Finding Diverse Optima and Near-Optima to Binary Integer Programs

Andrew C. Trapp, Renata A. Konrad
School of Business, Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA 01609, USA,
{atrapp@wpi.edu, rkonrad@wpi.edu}

Abstract: Typical output from an optimization solver is a single optimal solution. There are contexts, however, where a set of high-quality and diverse solutions may be beneficial, for example problems involving imperfect information, or those for which the structure of high-quality solution vectors can reveal meaningful insights. In view of this, we discuss a novel method to obtain multiple high-quality yet diverse solutions to pure binary (0–1) integer programs, employing fractional programming techniques to manage these typically competing goals. Specifically, we develop a general approach that makes use of Dinkelbach's algorithm to sequentially generate solutions that evaluate well with respect to both i) individual performance and, as a whole, ii) mutual variety. We assess the performance of our approach on a number of MIPLIB test instances from the literature. Using two diversity metrics, computational results show that our method provides an efficient way to optimize the fractional objective while sequentially generating multiple high-quality and diverse solutions.

Keywords: Binary (0–1) integer programming, fractional programming, Dinkelbach's algorithm, decision making, multiple solutions, solution diversity, solution quality

---

1. Introduction

The convention of mathematical programming solvers is to generate, whenever possible, a single optimal solution. While this is essential from an optimality point of view, there are scenarios where it may be advantageous to consider multiple solutions, and particularly if they are both high-quality and diverse. To give a few examples, one could imagine the following situations. A project manager wishes to fund several R&D projects subject to a limited budget, and is interested in examining dissimilar solutions with respect to risk exposure. A personnel supervisor has committed to a monthly schedule, and because of shifting employee availability, impromptu adjustments are required. In the context of production planning, several viable job assignments are available for review, and the most appropriate is selected according to domain expertise or other subjective criteria. In each of these examples there could be great value in having a readily available set of high-quality and yet diverse solutions.

Mathematical programs are abstract representations of real-world problems. Their parameters often include "best-guess" estimates and approximations of uncertain or fluctuating data (Sharda et al., 1988; Hoch and Schkade, 1996; Williams, 1999; Laguna et al., 2012), and further may entirely omit factors that are mathematically difficult to express, for instance the quality of work and service (Schittekat and Sörensen, 2009). As such, multiple solutions of high quality may offer more flexibility for the decision maker, especially when they are mutually diverse. Additionally, having several diverse solutions that evaluate well can provide insight into attractive features and inherent tradeoffs in the solution space. This is important, for example, in situations that are time-sensitive or cost-sensitive, or for one-time decisions having significant implications (see, e.g., Takriti et al., 1996; Fagerholt et al., 2009).

Many studies (Murthy et al., 1999; Glover et al., 2000; Danna et al., 2007; Greistorfer et al., 2008; Danna and Woodruff, 2009; Camm, 2014) discuss the merit of multiple high-quality solutions. While quality is clearly important, solution diversity seems to receive somewhat less exposure, yet has appeared in domains such as mixed integer programming (Glover et al., 2000; Danna et al., 2007; Danna and Woodruff, 2009), constraint programming (Hebrard et al., 2005), heuristic search (Deb, 1999; Zitzler et al., 2001; Martí et al., 2011) and multiobjective optimization (Masin and Bukchin, 2008).

Although a number of effective approaches exist to generate either diverse or high-quality solutions independently, few can accomplish both simultaneously (Greistorfer et al., 2008). Solutions of relatively high quality tend to display strong structural similarities among the variable values. On the other hand, solutions that exhibit higher diversity may well have disparate objective function values given their lack of proximity in the feasible region. Thus, an inherent tension exists when simultaneously considering both quality and diversity.

Even so, the ability to generate multiple solutions of high quality in reasonable time clearly exists given contemporary computational capabilities and industrial-strength solvers. While too many solutions may prove overwhelming to decision makers (Glover et al., 2000), a handful of such solutions, properly chosen, is of great benefit in the decision-making process (Glover et al., 2000; Danna et al., 2007), and can serve to paint a more complete landscape of the possibilities that a decision maker may consider. This is the topic we address in this paper. Namely, we discuss how to effectively generate a manageable set of diverse yet high-quality solutions for pure binary (0–1) integer programs. This is an important class of mathematical programs, including combinatorial optimization problems such as the knapsack, assignment, bin-packing, traveling salesman, and various scheduling problems, among others (Nemhauser and Wolsey, 1988; Schrijver, 2003).

Our contribution is a relatively fast and efficient method to dynamically generate multiple high-quality and diverse solutions to binary integer programs. The method is straightforward to implement and works in conjunction with any integer programming solver. Moreover, our approach, by construction, yields an optimal solution to the original problem – an important consideration in instances that feature exact problem data.

The remainder of the paper is organized as follows. We formally define the problem we solve in Section 2, and propose metrics for solution quality and diversity. In Section 3 we describe our approach for constructing a reasonably-sized set of such solutions. Section 4 highlights our computational experiments and results on test instances from the literature, while Section 5 concludes with directions for future work.

2. Problem Description

Consider the general pure binary integer programming optimization problem:

$$\begin{aligned} \max \qquad & \left\{ c^\top x \mid x \in \mathcal{S} \right\}, \\ \text{where} \qquad & \mathcal{S} = \left\{ x \in \{0,1\}^n : Ax \leq b \right\}. \end{aligned} \tag{BIP}$$

Problem (BIP) consists of a vector x of binary variables of dimension n, a right-hand side vector $b \in \mathbb{R}^m$, together with a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $c \in \mathbb{R}^n$. We assume $\mathcal{S} \neq \emptyset$, and note that the feasibility set $\mathcal{S}$ is bounded. Without loss of generality we consider only the maximization case.

Let $x^* \in \mathcal{S}$ be an optimal solution to (BIP) and $z^* = c^\top x^*$ be its optimal objective function value. For the sake of exposition, we assume there exists at least one alternative feasible solution in addition to x^* in $\mathcal{S}$, i.e., $\mathcal{S} \setminus \{x^*\} \neq \emptyset$, so that the discussion on alternate solutions is relevant. With respect to x^* and z^*, we next briefly review some common ways to assess diversity as well as quality.

2.1 Assessing Diversity, Assessing Quality

Measuring diversity between two binary vectors (e.g., $x^*, x \in \mathcal{S}$) can be accomplished using a variety of metrics, for example the L_1 (taxicab) norm and the L_2 (Euclidean) norm.

The quality of any $x \in \mathcal{S}$ is measured with objective function $c^\top x$, taking a maximum value at $z^* = c^\top x^*$. The distance from the optimal objective function value z^* can be expressed as:

$$z^* - c^\top x, \tag{1}$$

which is nonnegative for any $x \in \mathcal{S}$ as $z^* = c^\top x^* \geq c^\top x$. The quality of the solution x deteriorates as this difference grows larger.

2.2 Simultaneously Addressing Diversity and Quality of Solutions

While for any $x \in \mathcal{S}$ we can independently assess both diversity and quality, as mentioned in Section 1 solutions that perform well tend to lack diversity due to structural similarities. Likewise, solutions that score rather high in diversity are likely to be from a remote area of the feasible region, and thus may not evaluate well in terms of the objective function. This situation produces tension when both are desirable.

Multiobjective optimization is one approach used to handle the conflicting objectives of achieving diversity while ensuring quality (Miettinen, 1999; Løkketangen and Woodruff, 2005). Metaheuristic approaches are also considered to bring about diversity in solution sets of multiobjective problems (Fonseca and Fleming, 1993; Deb, 1999; Zitzler et al., 2001), including approaches that aim to improve the distribution of nondominated solutions (Murata et al., 2001; Gunawan et al., 2003). Another common way to resolve two conflicting objectives is by optimizing one of the objectives while constraining the other, for example as in classical portfolio theory (Markowitz, 1952).

We propose an alternate methodology to achieve balance between solution quality and diversity. Given problem (BIP) and x^*, z^*, consider the modified objective:

$$\mathcal{R} = \frac{N(x)}{D(x)} = \frac{\text{relative solution diversity}}{\text{relative deterioration in objective function quality}}. \tag{2}$$

With respect to x^* and z^*, given any $x \in \mathcal{S}$, objective $\mathcal{R}$ expresses the ratio of relative solution diversity to relative deterioration in the objective function corresponding to x. To measure solution diversity, we can consider the L_1 norm:

$$\|x - x^*\|_1 = \sum_{i=1}^{n} |x_i - x_i^*|, \tag{3}$$

whereas (1) captures the deterioration of the objective function corresponding to x. To eliminate scaling issues, we normalize these diversity and quality measures (discussed in Section 2.4) to create relative measures. However, nonlinearities remain in objective (2) arising from both the diversity measure as well as the objective's fractional nature.

2.3 Expressing Diversity in a Linear Fashion

The following involves an optimal solution x^* and any $x \in \{0,1\}^n$ and is a linear expression of diversity (Fischetti et al., 2005):

$$\sum_{i:x_i^*=0} x_i + \sum_{i:x_i^*=1} (1 - x_i). \tag{4}$$

Proposition 1 *With respect to an optimal solution x^* to* (BIP) *and any $x \in \{0,1\}^n$, expressions* (3) *and* (4) *are equivalent over all $x \in \{0,1\}^n$.*

Proof. For any $x_i, i = 1, \ldots, n$, there are four cases to consider; two where x_i and x_i^* are not equal, and two where they are equal. If $x_i \neq x_i^*$, then either i) $x_i = 0$ and $x_i^* = 1$, so that both (3) and (4) evaluate to 1 because $|x_i - x_i^*| = 1$ and $(1 - x_i) = 1$; or else ii) $x_i = 1$ and $x_i^* = 0$, where again both (3) and (4) evaluate to 1 because $|x_i - x_i^*| = 1$ and $x_i = 1$. If $x_i = x_i^*$, then both (3) and (4) evaluate to 0. ■

Metric (4) is linear, and introducing it as the diversity measure in (2) gives an advantage over perhaps more conventional but nonlinear norm metrics. We note that the linearity is directly due to the binary nature of the decision variables.

2.4 Normalizing Objective Terms

Diversity expression (4) and quality expression (1) may be of widely varying scale, and can thereby introduce a predisposed bias into the ratio $\mathcal{R}$. To offset any disparate magnitudes in these metrics, we normalize both expressions.

For the diversity metric, we use the length of the interval over which diversity can vary from x^* within $\mathcal{S}$ (see, e.g., Grodzevich and Romanko, 2006):

$$\text{max diversity} - \text{min diversity} = \max_{x \in \mathcal{S}} \left\{ \sum_{i:x_i^*=0} x_i + \sum_{i:x_i^*=1} (1 - x_i) \right\}. \tag{5}$$

By definition the minimum diversity value is zero, so (5) can be determined by solving a single binary integer program. The relative solution diversity can be found by dividing diversity measure (4) by (5).

To normalize quality, we use the length of the interval over which the solution quality can deteriorate from x^* within $\mathcal{S}$ (as measured by the objective):

$$\text{max deterioration} - \text{min deterioration} = \max_{x \in \mathcal{S}} \left\{ z^* - c^\top x \right\}. \tag{6}$$

As in (5), we solve a single binary integer program, and similarly note that the minimum deterioration value is zero. Thus the relative objective function deterioration can be found by dividing (1) by (6).

We merge normalization factors (5) and (6) into a single factor:

$$\mathcal{F} = \frac{\max\limits_{x \in \mathcal{S}} \left\{ z^* - c^\top x \right\}}{\max\limits_{x \in \mathcal{S}} \left\{ \sum\limits_{i:x_i^*=0} x_i + \sum\limits_{i:x_i^*=1} (1 - x_i) \right\}}. \tag{7}$$

The nonnegativity of (5) and (6) imply that normalization factor $\mathcal{F}$ is nonnegative. Additionally, should the cost of solving two additional binary integer programs be very high for a given problem instance, it may be possible to compute an approximate value for $\mathcal{F}$ by relaxing the binary variables and solving the linear programming relaxations.

2.5 Putting It All Together

With respect to x^* and z^*, the following optimization problem over $\mathcal{S}$ identifies an $x \in \mathcal{S}$ that maximizes the ratio of relative diversity to relative objective function deterioration:

$$\max_{x \in \mathcal{S}} \quad \mathcal{R} = \mathcal{F}\, \frac{N(x)}{D(x)} = \mathcal{F}\, \frac{\left(\sum\limits_{i:x_i^*=0} x_i + \sum\limits_{i:x_i^*=1} (1 - x_i) \right)}{(z^* - c^\top x) + \epsilon}, \tag{8}$$

where ϵ is a small positive constant that serves to ensure the denominator remains positive, an important consideration given the tendency for some binary integer programs to exhibit multiple optimal solutions (see, e.g., Margot, 2010).

3. Solution Approach: Incorporating Diversity and Quality

Formulation (8) is a specific type of mathematical program known as a fractional, or hyperbolic, binary programming problem (Martos et al., 1964), an area that has been studied extensively (Saipe, 1975; Tawarmalani et al., 2002; Prokopyev et al., 2005, among others). To establish a convention for the ensuing discussion, we now denote an original optimal solution of (BIP) as $x^{(0)^*}$ in lieu of x^*. We first discuss how to obtain a single diverse and high-quality solution, and then elaborate on finding multiple such solutions.

3.1 Finding A Single Solution of High Quality, yet Diverse from $x^{(0)^*}$

Dinkelbach's Algorithm (Dinkelbach, 1967) can solve the fractional binary program in (8). It does so by solving a sequence of linearized problems that are related to the original nonlinear fractional programming problem. Algorithm 1 details our implementation, with $\mathcal{S}$ as in (BIP), some initial feasible solution x^0 (e.g., $x^{(0)^*}$), $\mathcal{F}$ as in (7), and $N(x)$ and $D(x)$ as in (8).

Algorithm 1 Dinkelbach: Find Diverse, High-Quality Solution via (8)

Input: Feasibility set $\mathcal{S}$, x^0, normalization factor $\mathcal{F}$, numerator $N(x)$, and denominator $D(x)$.
Output: Optimal solution x^k.

1: **DINKELBACH**($\mathcal{S}, x^0, \mathcal{F}, N(x), D(x)$)
2: Let $k \leftarrow 0$.
3: **repeat**
4:     Compute $\lambda^{(k+1)} \leftarrow \mathcal{F}\frac{N(x^k)}{D(x^k)}$, and let $k \leftarrow k+1$.
5:     Determine $x^k \leftarrow \underset{x\in\mathcal{S}}{\operatorname{argmax}} \left\{\mathcal{F}N(x) - \lambda^k D(x)\right\}$.
6: **until** $\mathcal{F}N(x^k) - \lambda^k D(x^k) = 0$.
7: **return** Optimal solution x^k.

Algorithm 1 operates by computing the value for parameter λ^k with respect to solution x^k in Step 4. It subsequently uses that value to solve the parametrized optimization problem in Step 5, returning x^k. When the parameter λ^k reaches a value such that the resulting x^k satisfies $\mathcal{F}N(x^k) - \lambda^k D(x^k) = 0$, Algorithm 1 terminates. Upon completion at iteration k, it returns an optimal solution x^k that maximizes the ratio of relative solution diversity to relative deterioration in objective function quality. While each subproblem in Step 5 is, in general, an $\mathcal{NP}$-hard problem, Dinkelbach's algorithm itself has been shown to have superlinear convergence (Schaible, 1976).

3.2 Finding Multiple Diverse Solutions of High Quality

While Algorithm 1 generates a single high-quality and diverse solution with respect to $x^{(0)^*}$, it may very well be desirable to identify a set $\mathcal{X}$ containing several high-quality and diverse solutions. We can construct such a set $\mathcal{X}$ by iteratively calling Algorithm 1 from Algorithm 2, which will be presented after having addressed some preliminary considerations.

Consider constructing $\mathcal{X}$ by successively generating solutions that exhibit both high quality and diversity. Beginning with $\mathcal{X}$ empty, we first add $x^{(0)^*}$ to $\mathcal{X}$, so that $\mathcal{X} = \left\{x^{(0)^*}\right\}$. An initial call to Algorithm 1 generates a solution that is optimal with respect to $\mathcal{S}$ and the fractional objective in (8); however, to repeat this process, two additional considerations must be taken into account.

First, whereas the previous diversity measure (4) measures the distance from a single binary vector, we are now interested in solutions that are diverse – not solely from $x^{(0)^*}$ – but also from all elements presently in $\mathcal{X}$. Consider this process for iteration $\ell > 1$. The diversity measure for each subsequent solution should reflect the relative distance from all elements contained in $\mathcal{X} = \{x^{(0)^*}, \ldots, x^{(\ell-1)^*}\}$; otherwise maximizing a distance metric such as (4) continues to emphasize movement away from only $x^{(0)^*}$. So we propose the following modified diversity metric that makes use of the centroid $\mathbf{c} = \left(\mathbf{c}_i = \frac{1}{\ell}\sum_{j=0}^{\ell-1} x_i^{(j)^*}, i = 1, \ldots, n\right)$ to capture collective movement away from all elements presently contained in $\mathcal{X}$:

$$\sum_{i=1}^{n} \{(1-\mathbf{c}_i)x_i + \mathbf{c}_i(1-x_i)\} = \sum_{i=1}^{n} \mathbf{c}_i + \sum_{i=1}^{n} (1-2\mathbf{c}_i)x_i. \tag{9}$$

Nonnegative expression (9) generalizes (4), as can be seen in the case of $\mathcal{X} = \{x^{(0)^*}\}$. It computes the average L_1 distance from the elements of $\mathcal{X}$, uniformly measuring diversity away from all elements of $\mathcal{X}$. It can also be seen as a voting scheme that places varying emphasis upon each x_i based upon the cumulative weighting over values of $x_i^{(j)^*}$ from previous iterations.

Second, to construct a set $\mathcal{X}$ of diverse solutions, we need to enforce that a unique solution, if one exists, is returned upon each call to Algorithm 1. That is, we want to ensure that previous solutions are not revisited on subsequent iterations. We can accomplish this by prohibiting previous solutions $x^{(\ell)^*}$ through the following inequality (Balas and Jeroslow, 1972):

$$\sum_{i: x_i^{(\ell)^*}=0} x_i + \sum_{i: x_i^{(\ell)^*}=1} (1-x_i) \geq 1. \tag{10}$$

We use (10) to define an updated feasibility set in Step 3 of Algorithm 2.

For iteration ℓ, we can also define normalization factor $\mathcal{F}_\ell$ as:

$$\mathcal{F}_\ell = \frac{\max\limits_{x \in \mathcal{S}_\ell} \{z^* - c^\top x\}}{\max\limits_{x \in \mathcal{S}_\ell} \left\{\sum_{i=1}^{n} \mathbf{c}_i + \sum_{i=1}^{n} (1-2\mathbf{c}_i)x_i\right\} + \epsilon}, \tag{11}$$

where we introduce some small constant $\epsilon > 0$ to prevent division by zero in the unusual case where there are no additional feasible solutions in $\mathcal{S}_\ell$ distinct from the element(s) of $\mathcal{X}$.

We note that the deterioration in the objective function as expressed in (1) remains a valid measure for every solution $x \in \mathcal{S}_\ell$, and so no adjustment is necessary to account for multiple solutions. Taking $N_\ell(x)$ as in (9), this leads to the following optimization problem:

$$\max_{x \in \mathcal{S}_\ell} \quad \mathcal{R}_\ell = \mathcal{F}_\ell \, \frac{N_\ell(x)}{D(x)} = \mathcal{F}_\ell \, \frac{\left(\sum_{i=1}^{n} \mathbf{c}_i + \sum_{i=1}^{n} (1-2\mathbf{c}_i)x_i\right)}{(z^* - c^\top x) + \epsilon}. \tag{12}$$

Given a finite positive integer $\mathcal{K} \geq 2$, Algorithm 2 will return a set $\mathcal{X}$ of the $\mathcal{K}$-best solutions, each of which is obtained by solving (12) in an iterative manner. Of course, it may occur that fewer than $\mathcal{K}$ solutions are returned, for example if $\mathcal{S}$ contains only a small number of feasible solutions.

Algorithm 2 Finding $\mathcal{K}$-best Solutions via Dinkelbach's Algorithm

Input: Feasibility set $\mathcal{S}$, $x^{(0)^*}$, z^*, and finite integer $\mathcal{K} \geq 2$.
Output: Set $\mathcal{X}$ of $\mathcal{K}$-best solutions (or fewer, if $\mathcal{S}$ contains fewer than $\mathcal{K}$ feasible solutions).

1: Initialize $\mathcal{X} \leftarrow x^{(0)^*}$, let $\ell = 0$, and let $\mathcal{S}_\ell \leftarrow \mathcal{S}$.
2: **repeat**
3: $\quad \mathcal{S}_{\ell+1} \leftarrow \mathcal{S}_\ell \cap \left\{ \sum_{i:x_i^{(\ell)^*}=0} x_i + \sum_{i:x_i^{(\ell)^*}=1} (1 - x_i) \geq 1 \right\}$.
4: $\quad \ell \leftarrow \ell + 1$.
5: $\quad$ **if** $\mathcal{S}_\ell = \emptyset$ **then**
6: $\quad\quad$ **return** Proof of infeasibility and (partial) set $\mathcal{X}$.
7: $\quad$ **else**
8: $\quad\quad$ Compute current normalization factor $\mathcal{F}_\ell$ based on $\mathcal{X}$ and (11).
9: $\quad\quad$ Derive current numerator $N_\ell(x)$ for (12) based on $\mathcal{X}$ and (9).
10: $\quad\quad$ Identify initial solution x^0 to pass into Algorithm 1.
11: $\quad\quad$ $x^{(\ell)^*} \leftarrow$ **DINKELBACH**$(\mathcal{S}_\ell, x^0, \mathcal{F}_\ell, N_\ell(x), D(x))$.
12: $\quad\quad$ Add optimal solution $x^{(\ell)^*}$ to $\mathcal{X}$, i.e., $\mathcal{X} \leftarrow \mathcal{X} \cup \{x^{(\ell)^*}\}$.
13: **until** $|\mathcal{X}| = \mathcal{K}$.
14: **return** $\mathcal{X}$.

We compute the collective diversity of solutions in $\mathcal{X}$ along the lines of Danna and Woodruff (2009) and Prokopyev et al. (2009), that is:

$$D_{bin}(\mathcal{X}) = \frac{2}{n|\mathcal{X}|(|\mathcal{X}|-1)} \sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{X}|-1} \sum_{\ell=j+1}^{|\mathcal{X}|} \left| x_i^{(j)^*} - x_i^{(\ell)^*} \right|. \tag{13}$$

We use (13) to evaluate the collective diversity of $\mathcal{X}$ after its return from Algorithm 2.

3.3 Additional Algorithmic Considerations

We next discuss an enhancement to our approach, as well as introduce a second diversity metric for purposes of computational comparisons. The enhancement reduces the total number of iterations in every call to Algorithm 1, which in turn improves the overall computational performance of Algorithm 2. The additional diversity metric measures the minimum distance from all previously identified solutions in $\mathcal{X}$, and enables a comparison of distance metrics in Section 4.

Step 10 of Algorithm 2 requires an initial solution to be passed to Algorithm 1. Some candidates are feasible solutions from previously solved integer programs, such as those arising from the integer programs in (11), as well intermediate solutions from Step 5 of Algorithm 1. We propose to maintain a list $\mathcal{L}$ of such solutions. This choice is important for the overall efficiency of Algorithm 2, because every call to Algorithm 1 results in an integer program being solved in Step 5 for each iteration. For each iteration ℓ, in Step 10 of Algorithm 2 we choose $x^0 \in \underset{x \in \mathcal{L}}{\operatorname{argmax}} \left\{ \mathcal{F}_\ell \frac{N_\ell(x)}{D(x)} \right\}$. By reducing the overall number of iterations per call to Algorithm 1, significant computational savings can be obtained in the performance of Algorithm 2.

An alternate way to evaluate diversity is to measure the minimum distance from previously identified solutions in $\mathcal{X}$:

$$\min_{j=0,\ldots,\ell-1} \left\{ \sum_{i:x_i^{(j)*}=0} x_i + \sum_{i:x_i^{(j)*}=1} (1 - x_i) \right\}. \tag{14}$$

Metric (14) can replace diversity measure (9) in Algorithm 2, and can also be maximized over all $x \in \mathcal{S}_\ell$ to find a most diverse element:

$$\max_{x \in \mathcal{S}_\ell} \mathcal{R}_\ell = \mathcal{F}_\ell \, \frac{N_\ell(x)}{D(x)} = \mathcal{F}_\ell \, \frac{\left(\min\limits_{j=0,\ldots,\ell-1} \left\{ \sum\limits_{i:x_i^{(j)*}=0} x_i + \sum\limits_{i:x_i^{(j)*}=1} (1 - x_i) \right\} \right)}{(z^* - c^\top x) + \epsilon}. \tag{15}$$

The expression in (14) is nonlinear but has a straightforward linearization, namely, by adding a single unrestricted variable q which is maximized over all $x \in \mathcal{S}_\ell$, and having q lower bound, for every element of $\mathcal{X}$, the inner diversity metric in (14) (i.e., expression (4)). Thus, for every iteration of Algorithm 2, one additional constraint is added to $\mathcal{S}_\ell$ to ensure this relationship. We use this second diversity metric in our computational comparisons in Section 4.

4. Computational Discussions

We begin this section by comparing and contrasting relevant computational experiments, and subsequently discuss the computational performance of Algorithm 2.

4.1 Comparing and Contrasting Related Methods in the Literature

Two of the most relevant studies to our work are Danna et al. (2007) and Danna and Woodruff (2009). The main emphasis of Danna et al. (2007) is finding high-quality solutions to mixed-integer programs in an efficient manner; diversity is only tangentially discussed in Algorithm 5. However, diversity is a central focus in Danna and Woodruff (2009), where after pre-populating a set S of high-quality solutions, the authors use both exact and heuristic approaches to identify a subset of p solutions that are of maximum collective diversity. Similarly, our approach implements exact approaches to identify a set of several high-quality and diverse solutions. Whereas Danna et al. (2007) and Danna and Woodruff (2009) operate on mixed-integer programs, at present we concern ourselves with only binary integer programs.

Both aforementioned studies use a proprietary method (namely, the *one-tree* algorithm (IBM, 2012)) to identify a large set S containing solutions within $q\%$ of the optimum. While a threshold of 1% is reasonable, it is also somewhat subjective, potentially excluding a valuable solution having much higher diversity lying just outside the quality threshold. Our approach, on the other hand, sequentially generates the most attractive high-quality and diverse solutions that maximize ratio $\mathcal{R}_\ell$ in (12) or (15). Further, our approach is free to operate in conjunction with any integer programming solver, and is likewise free of the up-front requirements to pre-identify and store potentially large amounts of information in memory, a significant limitation in the computational findings of Danna and Woodruff (2009). Finally, our approach explicitly includes an actual optimal solution of (BIP). Depending upon the problem context and corresponding data accuracy, this may be of critical importance to decision makers, e.g., some combinatorial optimization problems such as the traveling salesman problem can have precise data.

We further compare our approach to two more conventional methods, in addition to that of Danna and Woodruff (2009). The first approach is a common multi-objective framework that simultaneously handles two objectives, here the competing (normalized) objectives of quality and diversity, which we refer to as the *multiobjective* method. Specifically, we implement this method by constructing a merged objective, to be maximized, with an additional parameter α on the (centroid) quality metric ($c^\top x$ for maximization and $-c^\top x$ for minimization) together with $(1-\alpha)$ on the (centroid) diversity metric as expressed in (9). The second approach is a scheme of randomly perturbing the original objective function coefficients c (see, e.g., Ben-Tal and Nemirovski, 2000), which we will refer to as the *perturbation* method. This is a technique that may commonly be employed in practice to identify solutions that are in the vicinity of the optimal solution and, ideally, of relatively high-quality. As per Ben-Tal and Nemirovski (2000), we perturb each c_i objective coefficient as $\tilde{c}_i = (1 + \varepsilon\xi_i)c_i$, where $\varepsilon > 0$ is a given uncertainty level, and uniformly distribute each $\xi_i \in [-1, 1]$.

4.2 Computational Setup

In advance of discussing our computational findings, we next detail the environment and test sets used to evaluate our algorithmic approaches.

4.2.1 Computational Environment

We developed our algorithms in C++ and compiled the source code using g++ version 4.4.6 20110731 (Red Hat 4.4.6-3) on a Dell R610 server with 2 Intel Xeon X5690 CPUs each with 6 cores running at 3.47 GHZ and 48GB of RAM. We used the callable library of IBM ILOG CPLEX 12.4 (IBM, 2012) to perform the optimization, and implemented a time limit of one hour for the solution to any binary integer program (e.g., Step 8 of Algorithm 2; Step 5 of Algorithm 1). Given the large potential numerator to denominator ratios for our objective function, we set the CPX_NUMERICAL_EMPHASIS parameter to CPX_ON to prioritize numerical stability. Finally, analogous to Danna and Woodruff (2009), we set our algorithm to retrieve $\mathcal{K} = 10$ solutions.

4.2.2 Computational Test Sets

We considered two test sets. The first set, comprised of seven basic multiple knapsack instances (Petersen, 1967; Beasley, 2015), was used to evaluate our approach and compare it with the more conventional *multiobjective* and *perturbation* methods described in Section 4.1. Results from the first test set motivate subsequent comparisons with the more sophisticated method of Danna and Woodruff (2009) on a secondary test set that consisted of 72 more difficult pure binary integer programming instances from MIPLIB 2003 (Achterberg et al., 2006) and MIPLIB 2010 (Koch et al., 2011) having an "easy" categorization and nonempty feasible regions. Notably, twelve of the MIPLIB 2003 instances coincided with those already appearing in Danna and Woodruff (2009).

4.3 Computational Results on First Test Set

Table 1 displays computational findings of Algorithm 2 applied to the first test set of seven multiple knapsack instances from Beasley (2015). The column "Instance" indicates the problem, while the column "$m \times n$" indicates the number of rows and columns. The column "Mean Iterations" reports the average number of iterations for a given call to Algorithm 1 using diversity measures (9) and (14), while the column $D_{bin}(\mathcal{X})$ reports the value of (13) using diversity measures (9) and (14). Our approach identified $|\mathcal{X}| = 10$ solutions to all seven multiple knapsack instances using each of diversity measures (9) and (14). For (9), the mean number of iterations was 2.5, the mean runtime for the algorithm was 95 seconds, and the mean value of $D_{bin}(\mathcal{X})$ was 25.4%, while for measure (14) these values were 2.9 iterations, 94 seconds, and 27.3%, respectively.

We now consider the performance of more conventional *multiobjective* and *perturbation* methods on these seven mknap instances. For the *multiobjective* method, we conducted nine runs, one for each level of the parameter α in $[0.1, 0.2, \ldots, 0.9]$, thereby emphasizing either diversity (low α) or

Table 1: Algorithm 2 on Multidimensional Knapsack Problems (Beasley, 2015)

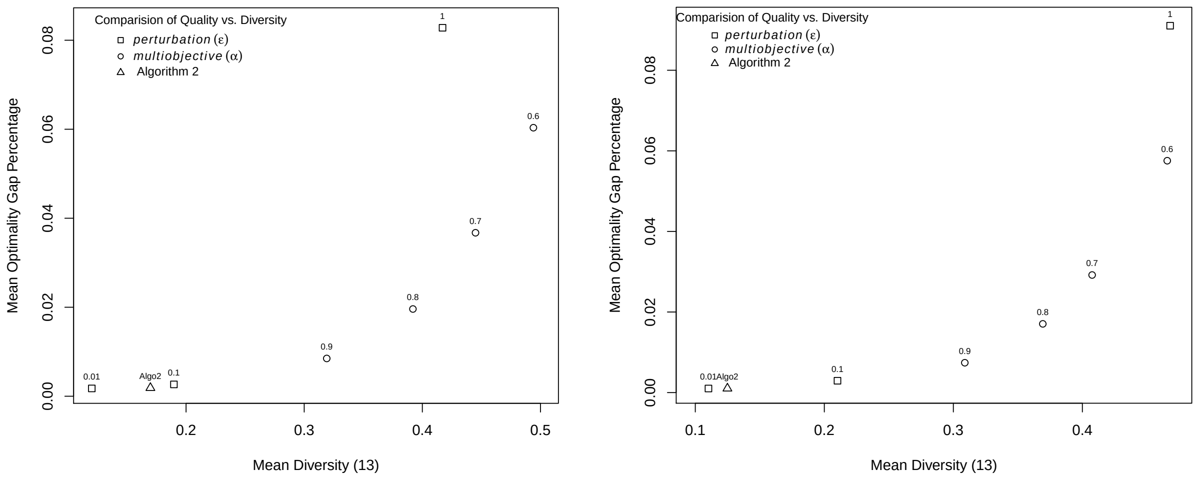
			$\|\mathcal{X}\|$		Mean Iterations		Time (seconds)		$D_{bin}(\mathcal{X})$	
Instance	m	n	(9)	(14)	(9)	(14)	(9)	(14)	(9)	(14)
mknap_1	10	6	10	10	2.3	2.7	<1	<1	0.4556	0.4556
mknap_2	10	10	10	10	2.3	2.6	<1	6	0.3622	0.4044
mknap_3	10	15	10	10	2.4	2.6	143	26	0.2548	0.2548
mknap_4	10	20	10	10	2.6	2.8	127	134	0.2456	0.2556
mknap_5	10	28	10	10	2.9	2.6	169	149	0.1643	0.1579
mknap_6	5	39	10	10	2.6	3.1	94	145	0.1698	0.2285
mknap_7	5	50	10	10	2.3	3.7	128	199	0.1249	0.1587

quality (high α). As for the *perturbation* method, we conducted three runs, one for each value of ε suggested in Ben-Tal and Nemirovski (2000), i.e. $\varepsilon \in [0.01, 0.1, 1]$, effectively causing each run to experience increased levels of perturbation in the objective coefficients c_i. We compared the results of these $9 + 3 = 12$ approaches with the approach of our Algorithm 2 in terms of both quality (mean optimality gap percentage) and diversity (as expressed in (13)).

Figure 1 plots the performance for the two largest mknap instances, mknap_6 and mknap_7, in terms of quality versus diversity. Note that lower values in the y-dimension are preferable, because smaller distances from z^* (i.e., the gap) indicate superior quality; whereas for the collective diversity displayed on the x-axis, larger values are more attractive. Thus the lower right corner of Figures 1a and 1b are more desirable.

Both plots demonstrate a noticeable, classical trend illustrating a frontier of nondominated solutions, with the only outlier being the dominated solution corresponding to $\varepsilon = 1$ in the mknap_6 instance. Intuitively, this finding is likely the result of large random perturbations of the c_i coefficients adversely influence the quality outcome. For the *multiobjective* method, we chose only to display the values of $\alpha \in [0.6, 0.7, 0.8, 0.9]$, as the other values of α continue the observable trend of deterioration in quality. Moreover, smaller values of α are not meaningful because we are, at the core, solving an optimization problem with an objective function that presumably has an actual interpretation in practice.

As can be seen from Figures 1a and 1b, with respect to any run of either of the *multiobjective* and *perturbation* methods, the performance of Algorithm 2 (triangle symbol) is nondominated. In particular, for the obtained level of quality, no other run can achieve as high a level of diversity, and vice versa. This behavior was consistent not only for the two largest instances mknap instances in Figure 1, but across all seven, as well as approximately a dozen MIPLIB instances chosen from the second test set. What is also clear is that our approach tends to emphasize quality over diversity,



(a) mknap_6 Instance

(b) mknap_7 Instance

Figure 1: Contrasting Quality and Diversity: Algorithm 2 versus *multiobjective* and *perturbation* Methods on Two Larger mknap Instances

which is inherently due to the denominator in (12) that, given the choice, favors identifying multiple optimal solutions contributing no deterioration in the objective.

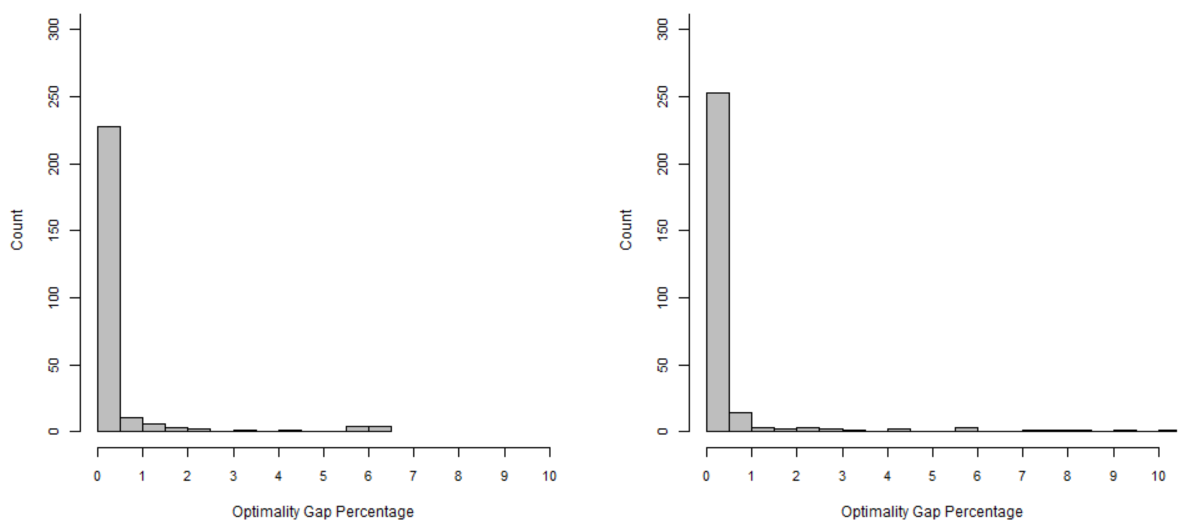
It is also important to keep in mind that these comparisons were somewhat uneven – our results were obtained with no additional weighting of the competing quality and diversity metrics, which is the purpose of α, as well as, for all practical purposes, ε. In fact we have discussed only the essential components of our approach. In particular, it lacks the additional weighting schemes present in the *multiobjective* and *perturbation* methods, which we feel could only improve our results, an idea we revisit as a possible future extension in Section 5. On the other hand, fine-tuning is also a limitation of any method that relies on such parameters (e.g., α and ε from, respectively, the *multiobjective* and *perturbation* methods). In any case, we are confident in these initial findings, because in the context of optimization, clearly the argument for quality is stronger than the argument for diversity, and our research supports this fact.

Hence, given the performance of our approach, we next compared it to Danna and Woodruff (2009) on a second, larger and more challenging test set of instances from MIPLIB.

4.4 Computational Results on Second Test Set

Table 2 displays the results of our approach, as well as that of Danna and Woodruff (2009)[‡], on the second test set consisting of more challenging MIPLIB instances. The layout is similar to that of Table 1, with additional information in the first column concerning the origin of the instance, i.e., whether it came from MIPLIB 2003 or MIPLIB 2010†. Of the 72 total MIPLIB test instances in the second test set, Table 2 displays the 40 instances for which $|\mathcal{X}| \geq 2$ for at least one of diversity measures (9) and (14). Of these, there were 29 MIPLIB instances for which we were able to obtain $|\mathcal{X}| = 10$ solutions using both diversity measures (9) and (14). For (9), the mean number of iterations was 2.3, the mean runtime was 4,995 seconds, and the mean value of $D_{bin}(\mathcal{X})$ was 7.1%, whereas for measure (14) these values were 2.6 iterations, 6,010 seconds, and 7.3%, respectively.

We observe some interesting and consistent behavior from these aggregate results. It appears that diversity measure (9) is able to find the full number of solutions (ten), on average, with slightly fewer iterations and in less overall time (though all of these instances completed in under 12 hours for either of the two metrics). With respect to the overall diversity of the set $|\mathcal{X}|$ using (13), metric (14) slightly outperformed (9) across both test sets. Thus both measures appear to have their respective merits.



(a) 270 Alternate Solutions Using (9)

(b) 270 Alternate Solutions Using (14)

Figure 2: Solution Quality – Highlighting Optimality Gap of 270 Alternate Solutions from 30 MIPLIB Instances for which $|\mathcal{X}| = 10$

Further investigations into the instances for which we found a full ten solutions yield some intriguing insights. For diversity measure (9), of the 270 total alternative solutions (i.e., not x^*) returned from Algorithm 2, over 56% were additional optimal solutions (i.e., no gap in optimality). The distribution of these solutions with respect to their distance from optimality are displayed in

Table 2: Algorithm 2 on BIP Instances from MIPLIB (Achterberg et al., 2006; Koch et al., 2011)

			$\lvert\mathcal{X}\rvert$		Mean It.		Time (sec)		$D_{bin}(\mathcal{X})$		
Instance	m	n	(9)	(14)	(9)	(14)	(9)	(14)	(9)	(14)	DW‡
acc-tight5†	3,052	1,339	10	10	2.0	2.0	15,065	21,616	0.1431	0.1416	
acc-tight6†	3,047	1,335	10	10	1.0	1.0	10,150	11,734	0.1512	0.1478	
air03	124	10,757	10	10	2.7	3.0	67	210	0.0006	0.0009	
air04	823	8,904	10	10	2.7	2.8	23,188	5,867	0.0029	0.0028	
air05	426	7,195	10	10	2.9	3.9	1,148	3,666	0.0052	0.0070	
bley_xl1†	175,620	5,831	10	10	2.3	2.7	1,136	1,381	0.0064	0.0056	
cap6000	2,176	6,000	10	10	3.1	3.0	253	452	0.0022	0.0022	0.0088
eil33-2†	32	4,516	10	10	2.2	2.8	1,593	4,212	0.0018	0.0020	
eilB101†	100	2,818	10	10	3.1	3.6	17,863	24,637	0.0082	0.0096	
l152lav	97	1,989	10	10	2.2	2.6	198	426	0.0112	0.0121	0.0225
lseu	28	89	10	10	2.6	2.8	198	373	0.1333	0.1498	
mine-166-5†	8,429	830	10	10	3.0	4.1	8,202	14,522	0.0052	0.0056	
mitre	2,054	10,724	10	10	2.3	2.3	24	63	0.0037	0.0037	
mod008	6	319	10	10	2.3	2.7	67	932	0.0145	0.0170	0.0220
mod010	146	2,655	10	10	2.3	2.3	147	198	0.0059	0.0059	0.0230
neos-777800†	479	6,400	10	10	1.0	1.0	383	88	0.0250	0.0250	
neos-957389†	5,115	6,036	10	10	2.0	2.0	72	89	0.0250	0.0249	
ns1688347†	4,191	2,685	10	10	2.1	2.1	12,298	16,303	0.0320	0.0294	
nw04	36	87,482	10	10	2.6	3.1	171	432	0.0001	0.0001	0.0001
opm2-z7-s2†	31,798	2,023	10	10	2.6	2.9	12,952	19,398	0.0375	0.0797	
p0033	16	33	10	10	2.2	2.2	74	371	0.1778	0.1758	0.2707
p0201	133	201	10	10	2.3	2.6	110	355	0.1612	0.1650	0.1705
p0282	241	282	10	10	3.0	3.4	251	1,253	0.0072	0.0071	0.1118
p0548	176	548	10	10	2.6	2.6	444	966	0.0371	0.0360	0.0587
p2756	755	2,756	10	10	2.7	2.6	389	1,436	0.0627	0.0624	
reblock67†	2,523	670	10	10	3.6	3.6	37,656	42,174	0.0051	0.0110	
stein27	118	27	10	10	1.0	1.0	35	105	0.4889	0.4840	0.4889
stein45	331	45	10	10	1.6	1.9	585	805	0.4840	0.4602	0.4840
tanglegram2†	8,980	4,714	10	10	2.0	2.3	123	239	0.0318	0.0306	
neos-1440225†	330	1,285	6	10	1.0	1.0	12,802	12,115	0.0546	0.0538	
ex9†	40,962	10,404	8	8	1.0	1.0	89	91	0.0134	0.0134	
tanglegram1†	68,342	34,759	10	5	2.2	2.8	32,073	29,161	0.0114	0.0119	
neos808444†	18,329	19,846	8	3	1.0	1.0	16,571	7,354	0.0499	0.0508	
acc-tight4†	3,285	1,620	6	3	1.0	1.5	12,712	5,159	0.1686	0.1778	
enigma	21	100	4	4	2.0	2.0	45	138	0.1133	0.1133	
neos18†	11,402	3,312	3	2	2.5	3.0	7,782	5,240	0.1222	0.1437	
neos-1109824†	28,979	1,520	–	–	–	–	–	–	–	–	
neos-941313†	13,189	167,910	2	–	3.0	–	6,860	–	0.0081	–	
ns894788†	2,279	3,463	–	2	–	1.0	–	7,317	–	0.0684	

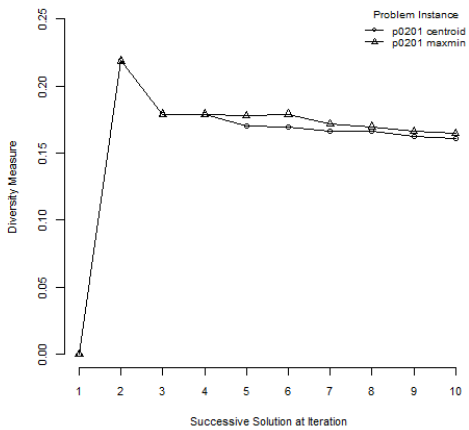
Figure 2(a). A similar distribution for measure (14) appears in Figure 2(b) depicting 151 optimal solutions of the 270 total alternate solutions (greater than 55%). From these figures we can see that, while a majority of solutions appear within 1% of optimal, roughly a tenth of all alternative solutions were outside of that range.

Figure 3 illustrates the progression of diversity for metrics (9) and (14) as the set $\mathcal{X}$ is constructed for several instances. For each successive solution on the y-axis, it displays the corresponding values of metric (13). The first, Figure 3(a), exemplifies most instances, in that the diversities obtained with either of metrics (9) or (14) track in a similar manner. However, there are several instances for which this differs. Figures 3(b) and 3(c) demonstrate a few of the instances where metric (9) outperforms (14), while Figure 3(d) illustrates a more exceptional case where (14) outperforms (9). Note that in each of the instances, both diversity measures identify the same first alternate solution (i.e., $x^{(1)^*}$).

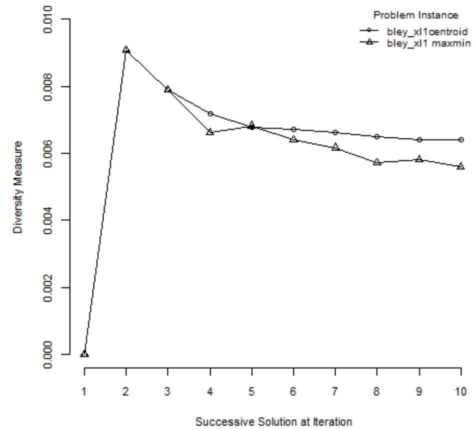
4.4.1 Computational Results: Comparison with Danna and Woodruff (2009)

In our opinion, the computational runtimes reported in Table 2 to generate a small but reasonably-sized set of $\mathcal{K}$ solutions are modest. For most instances we obtained cumulative runtimes in well-under half an hour, while for others (mostly from the more challenging MIPLIB2010) the runtimes were somewhat longer. As depicted in Figure 2, a vast majority of the solutions to individual MIPLIB test instances were very near optimal (i.e. within 0.5%, and included a number of multiple optimal instances). As an example, solving the stein27 instance using (9) took less than a minute to identify a set of ten solutions, all optima, that had a collective diversity value of 0.4889. This diversity value matched that obtained by Danna and Woodruff (2009) using a heuristic local search. Only a few instances exhibited solutions with optimality gaps greater than 5%, illustrating that some solutions contribute to the fractional objective by emphasizing diversity.

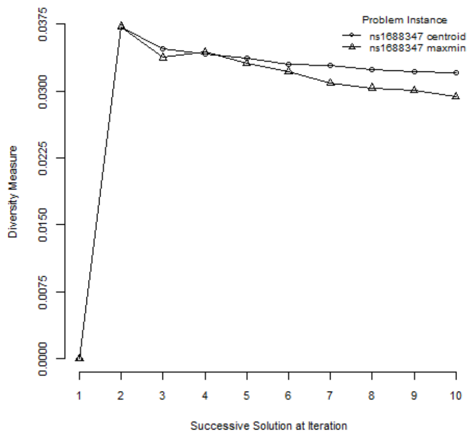
In Danna and Woodruff (2009), the task of finding high-quality and diverse solutions is separated into two components: first running the one-tree algorithm for one hour to identify a large set of high-quality solutions (within 1% of optimal), and subsequently finding diverse solutions in this extensive set using either a distinct integer program or via heuristic methods. This contrasts with our approach to simultaneously generate solutions on the fly, and appears to be one of its main advantages. Indeed, for eleven of the twelve MIPLIB 2003 instances we had solved in common, we identified ten solutions having competitive values for $D_{bin}(\mathcal{X})$ and averaging just over three minutes in runtime for diversity measure (9), and in just over nine minutes for (14). The longest runtime for these instances was cap6000 using diversity measure (14), lasting almost 30 minutes. In



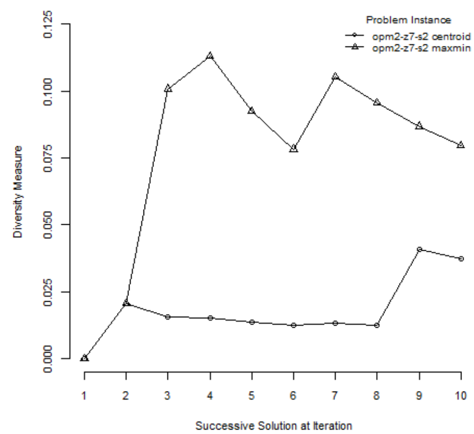
(a) Progression for p0201 instance



(b) Progression for bley_xl1 instance



(c) Progression for ns1688347 instance



(d) Progression for opm2-z7-s2 instance

Figure 3: Progression in Solution Diversity in Successive Iterations of Algorithm 2 Using Metrics (9) and (14)

contrast, for each of the cap6000, l152lav, mod010, nw04, p0282, p0548, and stein27 instances, the exact algorithm of Danna and Woodruff (2009) did not finish within ten days (although they did obtain superior results in terms of $D_{bin}(\mathcal{X})$ with various heuristic approaches). There was only one instance (disctom) which our approach could not solve that Danna and Woodruff (2009) solved.

Thus, while our algorithm takes less time to complete on the eleven solved MIPLIB instances that we had in common, this appears to come at the expense of some solution diversity, a strength of Danna and Woodruff (2009) given that they explicitly optimize for diversity over a set S of already identified high-quality solutions. Although on three of these instances (cap6000, mod010, p0282) our approach did not compare well with the heuristic results of Danna and Woodruff (2009), we did obtain respectable diversity results in the remaining eight instances. One factor likely contributing to this observation is that, by construction, our approach requires an optimal solution $x^{(0)^*}$ to be included into the solution set $\mathcal{X}$, whereas no such restriction is required in the approaches of Danna and Woodruff (2009). Also, as touched upon in Section 4.3, our method tends to prefer multiple optimal solutions, as in the denominator of the ratio they contribute no deterioration in the objective. Given these tradeoffs in solution time and overall diversity, it may be best to regard our methodology and that of Danna and Woodruff (2009) as complementary.

5. Conclusions

We present a new approach to generate multiple high-quality yet diverse solutions to pure binary (0–1) integer programs. Our method simultaneously obtains such solutions to pure binary program (BIP) with a modified, fractional objective over the same feasible region. Given an optimal solution x^* and optimal objective function value z^* to (BIP), we propose two measures of diversity from x^* for the numerator, while the denominator measures the deterioration in objective quality from the optimal value z^*. Our algorithmic approach uses an implementation of Dinkelbach's Algorithm (Dinkelbach, 1967) to handle the fractional objective and sequentially generates multiple high-quality and diverse solutions. Computational experiments indicate that the method is efficient given the rather modest runtimes on a variety of test instances from the literature. We also note that our approach is independent of the integer programming solver, and thus could be coupled with, for example, an open source solver such as CBC (Lougee-Heimer, 2003). Additionally, the source has been implemented in Python and submitted to the open source software library Pyomo (Python Optimization Modeling Objects), which includes a flexible framework for applying optimizers to analyze optimization models (Hart et al., 2011).

It is worth mentioning that certain nonlinear binary integer programs may also be amenable to the proposed methodology, for example if such functions can be represented in a piecewise-linear fashion using binary variables (see, e.g., Vielma et al., 2010). Furthermore, our methodology is not limited to calling an IP solver, per se; a call to any blackbox solver or solution approach could be incorporated in Step 5 of Algorithm 1.

Future extensions include allowing weights on the numerator and denominator to promote either greater diversity or quality; indeed, it would be interesting to view the tradeoffs in quality as the diversity influence is increased. Additionally, it may be of interest in the diversification process to weight individual subsets of variables from which subsequent solutions should be particularly far removed, for example, a group of 0–1 variables on which many other model decisions hinge. Still other diversity measures could be incorporated into the numerator of (12). Another avenue of research is extending the problem setting to integer and mixed integer cases, though as noted in both Danna et al. (2007) and Danna and Woodruff (2009) expressing diversity involving a set of continuous variables can be challenging.

6. Acknowledgments

The authors are grateful to R. Killea for assistance in implementing this methodology in Python for subsequent submission to the open source software library Coopr.

References

Achterberg, T., Koch, T., and Martin, A. MIPLIB 2003. *Operations Research Letters*, 34(4): 361–372, 2006.

Balas, E. and Jeroslow, R. Canonical cuts on the unit hypercube. *SIAM Journal on Applied Mathematics*, 23(1):61–69, 1972.

Beasley, J. E. Multidimensional knapsack test problems. `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html`, 2015.

Ben-Tal, A. and Nemirovski, A. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000.

Camm, J. D. ASP, the art and science of practice: A (very) short course in suboptimization. *Interfaces*, 44(4):428–431, 2014.

Danna, E., Fenelon, M., Gu, Z., and Wunderling, R. Generating multiple solutions for mixed integer programming problems. In Fischetti, M. and Williamson, D., editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 280–294. Springer Berlin Heidelberg, 2007.

Danna, E. and Woodruff, D. L. How to select a small set of diverse solutions to mixed integer programming problems. *Operations Research Letters*, 37(4):255–260, 2009.

Deb, K. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

Dinkelbach, W. On non-linear fractional programming. *Management Science*, 13(7):492–498, 1967.

Fagerholt, K., Korsvik, J. E., and Løkketangen, A. Ship routing scheduling with persistence and distance objectives. In Nunen, J. A. E. E., Speranza, M. G., and Bertazzi, L., editors, *Innovations in Distribution Logistics*, volume 619 of *Lecture Notes in Economics and Mathematical Systems*, pages 89–107. Springer Berlin Heidelberg, 2009.

Fischetti, M., Glover, F., and Lodi, A. The feasibility pump. *Mathematical Programming*, 104(1): 91–104, 2005.

Fonseca, C. M. and Fleming, P. J. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423. Morgan Kaufmann Publishers Inc., 1993.

Glover, F., Løkketangen, A., and Woodruff, D. L. Scatter search to generate diverse MIP solutions. In Laguna, M. and González-Velarde, J. L., editors, *OR Computing Tools for Modeling, Optimization and Simulation*, volume 12 of *Interfaces in Computer Science and Operations Research*, pages 299–317. Kluwer Academic Publishers, 2000.

Greistorfer, P., Løkketangen, A., Voß, S., and Woodruff, D. L. Experiments concerning sequential versus simultaneous maximization of objective function and distance. *Journal of Heuristics*, 14 (6):613–625, 2008.

Grodzevich, O. and Romanko, O. Normalization and other topics in multiobjective optimization. In *Proceedings of the First Fields-MITACS Industrial Problems Workshop*, pages 89–102. The Fields Institute, 2006.

Gunawan, S., Farhang-Mehr, A., and Azarm, S. Multi-level multi-objective genetic algorithm using entropy to preserve diversity. In Fonseca, C., Fleming, P., Zitzler, E., Thiele, L., and Deb, K., editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 148–161. Springer Berlin Heidelberg, 2003.

Hart, W. E., Watson, J.-P., and Woodruff, D. L. Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

Hebrard, E., Hnich, B., O'Sullivan, B., and Walsh, T. Finding diverse and similar solutions in constraint programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 372–377. American Association for Artificial Intelligence, 2005.

Hoch, S. J. and Schkade, D. A. A psychological approach to decision support systems. *Management Science*, 42(1):51–64, 1996.

IBM. *IBM ILOG CPLEX 12.4 User's Manual*. IBM ILOG CPLEX Division, Incline Village, NV, 2012.

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A. M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D. E., and Wolter, K. MIPLIB 2010. *Mathematical Programming Computation*, 3(2): 103–163, 2011.

Laguna, M., Gortázar, F., Gallego, M., Duarte, A., and Martí, R. A black-box scatter search for optimization problems with integer variables. Technical report, Universitat de Valéncia, Spain, 2012.

Løkketangen, A. and Woodruff, D. L. A distance function to support optimized selection decisions. *Decision Support Systems*, 39(3):345–354, 2005.

Lougee-Heimer, R. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

Margot, F. Symmetry in integer linear programming. In Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., editors, *50 Years of Integer Programming 1958–2008*, pages 647–686. Springer Berlin Heidelberg, 2010.

Markowitz, H. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

Martí, R., Gallego, M., Duarte, A., and Pardo, E. G. Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics*, pages 1–25, 2011.

Martos, B., Whinston, A., and Whinston, V. Hyperbolic programming. *Naval Research Logistics Quarterly*, 11(2):135–155, 1964.

Masin, M. and Bukchin, Y. Diversity maximization approach for multiobjective optimization. *Operations Research*, 56(2):411–424, 2008.

Miettinen, K. *Nonlinear Multiobjective Optimization*. Springer, 1999.

Murata, T., Ishibuchi, H., and Gen, M. Specification of genetic search directions in cellular multi-objective genetic algorithms. In Zitzler, E., Thiele, L., Deb, K., Coello Coello, C., and Corne, D., editors, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 82–95. Springer Berlin Heidelberg, 2001.

Murthy, S., Akkiraju, R., Goodwin, R., Keskinocak, P., Rachlin, J., Wu, F., Yeh, J., Fuhrer, R., Kumaran, S., Aggarwal, A., Sturzenbecker, M., Jayaraman, R., and Daigle, R. Cooperative multiobjective decision support for the paper industry. *Interfaces*, 29(5):5–30, 1999.

Nemhauser, G. L. and Wolsey, L. A. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.

Petersen, C. C. Computational experience with variants of the Balas algorithm applied to the selection of R&D projects. *Management Science*, 13(9):736–750, 1967.

Prokopyev, O. A., Huang, H. X., and Pardalos, P. M. On complexity of unconstrained hyperbolic 0–1 programming problems. *Operations Research Letters*, 33(3):312–318, 2005.

Prokopyev, O. A., Kong, N., and Martinez-Torres, D. L. The equitable dispersion problem. *European Journal of Operational Research*, 197(1):59–67, 2009.

Saipe, A. L. Solving a (0, 1) hyperbolic program by branch and bound. *Naval Research Logistics Quarterly*, 22(3):497–515, 1975.

Schaible, S. Fractional programming. II, on Dinkelbach's algorithm. *Management Science*, 22(8): 868–873, 1976.

Schittekat, P. and Sörensen, K. Supporting 3PL decisions in the automotive industry by generating diverse solutions to a large-scale location-routing problem. *Operations Research*, 57(5):1058–1067, 2009.

Schrijver, A. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer, 2003.

Sharda, R., Barr, S. H., and McDonnell, J. C. Decision support system effectiveness: A review and an empirical test. *Management Science*, 34(2):139–159, 1988.

Takriti, S., Birge, J. R., and Long, E. A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, 11(3):1497–1508, 1996.

Tawarmalani, M., Ahmed, S., and Sahinidis, N. V. Global optimization of 0–1 hyperbolic programs. *Journal of Global Optimization*, 24(4):385–416, 2002.

Vielma, J. P., Ahmed, S., and Nemhauser, G. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.

Williams, H. P. *Model Building in Mathematical Programming*, volume 4. Wiley, 1999.

Zitzler, E., Laumanns, M., and Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), 2001.