Scatter Search to Generate Diverse MIP Solutions

Fred Glover
School of Business, CB 419, Univ. Colorado, Boulder CO 80309 USA

Arne Løkketangen
Molde College, Britvn. 2, 6411 Molde Norway

David L. Woodruff
Graduate School of Management, U.C. Davis, Davis CA 95616 USA

Key words: Optimization, Heuristics, Tabu Search, Diversity Metrics, Chunking

Abstract: An objective function often is only a rough approximation of the actual goals of the organization and its stakeholders. Consequently, an optimal solution may be no more interesting than other solutions that provide good values. Beyond this, a set of "good" solutions whose members have diverse characteristics can be significantly more valuable for practical analysis and planning than any single, "best" solution. Yet the challenge of systematically uncovering such a *diverse set* of solutions, or even postulating what its defining features may be, has been conspicuously neglected. We address this challenge for 0-1 mixed integer programming problems and provide demonstrations of the computational efficiency of our approach.

Published in: *OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, Eds. M. Laguna and J.L. González-Velarde, Kluwer Academic Publishers, 2000, pp. 299 – 317.

1. INTRODUCTION

Mathematical programming problems with some or all variables constrained to take on zero-one values are important in a large number of applications. Many critical strategic and tactical decisions such as which product lines to develop, which factories to open and close, which projects to invest in, which sites to choose for locating new facilities, and so forth, all directly or indirectly require reference to binary variables. In order to make use of computers to help find good solutions for such problems, models that incorporate discrete-valued variables must be developed.

Such models must always be based on simplifying assumptions. The modeler often, by necessity, leaves out details that are hard to express mathematically or details that will render the problem too hard to solve. An important assumption often made is that costs and constraints can be expressed as linear functions of the decision variables. This enables the use of powerful algorithms to help solve the problem. In fact, the modeling power of zero-one variables enables non-linear function to be approximated to great degree of accuracy using such linear functions.

The abstract formulation for linear problems with binary variables takes as data a row vector c of length n, an $m \times n$ matrix T, a column vector b of length m and lower and upper bound vectors l and u of length n. We assume that the data has been scaled so that the values in the vector l are near zero and the values in u are near one. The goal is to select a column vector, x of length n to solve the problem.

$$\text{(P)} \quad \text{Minimize} \quad cx$$

subject to

$$Tx \geq b \tag{1}$$

$$x_i \in \{0, 1\} \quad i \in I \tag{2}$$

$$l \leq x \leq u \tag{3}$$

where the index set I gives the variables that must take on integer values. We refer to solutions that satisfy the constraints, except perhaps constraints (2) as *LP feasible*, and to solutions that satisfy all constraints, including constraints (2), as *MIP feasible* or as *MIP solutions*.

Our goal in this paper is to characterize and find a diverse set of solutions to mixed integer programming problems. There are a number of reasons why decision makers are interested in seeing a variety of solutions to (P), and not just those that achieve the minimum. MIP solutions are sometimes used as starting points for more detailed models (e.g. simulations). Such automated, but stochastic, optimization systems often benefit from a variety of starting

points. In the application area of decision support systems (DSS), practical mixed integer programming models typically leave out a lot of details and make use of very approximate data [9, 12]. The user of a DSS would often like to look at a variety of solutions and mull over the so-called intangibles omitted from the model.

These examples do not represent the same concerns as sensitivity analysis, where one considers the effect on the optimal solution of parametric changes in the data. The rich literature on MIP sensitivity analysis, which is the subject of an extensive annotated bibliography developed and maintained by Greenberg [8], discloses that sensitivity analyses can be extremely valuable in some settings. But in many other situations, analysis of the conditions of optimality can be premature given the state of the data and the model. Sensitivity information can be a form of information overload.

When the objective function is only an approximation of the actual goals of the organization and its stakeholders, the one solution that optimizes it may be no more interesting than other solutions that provide good values. However, information overload can be a problem here as well. It is not desirable to swamp the decision maker with solutions. Highly preferable is to identify a set of solutions that are decently good and, especially, diverse.

We can reasonably rely on the objective function to quantify the notion of "decently good". Methods for quantifying the notion of "mutually diverse" are given in §3. We devote §2 to a description of methods for obtaining a diverse set of solutions for MIPs. In §4 we provide computational experiments that show that our methods do indeed produce a diverse set of solutions and accomplish this much more effectively than branch and bound alone. The paper closes with some concluding remarks and directions for further research.

2. SCATTER SEARCH AND STAR PATHS

Our diversification methods are based on the idea of generating extreme points in a polyhedral region of interest and then using these points and the paths between them in a variety of ways. The methods examine points on the polyhedron, within and "near" it. It is natural, therefore, to use the constraints (1) and (3) provided by problem (P) to define edges of the polyhedron. The LP feasible point, x^{**}, obtained by solving the *relaxation* of (P), which is obtained by removing constraints (2) is also particularly useful. We proceed in two phases: first we generate a set of *centers* and then we connect them using *star paths*.

2.1 Generating Centers

The description of the generation of centers can also be broken into two phases. First we use a diversification generator to create points. In the second phase, these points are provided as data to an optimization problem that results in extreme points that are averaged to create the centers.

The diversification generator we employ creates a sequence wherein for each new vector the minimum Hamming distance from all previously generated vectors is maximized. We define the *complement over an index set* $J \in I$ of a solution vector x to be y where $y_j = 1 - x_j$ and $y_j = x_j$ for $j \in I \setminus J$. The generator first described in [5] proceeds as shown in Figure 1 when provided with a vector r of length n that satisfies constraints (2). This generator creates approximately $2(1 + \log |I|)$ solutions. Of course, the algorithm can be terminated when fewer vectors have been generated by a simple modification to step 3. We round x^{**} to serve as the vector r for the procedure and refer to this rounded solution as r^{**}. To use the terminology of branch and bound algorithms, it is the result of rounding the root node solution.

Each of the solutions created by the generator is used for a *construction by objective* to create points within and on the polyhedron that are called *primary centers* and *subcenters*.

To generate primary centers, we modify the problem (P) by removing constraints (2) and replacing the objective function with

$$\sum_{i \in I} 2\,\mathrm{r}_i - 1$$

where the vector r is provided as data with the property that it satisfies constraints (2). To avoid exploring the regions that are extremely large and likely to be uninteresting we add a constraint to those required by (P),

$$cx \le cx^{**} + a\left|cx^{**}\right| \qquad (4)$$

1. The solution vector r and its complement over I are the first two solutions generated. Form an arbitrary partition of I into two equal-sized cells and call these sets I' and I''; place I' in a set of subsets called K' and I'' in one called K''.
2. The next two solutions generated are the complement of r over I' and I''.
3. Move all of the subsets in K' and K'' that were created by the most recent step into a set of subsets and call it K and consider K' and K'' to be empty. If all of the subsets in K are empty or contain only one element, stop. Otherwise, partition each subset in K, into two equal-sized sets and add the new sets to K' and K'', respectively. As sets in K are encountered with an odd number of elements, alternate between assigning more elements to those sets that will be placed in K' and those that will go into K''.
4. Let I' be the union of all subsets in K' and let I'' be the union of the K'' sets. Go to Step 2.

Figure 1. The Sequential Diversification Algorithm Given the Data for Problem (P) and a Solution Vector r as an Input Parameter

that bounds the polyhedron to be "not too far" from x^{**}. Bearing in mind that our methods will examine points outside the polyhedron, one can believe that the choice of a is not critical. Computational experiments reveal that values such as 0.1 or 0.2 seem to work well. Of course, we can refine the representation of (4) by considering issues of scaling and accounting for the possibility that cx^{**} is zero. The problem we have constructed to correspond to problem (P) is problem

$$(\mathrm{P}') \quad \text{Minimize or Maximize} \quad \sum_{i \in I} 2\mathrm{r}_i - 1$$

subject to

$$Tx \geq b$$
$$cx \leq cx^{**} + a\left|cx^{**}\right|$$
$$l \leq x \leq u$$

Refer to the solution to the minimization problem as $\bar{x}(r)$ and the result of maximizing $\vec{x}(r)$. The primary center associated with r is $\left(\bar{x}(r) + \vec{x}(r)\right)/2$ and the set of subcenters is $\bar{x}(r) + w\left(\bar{x}(r) - \vec{x}(r)\right)$ for w = 0, 1/4, 3/4, and 1.

We use the result of the sequential generator shown in Figure 1 with $r = r^{**}$ to generate a set of vectors. The primary and secondary centers associated with these vectors are collected to be connected by a series of *star paths* that are used to generate potentially feasible, diverse solutions for (P).

2.2 Star Paths

Before proceeding to define star paths, we must first describe *directional rounding* [4] upon which the notion of star paths is based. Directional rounding is defined relative to a base point, x^0, and a focal point x'. The rounding is defined component-wise as

$$d(x'_j; x^0_j) = \begin{cases} 1, & \text{if } x'_j > x^0_j \\ 0, & \text{if } x'_j < x^0_j \\ x^0_j, & \text{if } x'_j = x^0_j \text{ and } x^0_j \in \{0,1\} \\ \text{round}(x^0_j) & \text{otherwise} \end{cases}$$

where round (·) refers to simple rounding. The motivation for directional rounding comes from the fact that if x^0 is LP feasible, then every feasible 0-1 solution can be obtained by directional rounding relative to focal points that lie within the LP cone defined by non-negative values for the current nonbasic variables. Moreover, attention can be restricted to focal points that lie on any chosen hyperplane that intersects the LP cone and excludes x^0, hence that passes through points that correspond to positive values for each of the nonbasic variables.

A star path, $L(x^0; x', x'')$ is defined as an ordered set of vectors that satisfy constraints (2) associated with a line between two arbitrary vectors, x' and x'' obtained from a sequence of directional roundings using x^0 as a *base point*. For every real value of λ, the point $x' + (x'' - x')\, l$ has a vector in $L(x^0; x', x'')$ which we write as $\ell(L(x^0; x', x''); l)$ or just as $\ell(l)$, which is understood to be shorthand for the same thing. Define $\Delta = x'' - x'$ and the index sets I^0, I^+, I^- to be, respectively those $j \in I$ such that Δ_j is equal to, greater than, and less than zero. For $j \in I^+ \cap I^-$ we define $l(j) = (x^0_j - x'_j)/\Delta_j$.

1. Let $l = 0$, and generate a solution vector, $\hat{x} = \ell(l)$ by the method described in Lemma C. If $l \geq l(\theta(t))$, it is the only vector to be generated and the procedure stops. Otherwise, identify $p = \theta(q)$, where $q = \min(k : l(\theta(k)) > l)$.
2. Generate the next $\hat{x}$ vector by setting $\hat{x}_p = 1 - \hat{x}_p$, without changing any other elements of the vector.
3. Set $q = q + 1$. If $q > t$, stop. Otherwise, set $p = \theta(q)$ and go to Step 2.

Figure 2. The Star Path Generation Algorithm, Which Generates a Sequence of Solution Vectors given $x^0, x', and\ x''$

These definitions allow a precise characterization of $\ell(l)$ as follows.

Lemma C. Given $x^0, x', and\ x''$ the elements of $\ell(l)$ are given by

$$\ell_j(l) = d(x'_j; x^0_j) \quad \text{for } j \in I^0$$

$$\ell_j(l) = \begin{cases} 0 & l < l(j) \\ 1 & l \geq l(j) \end{cases} \quad \text{for } j \in I^+, \text{ and}$$

$$\ell_j(l) = \begin{cases} 0 & l \geq l(j) \\ 1 & l < l(j) \end{cases} \quad \text{for } j \in I^-.$$

To take advantage of Lemma C, let $q(1), \ldots, q(t)$ be a permutation of the indexes of $I \setminus I^0$ so that $l(q(1)) \leq l(q(2)) \leq \ldots \leq l(q(t))$ where $t = |I \setminus I^0|$. In addition, let $l(0)$ be an arbitrary value less than $l(\theta(1))$. By convention, we will suppose that the values $l(q(q)), q = 1, \ldots, t$ are all distinct so that $l(q(q)) < l(q(q+1))$ for all $q < t$.

This convention allows a maximum number of elements of the star-path to be created, and also leads to characterizing these elements as adjacent vertices of the unit hypercube established by constraints (2). We will show that this convention is trivially easy to impose; that is, no explicit perturbation needs to be introduced to allow the $l(q(q))$ values to be treated as distinct in case there are tied values.

The star-path theorem given originally in [4] states that a star-path generated by the rule of Lemma C when λ takes the values $l(0), l(q(1)),\ldots, l(q(t))$ contains precisely $t + 1$ distinct points. The points constitute successively adjacent vertices of the unit hypercube, linked to each other by the following relationship. For the arbitrary value of $l < l(q(t))$, let $l_{\text{next}} = l(p)$, where $p = q$ (q) for $q = \min(k : l(q(k)) > l)$. Then $\ell(l)$ and $\ell(l_{\text{next}})$ are associated by the rule $\ell_j(l_{\text{next}}) = \ell_j(l)$ for $j \neq p, j \in I$ and $\ell_p(l_{\text{next}}) = 1 - \ell_p(l)$.

The fact that the points of the star path are successively adjacent vertices of the unit hypercube implies that it is unnecessary to create a numerical shift of tied values of $l(q(q))$ by an explicit perturbation in order to allow the specified points of the star-path to be generated. The star path generation algorithm is shown in Figure 2.

Our full procedure for generating a set of diverse, feasible solutions for MIP problem (P) is summarized in Figure 3, which we refer to as the *scatter-path* method.

1. Solve the relaxation of (P) to obtain x^{**} and apply simple rounding to obtain a solution of r^{**} that satisfies constraints (2).
2. Provide r^{**} to the sequential diversification algorithm shown in Figure 1. Use each vector created by the diversification algorithm as a vector r for problem (P′) to create a primary center and subcenters.
3. Generate star paths as described in Figure 2 using $x^0 = x^{**}$ and combinations of centers, subcenters and the average of the centers as x' and x''. Check the solutions along the star path for feasibility with respect to (P) perhaps by solving the LP that results from fixing the integers as indicated by the vectors $\hat{x}$ generated by the star path algorithm. Retain feasible solutions.
4. Sort the set of retained solutions to remove duplicates.

Figure 3. Summary of the Scatter-Path Method to Generate a Diverse Set of Feasible Vectors for (P)

3. MEASURING DIVERSITY OF MIP SOLUTIONS

Although a diverse set of good solutions is clearly desirable, it is not clear in advance how to measure the property of diversity. In spite of the fact that the objective function is not exact, it presumably gives a reasonable way to assess the relative "goodness" of a set of solutions. No such simple mapping is known from solution vectors to a one-dimensional measure of diversity.

3.1 Diversity Metrics

We need diversity measures both for the design of practical software and for research purposes. For practical software, we want to know if the user should be "bothered" with a particular solution vector. That is, we want to know if a vector adds enough diversity to warrant adding it to the set of solutions that are displayed. For research purposes, we might want to compare the set of vectors generated by one method with a set of vectors generated by another. These issues are intertwined with solution quality and are ultimately a matter of taste, but we can shed some light by briefly considering measures of diversity in isolation.

Vector	L.A.	S.F.	Hamburg	Contingency
$A^{(1)}$	1	0	0	94.8
$A^{(2)}$	0	1	0	32.9
$A^{(3)}$	1	0	1	4.3
$A^{(4)}$	0	0	1	2.9
$A^{(5)}$	1	0	1	12.9
Mean	0.6	0.2	0.6	29.56
Std. Dev.	0.5	0.4	0.5	38.39

Figure 4: A Small Example of a Set of Solution Vectors

Some of the issues can be seen easily using a contrived example. Suppose we have a problem whose first three variables take zero-one values that indicate whether there is a production facility in Los Angeles, San Francisco, and Hamburg respectively. A fourth and final value indicates the amount of money to place in an exchange rate contingency fund. For example, the solution vector (0,1,1,6.3) indicates that there will be factories in San Francisco and Hamburg, and there will be 6.3 million dollars in the contingency fund. Suppose we have a set of vectors and the size of the contingency fund is negatively correlated with the existence of a factory in Hamburg, as shown in Figure 4. The addition of a vector B=(0,0,1,31.1) would clearly be more diversifying than C=(0,0,1,3.3). The only column that

varies between *B* and *C* is the contingency fund. Both values are within a standard deviation of the mean for set *A* so neither would be considered an outlier. However, if one conditions the statistical analysis on the value of the Hamburg column, a straightforward t-test would suggest to a reasonable person that vector *B* does not seem to come from the same population as *A*, i.e., its addition would be diversifying. What we desire is an automatic way of discovering such a relationship in large solution vectors without knowing in advance what columns to consider. Furthermore, in addition to an indication of whether vectors are diversifying, it would be useful to have metrics that allow comparison of vectors and sets of vectors.

Before proceeding with our proposal, we show why some fairly obvious ideas do not work well in general. The first thing to notice is that column-wise methods will generally not produce satisfactory results because, for one thing, they fail to consider interactions. For example, the addition of vector *C* to set *A* increases both the variance and coefficient of variation for the last column more than *B*. Of course, one can contrive examples where univariate methods are effective, but for MIP solutions there are typically significant interactions between the columns; when there are no interactions, the problem can be separated into smaller problems.

This leads to a desire for a metric that considers entire vectors. We quickly reject Hamming distances because they ignore the real-valued variables. For example, the Hamming distance between vectors *B* and *C* is zero. Euclidean distances are not scale invariant (the currency used to denominate the contingency fund controls the pairwise distances). Their more important shortcoming, shared by Hamming distances, is that they fail to take correlations into account. For the example shown in Figure 4 the Euclidean distance from the mean of *A* to *B* is much less than the distance to *C*. This remains true even if the vectors are scaled so that the mean of all vector elements in *A* are 1 and the scaling is applied to *B* and *C*. The rejection of Euclidean distances leads to rejection of algorithms such as K-means to cluster solutions.

There are a number of advantages to the quadratic metric known in this context as *Mahalanobis* distances. A *p*-vector *x* can be said to have a squared Mahalanobis distance from a *p*-vector m *under* a $p \times p$ positive, definite symmetric matrix *S* that is given by

$$d^2\left(m,x;\mathrm{S}\right) \equiv \left(x-m\right)^{\mathrm{T}} S^{-1}\left(x-m\right)$$

This metric is scale invariant and can take correlations into account if the matrix *S* is a covariance matrix. An estimate of the covariance matrix for the population from which a sample, *A* containing *h* vectors is

$$\frac{1}{h-1}\sum_{j=1}^{h}\left(A^{(j)}-\overline{A}\right)\left(A^{(j)}-\overline{A}\right)^{\mathrm{T}},$$

where $\overline{A}$ is the usual mean of the set of vectors, which is indexed here by *j*. Note that the Euclidean metric is a special case that uses the identity matrix as *S*. The motivation for the Mahalanobis distance comes from multivariate probability theory. If a multivariate normal distribution is defined by mean m and covariance *S*, then for points v governed by the distribution, the Mahalanobis distances, $d^2(m,v;S)$ have a X^2 distribution (see e.g., [1, pages 72-75]).

Furthermore, this type of distance connects naturally with a scalar measure of the diversity of a set of vectors, which is the determinant of the covariance matrix of the set. Under the assumption of multivariate normality, the covariance matrix defines ellipsoids of constant Mahalanobis distances that constitute probability contours. Large covariance determinants correspond to large volumes in the ellipsoids. The assumption of multivariate normality is not needed to use the covariance determinant to put an order on sets of vectors and furthermore it is not needed to see that adding points with large Mahalanobis distances will increase the covariance determinant.

To summarize, we can use the covariance determinant to put an ordering on the diversity of sets of solution vectors and the Mahalanobis distance to put an order on the diversifying effect of vectors on a set. In the simple example given, this works nicely. The Mahalanobis distance from the mean of the set *A* under its covariance matrix to vector *B* is much greater than the distance to *C* as we would expect. Consequently, we would correctly conclude that vector *B* would be more diversifying than vector *C* if we used Mahalanobis distances.

However, there is a major difficulty. In order to calculate a covariance matrix for a set of vectors of length $p = n$ one must have a set of vectors that does not lie entirely in a subspace. This means that at a minimum the set must contain $n + 1$ vectors and for MIP solutions, more vectors will often be required to span the full n dimensions. For even modest sized MIPs this is not good. In order to have a working definition of diversity, one must have thousands of solution vectors. A remedy for this difficulty that also increases the plausibility of multivariate normality has been referred to as *chunking.*

3.2 Chunking Solution Vectors

The specific chunking method that we use is an application of more general constructs described by Woodruff [13]. We refer readers to that

paper for background information on chunking. We will refer to a non-empty subset of the solution vector indexes 1,...,*n* as a *chunk*. An instance of a partial vector corresponding to a chunk is referred to as a *chunk instantiating value*. That is, a chunk instantiating value for a given chunk $C \subseteq \{1,\ldots, p\}$ depends on the specific vector *x* whose values are under consideration. Let *R* be a partition of {1,...,*n*} with *p* cells. The mappings from chunk instantiating values to reals are referred to as *chunk valuation* functions, *v*. We will deal with a fairly specific valuation function. Each element of the valuation vector is defined as

$$v_C(x) = \frac{1}{|C|}\sum_{i \in C} x_i$$

where |*C*| is the number of indexes in the chunk for each chunk $C \in \mathrm{P}$. If we index the chunks $C \in \mathrm{P}$, using i we can write v_i in the same way for $i = 1,\ldots, p$. The unweighted summation makes intuitive sense only in the presence of our requirement that the problem data be scaled so that the bounds vectors *l* and *u* specified in problem (P) are approximately zero and one respectively.

We can compute the mean and covariance matrix for the valuation vectors that correspond to a set of vectors and use those to characterize the location (mean), shape (covariance), and diversity (covariance determinant) of the set. Valuation vectors for candidate vectors can be computed and their Mahalanobis distance from the set can be used to characterize the effect on the diversity of the set if they were to be added to it. If the same chunking is applied to other sets of vectors, their diversity can be compared. The question is, how should we choose the chunk partition?

3.3 Finding Good Chunks

The idea is to find a way to group the solution vector indexes so that the resulting valuation vectors are as useful as possible for measuring the diversity of a set of solution vectors and for anticipating the diversifying effects of adding a vector to the set. We cast these as optimization problems of finding a chunking for a given vector set subject to constraints on the number of cells in the partition and the number of elements in each cell. We have two related objectives and therefore two related objective functions.

For both problems, we want to specify the number of cells, *p*, in the partition and a minimum number of indexes in each cell, *H*. Based on the desire for normality, *H* should usually be significantly greater than 20. We

want to use a collection of solution vectors of length *n*, that are given as data in a set *A* to find a partition *P* of the indexes $1,\ldots,n$, with cells P_i, i = 1,..., p so as to solve a problem

(*MW*) minimize or maximize $|W|$

subject to

$$
\begin{aligned}
W &= \sum_{j=1}^{a} \left(v^{(j)} - \bar{v}\right)\left(v^{(j)} - \bar{v}\right)^{\mathrm{T}} \\
\bar{v} &= \left(\sum_{j=1}^{a} v^{(j)}\right) / a \\
v_i^{(j)} &= \frac{1}{|P_i|} \sum_{k \in P_i} A_k^{(j)} \qquad i = 1,\ldots, p, \quad j = 1,..., a \\
|P_i| &\geq H \qquad\qquad i = 1,\ldots, p
\end{aligned}
$$

where $|W|$ is the determinant of W, $|\mathrm{P}_i|$ is the number of indexes in cell i of the partition P and a is the number of vectors in A. Note that $v^{(j)}$ and $\bar{v}$ are column vectors of length P and that W is $(a-1)$ times an unbiased covariance estimate of the valuation vectors associated with A.

One objective is to be able to maximize the ability to distinguish vectors from each other. For example, it would be desirable from some DSS applications to distinguish new solutions that are encountered from those that have already been displayed. This objective is beyond the scope of this paper, but one possibility would be to minimize the covariance determinant as shown in problem *(MW)*.

For research purposes one would also like to be able to compare methods of measuring and creating diversity.

For the purpose of comparing two methods of generating vectors, the method that produces the set with the largest covariance determinant has produced the set that occupies the greatest volume and hence is the most diverse. If we are testing the hypothesis that method B, which produced vector set *B*, is better than method C, which produced vector set *C*, we would want to be conservative and select a chunking that tends to maximize the volume occupied by the valuation vectors created from set *C* and then apply that chunking to the vectors in *B*. If the determinant of the covariance matrix of valuation vectors for *B* is greater than that of the valuation vectors for *C* even under a chunking that favors method C, then this would be strong evidence that method B is better for the purpose of producing a diverse set of vectors.

Unfortunately, there is no free lunch. Apart from the obvious risks that accompany dimension reduction and the associated loss of information, there are some other difficulties. Under fairly reasonable assumptions, and

typically as a practical matter, the objective function is decreasing in the number of chunks; so it is not possible to automate the selection of *p*. The only mathematical guidance is that one should reduce *p* or increase *H* if a zero determinant is found (i.e., if *H* or more co-planar evaluation vectors are present).

Furthermore, problem (*MW*) is a highly nonlinear, combinatorial optimization problem. Instances of realistic size are not easily solved to optimality. This difficulty is mitigated by the fact that the optimization is primarily of importance for theoretical reasons and approximate solutions suffice in practice.

4. COMPUTATIONAL EXPERIMENTS

Our primary goal in this section is to demonstrate that the scatter-path method is useful for generating good, diverse vectors for MIP problems (P) and that it offers particular advantages when used in conjunction with branch and bound. A secondary goal is to demonstrate the value of chunking as a means of measuring diversity. For both goals we need heuristic solutions to problems *(MW)*. The methods we used for obtaining approximate solutions for these problems are described in the Appendix.

In keeping with the recommended practice for computational experiments [2, 7, 10, 11] we use a measure of computational effort that is independent of the computer or programming language employed, which is the number of relaxations that must be solved. This is conservative in that the branch and bound relaxations typically have more free variables than the scatter-path relaxations (since the star path process fixes the value of all integer variables). For a fixed objective function threshold and total number of relaxations, we compare the diversity of solutions generated by branch and bound with those generated by adding the scatter-path method to the branch and bound procedure. To be conservative, the chunking scheme was selected so as to maximize the objective function for (*MW*) using the feasible vectors generated by branch and bound. This chunking scheme was then applied to the vectors generated by the combined methods. Our experiments were conducted using XPress-MP [3] version 10. The value of α for problem (P´) was fixed at 0.1.

The results are strong. Table 1 shows the ratio of the determinant of the *W* matrix for the valuation vectors corresponding to the two sets of solution vectors. The first set is formed by the first 500 branch and bound relaxations plus the first 500 relaxations solved by the scatter-path method. The second is 1000 branch and bound relaxations alone. In both cases, the chunking scheme is created with the goal of maximizing the determinant for branch

and bound alone. Only those solutions with objective functions within 20 per cent of the best solution found are considered.

There are a number of issues to discuss. Note that there is a tendency for the margin by which the combined method dominates to be decreasing in *p*. This is an artifact of our conservative comparison methods. The optimization algorithms can simply do a better job fitting a chunking to the branch and bound method with more degrees of freedom. If the chunking were optimized for the combined methods, the trend would be reversed. For MISC04, the branch and bound finds very few solutions, so values of *p* greater than 2 are not sensible so these entries in the table contain a dash.

A more important issue is raised by the instances marked with an asterisk, which are those for which branch and bound terminates before it solves 1000 relaxations. The relationship between diversification methods and branch and bound in practice would not be one of competing with each other but of complementing each other. A practitioner would still find it valuable to use branch and bound in order to have information about the quality of solutions obtained, and in order to find solutions for classes of problems that are especially susceptible to solution by this approach. We have used a branch and bound algorithm as the "competition" in order to have a well-known frame of reference. There are some instances in MIPLIB for which the logical partitioning of branch and bound is highly effective, and indeed indispensable for finding optimal solutions. Thus our preceding results – which disclose the advantages of joining the scatter-path method with branch and bound, by comparison with branch and bound itself – should not be interpreted to derive from the scatter-path method in isolation, because the method is incomplete as a search engine to find best solutions. Moreover, the logical partitioning of branch and bound also appears essential in some cases to isolate the regions in which good diverse solutions can be generated.

	p		
Instance	2	3	4
DCMULT	815.9	117.1	37.4
EGOUT*	5.6	2.8	2.9
MISC04*	9942	-	-
MISC05	1.6	1.1	1.0
MISC06	89.5	21.3	11.4
RGN	14.5	5.2	3.5

Table 1. The p^{th} Root of the Ration of Determinant of the W Matrix of the Valuation Vectors of a Combination of Scatter-Path and Branch and Bound Compared with Branch and Bound Alone for Some MIPLIB Instances

To illustrate, applied to the DANOIN problem from MIPLIB, the scatter-path method by itself cannot find solutions that are within 20% of the best

branch and bound solution in a reasonable amount of time. There are other instances such as PP08AC, VPM1 and VPM2 for which the scatter-path method on its own requires more than 1000 relaxations to find good diversifying solutions. For example, after 20,000 relaxations, the entries for PP08AC that would go in Table 1 if the scatter-path method were applied in exclusion from branch and bound would be 7.8, 5.0 and 3.5. Of course, if a practitioner has the luxury of examining larger numbers of relaxations as a basis for finding good diversifying solutions, then over time the scatter-path approach may find them. But the chief message from these outcomes is that value derives from the interaction between diversification and partitioning, and that the advantages go in both directions. That is, the diversification of the scatter-path method can significantly multiply the number of high quality (and appreciably distinct) solutions found by branch and bound, while the partitioning of branch and bound can significantly enhance the effectiveness of the scatter-path process.

5. CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

In this paper we have presented methods of systematically uncovering a diverse set of solutions for 0-1 mixed integer programming problems. These methods can be applied to instances without any special foreknowledge concerning the characteristics of the instances, but the absence of such knowledge gives rise to a need for general methods to assess diversity. We have also presented methods based on chunking for this purpose.

More research is needed concerning the mapping between the order put on solution sets by covariance determinants and the diversity perceived by a user. For example, we have compared methods by looking at the ratio of the determinants of the W matrix which compares the volume occupied by the vectors, but the covariance matrices (which are the average W matrix) might be interesting in some applications because the measure of dispersion is then weighted by the number of vectors. Objectives other than minimizing problem (*MW*) also warrant further research.

Optimization methods are enjoying a resurgence in fields such as data mining, supply chain management and finance. As general purpose optimization methods are embedded in decision support systems we anticipate an increased need not only for optimal solutions, but also for a diverse set of good solutions. This paper has presented methods that serve this purpose as well as methods that facilitate on-going research in this area.

ACKNOWLEDGEMENTS

This research was supported in part by grants from the National Science Foundation, the National Partnership for Academic Computing Infrastructure and the Norwegian Research Council.

APPENDIX — METHODS USED TO GENERATE INDEX PARTITIONS

We proceed in two stages. First we use a fast heuristic to get an initial solution, then a simple, general purpose tabu search is used to improve it.

To quickly generate an index partition that tends to maximize (minimize) the covariance determinant for the set of valuation vectors that correspond to a particular set of full solution vectors, *A*, we make use of the idea that this will tend to correspond to chunkings where the covariation of full vector values within a cell of the partition is minimized (maximized) thereby tending to maximize (minimize) the covariance between the valuation vectors. This does not characterize optimal solutions in any rigorous way, but is a reasonable basis for design of a heuristic.

We provide one fast heuristic for the maximization problem and a similar one for the minimization problem. Both heuristics begin by calculating the variances of all vector elements in the set for each index. For a particular index *i*, we will abuse the notation s_{ii}^2 to represent the variance by defining

$$s_{ij}^2 \equiv \frac{\sum a \in A \left(x_i^{(a)} - \bar{x}_i\right) \left(x_j^{(a)} - \bar{x}_j\right)}{|A|}$$

where $|A|$ is the number of vectors in A and $\bar{x}_i \equiv \frac{\sum a \in A\, x_i^{(a)}}{|A|}$. Those indexes corresponding to zero variances are randomly assigned to the partitions and removed from further consideration. This leaves a set of indexes to be assigned that we will refer to as *I*.

The basic data structure is a list of pairs of indexes (*i, j*), which initially covers all of $I \times I$ along with the sample covariance, s_{ij} for the values in A of the vector elements for the index pair. We define the next member in a *chains* for a particular list and a particular list element characterized by

index pair (i, i') as the list element characterized by (i', i'') that has highest covariance from among all $i'' \in I$ with ties broken arbitrarily.

For the maximization problem, repeat the following until the list is empty:

> Begin with the list element that corresponds to the highest covariance value, call the index pair (i, j). Repeat the following step with iterations indexed by k until the list is empty or there is no next chain member for the list element characterized by (i, j):
>
> Assign i to chunk (k mod p) + 1, remove i from I and for all $i' \in I$, remove elements for (i, i') from the list. If possible, proceed to the next member of the chain for (i, j), whose index pair is then called *(i, j)*.

This is similar to sorting a deck of playing cards and then dealing out p hands. The effect is that each hand has as much variety as possible. These groups will be roughly equal in size. We use the results of these heuristics as starting points for a simple, first improving tabu search. Because these chunkings will be used to analyze vectors that do not come from the population used to generate them, we want to avoid overfitting so we use $H = n/2p$, which keeps all of the chunks fairly large. The search uses a neighborhood formed by moving an index from one chunk to another and terminates when 10 passes through the indexes have not resulted in a new best solution. The tabu tenure is also set at 10. For more information about tabu search, see [6].

BIBLIOGRAPHY

[1] Anderson, T.W., *An Introduction to Multivariate Statistical Analysis*, John Wiley and Sons, New York 1984.

[2] Barr, R., B. Golden, J. Kelly, M. Resende, and W. Stewart, "Designing and Reporting on Computational Experiments with Heuristic Methods," *Journal of Heuristics* 1 (1995) 9-32.

[3] Dash Assoc., *XPRESS-MP Reference Manual*, Version 10, Binswood Ave, Leamington Spa, Warwickshire, CV32 5TH, UK, (1997).

[4] Glover, F., "Scatter Search and Star Paths: Beyond the Genetic Metaphor," *OR Spektrum*, 17 (1995) 125-137.

[5] Glover, F., "A Template for Scatter Search and Path Relinking," in Hao, et al. (eds) *Artificial Evolution, Lecture Notes in Computer Science*, Springer (1997) 13-54.

[6] Glover, F., and M. Laguna *Tabu Search*, Kluwer, Boston (1997).

[7] Greenberg H.J., "Computational Testing: Why, How and How Much," *ORSA Journal on Computing* 2 (1990), 7-11.

[8] Greenberg, H.J., "An Annotated Bibliography for Post-solution Analysis in Mixed Integer Programming and Combinatorial Optimization," in *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*, D.L. Woodruff (ed), pp 97-148, Kluwer, Boston, MA, 1998.

[9] Hoch, S.J., and D.A. Schkade, "A Psychological Approach to Decision Support Systems," *Management Science*, 42 (1996), 51-64.

[10] Hooker J.N., "Needed: An Empirical Science of Algorithms," *Operations Research* 42 (1994), 201-212.

[11] Hooker J.N., "Testing Heuristics: We Have it all Wrong," *Journal of Heuristics* 1 (1996), 33-42.

[12] Sharda, R., S.H. Barr, and J.C. McDonnell, "Decision Support System Effectiveness: A Review and an Empirical Test," *Management Science*, 34 (1988), 139-159.

[13] Woodruff, D.L., "Proposals for Chunking and Tabu Search," *European J. Oper. Res.*, 106 (1998) 585-598.