An Improved Algorithm for Biobjective Integer Programs

Ted K. Ralphs* Matthew J. Saltzman† Margaret M. Wiecek‡

Revised January 15, 2005

Abstract

A parametric algorithm for identifying the Pareto set of a biobjective integer program is proposed. The algorithm is based on the weighted Chebyshev (Tchebycheff) scalarization, and its running time is asymptotically optimal. A number of extensions are described, including: a technique for handling weakly dominated outcomes, a Pareto set approximation scheme, and an interactive version that provides access to all Pareto outcomes. Extensive computational tests on instances of the biobjective knapsack problem and a capacitated network routing problem are presented.

1 Introduction

Biobjective integer programming (BIP) is an extension of the classical single-objective integer programming problem motivated by a variety of real world applications in which it is necessary to consider two or more criteria when selecting a course of action. Examples may be found in business and management, engineering, and many other areas where decision-making requires consideration of competing objectives. Examples of the use of BIPs can be found in capital budgeting [4], location analysis [14], and engineering design [25].

1.1 Terminology and Definitions

A general biobjective or bicriterion integer program (BIP) is formulated as

$$\begin{array}{ll} \text{vmax} & f(x) = [f_1(x), f_2(x)] \\ \text{subject to} & x \in X \subset \mathbb{Z}^n, \end{array} \tag{1}$$

where $f_i(x)$, $i = 1, 2$ are real-valued criterion functions. The set X is called the set of *feasible solutions* and the space containing X is the *solution space*. Generally, X is the subset of $\mathbb{Z}^n$ contained in a region defined by a combination of equality and inequality constraints, as well as explicit bounds on individual variables. We define the set of *outcomes* as $Y = f(X)$, and call the space containing Y the *objective space* or *outcome space*.

A feasible solution $x \in X$ is *dominated* by $\hat{x} \in X$, or $\hat{x}$ *dominates* x, if $f_i(\hat{x}) \geq f_i(x)$ for $i = 1, 2$ and at least one of the two inequalities is strict. The same terminology can be applied to points in outcome space, so that $y = f(x)$ is dominated by $\hat{y} = f(\hat{x})$ and $\hat{y}$ dominates y. If $\hat{x}$ dominates x and $f_i(\hat{x}) > f_i(x)$ for $i = 1, 2$, then the dominance relation is *strong*, otherwise it is *weak* (and correspondingly in outcome space).

A feasible solution $\hat{x} \in X$ is said to be *efficient* if there is no other $x \in X$ such that x dominates $\hat{x}$. Let X_E denote the set of efficient solutions of (1) and let Y_E denote the image of X_E in the outcome space, that is $Y_E = f(X_E)$. The set Y_E is referred to as the set of *Pareto outcomes* of (1). An outcome $y \in Y \setminus Y_E$ is called *non-Pareto*. An efficient solution $\hat{x} \in X$ is *weakly efficient* if there exists $x \in X$ weakly dominated by

*Dept. of Industrial and Systems Engineering, Lehigh University, Bethlehem PA, tkr2@lehigh.edu
†Dept. of Mathematical Sciences, Clemson University, Clemson SC, mjs@clemson.edu
‡Dept. of Mathematical Sciences, Clemson University, Clemson SC, wmalgor@clemson.edu

$\hat{x}$, otherwise $\hat{x}$ is *strongly efficient*. Correspondingly, $\hat{y} = f(\hat{x})$ is *weakly* or *strongly* Pareto. The Pareto set Y_E is *uniformly dominant* if all points in Y_E are strongly Pareto.

The operator *vmax* means that solving (1) is understood to be the problem of generating efficient solutions in X and Pareto outcomes in Y. Note that in (1), we require all variables to have integer values. In a *biobjective mixed integer program*, not all variables are required to be integral. The results of this paper apply equally to mixed problems, as long as Y_E remains a finite set.

Because several members of X may map to the same outcome in Y, it is often convenient to formulate a multiobjective problem in the outcome space. For BIPs, problem (1) then becomes

$$\begin{array}{ll} \text{vmax} & y = [y_1, y_2] \\ \text{subject to} & y \in Y \subset \mathbb{R}^2. \end{array} \tag{2}$$

Depending upon the form of the objective functions and the set X, BIPs are classified as either linear or nonlinear. In linear BIPs, the objective functions are linear and the feasible set is the set of integer vectors within a polyhedral set. All other BIPs are considered nonlinear.

1.2 Previous Work

A variety of solution methods are available for solving BIPs. These methods have typically either been developed for (general) multiobjective integer programs, and so are naturally applicable to BIPs, or they have been developed specifically for the biobjective case. Depending on the application, the methods can be further classified as either interactive or non-interactive. Non-interactive methods aim to calculate either the entire Pareto set or a subset of it based on an a priori articulation of a decision maker's preferences. Interactive methods also calculate Pareto outcomes, but they do so based on a set of preferences that are revealed progressively during execution of the algorithm.

Overviews of different approaches to solving multiobjective integer programs are provided by Climaco et al. [9] and more recently by Ehrgott and Gandibleux [10, 11] and Ehrgott and Wiecek [12]. In general, the approaches can be classified as exact or heuristic and grouped according to the methodological concepts they use. Among others, the concepts employed in exact algorithms include branch and bound techniques [1, 32, 40, 41, 42, 46], dynamic programming [51, 52], implicit enumeration [27, 35], reference directions [22, 33], weighted norms [2, 3, 13, 23, 34, 44, 47, 49], weighted sums with additional constraints [8, 14, 34], and zero-one programming [5, 6]. Heuristic approaches such as simulated annealing, tabu search, and evolutionary algorithms have been proposed for multiobjective integer programs with an underlying combinatorial structure [11]. Lee and Pulat [29] and Sedeño-Noda and González-Martín [45] have developed methods for bicriterion integer network flows.

The algorithms of particular relevance to this paper are specialized approaches for biobjective programs based on a parameterized exploration of the outcome space. In this paper, we focus on a new algorithm, called the WCN algorithm, for identifying the complete Pareto set that takes this approach. The WCN algorithm builds on the results of Eswaran et al. [13], who proposed an exact algorithm to compute the complete Pareto set of BIPs based on Chebyshev norms, as well as Solanki [47], who proposed an approximate algorithm also using Chebyshev norms, and Chalmet et al. [8], who proposed an exact algorithm based on weighted sums.

The specialized algorithms listed in the previous paragraph reduce the problem of finding the set of Pareto outcomes to that of solving a parameterized sequence of single-objective integer programs (called *subproblems*) over the set X. Thus, the main factor determining the running time is the number of such subproblems that must be solved. The WCN algorithm is an improvement on the work of Eswaran et al. [13] in the sense that all Pareto outcomes are found by solving only $2|Y_E| - 1$ subproblems. The number of subproblems solved by Eswaran's algorithms depends on a tolerance parameter and can be much larger (see (8)). In addition, our method properly identifies weakly dominated outcomes, excluding them from the Pareto set. The algorithm of Chalmet et al. [8] solves approximately the same number of subproblems (as does an exact extension of Solanki [47]'s approximation algorithm), but the WCN algorithm (and Eswaran's) also finds the exact values of breakpoints (with respect to the weighted Chebyshev norm) between adjacent Pareto outcomes, where no such parametric information is available from either [8] or [47] and only approximate information is available with Eswaran's algorithm.

Although we focus mainly on generating the entire Pareto set, we also investigate the behavior of the WCN algorithm when used to generate approximations to the Pareto set, and we present an interactive version based on pairwise comparison of Pareto outcomes. The interactive WCN algorithm can generate any Pareto outcomes (as compared to Eswaran's interactive method which can only generate outcomes on the convex upper envelope of Y). The comparison may be supported with tradeoff information. Studies on tradeoffs in the context of the augmented (or modified) weighted Chebyshev scalarization have been conducted mainly for continuous multiobjective programs [19, 20, 21]. A similar view of global tradeoff information applies in the context of BIPs.

The remainder of this paper is organized as follows: In Section 2, we briefly review the foundations of the weighted-sum and Chebyshev scalarizations in biobjective programming. The WCN algorithm for solving BIPs is presented in Section 3. Section 4 describes details of the implementation using the SYMPHONY framework [39]. Results of a computational study are given in Section 5. Section 6 recaps our conclusions.

2 Fundamentals of Scalarization

The main idea behind what we term *probing algorithms* for biobjective discrete programs is to combine the two objectives into a single criterion, i.e., to *scalarize* the objective.. The combination is parameterized in some way so that as the parameter is varied, optimal outcomes for the single-objective programs correspond to Pareto outcomes for the biobjective problem. The main techniques for constructing parameterized single objectives are weighted sums (i.e., convex combinations) and weighted Chebyshev norms (and variations). The algorithms proceed by solving a sequence of subproblems (*probes*) for selected values of the parameters.

2.1 Weighted Sums

A single-objective mathematical program can be derived from a multiobjective program by taking a non-negative linear combination of the objective functions [16]. Without loss of generality, the weights can be scaled so they sum to one. Each selection of weights produces a different single-objective problem, and optimizing the resulting problem produces a Pareto outcome. For biobjective problems, the combined criterion is parameterized by a single scalar $0 \leq \alpha \leq 1$:

$$\max_{y \in Y} (\alpha y_1 + (1 - \alpha) y_2). \tag{3}$$

An optimal outcome for any single-objective program (3) lies on the *convex upper envelope* of outcomes, i.e., the Pareto portion of the boundary of conv(Y). Such an outcome is said to be *supported*. Not every Pareto outcome is supported. In fact, the existence of unsupported Pareto outcomes is common in practical problems. Thus, no algorithm that solves (3) for a sequence of values of α can be guaranteed to produce all Pareto outcomes, even in the case where f_i is linear for $i = 1, 2$. A Pareto set for which some outcomes are not supported is illustrated in Figure 1. In the figure, y^p and y^r are Pareto outcomes, but any convex combination of the two objective functions (linear in the example) produces one of y^s, y^q, and y^t as the optimal outcome. The points y^p and y^r are referred to as *convex dominated* solutions. The convex upper envelope of the outcome set is marked by the dashed line.

The algorithm of Chalmet et al. [8] searches for Pareto points over subregions of the outcome set. These subregions are generated in such a way as to guarantee that every Pareto point lies on the convex upper envelope of some subregion, ensuring that every Pareto outcome is eventually identified. The algorithm begins by identifying outcomes that maximize y_1 and y_2, respectively. Each iteration of the algorithm then searches an unexplored region between two known Pareto points, say y^s and y^t. The exploration (or *probe*) consists of solving the problem with a weighted-sum objective and "optimality constraints" that enforce a strict improvement over $\min\{y_1^s, y_1^s\}$ and $\min\{y_2^t, y_2^t\}$. If the constrained problem is infeasible, then there is no Pareto outcome in that region. Otherwise the optimal outcome y^q is generated and the region is split into the parts between y^p and y^q and between y^s and y^t. The algorithm continues until all subregions have been explored in this way.

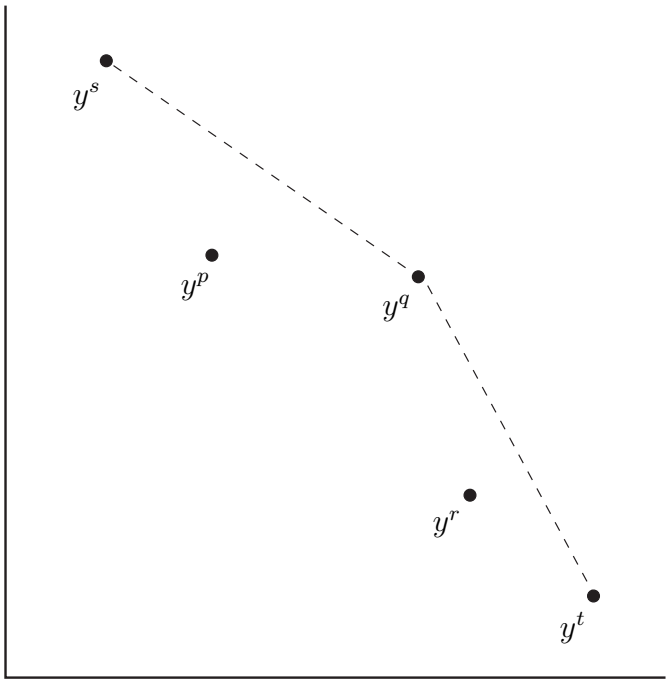

Figure 1: Example of the convex upper envelope of outcomes.

Note that y^q need not lie on the convex upper envelope of all outcomes, only of those outcomes between y^s and y^t, so all Pareto outcomes are generated. Also note that at every iteration, a new Pareto outcome is generated or a subregion is proven empty of outcomes. Thus, the total number of subproblems solved is $2|Y_E| + 1$.

2.2 Weighted Chebyshev Norms

The *Chebyshev norm* in $\mathbb{R}^2$ is the max norm (l_∞ norm) defined by $\|y\|_\infty = \max\{|y_1|, |y_2|\}$. The related distance between two points y^1 and y^2 is

$$d(y^1, y^2) = \|y^1 - y^2\|_\infty = \max\{|y_1^1 - y_1^2|, |y_2^1 - y_2^2|\}.$$

A *weighted* Chebyshev norm in $\mathbb{R}^2$ with weight $0 \leq \beta \leq 1$ is defined as $\|(y_1, y_2)\|_\infty^\beta = \max\{\beta|y_1|, (1-\beta)|y_2|\}$. The *ideal point* y^* is (y_1^*, y_2^*) where $y_i^* = \max_{x \in X} f_i(x)$ maximizes the single-objective problem with criterion f_i. Methods based on weighted Chebyshev norms select outcomes with minimum weighted Chebyshev distance from the ideal point. Figure 2 shows the southwest quadrant of the level lines for two values of β for an example problem.

The following are well-known results for the weighted Chebyshev scalarization [49].

Theorem 1 *If $\hat{y} \in Y_E$ is a Pareto outcome, then $\hat{y}$ solves*

$$\min_{y \in Y} \{\|y - y^*\|_\infty^\beta\} \tag{4}$$

for some $0 \leq \beta \leq 1$.

The following result of Bowman [7], used also in [13], was originally stated for the efficient set but it is useful here to state the equivalent result for the Pareto set.

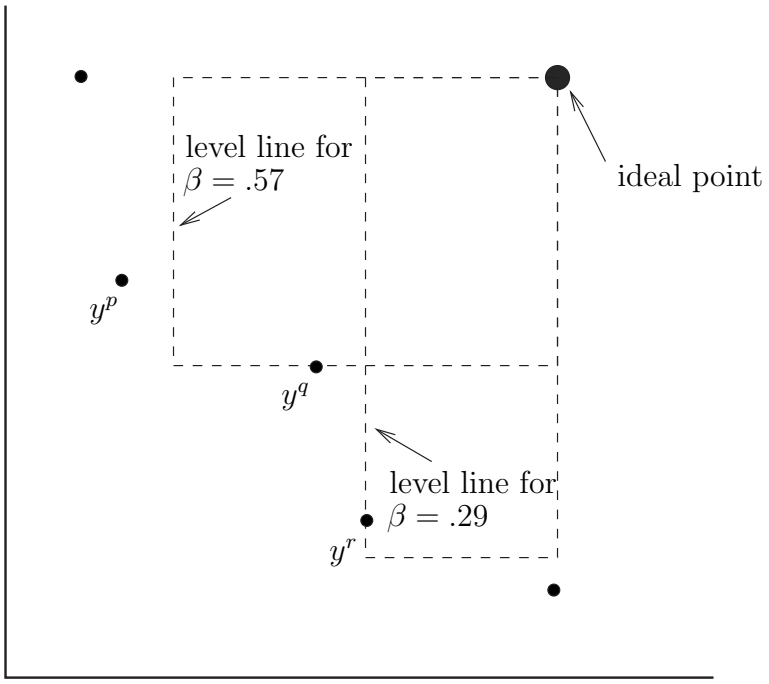

Figure 2: Example of weighted Chebyshev norm level lines.

Theorem 2 *If the Pareto set for (2) is uniformly dominant, then any solution to (4) corresponds to a Pareto outcome.*

For the remainder of this section, we assume that the Pareto set is uniformly dominant. Techniques for relaxing this assumption are discussed in Section 3.2 and their computational properties are investigated in Section 5.

Problem (4) is equivalent to

$$\begin{array}{lrcl} \text{minimize} & z & & \\ \text{subject to} & z & \geq & \beta(y_1^* - y_1), \\ & z & \geq & (1-\beta)(y_2^* - y_2), \\ & y & \in & Y, \end{array} \tag{5}$$

where $0 \leq \beta \leq 1$.

As in [13], we partition the set of possible values of β into subintervals over which there is a single unique optimal solution for (5). More precisely, let $Y_E = \{y^p \mid p \in 1, \ldots, N\}$ be the set of Pareto outcomes to (2), ordered so that $p < q$ if and only if $y_1^p < y_1^q$. Under this ordering, y^p and y^{p+1} are called *adjacent* Pareto points. For any Pareto outcome y^p, define

$$\beta_p = (y_2^* - y_2^p)/(y_1^* - y_1^p + y_2^* - y_2^p), \tag{6}$$

and for any pair of Pareto outcomes y^p and y^q, $p < q$, define

$$\beta_{pq} = (y_2^* - y_2^q)/(y_1^* - y_1^p + y_2^* - y_2^q). \tag{7}$$

Equation (7) generalizes the definition of $\beta_{p,p+1}$ in [13]. We obtain:

1. For $\beta = \beta_p$, y^p is the unique optimal outcome for (4), and

$$\beta_p(y_1^* - y_1^p) = (1-\beta_p)(y_2^* - y_2^p) = \|y^* - y^p\|_\infty^\beta.$$

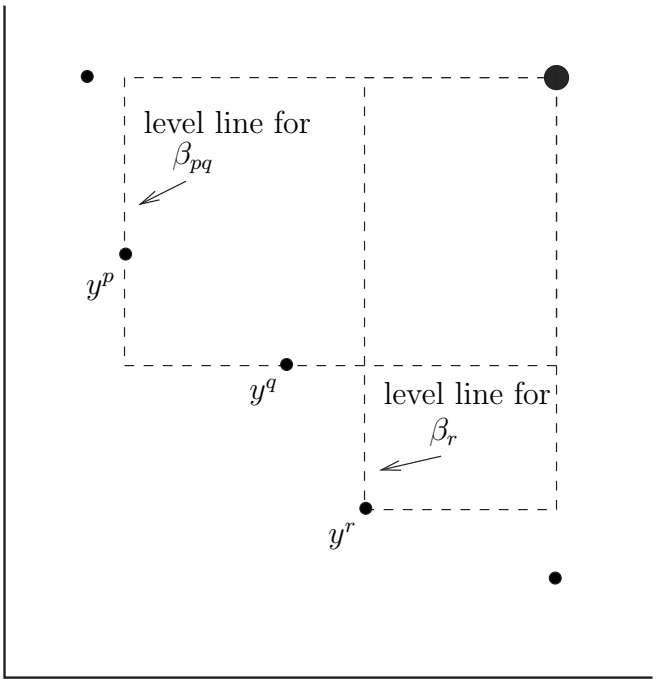

Figure 3: Relationship between Pareto points y^p, y^q, and y^r and the weights β_r and β_{pq}.

2. For $\beta = \beta_{pq}$, y^p and y^q are both optimal outcomes for (4), and

$$\beta_{pq}(y_1^* - y_1^p) = (1 - \beta_{pq})(y_2^* - y_2^q) = \|y^* - y^p\|_\infty^\beta = \|y^* - y^q\|_\infty^\beta.$$

This relationship is illustrated in Figure 3. This analysis is summarized in the following result [13].

Theorem 3 *If we assume the Pareto outcomes are ordered so that*

$$y_1^1 < y_1^2 < \cdots < y_1^N$$

and

$$y_2^1 > y_2^2 > \cdots > y_2^N$$

then

$$\beta_1 > \beta_{12} > \beta_2 > \beta_{23} > \cdots > \beta_{N-1,N} > \beta_N.$$

Also, y^p is an optimal outcome for (5) with $\beta = \hat{\beta}$ if and only if $\beta_{p-1,p} \leq \hat{\beta} \leq \beta_{p,p+1}$.

If y^p and y^q are adjacent outcomes, the quantity β_{pq} is the *breakpoint* between intervals containing values of β for which y^p and y^q, respectively, are optimal for (5). Eswaran et al. [13] describe an algorithm for generating the complete Pareto set using a bisection search to approximate the breakpoints. The algorithm begins by identifying an optimal solution to (5) for $\beta = 1$ and $\beta = 0$. Each iteration searches an unexplored region between pairs of consecutive values of β that have been probed so far (say, β_p and β_q). The search consists of solving (5) with $\beta_p < \beta = \hat{\beta} < \beta_q$. If the outcome is y^p or y^q, then the interval between $\hat{\beta}$ and β_p or β_q, respectively, is discarded. If a new outcome y^r is generated, the intervals from β_p to β_r and from β_r to β_q are placed on the list to investigate. Intervals narrower than a preset tolerance ξ are discarded. If $\hat{\beta} = (\beta_p + \beta_q)/2$, then the total number of subproblems solved in the worst case is approximately

$$|Y_E|(1 - \lg(\xi(|Y_E| - 1))). \tag{8}$$

Eswaran also describes an interactive algorithm based on pairwise comparisons of Pareto outcomes, but that algorithm can only reach supported outcomes.

Solanki [47] proposed an algorithm to generate an approximation to the Pareto set, but it can also be used as an exact algorithm. The algorithm is controlled by an "error measure" associated with each subinterval examined. The error is based on the relative length and width of the unexplored interval. This algorithm also begins by solving (5) for $\beta = 1$ and $\beta = 0$. Then for each unexplored interval between outcomes y^p and y^q, a "local ideal point" is $(\max\{y_1^p, y_1^q\}, \max\{y_2^p, y_2^q\})$. The algorithm solves (5) with this ideal point and constrained to the region between y^p and y^q. If no new outcome to this subproblem is found, then the interval is explored completely and its error is zero. Otherwise a new outcome y^r is found and the interval is split. The interval with largest error is selected to explore next. The algorithm proceeds until all intervals have error smaller than a preset tolerance. If the error tolerance is zero, this algorithm requires solution of $2|Y_E| - 1$ subproblems and generates the entire Pareto set.

3 An Algorithm for Biobjective Integer Programming

This section describes an improved version of the algorithm of Eswaran et al. [13]. Eswaran's method has two significant drawbacks:

- It cannot be guaranteed to generate all Pareto points if several such outcomes fall in a β-interval of width smaller than the tolerance ξ. If ξ is small enough, then all Pareto outcomes will be found (under the uniform dominance assumption). However, the algorithm does not provide a way to bound ξ to guarantee this result.
- As noted above, the running time of the algorithm is heavily dependent on ξ. If ξ is small enough to provide a guarantee that all Pareto outcomes are found, then the algorithm may solve a significant number of subproblems that produce no new information about the Pareto set.

Another disadvantage of Eswaran's algorithm is that it does not generate an exact set of breakpoints. The WCN algorithm generates exact breakpoints, as described in Section 2.2, and guarantees that all Pareto outcomes and the breakpoints are found by solving a sequence of $2|Y_E| - 1$ subproblems. The complexity of our method is on par with that of Chalmet et al. [8], and the number of subproblems solved is asymptotically optimal. However, as with Eswaran's algorithm, Chalmet's method does not generate or exploit the breakpoints. One potential advantage of weighted-sum methods is that they behave correctly in the case of non-uniformly dominant Pareto sets, but Section 3.2.2 describes techniques for dealing with such sets using Chebyshev norms.

3.1 The WCN Algorithm

Let $P(\hat{\beta})$ be the problem defined by (5) for $\beta = \hat{\beta}$ and let $N = |Y_E|$. Then the WCN (weighted Chebyshev norm) algorithm consists of the following steps:

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 $I = \{(y^1, y^N)\}$ and $S = \{(x^1, y^1), (x^N, y^N)\}$ (where $y^j = f(x^j)$).

Iteration While $I \neq \emptyset$ do:

1. Remove any (y^p, y^q) from I.
2. Compute β_{pq} as in (7) 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 (x^r, y^r) to S. Add (y^p, y^r) and (y^r, y^q) to I.

By Theorem 3, every iteration of the algorithm must identify either a new Pareto point or a new breakpoint $\beta_{p,p+1}$ between adjacent Pareto points. Since the number of breakpoints is $N-1$, the total number of iterations is $2N-1 = O(N)$. Any algorithm that identifies all N Pareto outcomes by solving a sequence of subproblems over the set X must solve at least N subproblems, so the number of iterations performed by this algorithm is asymptotically optimal among such methods.

3.2 Algorithmic Enhancements

The WCN algorithm can be improved in a number of ways. We describe some global improvements in this section.

3.2.1 A Priori Upper Bounds

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. In Eswaran's algorithm, this step is necessary, as which of y^p and y^q is optimal for $P(\hat{\beta})$ determines which half of the unexplored interval can be discarded. In the WCN algorithm, generating either y^p or y^q indicates that the entire interval can be discarded. No additional information is gained by knowing which of y^p or y^q was generated.

Using this fact, the WCN algorithm can be improved as follows. Consider an unexplored interval between Pareto outcomes y^p and y^q. Let ϵ_1 and ϵ_2 be positive numbers 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 $i = 1, 2$. For example, if $f_1(x)$ and $f_2(x)$ are integer-valued, then $\epsilon_1 = \epsilon_2 = 1$. Then it must be the case that

$$\|y^* - y^r\|_\infty^{\beta_{pq}} + \min\{\beta_{pq}\epsilon_1, (1-\beta_{pq})\epsilon_2\} \leq \|y^* - y^p\|_\infty^{\beta_{pq}} = \|y^* - y^q\|_\infty^{\beta_{pq}} \tag{9}$$

Hence, we can impose an a priori upper bound of

$$\|y^* - y^p\|_\infty^{\beta_{pq}} - \min\{\beta_{pq}\epsilon_1, (1-\beta_{pq})\epsilon_2\} \tag{10}$$

when solving the subproblem $P(\beta_{pq})$. This upper bound effectively eliminates all outcomes that do not have strictly smaller Chebyshev norm values from the search space of the subproblem. The outcome of Step 2 is now either a new outcome or infeasibility. Detecting infeasibility generally has a significantly lower computational burden than verifying optimality of a known outcome, so this modification generally improves overall performance.

3.2.2 Relaxing the Uniform Dominance Requirement

Many practical problems violate the assumption of uniform dominance of the Pareto set made in the WCN algorithm. While probing algorithms based on weighted sums (such as that of Chalmet et al. [8]) do not require this assumption, algorithms based on Chebyshev norms must be modified to take non-uniform dominance into account. If the Pareto set is not uniformly dominant, problem $P(\beta)$ may have multiple optimal outcomes, some of which are not Pareto.

An outcome that is weakly dominated by a Pareto outcome is problematic, because both may lie on the same level line for some weighted Chebyshev norms, hence both may solve $P(\beta)$ for some β encountered in the course of the algorithm. For example, in Figure 4, the dashed rectangle represents the optimal level level of the Chebyshev norm for a given subproblem $P(\beta)$. In this case, both y^p and y^q are optimal for $P(\beta)$, but y^p weakly dominates y^q. The point y^r, which is on a different "edge" of the level line is also optimal, but is neither weakly dominated by nor a weak dominator of either y^p or y^q. If an outcome y is optimal for some $P(\beta)$, it must lie on an edge of the optimal level line and cannot be strongly dominated by any other outcome. Solving (5) using a standard branch and bound approach only determines the optimal level line and returns one outcome on that level line. As a secondary objective, we must also ensure that the outcome generated is as close as possible to the ideal point, as measured by an l_p norm for some $p < \infty$. This ensures that the final outcome is Pareto. There are two approaches to accomplishing this goal, which we cover in the next two sections.

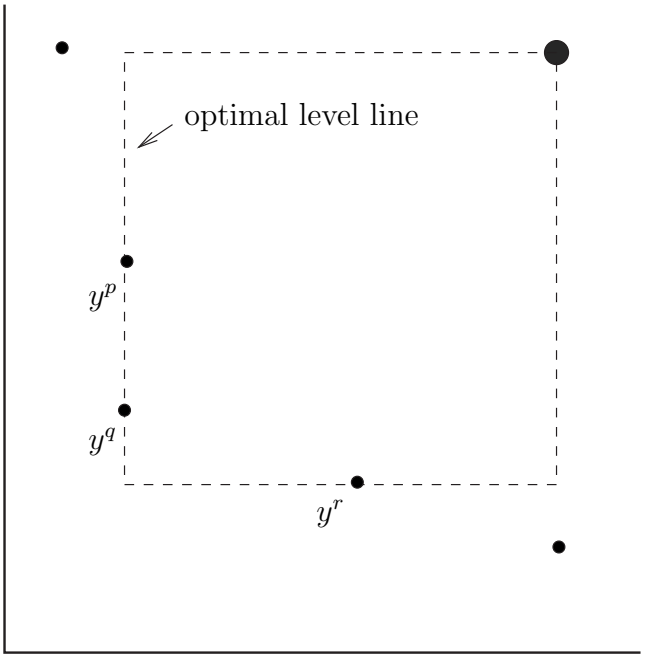

Figure 4: Weak domination of y^r by y^p.

Augmented Chebyshev norms. One way to guarantee that a new outcome found in Step 2 of the WCN algorithm is in fact a Pareto point is to use the augmented Chebyshev norm defined by Steuer [48].

Definition 1 *The* augmented Chebyshev norm *is 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.

The idea is to ensure that we generate the outcome closest to the ideal point along one edge of the optimal level line, as measured by both the l_∞ norm *and* the l_1 norm. This is done by actually adding a small multiple of the l_1 norm distance to the Chebyshev norm distance. A graphical depiction of the level lines under this norm is shown in Figure 5. The angle between the bottom edges of the level line is

$$\theta_1 = \tan^{-1}[\rho/((1-\beta+\rho)],$$

and the angle between the left side edges is

$$\theta_2 = \tan^{-1}[\rho/((\beta+\rho)].$$

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

$$\begin{array}{lrcl} \min & z & + & \rho(|y_1^* - y_1| + |y_2^* - y_2|) \\ \text{subject to} & z & \geq & \beta(y_1^* - y_1) \\ & z & \geq & (1-\beta)(y_2^* - y_2) \\ & y & \in & Y. \end{array} \tag{11}$$

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

$$\min z - \rho(y_1 + y_2). \tag{12}$$

For fixed $\rho > 0$ small enough:

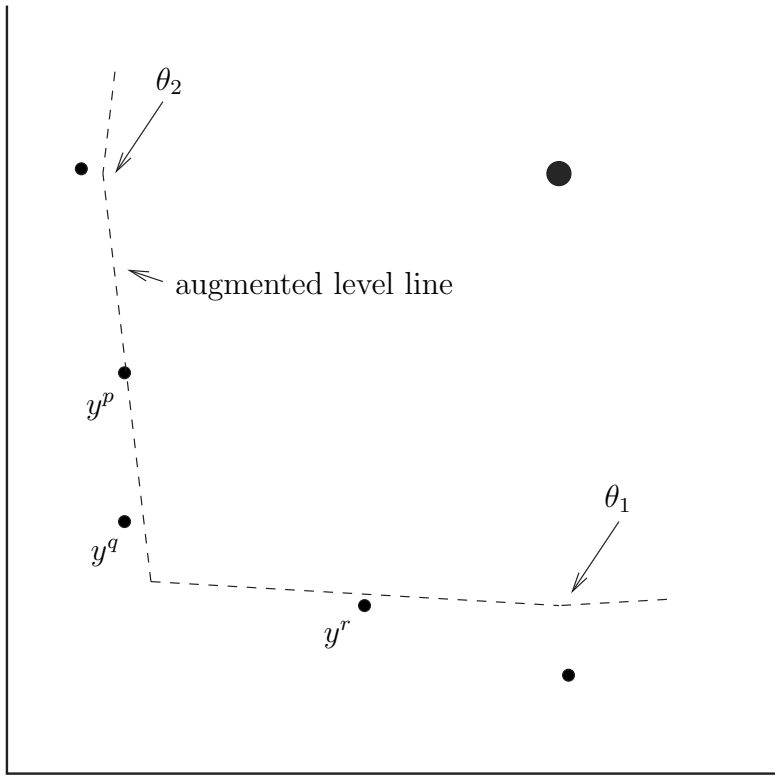

Figure 5: Augmented Chebyshev norm. Point y^p is the unique minimizer of the augmented-norm distance from the ideal point.

- all optimal outcomes for problem (11) are Pareto (in particular, they are not weakly dominated); and
- for a given Pareto outcome y for problem (11), there exists $0 \leq \hat{\beta} \leq 1$ such that y is the unique outcome to problem (11) with $\beta = \hat{\beta}$.

In practice, choosing a proper value for ρ can be problematic. Too small a ρ can cause numerical difficulties because the weight of the secondary objective can lose significance with respect to the primary objective. This situation can lead to generation of weakly dominated outcomes despite the augmented objective. On the other hand, too large a ρ can cause some Pareto outcomes to be unreachable (i.e., not optimal for problem (11) for any choice of β). Steuer [48] recommends $0.001 \leq \rho \leq 0.01$, but these values are completely ad hoc. The choice of ρ that works properly depends on the relative size of the optimal objective function values and cannot be computed a priori. In some cases, values of ρ small enough to guarantee detection of all Pareto points (particularly for β close to zero or one) may already be small enough to cause numerical difficulties.

Combinatorial methods. An alternative strategy for relaxing the uniform dominance assumption is to implicitly enumerate all optimal outcomes to $P(\beta)$ and eliminate the weakly dominated ones using cutting planes. This increases the time required to solve $P(\beta)$, but eliminates the numerical difficulties associated with the augmented Chebyshev norm. To implement this method, the subproblem solver must be allowed to continue to search for alternative optimal outcomes to $P(\beta)$ and record the best of these with respect to a secondary objective. This is accomplished by modifying the usual pruning rules for the branch and bound algorithm used to solve $P(\beta)$. In particular, the solver must not prune any node during the search unless it is either proven infeasible or its upper bound falls *strictly* below that of the best known lower bound, i.e., the

best outcome seen so far with respect to the weighted Chebyshev norm. This technique allows alternative optima to be discovered as the search proceeds.

An important aspect of this modification is that it includes a prohibition on pruning any node that has already produced an integer feasible solution (corresponding to an outcome in Y). Although such a solution must be optimal with respect to the weighted Chebyshev norm (subject to the constraints imposed by branching), the outcome may still be weakly dominated. Therefore, when a new outcome $\hat{y}$ is found, its weighted Chebyshev norm value is compared to that of the best outcome found so far. If the value is strictly larger, the solution is discarded. If the value is strictly smaller, it is installed as the new best outcome seen so far. If its norm value is equal to the current best outcome, it is retained only if it weakly dominates that outcome. After determining whether to install $\hat{y}$ as the best outcome seen so far, we impose an *optimality cut* that prevents any outcomes that are weakly dominated by $\hat{y}$ from being subsequently generated in further processing of the current node. To do so, we determine which of the two constraints

$$z \geq \beta(y_1^* - y_1) \tag{13}$$
$$z \geq (1-\beta)(y_2^* - y_2) \tag{14}$$

from problem (4) is binding at $\hat{y}$. This determines on which "edge" of the level line the outcome lies. If only the first constraint is binding, then any outcome $\bar{y}$ that is weakly dominated by $\hat{y}$ must have $\bar{y}_1 < \hat{y}_1$. This corresponds to moving closer to the ideal point in l_1 norm distance along the edge of the level line. Therefore, we impose the optimality cut

$$y_2 \geq \hat{y}_2 + \epsilon_2, \tag{15}$$

where ϵ_i is determined as in Section 3.2.1. Similarly, if only the second constraint is binding, we impose the optimality cut

$$y_1 \geq \hat{y}_1 + \epsilon_1. \tag{16}$$

If both constraints are binding, this means that the outcome lies at the intersection of the two edges of the level line. In this case, we arbitrarily impose the first cut to try to move along that edge, but if we fail, then we impose the second cut. After imposing the optimality cut, the current outcome becomes infeasible and processing of the node (and possibly its descendants) is continued until either a new outcome is determined or the node proves to be infeasible.

One detail we have glossed over is the possibility that the current value of β may be a breakpoint between two previously undiscovered Pareto outcomes. This means there is a distinct outcome on *each* edge of the optimal level line. In this case, it does not matter which of these outcomes is produced—only that the outcome produced is not weakly dominated. Therefore, once we have found the optimal level line, we confine our search for a Pareto outcome to only one of the edges (the one on which we discover a solution first). This is accomplished by discarding any outcome discovered that has the same weighted Chebyshev norm value as the current best, but is incomparable to it, i.e., is neither weakly dominated by nor a weak dominator of it.

Hybrid methods. A third alternative, which is effective in practice, is to combine the augmented Chebyshev norm method with the combinatorial method described above. To do so, we simply use the augmented objective function (12) while also applying the combinatorial methodology described above. This has the effect of guarding against values of ρ that are too small to ensure generation of Pareto outcomes, while at the same time guiding the search toward Pareto outcomes. In practice, this hybrid method tends to reduce running times over the pure combinatorial method. Computational results with both methods are presented in Section 5.

3.3 Approximation of the Pareto Set

If the number of Pareto outcomes is large, the computational burden of generating the entire set may be unacceptable. In that case, it may be desirable to generate just a subset of representative points, where a "representative" subset is one that is "well-distributed over the entire set" [47]. Deterministic algorithms

using Chebyshev norms have been proposed to accomplish that task for general multicriteria programs that subsume BIPs [24, 26, 28], but the works of Solanki [47] and Schandl et al. [44] seem to be the only specialized deterministic algorithms proposed for BIPs. None of the papers known to the authors offer in-depth computational results on the approximation of the Pareto set of BIPs with deterministic algorithms. (See Ruzika and Wiecek [43] for a recent review.)

Solanki's method minimizes a geometric measure of the "error" associated with the generated subset of Pareto outcomes, generating the smallest number of outcomes required to achieve a prespecified bound on the error. Schandl's method employs polyhedral norms not only to find an approximation but also to evaluate its quality. A norm method is used to generate supported Pareto outcomes while the lexicographic Chebyshev method and a cutting-plane approach are proposed to find unsupported Pareto outcomes.

Any probing algorithm can generate an approximation to the Pareto set by simply terminating early. (Solanki's algorithm can generate the entire Pareto set by simply running until the error measure is zero.) The representativeness of the resulting approximation can be influenced by controlling the order in which available intervals are selected for exploration. Desirable features for such an ordering are:

- the points should be representative, and
- the computational effort should be minimized.

In the WCN algorithm, both of these goals are advanced by selecting unexplored intervals in a first-in-first-out (FIFO) order. FIFO selection increases the likelihood that a subproblem results in a new Pareto outcome and tends to minimize the number of infeasible subproblems (i.e., probes that do not generate new outcomes, when terminating the algorithm early). It also tends to distribute the outcomes across the full range of β. Section 5 describes a computational experiment demonstrating this result.

3.4 An Interactive Variant of the Algorithm

After employing an algorithm to find all (or a large subset of) Pareto outcomes, a decision maker intending to use the results must then engage in a second phase of decision making to determine the one Pareto point that best suits the needs of the organization. In order to select the "best" from among a set of Pareto outcomes, the outcomes must ultimately be compared with respect to a single-objective utility function. If the decision maker's utility function is known, then the final outcome selection can be made automatically. Determining the exact form of this utility function for a particular decision maker, however, is a difficult challenge for researchers. The process usually involves restrictive assumptions on the form of such a utility function, and may require complicated input from the decision maker.

An alternative strategy is to allow the decision maker to search the space of Pareto outcomes interactively, responding to the outcomes displayed by adjusting parameters to direct the search toward more desirable outcomes.

An interactive version of the WCN algorithm consists of the following steps:

Initialization Solve $P(1)$ and $P(0)$ to identify optimal outcomes y^1 and y^N, respectively, and the ideal point $y^* = (y^1_1, y^N_2)$. Set $I = \{(y^1, y^N)\}$ and $S = \{(x^1, y^1), (x^N, y^N)\}$ (where $y^j = f(x^j)$).

Iteration While $I \neq \emptyset$ do:

1. Allow user to select (y^p, y^q) from I. Stop if user declines to select. Compute β_{pq} as in (7) and solve $P(\beta_{pq})$.
2. If no new outcome is found, then y^p and y^q are adjacent in the list $(y^1, y^2, \ldots, y^N)$. Report this fact to the user.
3. Otherwise, a new outcome y^r is generated. Report (x^r, y^r) to the user and add it to S. Add (y^p, y^r) and (y^r, y^q) to I.

This algorithm can be used as an interactive "binary search," in which the decision maker evaluates a proposed outcome and decides whether to give up some value with respect to the first objective in order to gain some value in the second or vice versa. If the user chooses to sacrifice with respect to objective f_1, the next probe finds an outcome (if one exists) that is better with respect to f_1 than any previously-identified outcome except the last. In this way, the decision maker homes in on a satisfactory outcome or on a pair of adjacent outcomes that is closest to the decision maker's preference. Unlike many interactive algorithms, this one does not attempt to model the decision maker's utility function. Thus, it makes no assumptions regarding the form of this function and neither requires nor estimates parameters of the utility function.

3.5 Analyzing Tradeoff Information

In interactive algorithms, it can be helpful for the system to provide the decision maker with information about the tradeoff between objectives in order to aid the decision to move from a candidate outcome to a nearby one. In problems where the boundary of the Pareto set is continuous and differentiable, the slope of the tangent line associated with a particular outcome provides local information about the rate at which the decision maker trades off value between objective functions when moving to nearby outcomes.

With discrete problems, there is no tangent line to provide local tradeoff information. Tradeoffs between a candidate outcome and another particular outcome can be found by computing the ratio of improvement in one objective to the decrease in the other. This information, however, is specific to the outcomes being compared and requires knowledge of both outcomes. In addition, achieving the computed tradeoff requires moving to the particular alternate outcome used in the computation, perhaps bypassing intervening outcomes (in the ordering of Theorem 3) or stopping short of more distant ones with different (higher or lower) tradeoff rates.

A global view of tradeoffs for continuous Pareto sets, based on the pairwise comparison described above, is provided by Kaliszewski [21]. For a decrease in one objective, the tradeoff with respect to the other is the supremum of the ratio of the improvement to the decrease over all outcomes that actually decrease the first objective and improve the second. Kaliszewski's technique can be extended to discrete Pareto sets as follows.

With respect to a particular outcome y^p, a pairwise tradeoff between y^p and another outcome y^q with respect to objectives i and j is defined as

$$T_{ij}(y^p, y^q) = \frac{y_i^q - y_i^p}{y_j^p - y_j^q}.$$

Note that $T_{ji}(y^p, y^q) = T_{ij}(y^p, y^q)^{-1}$. In comparing Pareto outcomes, we adopt the convention that objective j is the one that decreases when moving from y^p to y^q, so the denominator is positive and the tradeoff is expressed as units of increase in objective i per unit decrease in objective j. Then a global tradeoff with respect to y^p when allowing decreases in objective j is given by

$$T_{ij}^G(y^p) = \max_{y \in Y : y_j < y_j^p} \frac{y_i - y_i^p}{y_j^p - y_j}. \tag{17}$$

The tradeoff analysis is illustrated in Figure 6.

Computing $T_{ij}^G(y^p)$ requires knowledge of all Pareto outcomes in $\{y \in Y : y_j < y_j^p\}$. In an interactive setting, however, outcomes are generated dynamically. The best that can be done is to provide a lower bound on the global tradeoff based on the subset of $\{y \in Y : y_j < y_j^p\}$ generated up to that point. This approximate tradeoff information can be computed at each iteration of the interactive algorithm and reported to the decision maker.

This tradeoff measure must be taken with a grain of salt, as it can lead the decision maker away from convex dominated outcomes near the current one. An alternative measure can be constructed that minimizes the ratio in (17), but such a measure would tend to focus attention too much on those outcomes.

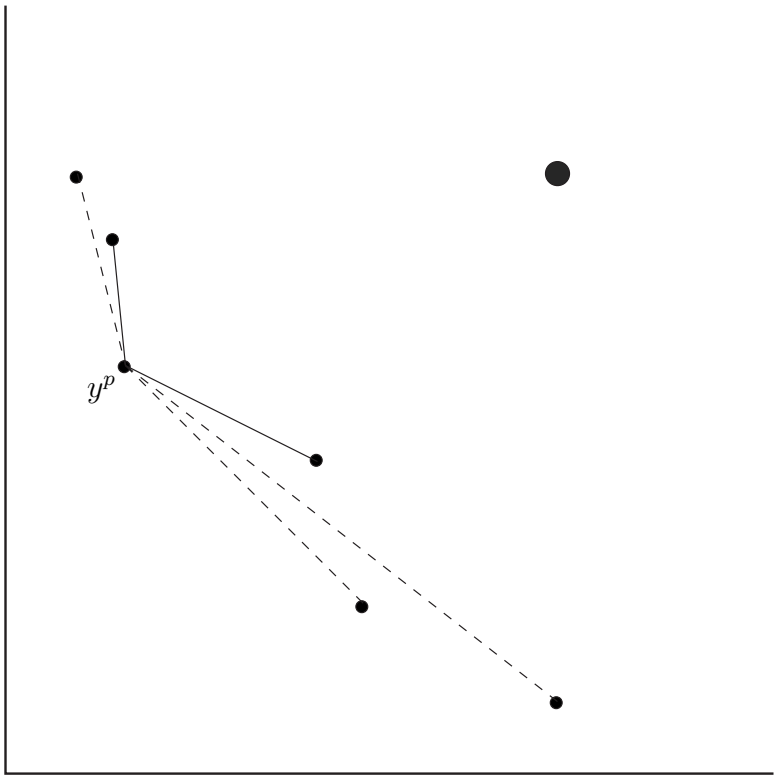

Figure 6: Tradeoff measures $T_{12}^G(y^p)$ and $T_{21}^G(y^p)$ illustrated.

4 Implementation

4.1 The SYMPHONY Callable Library

As part of this work, we have incorporated generic implementations of several of the algorithms discussed here into SYMPHONY [39], a customizable open-source callable library for solving mixed-integer linear programs available for download at `www.BranchAndCut.org`. These algorithms have been incorporated into version 5.0 of the library and can be accessed through either the native C interface or an extended version of the Open Solver Interface (OSI) available through the COIN-OR Foundation Web site [30]. Instance data is passed to SYMPHONY using the same interface as is used for single-criterion problems, except that the user is also required to pass a second objective function. SYMPHONY accepts input data in MPS format, using the COIN-OR MPS file parser, or in AMPL/GMPL format, using the parser available with GLPK [31]. It is also possible to pass data through a library call or to use the FlopC++ modeling language [17] to build models using C++ objects. This functionality is similar to ILOG's Concert technology [18]. More information on the library and its interface are available in the user's manual [37] and in a recent paper that discusses the new features in version 5.0 [39].

The main algorithms implemented in SYMPHONY are the WCN algorithm, the ACN algorithm, a hybrid version of these two algorithms, the bisection algorithm, and an algorithm for finding only the supported solutions. This latter algorithm can be thought of as a heuristic useful for obtaining an approximation of the complete set of Pareto outcomes. In addition, by setting various parameters, such as the probing order (LIFO or FIFO) and the maximum number of solutions to be generated, a wide variety of heuristic procedures can also be obtained. All algorithms are implemented within a single function consisting of a main loop that constructs the appropriate sequence of single-criterion MILPs and solves them by making a call to the underlying SYMPHONY library. In addition, some modifications to the underlying library were

needed. These are described in Section 4.2 below.

The underlying solver uses an implementation of the well-known branch and cut algorithm, which has been very successful in solving a wide variety of discrete optimization problems, including many notably difficult classes of combinatorial problems. SYMPHONY was originally developed to facilitate the implementation of custom solvers for such difficult combinatorial problems and has a number of customization features for this purpose, including the ability to generate application-specific valid inequalities and to define specialized branching rules. SYMPHONY has been used to develop customized solvers for a variety of combinatorial problem classes. Using the algorithms described here, all of these solvers can now be easily utilized with multiple objectives. The problem classes for which there are already solvers available include the traveling salesman and vehicle routing problems, the set partitioning problem, the mixed postman problem, the capacitated network routing problem (described below), and the knapsack problem. SYMPHONY can also be utilized as a generic MILP solver, employing the COIN-OR Cut Generation Library to generate valid inequalities and strong branching on fractional variables to perform the subproblem partitioning required by the branch and bound algorithm.

4.2 Customizing the Library

In order to support implementation of the WCN algorithm, we made a number of changes to SYMPHONY's internal library. To eliminate weakly dominated outcomes, we use the method described in Section 3.2.2. To accommodate this method, we had to modify SYMPHONY to allow the enumeration of alternative optima using optimality cuts, as described in section 3.2.2. We also added to SYMPHONY the ability to compare and track solutions according to two objective function values, rather than just one.

Numerical issues are particularly difficult when implementing algorithms for enumerating Pareto outcomes, as one is frequently forced to deal with small weights and objective function values defined on very different scales. It was therefore necessary to define a number of new error tolerances. As usual, we had to specify an integer tolerance for determining whether a given variable was integer valued or not. For this value, we used SYMPHONY's internal error tolerance, which in turn depends on the LP solver's error tolerance. We also had to specify the minimum Chebyshev norm distance between any two distinct outcomes. From this parameter and the parameter β, we determined the minimum difference in the value of the weighted Chebyshev norm for two outcomes, one of which weakly dominates the other. This was used to set SYMPHONY's granularity parameter, which is used to determine when a new best solution has been found as well as for node fathoming. We also added a parameter for specifying the weight ρ for the secondary objective in the augmented Chebyshev norm method. Selection of this parameter value is discussed below. Finally, we had to specify a tolerance for performing the bisection method of Eswaran. Selection of this tolerance is also discussed below.

4.3 Applications

We tested the algorithm on two different applications. The first is a solver for bicriterion knapsack problems and utilizes the generic SYMPHONY library with cutting planes generated by COIN-OR's Cut Generation Library. For this application, we wrote several parsers for reading files in the formats for which data files are readily available electronically. For all other aspects of the solver, the implementation was the generic one provided by SYMPHONY.

The second application is a solver for so-called *capacitated network routing problems* (CNRPs). This general class of models is described in [38] and is closely related to the single-source fixed-charge network flow problem, a well known and difficult combinatorial optimization problem, in which there is a tradeoff between the fixed cost associated with constructing a network and the variable cost associated with operating it. Here also, we wrote a customized parser for the input files (which use slight modifications of the TSPLIB format) and generating the formulation. In addition, we have implemented generators for a number of problem-specific classes of cutting planes. The implementation details of this solver will be covered in a separate paper.

5 Computational Study

5.1 Setup

Our computational study includes instances of the bicriterion knapsack problem from the test set described in [15] with between 10 and 50 variables. In addition, we solved instances of a CNRP model known as the *cable trench problem* (CTP) [50] constructed from data available in the library of vehicle routing problem instances maintained by author Ralphs [36]. Because the CTP is a difficult model, we created the instances by randomly sampling sets of 15 customers to create networks that were generally easy enough to solve repeatedly in the several tests that we ran, but varied enough to draw reasonably broad conclusions about the methods.

Although there are several articles that describe specialized exact algorithms for the bicriterion knapsack problem (see, e.g., those listed in the survey [11]), it is not our intent to draw direct comparisons with the computational performance of these methods. WCN is a general-purpose integer algorithm, and our main purpose is to draw comparisons with other methods in the same class. Thus, we do not use specialized subproblem solvers. Also, we found that the knapsack cut generators slowed the code down substantially and introduced severe numerical instabilities in the ACN and hybrid codes, so we turned these solver features off.

The computational platform was an SMP machine with four Intel Xeon 700MHz CPUs and 2G of memory (memory was never an issue). These experiments were performed with SYMPHONY 5.0. SYMPHONY is designed to work with a number of LP solvers through the COIN-OR Open Solver Interface. For the runs reported here, we used the OSI CPLEX interface with CPLEX 8.1 as the underlying LP solver.

In designing the computational experiments, there were several comparisons we wanted to make. First, we wanted to compare our exact approach to the bisection algorithm of Eswaran in terms of both computational efficiency and ability to produce all Pareto outcomes. Second, we wanted to compare the various approaches described in Section 3.2.2 for relaxing the uniform dominance assumption. Third, we wanted to test various approaches to approximating the set of Pareto outcomes. The results of these experiments are described in the next section.

5.2 Results

We report here on four experiments, each described in a separate table. In each table, the methods are compared to the WCN method (with the combinatorial method for eliminating weakly dominated outcomes), which is used as a baseline. All numerical data are reported as differences from the baseline method to make it easier to spot trends. On each chart, the group of columns labeled *Iterations* gives the total number of subproblems solved. The column labeled *Outcomes Found* gives the total number of Pareto outcomes reported by the algorithm. The *Max Missed* column contains the maximum number of missing Pareto outcomes in any interval between two Pareto outcomes that were found. This is a rough measure of how the missing Pareto outcomes are distributed among the found outcomes, and therefore indicates how well distributed the found outcomes are among the set of all Pareto outcomes. The entries in these columns in the *Totals* row are arithmetic means, with the exception of *Max Missed*, which is a maximum. Finally, the column labeled *CPU seconds* is the running time of the algorithm on the platform described earlier. For the knapsack problems, the results given are totals (maximums for *Max Missed*) for problems in each size class. Our test set included 20 problems of sizes 10 and 20, and 10 problems of sizes 30, 40, and 50. For the CNRPs, we summarize the totals (maximums for *Max Missed*) for 34 problems with 15 customers each. In addition, we show the individual results for one particularly difficult problem (`att48`). Because this particular instance is an outlier and because results for this instance are displayed separately, we have not included it in the totals shown for the CNRP.

In Table 1, we compare the WCN algorithm to the bisection search algorithm of Eswaran for three different tolerances, $\xi = 10^{-1}$, 10^{-2}, and 10^{-3}. (Our implementation of Eswaran's algorithm uses the approach described in Section 3.2.2 for eliminating weakly dominated outcomes.) Even at a tolerance of 10^{-3}, some outcomes are missed for the largest knapsack instances and the CNRP instance `att48`, which

has a large number of nonconvex regions in its frontier. It is clear that the tradeoff between tolerance and running time favors the WCN algorithm for this test set. The tolerance required in order to have a reasonable expectation of finding the full set of Pareto outcomes results in a running time far exceeding that for the WCN algorithm. This is predictable, based on the crude estimate of the number of iterations required in the worst case for Eswaran's algorithm given by (8) and we expect that this same behavior would hold for most classes of BIPs.

In Table 2, we compare the WCN algorithm with the ACN method described in Section 3.2.2 (i.e., the WCN method using augmented Chebyshev norms to eliminate weakly dominated solutions). Here, the columns are labeled with the secondary objective function weight ρ that was used. The ACN method is much faster for large secondary objective function weights (as one would expect). For the knapsack problems, $\rho = 10^{-4}$ results in identification of all Pareto outcomes. These problems are well-behaved numerically with well-scaled data and objective function values that are all of the same order of magnitude, so finding a value of ρ that works well was relatively easy. When we solved these instances using cutting planes to improve bounds, however, this introduced numerical instability sufficient to cause some instances to exhibit both missed solutions and generation of weakly dominated solutions. The CNRP instances were much less well behaved. The results demonstrate why it is not possible in general to determine a weight for the secondary objective function that both ensures the enumeration of all Pareto outcomes and protects against the generation of weakly dominated outcomes. For $\rho = 10^{-4}$, the ACN algorithm generates more outcomes than the WCN (which generates all Pareto outcomes) for some instances. This is because the ACN algorithm is producing weakly dominated outcomes in these cases, due to the value of ρ being set too small. Even setting the tolerance separately for each instance does not have the desired effect, as there are instances for which the algorithm produced both one more or more weakly dominated outcomes *and* missed Pareto outcomes. For these instances, it is not possible to choose a proper tolerance.

In Table 3, we compare WCN to the hybrid algorithm also described in Section 3.2.2. The value of ρ used is displayed above the columns of results for the hybrid algorithm. As described earlier, the hybrid algorithm has the advantages of both the ACN and the WCN algorithms and allows ρ to be set small enough to ensure correct behavior, as the combinatorial method will eliminate any spurious weakly dominated outcomes generated. For the knapsack instances, the results are identical to the non-hybrid ACN algorithm. This is because for these instances, the strongly dominated solutions are located naturally during the search process. In the cases where ACN produced weakly nondominated solutions, the dominating solutions were produced in another branch of the search tree, rather than in a descendant of the subproblem that produced the weakly dominated solution. Therefore, the optimality cuts that force the continued search have no effect. The CNRP instances exhibit a much more obvious effect from the hybrid algorithm, missing substantially fewer solutions than the non-hybrid, although at an increased cost of running time. As expected, the table shows that as ρ decreases, running times for the hybrid algorithm increase. However, it appears that choosing ρ approximately 10^{-5} results in a reduction in running time without a great loss in terms of accuracy. We also tried setting ρ to 10^{-6} and in this case, the full Pareto set is found for every problem, but the advantage in terms of running time is insignificant.

Finally, we experimented with a number of approximation methods. As discussed in Section 3.3, we chose to judge the performance of the various heuristics on the basis of both running time and the distribution of outcomes found among the entire set, as measured by the maximum number of missed outcomes in any interval between found outcomes. The results described in Table 1 indicate that Eswaran's bisection algorithm does in fact make a good heuristic based on our measure of distribution of outcomes, but the reduction in running times does not justify the loss of accuracy. The ACN algorithm with a relatively large value of ρ also makes a reasonable heuristic and the running times are much better. One disadvantage of these two methods is that it would be difficult to predict a priori the behavior of these algorithms, both in terms of running time and number of Pareto outcomes produced. To get a predictable number of outcomes in a predictable amount of time, we simply stopped the WCN algorithm after a fixed number of outcomes had been produced. The distribution of the resulting set of outcomes depends largely on the order in which the outcome pairs are processed, so we compared a FIFO ordering to a LIFO ordering. One would expect the FIFO ordering, which prefers processing parts of outcomes that are "far apart" from each other, to

Knapsack

	Iterations				Outcomes Found							CPU sec			
	WCN	Δ from WCN			WCN	Δ from WCN			Max Missed			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}	0	10^{-1}	10^{-2}	10^{-3}
10	278	12	300	679	149	−17	0	0	6	0	0	13.18	−2.04	10.17	30.27
20	364	−1	390	896	192	−22	−2	0	6	1	0	17.46	−2.70	16.87	46.18
30	324	−43	246	712	167	−25	0	0	4	0	0	24.93	−5.73	18.24	61.51
40	490	−108	235	898	250	−55	−11	0	5	2	0	65.88	−22.20	25.57	128.01
50	686	−138	235	1123	348	−69	−9	−1	11	1	1	139.42	−41.90	37.70	246.15
Totals	2142	−278	1406	4308	1106	−188	−22	−1	11	2	1	260.87	−74.57	108.55	512.12

CNRP

	Iterations				Outcomes Found							CPU sec			
	WCN	Δ from WCN			WCN	Δ from WCN			Max Missed			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}	0	10^{-1}	10^{-2}	10^{-3}
att48	147	−35	−9	104	74	−18	−15	−4	3	3	1	83.67	−24.51	−2.75	83.96
Totals	2381	−264	724	3794	1207	−135	−13	0	5	1	0	8122.16	−1198.45	2428.70	14520.00

Table 1: Comparing the WCN Algorithm with Bisection Search

Knapsack

	Iterations				Outcomes Found							CPU sec			
	WCN	Δ from WCN			WCN	Δ from WCN			Max Missed			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}	0	10^{-2}	10^{-3}	10^{-4}
10	278	−4	0	0	149	−2	0	0	1	0	0	13.18	0.06	−0.23	−0.10
20	364	−6	0	0	192	−3	0	0	1	0	0	17.46	−1.33	−0.41	−0.21
30	324	−6	0	0	167	−3	0	0	1	0	0	24.93	−1.28	−0.43	−0.43
40	490	−24	0	0	250	−12	0	0	1	0	0	65.88	−5.69	−1.70	−0.66
50	686	−28	-4	0	348	−24	−2	0	3	2	0	139.42	−27.18	−3.78	−1.35
Totals	2142	−70	0	0	1106	−34	−2	0	3	2	0	260.87	−35.42	−6.55	−2.75

CNRP

	Iterations				Outcomes Found							CPU sec			
	WCN	Δ from WCN			WCN	Δ from WCN			Max Missed			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}	0	10^{-2}	10^{-3}	10^{-4}
att48	147	−140	−106	−62	74	−70	−53	−31	44	17	8	83.67	−80.14	−59.83	−28.48
Totals	2381	−2056	−1012	−34	1207	−1028	−506	−17	18	5	1	8122.36	−7728.51	−5244.54	−1451.37

Table 2: Comparing the WCN Algorithm with the ACN Algorithm

Knapsack

	Iterations				Outcomes Found							CPU sec			
	WCN	Δ from WCN			WCN	Δ from WCN			Max Missed			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}	0	10^{-2}	10^{-3}	10^{-4}
10	278	−4	0	0	149	−2	0	0	1	0	0	13.18	0.34	0.12	0.16
20	364	−6	0	0	192	−3	0	0	1	0	0	17.46	−1.17	0.03	−0.16
30	324	−6	0	0	167	−3	0	0	1	0	0	24.93	−1.02	−0.11	0.10
40	490	−24	0	0	250	−12	0	0	1	0	0	65.88	−4.89	−1.09	−0.30
50	686	−28	-4	0	348	−14	-2	0	3	2	0	139.42	−13.04	−3.37	−1.17
Totals	2142	−68	−4	0	1106	−34	−2	0	3	2	0	260.87	−19.78	−4.42	−1.37

CNRP

	Iterations				Outcomes Found							CPU sec			
	WCN	Δ from WCN			WCN	Δ from WCN			Max Missed			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}	0	10^{-3}	10^{-4}	10^{-5}
att48	147	−106	−62	−6	74	−53	−31	−3	17	8	2	83.67	−59.34	−30.19	−1.12
Totals	2381	−1012	−44	−2	1207	−612	−22	−1	5	1	0	8122.36	−5481.53	−1531.35	−589.90

Table 3: Comparing the WCN Algorithm with the Hybrid ACN Algorithm

outperform the LIFO ordering, which prefers processing pairs of outcomes that are closer together. Table 4 shows that this is in fact the case. In these experiments, we stopped the algorithm after 15 outcomes were produced (the table only includes problems with more than 15 Pareto outcomes). The distribution of outcomes for the FIFO algorithm is dramatically better than that for the LIFO algorithm. Of course, other orderings are also possible. We also tried generating supported outcomes as a possible heuristic approach. This can be done extremely quickly, but the distribution of the outcomes produced was poor.

6 Conclusion

We have described an algorithm for biobjective discrete programs (BIPs) based on weighted Chebyshev norms. The algorithm improves on the similar method of Eswaran et al. [13] by providing a guarantee that all Pareto outcomes are identified with a minimum number of solutions of scalarized subproblems, as well as by producing the exact values of all breakpoints. The method thus matches the complexity of the best methods available for such problems. Our computational experience indicates that the WCN algorithm is robust across problem classes and is much less sensitive to numerical instabilities and tolerance settings than other probing methods. Overall, it works better "out of the box" and requires less tuning to work properly. To make the ACN algorithm work properly requires tuning and the proper selection of the Chebyshev weight parameter, which cannot be done a priori. The results also show that the hybrid algorithm with a small Chebyshev weight is a reasonable alternative that exhibits both the improved efficiency of the ACN algorithm and the stability of the WCN algorithm. The WCN algorithm also extends naturally to approximation of the Pareto set and to nonparametric interactive applications. We have described an extension of a global tradeoff analysis technique to discrete problems.

We incorporated the algorithm into the SYMPHONY branch-cut-price framework and demonstrated that it performs effectively on a test suite of biobjective knapsack and capacitated network routing problems. Topics for future research include studies of the performance of a parallel implementation of the WCN algorithm and extension to more than two objectives.

Acknowledgement. Authors Saltzman and Wiecek were partially supported by ONR Grant N00014-97-1-0784.

References

[1] Y. Aksoy. An interactive branch-and-bound algorithm for bicriterion nonconvex/mixed integer programming. *Naval Research Logistics*, 37(3):403–417, 1990.

[2] M. J. Alves and J. Climaco. Using cutting planes in an interactive reference point approach for multiobjective integer linear programming problems. *European Journal of Operational Research*, 117:565–577, 1999.

[3] M. J. Alves and J. Climaco. An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. *European Journal of Operational Research*, 124:478–494, 2000.

[4] K. Bhaskar. A multiple objective approach to capital budgeting. *Accounting and Business Research*, 9:25–46, 1979.

[5] G. R. Bitran. Linear multiple objective programs with zero-one variables. *Mathematical Programming*, 13:121–139, 1977.

[6] G. R. Bitran. Theory and algorithms for linear multiple objective programs with zero-one variables. *Mathematical Programming*, 17:362–390, 1979.

Knapsack

	Iterations			Outcomes Found			Max Missed		CPU sec		
	WCN	Δ from WCN		WCN	Δ from WCN				WCN	Δ from WCN	
Size		FIFO	LIFO		FIFO	LIFO	FIFO	LIFO		FIFO	LIFO
20	972	−948	−945	514	−484	−484	1	1	50.82	−50.82	−50.82
30	458	−443	−431	237	−132	−132	4	7	57.08	−57.08	−57.08
40	490	−474	−465	250	−100	−100	9	14	116.48	−116.48	−116.48
50	686	−671	−660	348	−198	−198	7	22	211.57	−211.57	−211.57
Totals	2606	−2536	−2501	1349	−914	−914	7	22	435.95	−435.95	−435.95

CNRP

	Iterations			Outcomes Found			Max Missed		CPU sec		
	WCN	Δ from WCN		WCN	Δ from WCN				WCN	Δ from WCN	
Name		FIFO	LIFO		FIFO	LIFO	FIFO	LIFO		FIFO	LIFO
att48	147	−132	−128	74	−59	−59	23	24	83.67	−69.08	−73.55
Totals	2492	−2003	−1709	1262	−782	−782	11	32	8191.62	−6224.13	−7244.55

Table 4: Comparing Limited WCN Algorithm with LIFO and FIFO Ordering

[7] V. J. Bowman. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In H. Thieriez, editor, *Multiple Criteria Decision Making*, pages 248–258. Springer, Berlin, 1976.

[8] L. G. Chalmet, L. Lemonidis, and D. J. Elzinga. An algorithm for the bi-criterion integer programming problem. *European Journal of Operational Research*, 25:292–300, 1986.

[9] J. Climaco, C. Ferreira, and M. E. Captivo. Multicriteria integer programming: an overview of different algorithmic approaches. In J. Climaco, editor, *Multicriteria Analysis*, pages 248–258. Springer, Berlin, 1997.

[10] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22:425–460, 2000.

[11] M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization—theory, methodology and applications. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization—State of the Art Annotated Bibliographic Surveys*, pages 369–444. Kluwer Academic Publishers, Boston, MA, 2002.

[12] M. Ehrgott and M. M. Wiecek. Multiobjective programming. In M. Ehrgott, J. Figueira, and S. Greco, editors, *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 667–722. Springer, Berlin, Germany, 2005.

[13] P. K. Eswaran, A. Ravindran, and H. Moskowitz. Algorithms for nonlinear integer bicriterion problems. *Journal of Optimization Theory and Applications*, 63(2):261–279, 1989.

[14] C. Ferreira, J. Climaco, and J. Paixão. The location-covering problem: a bicriterion interactive approach. *Investigación Operativa*, 4(2):119–139, 1994.

[15] X Gandibleaux. MCDM numerical instances library. http://www.univ-valenciennes.fr/ROAD/MCDM/ListMOKP.html.

[16] A. M. Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22:618–630, 1968.

[17] T. H. Hultberg. A presentation of FlopC++. http://www.mat.ua.pt/thh/flopc/orbit.pdf, 2003.

[18] ILOG, Inc., Mountain View CA. *ILOG Concert Technology Reference Manual*.

[19] I. Kaliszewski. Uisng tradeoff information in decision-making algorithms. *Computers and Operations Research*, 27:161–182, 2000.

[20] I. Kaliszewski. Dynamic parametric bounds on efficient outcomes in interactive multiple criteria decision making problems. *European Journal of Operational Research*, 147:94–107, 2003.

[21] I. Kaliszewski and W. Michalowski. Efficient solutions and bounds on tradeoffs. *Journal of Optimization Theory and Applications*, 94(2):381–394, 1997.

[22] J. Karaivanova, P. Korhonen, S. Narula, J. Wallenius, and V. Vassilev. A reference direction approach to multiple objective integer linear programming. *European Journal of Operational Research*, 81:176–187, 1995.

[23] J. Karaivanova, S. Narula, and V. Vassilev. An interactive procedure for multiple objective integer linear programming problems. *European Journal of Operational Research*, 68(3):344–351, 1993.

[24] K. Karaskal and M. Köksalan. Generating a representative subset of the efficient frontier in multiple criteria decision analysis. Working Paper 01-20, Faculty of Administration, University of Ottawa, 2001.

[25] P. Kere and J. Koski. Multicriterion optimization of composite laminates for maximum failure margins with an interactive descent algorithm. *Structural and Multidisciplinary Optimization*, 23(6):436–447, 2002.

[26] K. Klamroth, J. Tind, and M. M. Wiecek. Unbiased approximation in multicriteria optimization. *Mathematical Methods of Operations Research*, 56:413–437, 2002.

[27] D. Klein and E. Hannan. An algorithm for the multiple objective integer linear programming problem. *European Journal of Operational Research*, 93:378–385, 1982.

[28] M. M. Kostreva, Q. Zheng, and D. Zhuang. A method for approximating solutions of multicriteria nonlinear optimization problems. *Optimization Methods and Software*, 5:209–226, 1995.

[29] H. Lee and S. Pulat. Bicriteria network flow problems: Integer case. *European Journal of Operational Research*, 66:148–157, 1993.

[30] R. Lougee-Heimer et al. Computational infrastructure for operations research. http://www.coin-or.org.

[31] A. Makhorin. Introduction to GLPK. http://www.gnu.org/software/glpk/glpk.html, 2004.

[32] G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107(3):530–541, 1998.

[33] S. C. Narula and V. Vassilev. An interactive algorithm for solving multiple objective integer linear programming problems. *European Journal of Operational Research*, 79(3):443–450, 1994.

[34] P. Neumayer and D. Schweigert. Three algorithms for bicriteria integer linear programs. *OR Spectrum*, 16:267–276, 1994.

[35] H. Pasternak and U. Passy. Bicriterion mathematical programs with boolean variables. In J. L. Cochrane and M. Zeleny, editors, *Multiple Criteria Decision Making*, pages 327–348. University of South Carolina Press, 1973.

[36] T. K. Ralphs. Library of vehicle routing problem instances. www.BranchAndCut.org/VRP/data.

[37] T. K. Ralphs. SYMPHONY Version 5.0 user's manual. Technical Report 04T-011, Lehigh University Industrial and Systems Engineering, 2004.

[38] T. K. Ralphs and J. C. Hartman. Capacitated network routing (a preliminary progress report). Technical Report 01W-009, Lehigh University Industrial and Systems Engineering, 2001.

[39] T.K. Ralphs and M. Guzelsoy. The SYMPHONY callable library for mixed-integer linear programming. To appear in the Proceedings of the Ninth INFORMS Computing Society Conference, 2004.

[40] R. Ramesh, M. H. Karwan, and S. Zionts. Preference structure representation using convex cones in multicriteria integer programming. *Management Science*, 35(9):1092–1105, 1989.

[41] R. Ramesh, M. H. Karwan, and S. Zionts. An interactive method for bicriteria integer programming. *IEEE Transcations on Systems, Man, and Cybernetics*, 20(2):395–403, 1990.

[42] R. Ramesh, M. H. Karwan, and S. Zionts. Interactive bicriteria integer programming: a performance analysis. In M. Fedrizzi, J. Kacprzyk, and M. Roubens, editors, *Interactive Fuzzy Optimization and Mathematical Programming*, volume 368 of *Lecture Notes in Economics and Mathematical Systems*, pages 92–100. Springer-Verlag, 1991.

[43] S. Ruzika and M. M. Wiecek. Survey paper: Approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications*. to appear.

[44] B. Schandl, K. Klamroth, and M. M. Wiecek. Norm-based approximation in bicriteria programming. *Computational Optimization and Applications*, 20(1):23–42, 2001.

[45] A. Sedeño Noda and C. González-Martín. An algorithm for the biobjective integer minimum cost flow porblem. *Computers and Operations Research*, 28:139–156, 2001.

[46] W. S. Shin and D. B. Allen. An interactive paired-comparison method for bicriterion integer programming. *Naval Research Logistics*, 41(3):423–434, 1994.

[47] R. Solanki. Generating the noninferior set in mixed integer biobjective linear programs: an application to a location problem. *Computers and Operations Research*, 18:1–15, 1991.

[48] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application.* John Wiley & Sons, New York, 1986.

[49] R. E. Steuer and E.-U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26:326–344, 1983.

[50] F. J. Vasko, R. S. Barbieri, B. Q. Rieksts, K. L. Reitmeyer, and K. L. Stott, Jr. The cable trench problem: combining the shortest path and minimum spanning tree problems. *Comput. Oper. Res.*, 29(5):441–458, 2002.

[51] B. Villarreal and M. H. Karwan. Multicriteria integer programming: A (hybrid) dynamic programming recursive approach. *Mathematical Programming*, 21:204–223, 1981.

[52] B. Villarreal and M. H. Karwan. Multicriteria dynamic programming with an application to the integer case. *Journal of Mathematical Analysis and Applications*, 38:43–69, 1982.