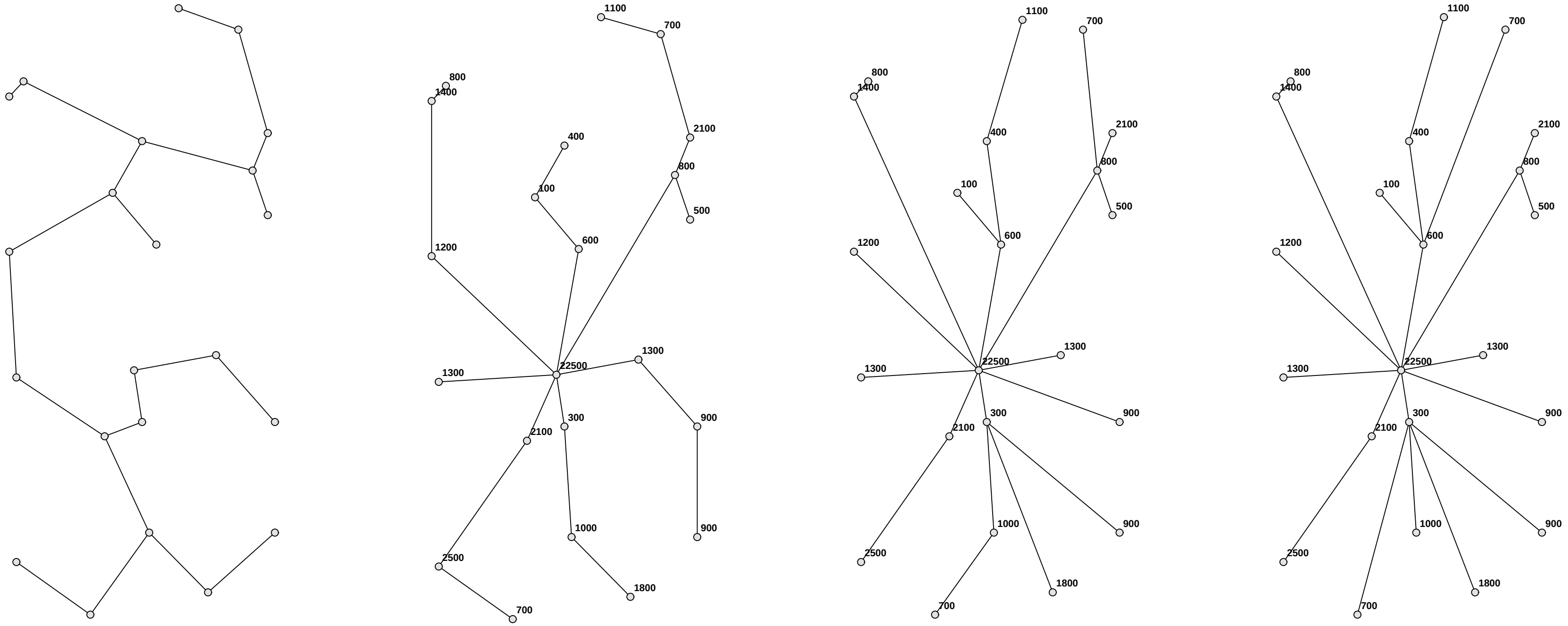
Biobjective Integer Programming

Ted Ralphs, Menal Guzelsoy, and Jeff Linderoth
Industrial and Systems Engineering, Lehigh University

Matthew Saltzman and Margaret Wiecek
Mathematical Sciences, Clemson University



The SAS Institute, Thursday, May 18, 2006

Outline of Talk

- Preliminaries
- The WCN Algorithm
- Variants
  - Interactive algorithm
  - Approximation algorithm
- Enhancements
  - Avoiding weakly dominated solutions
  - Improving efficiency
- Examples and Applications
  - Parametric Programming
  - Network Routing
- Computational Results

Biobjective Mixed-integer Programs

A *biobjective* or *bicriterion mixed-integer program* (BMIP) is an optimization problem of the form

$$\begin{array}{ll} \text{vmax} & f(x) \\ \text{subject to} & x \in X, \end{array}$$

where

- $f : \mathbb{R}^n \to \mathbb{R}^2$ is the *(bicriteria) objective function*, and
- $X \subset \mathbb{Z}^p \times \mathbb{R}^{n-p}$ is the *feasible region*, usually defined to be

$$\{x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \mid g_i(x) \leq 0, i = 1, \ldots, m\}$$

for functions $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$.

The vmax operator indicates that the goal is to generate the set of *efficient solutions* (defined next).

Some Definitions

- We define the set of *outcomes* to be $Y = f(X) \subset \mathbb{R}^2$.
- In outcome space, BMIP can be restated as

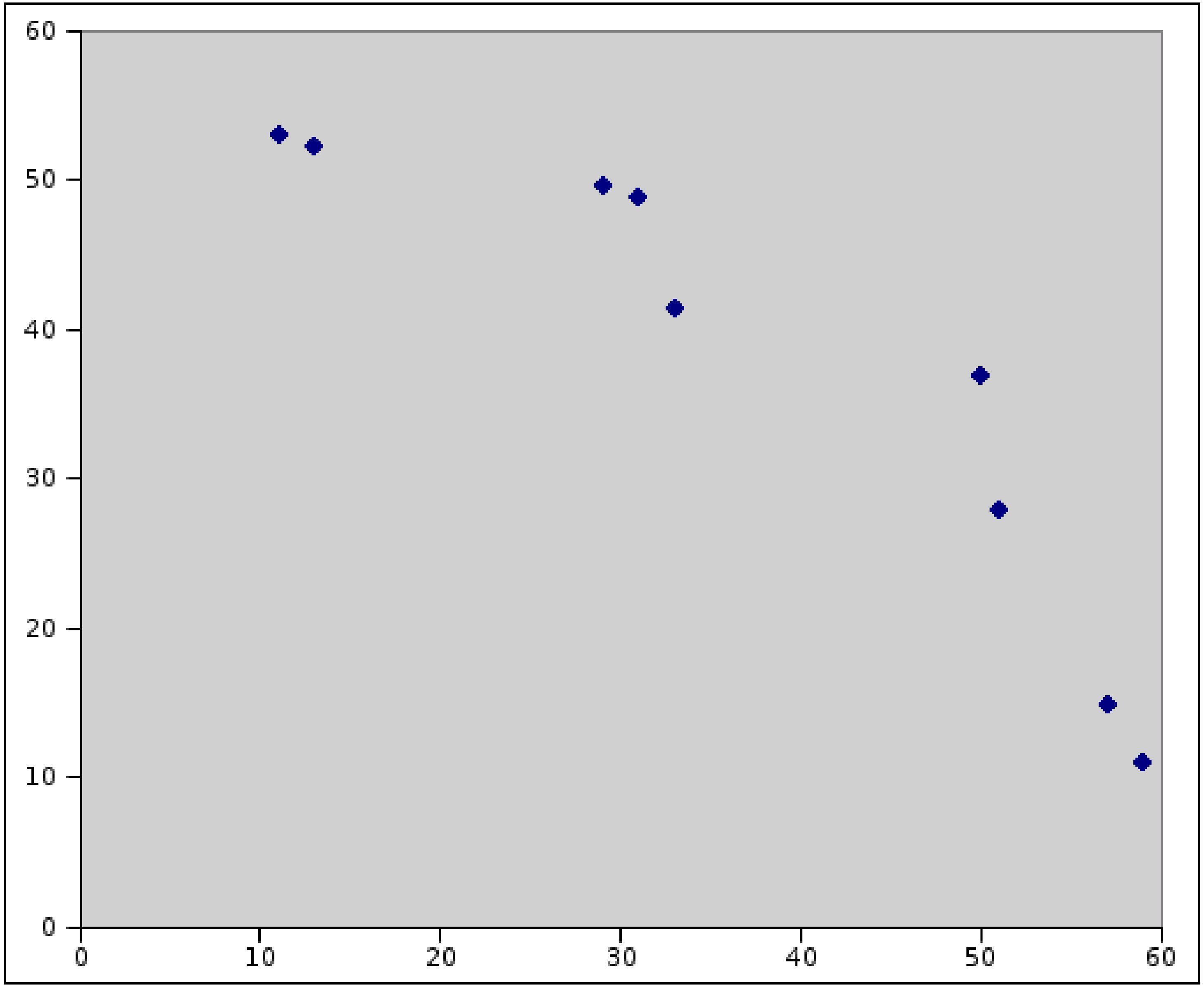
$$\begin{array}{ll} \text{vmax} & y \\ \text{subject to} & y \in f(X), \end{array}$$

- For convenience, we will work primarily in outcome space.
- $x^1 \in X$ *dominates* $x^2 \in X$ if $f_i(x_1) \geq f_i(x_2)$ for $i = 1, 2$ and at least one inequality is strict.
- If both inequalities are strict the dominance is *strong* (otherwise *weak*).
- Any $x \in X$ not dominated by another member of X is said to be *efficient*.
- If $x \in X$ is efficient, then $y = f(x)$ is a *Pareto outcome*.
- Our goal is to generate the set of all Pareto outcomes.

More Definitions

- We will denote the set of efficient solutions by X_E.
- The set of Pareto outcomes is then $Y_E = f(X_E)$.
- We assume that $|Y_E|$ is finite.
- If $x \in X_E$ strongly dominates all members of $X \setminus X_E$, then x is said to be *strongly efficient*.
- Likewise, if $x \in X_E$ is strongly efficient, then $y = f(x)$ is *strongly Pareto*.
- If all members of Y_E are strongly Pareto, then Y_E is said to be *uniformly dominant*.
- The assumption of uniform dominance simplifies computation substantially, but is not satisfied in most practical settings.

Illustrating Pareto Outcomes



Algorithms for Generating Pareto Outcomes

- A number of algorithms for generating Pareto outcomes have been proposed.
- These can be categorized in several ways:
  - By output: complete enumeration, partial enumeration, or heuristic enumeration of Y_E.
  - By user interaction: Interactive or non-interactive.
  - By methodology: branch and bound, dynamic programming, implicit enumeration, weighted sums, weighted norms, probing.
- We present an algorithm
  - that can either partially or completely enumerate the Pareto set,
  - has both interactive and non-interactive variants,
  - is based on a modified branch and bound algorithm.

Probing Algorithms

- We will focus on *probing algorithms* that *scalarize* the objective, i.e., replace it with a single criterion.
- Such algorithms reduce solution of a BMIP to a series of MIPs.
- The main factor in the running time is then the number of probes.
- The most obvious scalarization is the *weighted sum objective*.
- We replace the original objective with

$$\max_{y \in f(X)} \beta y_1 + (1 - \beta) y_2$$

to obtain a parameterized family of MIPs.

Supported Outcomes

- Optimal solutions to weighted sum MIPs are extreme points of $conv(Y_E)$.
- Such outcomes are called *supported outcomes*.
- The set of all supported outcomes can easily be generated by solving a sequence of MIPs.
- Every supported outcome is Pareto, but the converse is not true.
- This makes it difficult as a tool to generate all Pareto outcomes.
- Chalmet (1986) suggested restricting the subproblems so that each Pareto outcome is supported on some subregion.
- Using this technique, it is possibe to generate all Pareto outcomes.

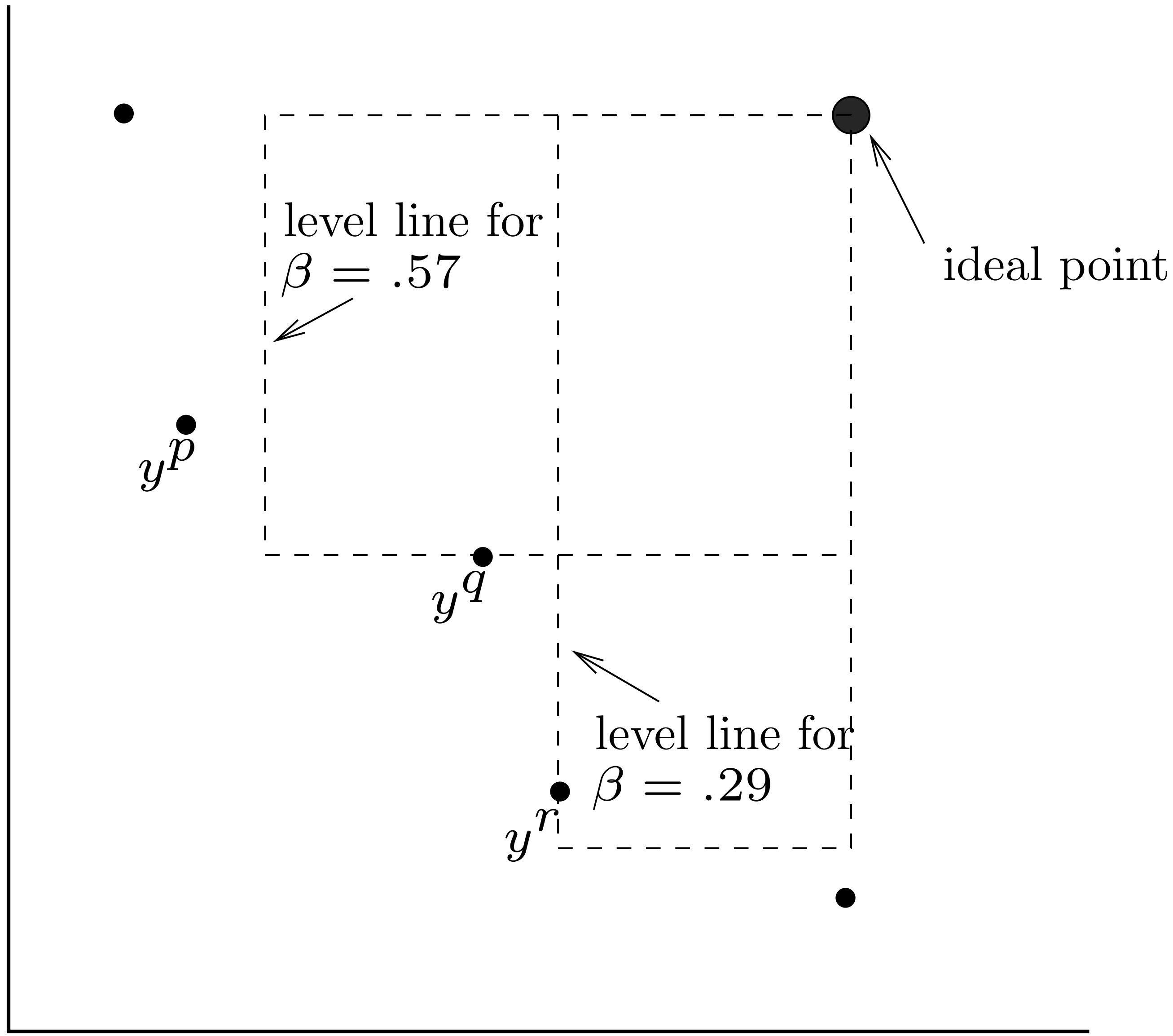
The Weighted Chebyshev Norm

- Another option is to replace the weighted sum objective with a *weighted Chebyshev norm* (WCN) objective.
- The *Chebyshev norm* (l_∞ norm) in $\mathbb{R}^2$ is defined by $\|y\|_\infty = \max\{|y_1|, |y_2|\}$.
- The *weighted Chebyshev norm* with weight $0 \leq \beta \leq 1$ is defined by $\|y\|_\infty = \max\{\beta|y_1|, (1-\beta)|y_2|\}$.
- The *ideal point* y^* is (y_1^*, y_2^*) where $y_i^* = \max_{x \in X}(f(x))_i$.
- Methods based on the WCN select outcomes with minimum WCN distance from the ideal point by solving

$$\min_{y \in f(X)} \{\|y^* - y\|_\infty^\beta\}. \tag{1}$$

- Bowman (1976) showed that every Pareto outcome is a solution to (1) for some $0 \leq \beta \leq 1$.
- The converse only holds if Y_E is uniformly dominant.

Illustrating the WCN



Ordering the Pareto Outcomes

- Eswaran (1989) suggested ordering the Pareto outcomes so that
  - $Y_E = \{y_p \mid 1 \leq p \leq N\}$, and
  - if $p < q$, then $y_1^p < y_1^q$ (and hence $y_2^p > y_2^q$).
- For any Pareto outcome y_p, if we define

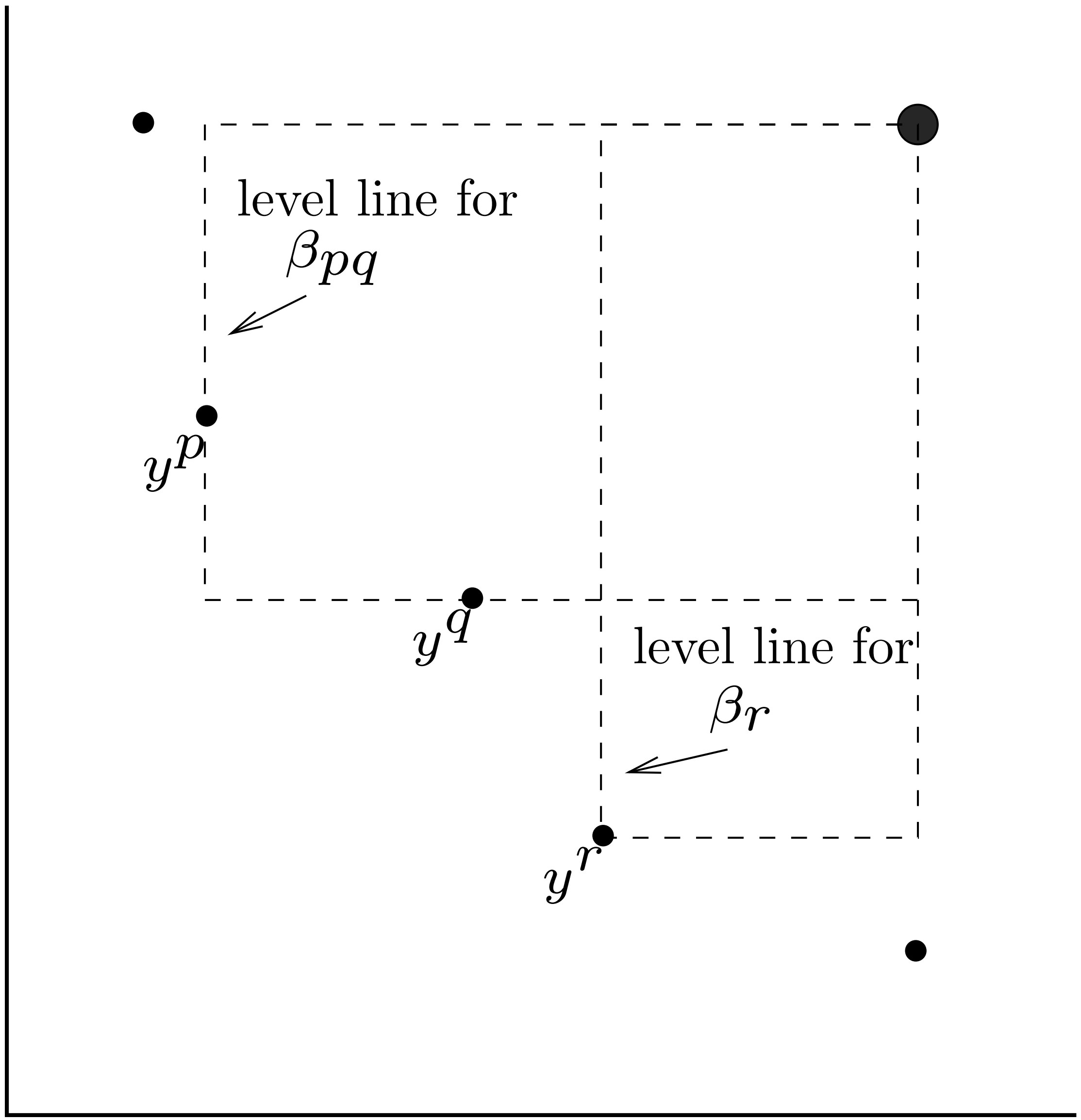
$$\beta_p = (y_2^* - y_2^p)/(y_1^* - y_1^p + y_2^* - y_2^p),$$

  then y^p is the unique optimal outcome for (1) with $\beta = \beta_p$.
- For any pair of Pareto outcomes y^p and y^q with $p < q$, if we define

$$\beta_{pq} = (y_2^* - y_2^q)/(y_1^* - y_1^p + y_2^* - y_2^q), \tag{2}$$

  then y^p and y^q are both optimal outcomes for (1) with $\beta = \beta_{pq}$.
- This provides us with a notion of *adjacency* and *breakpoints*.

Breakpoints Between Pareto Outcomes with the WCN



Algorithms Based on the WCN

- Solanki (1991) proposed an algorithm to generate an approximation to the Pareto set using the WCN.
  - The algorithm probes between pairs of known outcomes for new outcomes by restricting the domain ala Chalmet.
  - The search is controlled by an "error measure," which can be set to zero to get complete enumeration.
  - The number of probes is asymptotically optimal, but the algorithm does not produce breakpoints (directly).
- Eswaran (1989) proposed an algorithm based on binary search over the values of β.
  - In the worst case, the number of probes is

$$|Y_E|(1 - \lg(\xi(|Y_E| - 1))),$$

    where ξ is a chosen error parameter.
  - The algorithm produces only approximate breakpoint information.

The WCN Algorithm

Let $P(\beta)$ be the parameterized subproblem defined by (1) for a given weight β. The WCN algorithm is then:

Initialization Solve $P(1)$ and $P(0)$ to identify optimal outcomes y^1 and y^N, respectively, and the ideal point $y^* = (y_1^1, y_2^N)$. Set $I = \{(y^1, y^N)\}$.

Iteration While $I \neq \emptyset$ do:

1. Remove any (y^p, y^q) from I.
2. Compute β_{pq} as in (2) and solve $P(\beta_{pq})$. If the outcome is y^p or y^q, then y^p and y^q are adjacent in the list $(y^1, y^2, \ldots, y^N)$.
3. Otherwise, a new outcome y^r is generated. Add (y^p, y^r) and (y^r, y^q) to I.

This reduces solution of the original BMIP to solution of a sequence of $2N - 1$ MIPs, but still requires the assumption of uniform dominance.

Solving $P(\beta)$

- Problem (1) is equivalent to

$$\begin{array}{llll} \text{minimize} & z & & \\ \text{subject to} & z & \geq & \beta(y_1^* - y_1), \\ & z & \geq & (1-\beta)(y_2^* - y_2), \text{ and} \\ & y & \in & f(X). \end{array} \tag{3}$$

- This is a MIP, which can be solved by standard methods.
- This reformulation can still produce weakly dominated outcomes.

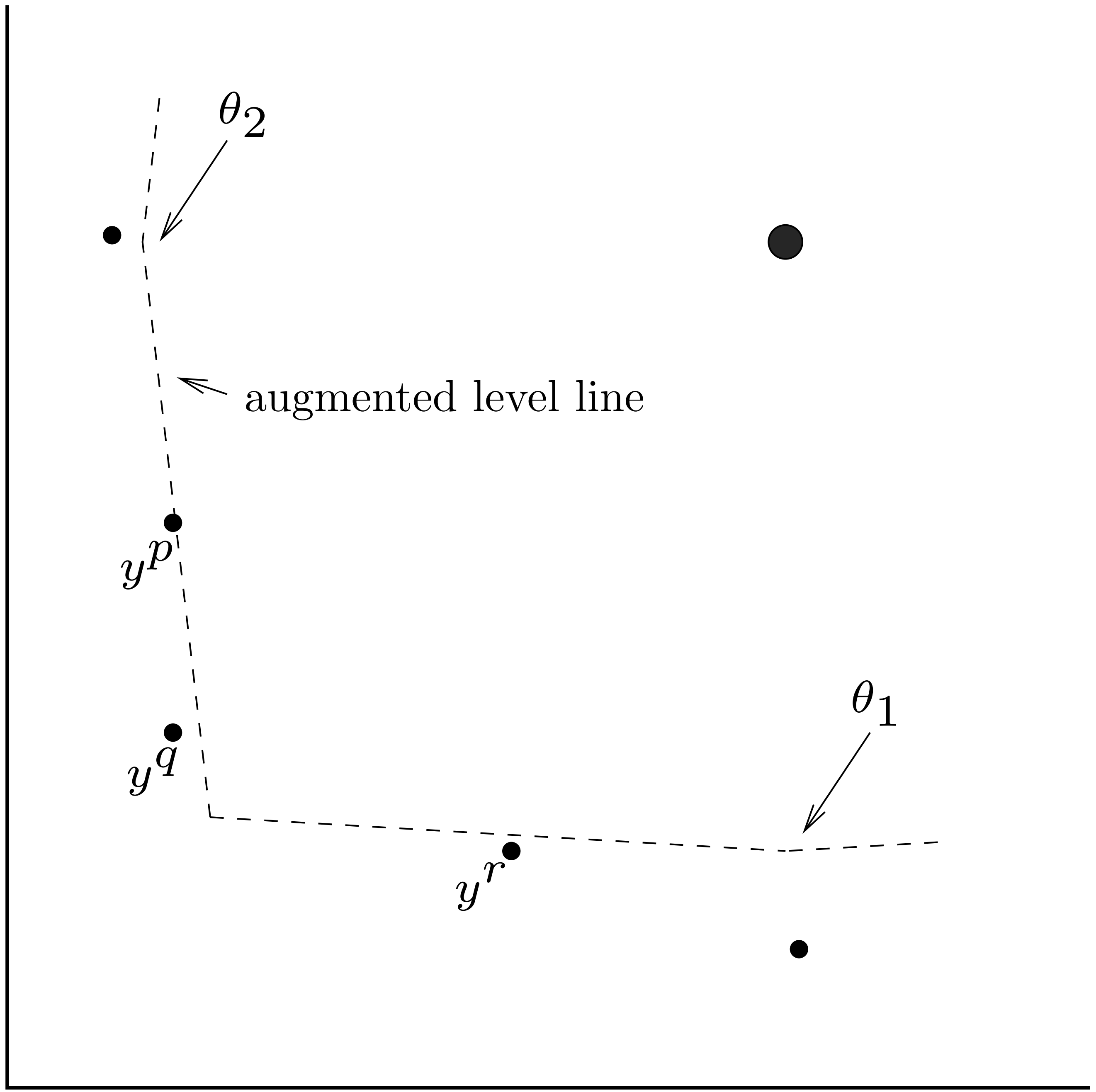
Relaxing the Uniform Dominance Requirement

- Dealing with weakly dominated outcomes is the most challenging aspect of these methods.
- We need a method of preventing $P(\beta)$ from producing weakly dominated outcomes.
- Weakly dominated outcomes are the same WCN distance from the ideal point as the outcomes they are dominated by.
- However, they are farther from the ideal point as measured by the l_p norm for $p < \infty$.
- One solution is to replace the WCN with the *augmented Chebyshev norm* (ACN), defined by

$$\|(y_1, y_2)\|_{\infty}^{\beta,\rho} = \max\{\beta|y_1|, (1-\beta)|y_2|\} + \rho(|y_1| + |y_2|),$$

where ρ is a small positive number.

Illustrating the ACN



Solving $P(\beta)$ with the ACN

- The problem of determining the outcome closest to the ideal point under this metric is

$$\begin{array}{llll} \min & z & + & \rho(|y_1^* - y_1| + |y_2^* - y_2|) \\ \text{subject to} & z & \geq & \beta(y_1^* - y_1) \\ & z & \geq & (1-\beta)(y_2^* - y_2) \\ & y & \in & f(X). \end{array} \tag{4}$$

- Because $y_k^* - y_k \geq 0$ for all $y \in f(X)$, the objective function can be rewritten as

$$\min z - \rho(y_1 + y_2).$$

- For fixed $\rho > 0$ small enough:
  - all optimal outcomes for problem (4) are Pareto (in particular, they are not weakly dominated), and
  - for a given Pareto outcome y for problem (4), there exists $0 \leq \hat{\beta} \leq 1$ such that y is the unique outcome to problem (4) with $\beta = \hat{\beta}$.
- In practice, choosing a proper value for ρ can be problematic.

Combinatorial Methods for Eliminating Weakly Dominated Solutions

- In the case of *biobjective linear integer programs* (BLIPs), we can employ combinatorial methods.
- Such a strategy involves implicitly enumerating alternative optimal solutions to $P(\beta)$.
- Weakly dominated outcomes are eliminated with cutting planes during the branch and bound procedure.
- Instead of pruning subproblems that yield feasible outcomes, we continue to search for alternative optima that dominate the current incumbant.
- To do so, we determine which of the two constraints

$$\begin{aligned} z &\geq \beta(y_1^* - y_1) \\ z &\geq (1-\beta)(y_2^* - y_2) \end{aligned}$$

from problem (1) is binding at $\hat{y}$.

Combinatorial Methods for Eliminating Weakly Dominated Solutions (cont'd)

- Let ϵ_1 and ϵ_2 be such that if y_r is a new outcome between y^p and y^q, then $y_i^r \geq \min\{y_i^p, y_i^q\} + \epsilon_i$, for $i = 1, 2$.
- If only the first constraint is binding, then the cut

$$y_1 \geq \hat{y}_1 + \epsilon_1$$

  is valid for any outcome that dominates $\hat{y}$.
- If only the second constraint is binding, then the cut

$$y_2 \geq \hat{y}_2 + \epsilon_2$$

  is valid for any outcome that dominates $\hat{y}$.
- If both constraints are binding, either cut can be imposed.

Hybrid Methods

- In practice, the ACN method is fast, but choosing the proper value of ρ is problematic.
- Combinatorial methods are less susceptible to numerical difficulties, but are slower.
- Combining the two methods improves running times and reduces dependence on the magnitude of ρ.

Other Enhancements to the Algorithm

- In Step 2, any new outcome y^r will have $y_1^r > y_1^p$ and $y_2^r > y_2^q$.
- If no such outcome exists, then the subproblem solver must still re-prove the optimality of y^p or y^q.
- Then it must be the case that

$$\|y^* - y^r\|_\infty^{\beta_{pq}} + \min\{\beta_{pq}\epsilon_1, (1-\beta_{pq})\epsilon_2\} \le \|y^* - y^p\|_\infty^{\beta_{pq}} = \|y^* - y^q\|_\infty^{\beta_{pq}}$$

- Hence, we can impose an a priori upper bound of

$$\|y^* - y^p\|_\infty^{\beta_{pq}} - \min\{\beta_{pq}\epsilon_1, (1-\beta_{pq})\epsilon_2\}$$

  when solving the subproblem $P(\beta_{pq})$.
- With this upper bound, each subproblem will either be infeasible or produce a new outcome.

Using Warm Starting

- We have been developing methodology for *warm starting* branch and bound computations.
- Because the WCN algorithm involves solving a sequence of slightly modified MILPs, warm starting can be used.
- Three approaches
  - Warm start from the result of the previous iteration.
  - Solve a "base" problem first and warm each subsequent problem from there.
  - Warm start from the "closest" previously solved subproblem.
- In addition, we can optionally save the global cut pool from iteration to iteration.

Approximating the Pareto Set

- If the number of Pareto outcomes is large, it may not be desirable to generate the entire set.
- If only part of the set is generated, it is important that the subset be *well-distributed* among the entire set.
- Any probing algorithm can generate an approximation to the Pareto set by terminating early.
  - In such case, the key is to avoid failed probes whenever possible.
  - The order in which the intervals are explored affects both the distribution of solutions and the number of failed probes.
  - Empirically, FIFO selection schemes tend to distribute the points well and also minimize the number of failed probes.
- Another approach is to generate the set of supported solutions.
- This can be an extremely bad approximation in some cases.

Interactive Algorithms

- Interactive algorithms offer another method of avoiding enumeration of the entire set.
- In an interactive algorithm, the user guides the solution process by providing real-time feedback.
- This feedback provides information about the user's unknown utility function.
- A simple feedback mechanism for the WCN algorithm is to allow the user to select the next interval to be explored.
- In this way, the user is able to zero in on the portion of the tradeoff curve that is most attractive.
- There are a number of mechanisms for providing estimated tradeoff information to the user as the algorithm progresses.

Implementation: A 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.
- Makes extensive use of the Computational Infrastructure for Operations Research (COIN-OR) libraries (`www.coin-or.org`).
- Complete documentation, code samples, data sets, and application plug-ins are available (`www.BranchAndCut.org`).
- Advanced features
  - Warm starting
  - Bicriteria solve
  - Sensitivity analysis
  - Parallel execution mode

Example: Bicriteria ILP

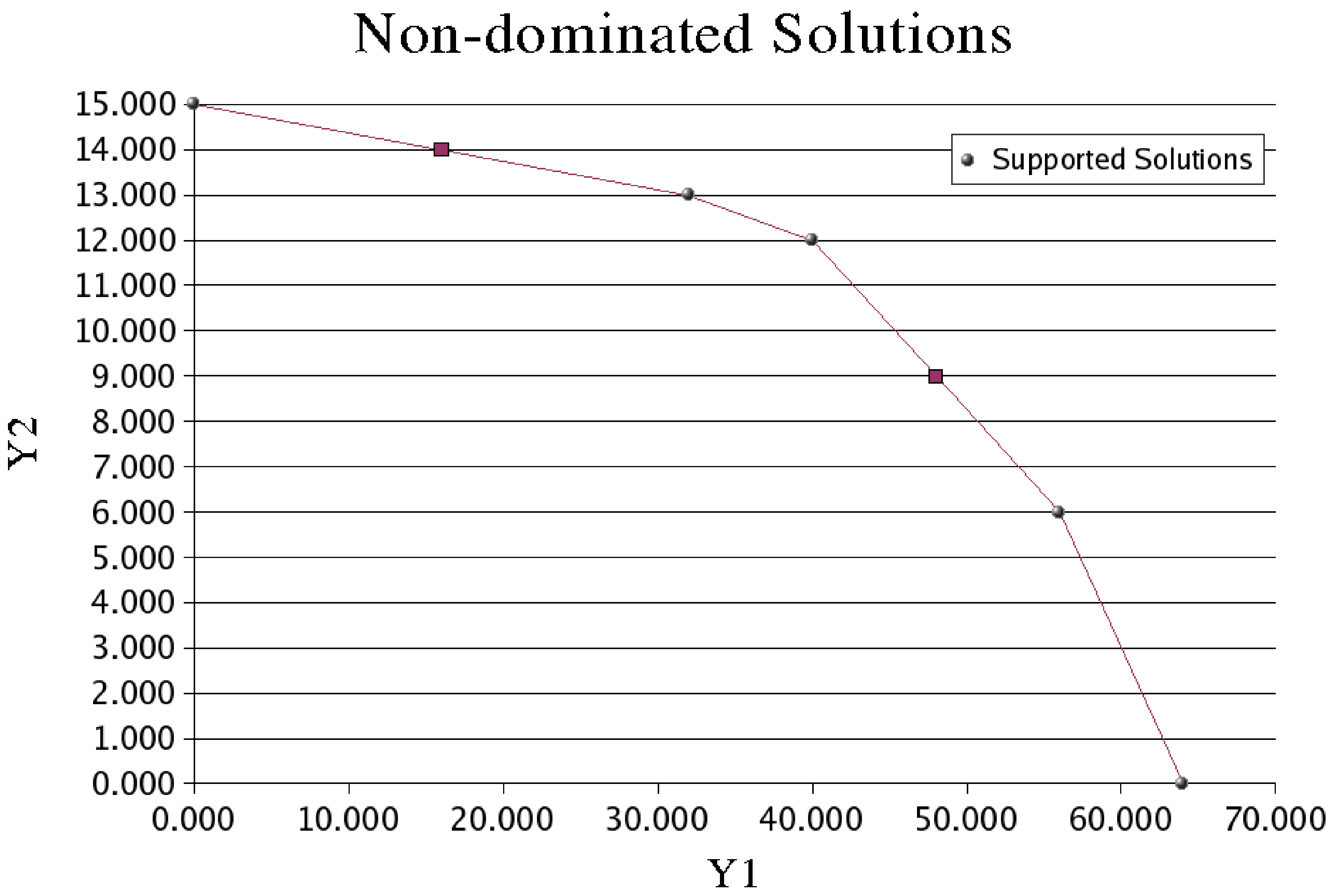
- Consider the following bicriteria ILP:

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

- The following code solves this model using SYMPHONY.

```
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

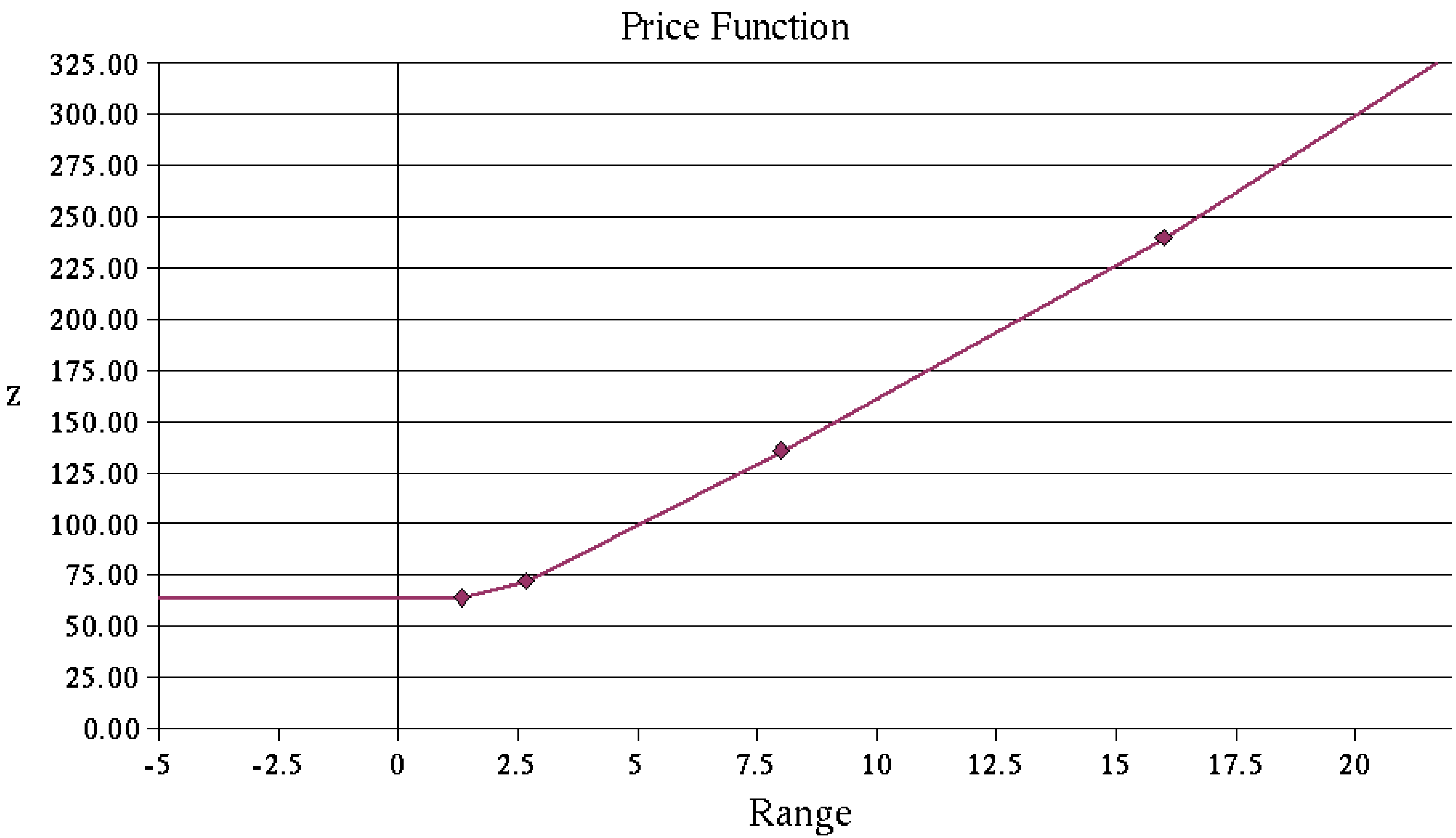


Example: Sensitivity Analysis

- By examining the supported solutions and break points, we can easily determine $p(\theta)$, the optimal solution to the ILP with objective $8x_1 + \theta x_2$.

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

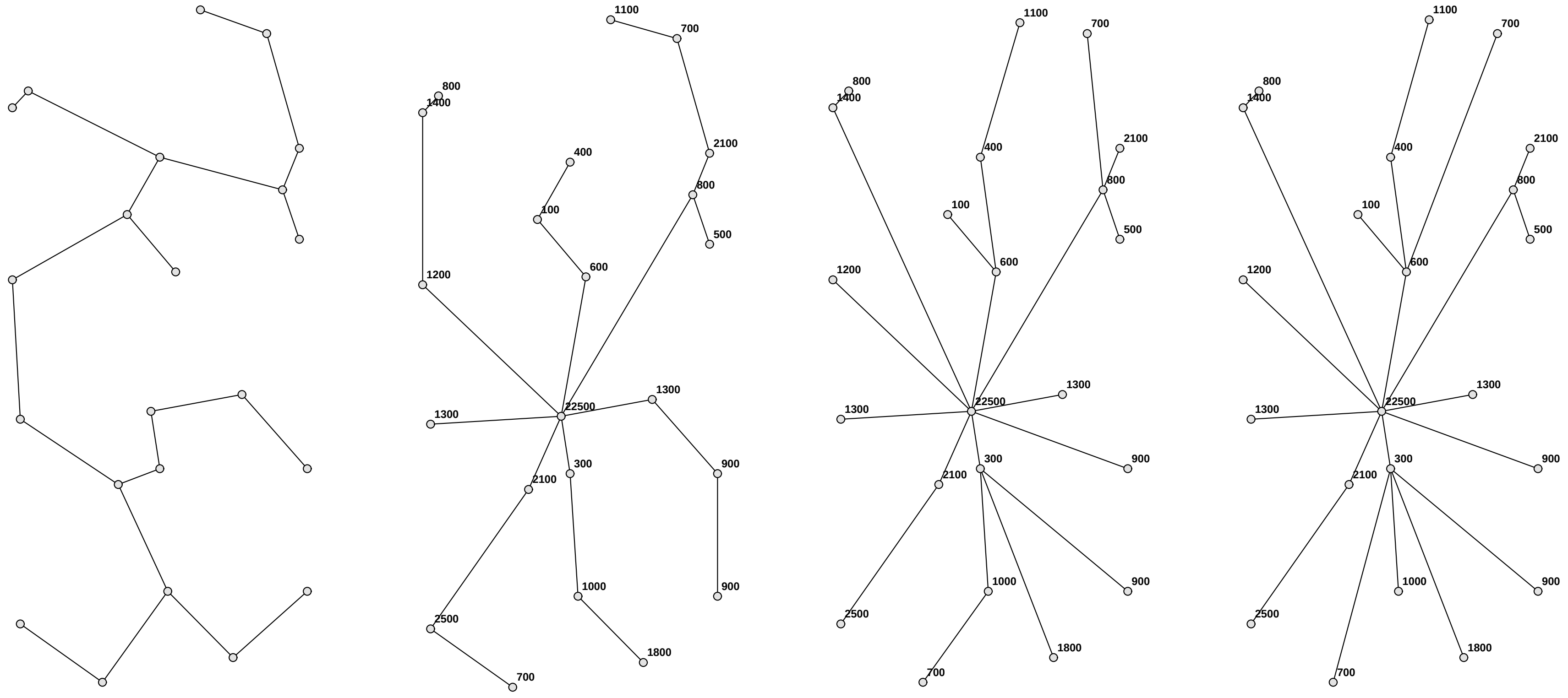
Example: Price Function



Application: Capacitated Network Routing Problems

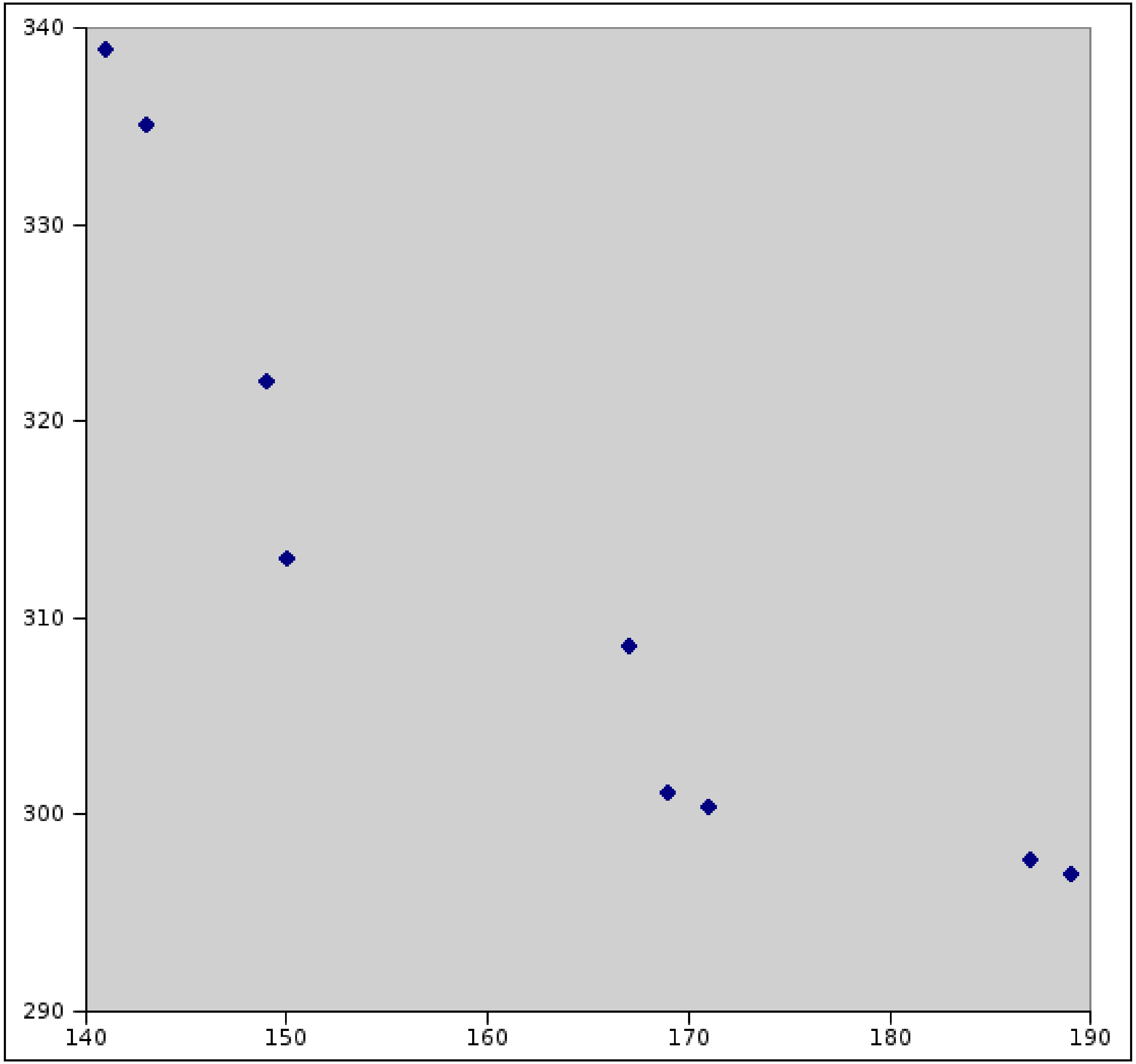
- Using SYMPHONY, we developed a custom solver for a class of capacitated network routing problems (CNRPs).
- A single commodity is supplied to a set of customers from a single supply point.
- We must design the network and route the demand, obeying capacity and other side constraints.
- We wish to consider both
  - the cost of construction (the sum of lengths of all links), and
  - the latency of the resulting network (the sum of length multiplied by demand carried for all links).
- These are competing objectives, so we can analyze the tradeoff by using the SYMPHONY multicriteria solver.

Application: Efficient Solutions for a Small CNRP

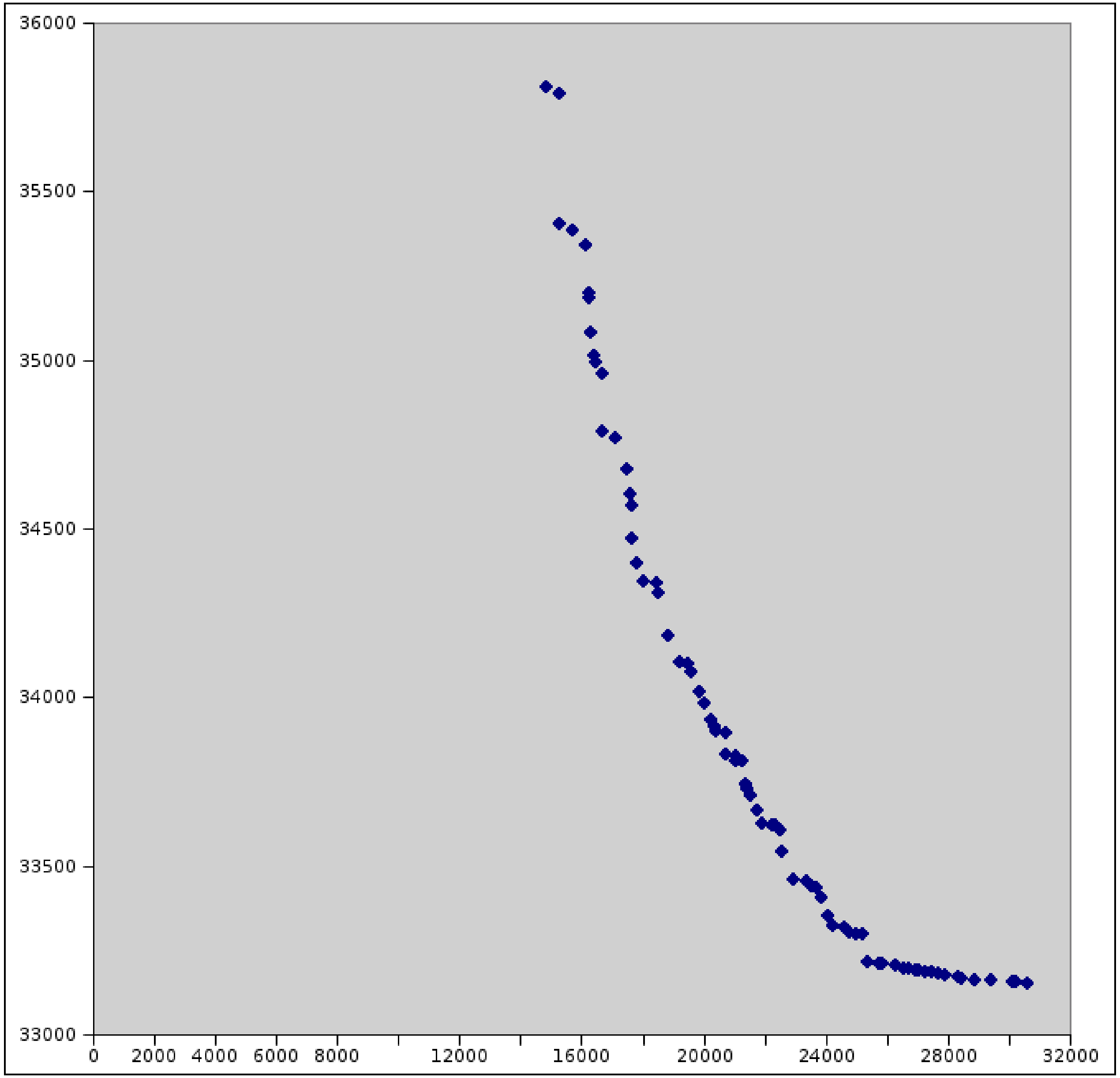


(a) (b) (c) (d)

Application: Pareto Outcomes for a Small CNRP



Application: Pareto Outcomes for a Larger CNRP



Computational Results: Comparing WCN with Bisection Search

Knapsack

	Iterations				Outcomes Found				Max Missed		
	WCN	Δ from WCN			WCN	Δ from WCN					
Size	0	10^{-1}	10^{-2}	10^{-3}	0	10^{-1}	10^{-2}	10^{-3}	10^{-1}	10^{-2}	10^{-3}
10	278	12	300	679	149	-17	0	0	6	0	0
20	364	-1	390	896	192	-22	-2	0	6	1	0
30	324	-43	246	712	167	-25	0	0	4	0	0
40	490	-108	235	898	250	-55	-11	0	5	2	0
50	686	-138	235	1123	348	-69	-9	-1	11	1	1
Totals	2142	-278	1406	4308	1106	-188	-22	-1	11	2	1

CNRP

	Iterations				Outcomes Found				Max Missed		
	WCN	Δ from WCN			WCN	Δ from WCN					
Name	0	10^{-1}	10^{-2}	10^{-3}	0	10^{-1}	10^{-2}	10^{-3}	10^{-1}	10^{-2}	10^{-3}
att48	147	-35	-9	104	74	-18	-15	-4	3	3	1
Totals	2381	-264	724	3794	1207	-135	-13	0	5	1	0

Computational Results: Comparing WCN with ACN

Knapsack

	Iterations				Outcomes Found				Max Missed		
	WCN	Δ from WCN			WCN	Δ from WCN					
Size	0	10^{-2}	10^{-3}	10^{-4}	0	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}
10	278	-4	0	0	149	-2	0	0	1	0	0
20	364	-6	0	0	192	-3	0	0	1	0	0
30	324	-6	0	0	167	-3	0	0	1	0	0
40	490	-24	0	0	250	-12	0	0	1	0	0
50	686	-28	-4	0	348	-24	-2	0	3	2	0
Totals	2142	-70	0	0	1106	-34	-2	0	3	2	0

CNRP

	Iterations				Outcomes Found				Max Missed		
	WCN	Δ from WCN			WCN	Δ from WCN					
Name	0	10^{-2}	10^{-3}	10^{-4}	0	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}
att48	147	-140	-106	-62	74	-70	-53	-31	44	17	8
Totals	2381	-2056	-1012	-34	1207	-1028	-506	-17	18	5	1

Computational Results: Comparing WCN with Hybrid ACN

Knapsack

	Iterations				Outcomes Found				Max Missed		
	WCN	Δ from WCN			WCN	Δ from WCN					
Size	0	10^{-2}	10^{-3}	10^{-4}	0	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}
10	278	-4	0	0	149	-2	0	0	1	0	0
20	364	-6	0	0	192	-3	0	0	1	0	0
30	324	-6	0	0	167	-3	0	0	1	0	0
40	490	-24	0	0	250	-12	0	0	1	0	0
50	686	-28	-4	0	348	-14	-2	0	3	2	0
Totals	2142	-68	-4	0	1106	-34	-2	0	3	2	0

CNRP

	Iterations				Outcomes Found				Max Missed		
	WCN	Δ from WCN			WCN	Δ from WCN					
Name	0	10^{-3}	10^{-4}	10^{-5}	0	10^{-3}	10^{-4}	10^{-5}	10^{-3}	10^{-4}	10^{-5}
att48	147	-106	-62	-6	74	-53	-31	-3	17	8	2
Totals	2381	-1012	-44	-2	1207	-612	-22	-1	5	1	1

Computational Results: Comparing WCN with ACN and Hybrid ACN (CPU Time)

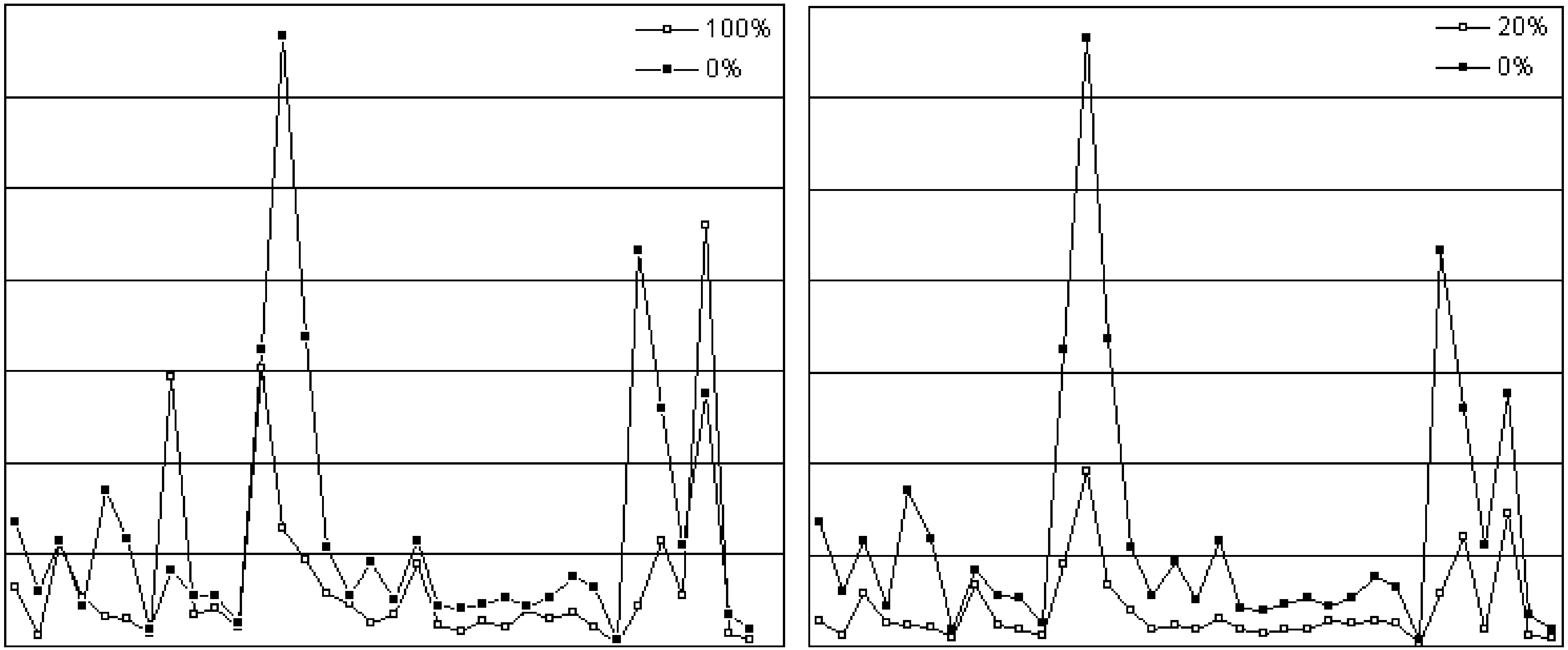
Knapsack

	CPU Time (ACN)				CPU Time (Hybrid)			
	WCN	Δ from WCN			WCN	Δ from WCN		
Size	0	10^{-2}	10^{-3}	10^{-4}	0	10^{-2}	10^{-3}	10^{-4}
10	13.18	0.06	-0.23	-0.10	13.18	0.34	0.12	0.16
20	17.46	-1.33	-0.41	-0.21	17.46	-1.17	0.03	-0.16
30	24.93	-1.28	-0.43	-0.43	24.93	-1.02	-0.11	0.10
40	65.88	-5.69	-1.70	-0.66	24.93	-1.02	-0.11	0.10
50	139.42	-27.18	-3.78	-1.35	65.88	-4.89	-1.09	-0.30
60	260.87	-35.42	-6.55	-2.75	139.42	-13.04	-3.37	-1.17
Totals	260.87	-35.42	-6.55	-2.75	260.87	-19.78	-4.42	-1.37

CNRP

	CPU Time (ACN)				CPU Time (Hybrid)			
	WCN	Δ from WCN			WCN	Δ from WCN		
Name	0	10^{-2}	10^{-3}	10^{-4}	0	10^{-2}	10^{-3}	10^{-4}
att48	83.67	-80.14	-59.83	-28.48	83.67	-59.34	-30.19	-1.12
Totals	8122.36	-7728.51	-5244.54	-1451.37	8122.36	-5481.53	-1531.35	-589.90

Computational Results: Using Warm Starting to Solve CNRP Instances



These are results using SYMPHONY to solve CNRP instances with two different warm starting strategies.

The Next Frontier: Using the Computational Grid

- Enumerating the entire Pareto set may be difficult for hard combinatorial problems.
- This algorithm is, however, naturally parallelizable.
- The order in which the subproblems are solved is not crucial, so there is little need for synchronization.
- Solution of the subproblems themselves can also be parallelized.
- Speedup will depend on the number of subproblems in the queue at any given time.
- Solving the subproblems in different orders may result in different parallel performance.
- We are currently using MW Blackbox to develop a grid-enabled implementation of this algorithm.
- Only the list of breakpoints and solutions generated so far are needed to restart the algorithm.

Conclusion

- Generating the complete set of Pareto outcomes is a challenging computational problem.
- We presented a new algorithm for solving bicriteria mixed-integer programs.
- The algorithm is
  - asymptotically optimal,
  - generates exact breakpoints,
  - has good numerical properties, and
  - can exploits modern solution techniques.
- We have shown how this algorithm is implemented in the SYMPHONY MILP solver framework.
- Future work
  - Improvements to warm starting procedures
  - Parallelization
  - More than two objective

More Information

- SYMPHONY
  - Prepackaged releases can be obtained from `www.BranchAndCut.org`.
  - Up-to-date source is available from `www.coin-or.org`.
  - Available Solvers
    - Generic MILP
    - Traveling Salesman Problem
    - Vehicle Routing Problem
    - Mixed Postman Problem
    - Bicriteria Knapsack Solver
    - Set Partitioning Problem
    - Matching Problem
    - Network Routing
- For references and further details, see *An Improved Algorithm for Biobjective Integer Programming*, to appear in Annals of OR, available from

  `www.lehigh.edu/~tkr2`

- Overviews of multiobjective integer programming
  - Climaco (1997)
  - Ehrgott and Gandibleux (2002)
  - Ehrgott and Wiecek (2005)