a snapshot of the search tree, with node descriptions including:
  - lists of active cuts and variables,
  - branching information,
  - warm start information, and
  - current status (candidate, fathomed, etc.).
- The tree is stored in a compact form by storing the node descriptions as differences from the parent.
- Other auxiliary information is also stored, such as the current incumbent.
- A warm start can be saved at any time and then reloaded later.
- The warm starts can also be written to and read from disk.

Warm Starting Procedure

- After modifying parameters
  - If only parameters have been modified, then the candidate list is recreated and the algorithm proceeds as if left off.
  - This allows parameters to be tuned as the algorithm progresses if desired.
- After modifying problem data
  - Currently, we only allow modification of rim vectors.
  - After modification, all leaf nodes must be added to the candidate list.
  - After constructing the candidate list, we can continue the algorithm as before.
- There are many opportunities for improving the basic scheme, especially when solving a known family of instances (Geoffrion and Nauss)

Warm Starting Example (Parameter Modification)

- The following example shows a simple use of warm starting to create a dynamic algorithm.

```
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.initialSolve();
   si.setSymParam(OsiSymFindFirstFeasible, false);
   si.setSymParam(OsiSymSearchStrategy, BEST_FIRST_SEARCH);
   si.resolve();
}
```

Warm Starting Example (Problem Modification)

- The following example shows how to warm start after problem modification.

```
int main(int argc, char **argv)
{
   OsiSymSolverInterface si;
   CoinWarmStart ws;
   si.parseCommandLine(argc, argv);
   si.loadProblem();
   si.setSymParam(OsiSymNodeLimit, 100);
   si.initialSolve();
   ws = si.getWarmStart();
   si.resolve();
   si.setObjCoeff(0, 1);
   si.setObjCoeff(200, 150);
   si.setWarmStart(ws);
   si.resolve();
}
```

Using Warm Starting: Generic Mixed-Intger Programming

- Applying the code from the previous slide to the MIPLIB 3 problem p0201, we obtain the results below.
- Note that the warm start doesn't reduce the number of nodes generated, but does reduce the solve time dramatically.

	CPU Time	Tree Nodes
Generate warm start	28	100
Solve orig problem (from warm start)	3	118
Solve mod problem (from scratch)	24	122
Solve mod problem (from warm start)	6	198

Using Warm Starting: Generic Mixed-Intger Programming

- Here, we show the effect of using warm starting to solve generic MILPs whose objective functions have been perturbed.
- The coefficients were perturbed by a random percentage between α and $-\alpha$ for $\alpha = 1, 10, 20$.

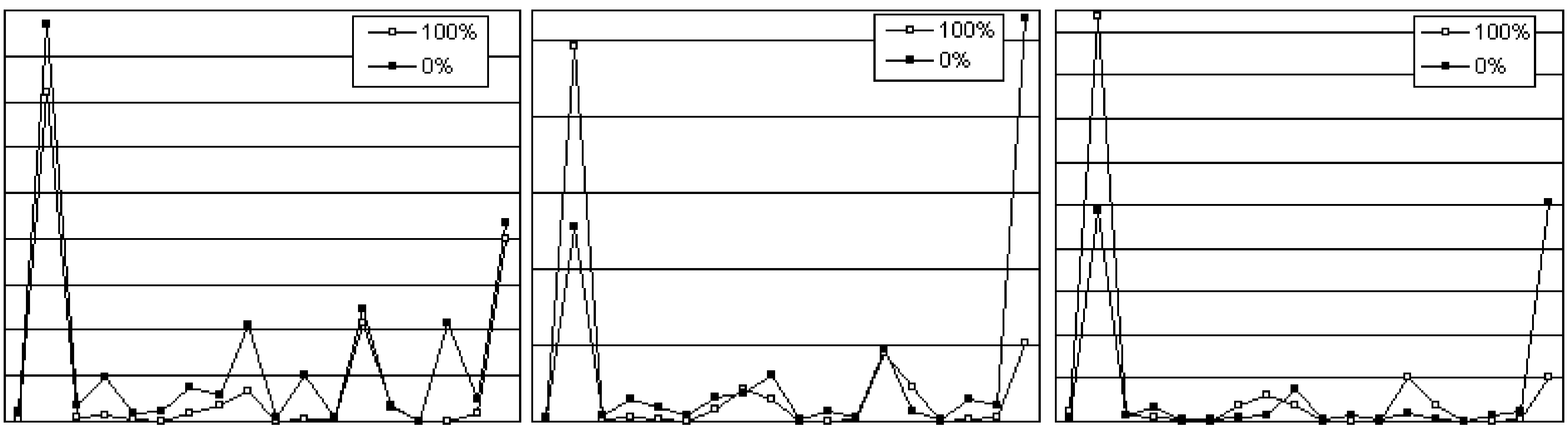


Table 1: Results of using warm starting to solve multi-criteria optimization problems.

Using Warm Starting: Network Routing

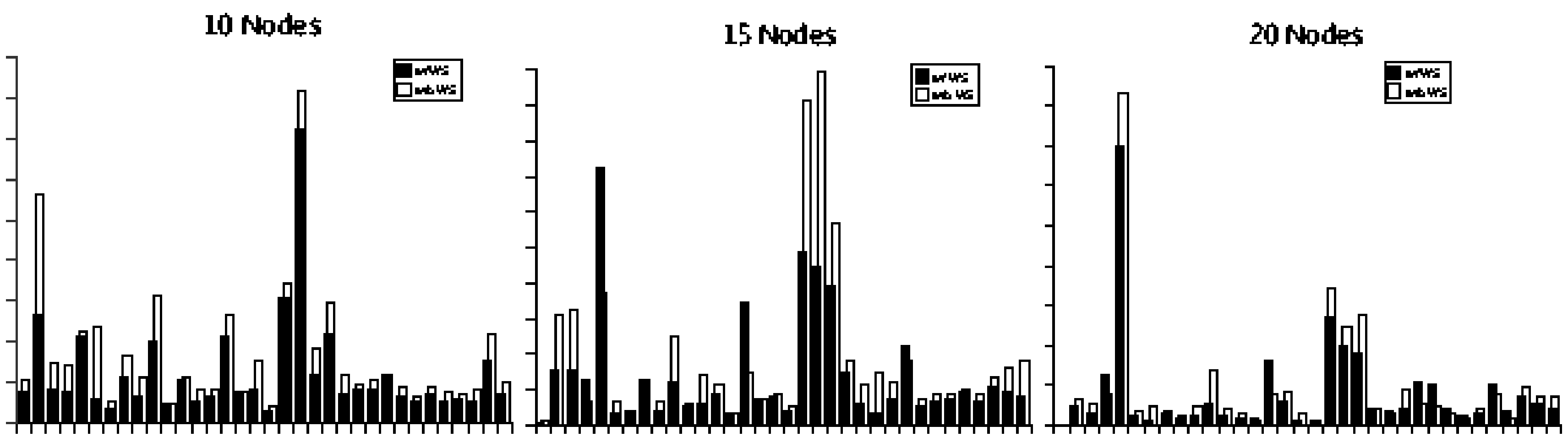


Table 2: Results of using warm starting to solve multi-criteria optimization problems.

Using Warm Starting: Stochastic Integer 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

Example: Bicriteria ILP

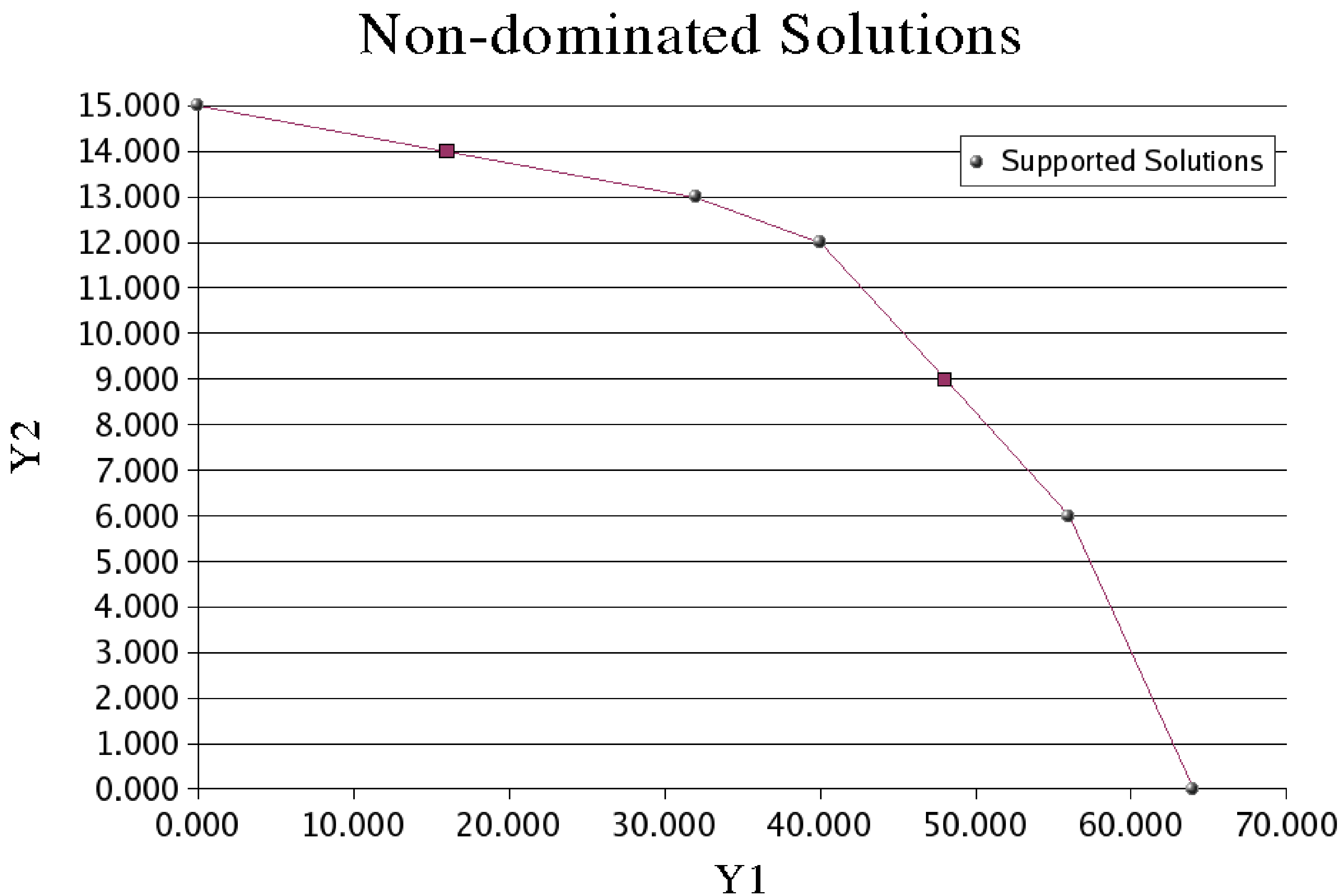
- Consider the following bicriteria ILP:

$$
\begin{aligned}
\text{vmax} \quad & [8x_1, x_2] \\
\text{s.t.} \quad & 7x_1 + x_2 \le 56 \\
& 28x_1 + 9x_2 \le 252 \\
& 3x_1 + 7x_2 \le 105 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

- The following code solves this model.

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

Example: Pareto Outcomes for Example



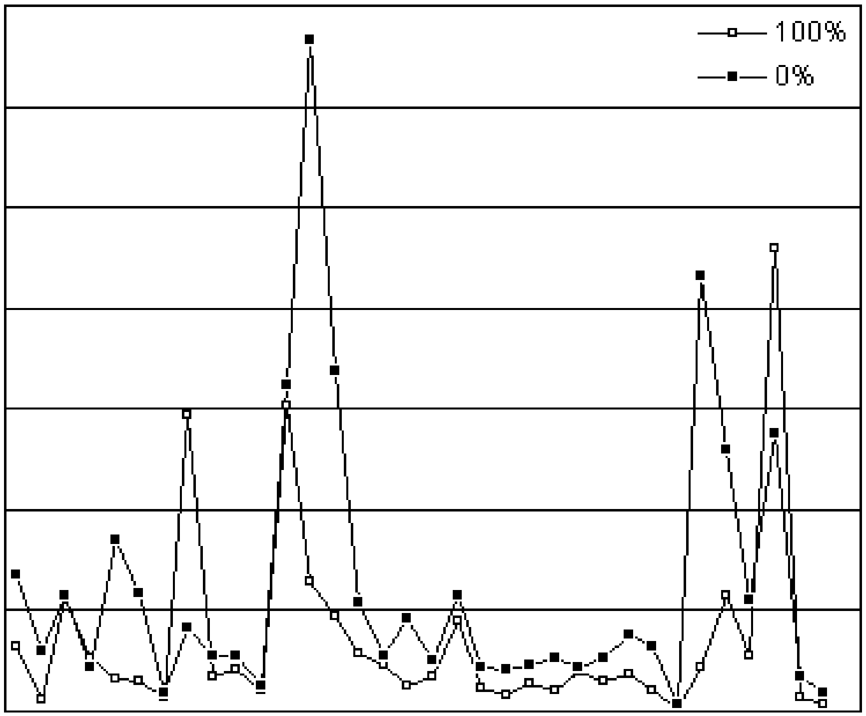
Example: Bicriteria Solver

- By examining the supported solutions and break points, we can easily determine $p(\theta)$, the optimal solution to the ILP with objective $8x_1 + \theta$.

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

Example: Graph of Price Function

Using Warm Starting: Bicriteria Optimization



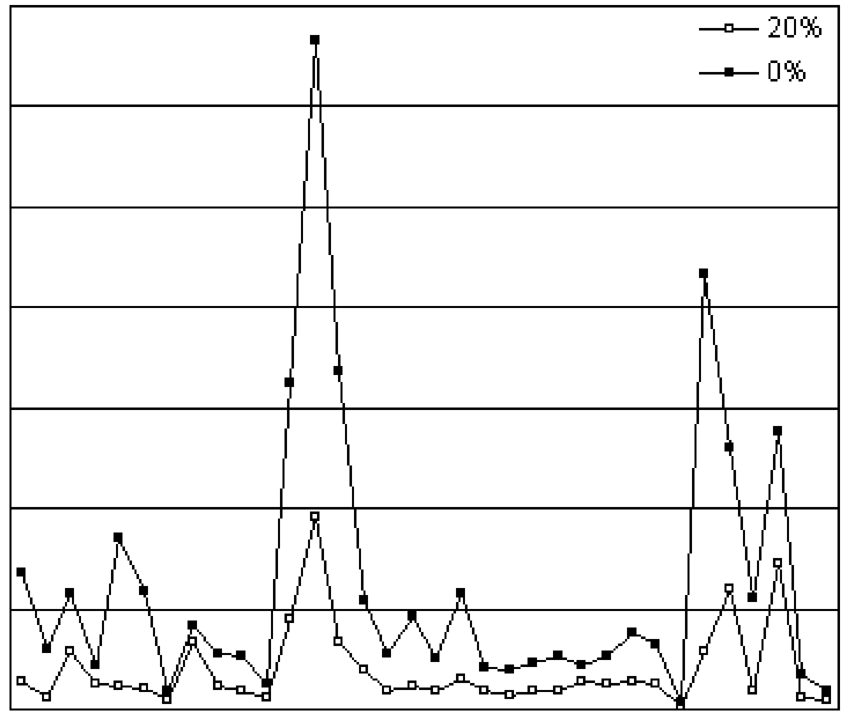


Table 3: Results of using warm starting to solve bicriteria optimization problems.

Conclusion

- We have briefly introduced the issues surrounding warm starting and sensitivity analysis for integer programming.
- An examination of early literature has yielded some ideas that can be useful in today's computational environment.
- We presented a new version of the SYMPHONY solver supporting warm starting and sensitivity analysis for MILPs.
- We have also demonstrated SYMPHONY's multicriteria optimization capabilities.
- This work has only scratched the surface of what can be done.
- In future work, we plan on refining SYMPHONY's warm start and sensitivity analysis capabilities.
- We will also provide more extensive computational results.