Final Report—Optimization Under Nonconvexity and Uncertainty: Algorithms and Software

Grant Award: DE-FG02-05ER25694

Jeff Linderoth, Principal Investigator

Department of Industrial and Systems Engineering
University of Wisconsin-Madison
3226 Mechanical Engineering Bldg
1513 University Avenue
Madison, WI 53706-157 USA

Phone: 608-890-1931
Fax: 608-262-8454

email: `linderoth@wisc.edu`

DOE/Office of Science Program Office: Advanced Scientific Computing Research
DOE/Office of Science Technical Program Manager: Homer Walker

Project Performance Site: Lehigh University

1 Overview

Numerical optimization is a pervasive tool for planning and controlling physical systems. A small sample of applications using optimization includes blending and process design, bioinformatics, environmental planning, national security, medical treatment planning, and financial asset and supply chain management. However, even with the advances made in optimization theory and algorithms, and with the increasing the power and utility of large-scale computational platforms, there still exist areas in which the needs of users of optimization technology exceed the tools and solution capabilities of modern solvers. This proposal addressed two of these areas: optimization problems containing nonconvexities and optimization problems under uncertainty.

Project Objectives: Develop new algorithmic techniques for solving large-scale numerical optimization problems, focusing on problems classes that have proven to be among the most challenging for practitioners: those involving uncertainty and those involving nonconvexity. This research advanced the state-of-the-art in solving mixed integer linear programs containing symmetry, mixed integer nonlinear programs, and stochastic optimization problems.

2 Stochastic Programming

In the context of stochastic programming, this project made two main contributions. First, we studied a stochastic version of the well-known network interdiction problem, where the successful destruction of an arc of the network is a Bernoulli random variable, and the objective is to minimize the maximum expected flow of the adversary. By using duality and linearization techniques, an equivalent deterministic mixed integer program was formulated. A specialized decomposition approach was used in combination with grid computing to solve instances much more efficiently that reported in the literature. This work appeared in [9].

The second contribution demonstrated the efficacy of using sampled approximations to stochastic programs. Stochastic programs can be solved approximately by drawing a subset of all possible random scenarios and solving the problem based on this subset. The value of the optimal solution to the sampled problem provides an estimate of the true objective function value. This estimator is known to be optimistically biased; the expected optimal objective function value for the sampled problem is lower (for minimization problems) than the optimal objective function value for the true problem. We investigated how two alternative sampling methods, antithetic variates and Latin Hypercube sampling, affect both the bias and variance, and thus the mean squared error (MSE), of this estimator. For a simple example, we analytically express the reductions in bias and variance obtained by these two alternative sampling methods. For test problems from the literature, we computationally investigated the impact of these sampling methods on bias and variance. We find that both sampling methods are effective at reducing mean squared error, with Latin Hypercube sampling outperforming antithetic variates. Whether the bias reduction or variance reduction plays a larger role in MSE reduction is problem and parameter specific. This work resulted in the publications [16, 4]. In this section we give details about these contributions.

2.1 Stochastic Network Interdiction

The Stochastic Network Interdiction Problem (SNIP) is defined on a capacitated directed network $G = (V, A)$, with specified source node $r \in V$ and sink node $t \in V$. The capacity of arc $(i, j) \in A$ is u_{ij}. The realizations of uncertainty are captured in a countable set of scenarios S, in which each element $s \in S$ occurs with probability p_s. There is a finite budget K available for interdiction, and the cost of interdicting on any arc $(i, j) \in A$ is h_{ij}. To write a mathematical program for SNIP, define the binary decision variables

$$x_{ij} = \begin{cases} 1 & \text{if interdiction occurs on arc } (i,j) \in A, \\ 0 & \text{otherwise.} \end{cases}$$

With these definitions, SNIP can be written as the formulation $\mathcal{F}_1$:

$$\begin{array}{lrl} (\mathcal{F}_1) & z_{\text{SNIP}} = \min \displaystyle\sum_{s \in S} p_s f_s(x) & \\ \text{subject to} & \displaystyle\sum_{(i,j) \in A} h_{ij} x_{ij} \leq K, & \\ & x_{ij} \in \{0,1\} & \forall (i,j) \in A, \end{array}$$

where $f_s(x)$ is the maximum flow from node r to node t in scenario s if interdictions occur on the arcs indicated by the $\{0,1\}$-vector x. The scenario is defined by a random vector

$$\xi_{ijs} = \begin{cases} 1 & \text{if interdiction on arc } (i,j) \in A \text{ in scenario } s \in S \text{ would be successful,} \\ 0 & \text{otherwise,} \end{cases}$$

Using duality and linearization arguments, we showed that the following mixed integer linear program is a valid reformulation of the stochastic network interdiction problem:

$$(\mathcal{F}) \qquad z_{\text{SNIP}} = \min \sum_{s \in S} p_s \left(\sum_{(i,j)\in A} u_{ij}\rho_{ijs} - \sum_{(i,j)\in A} u_{ij}\xi_{ijs} z_{ijs} \right)$$

subject to

$$\begin{aligned}
z_{ijs} - x_{ij} &\leq 0 && \forall (i,j) \in A, \forall s \in S,\\
z_{ijs} - \rho_{ijs} &\leq 0 && \forall (i,j) \in A, \forall s \in S,\\
\sum_{(i,j)\in A} h_{ij} x_{ij} &\leq K, && \\
\pi_{rs} - \pi_{ts} &\geq 1 && \forall s \in S,\\
\rho_{ijs} - \pi_{is} + \pi_{js} &\geq 0 && \forall (i,j) \in A, \forall s \in S,\\
x_{ij} &\in \{0,1\} && \forall (i,j) \in A,\\
z_{ijs}, \rho_{ijs} &\geq 0 && \forall (i,j) \in A, \forall s \in S.
\end{aligned}$$

Formulation $\mathcal{F}$ is a mixed integer linear program whose solution gives an optimal solution to the stochastic network interdiction problem. A desirable feature of this formulation is that if x is fixed, then the formulation decouples into $|S|$ independent linear programs. The decomposable structure of the formulation was exploited by the solution algorithm.

Table 1 shows how the lower and upper bounds on optimal solution value vary as a function of the sample size for each instance for different network sizes. The average wall clock time and average number of processors used to achieve the solution x_j^* to the sample average approximation is also listed for each instance and sample size, as are the average number of iterations required to solve the initial linear programming relaxation and the average number of integer programs necessary to solve the instance.

The solution time for most instances is quite moderate and shows a relatively low dependence on the number of scenarios N. The low dependence on N is primarily accounted for by the fact that the number of LP iterations and IP solves remains nearly constant regardless of N. Instances with larger values of N require more computational effort, but they can also be parallelized to a higher degree, as there are more scenario linear programs to solve at each iteration. Cormican, Morton, and Wood report solving an instance of the size of SNIP10x10 to 1% accuracy in 23,970 seconds on an RS/6000 Model 590. We achieve a similarly accurate solution in two minutes.

2.2 Bias Reduction for Sample Path Optimization

The Stochastic Network Interdiction Problem from Section 2.1 was an example of a two-stage stochastic optimization problem. At the first

At the first stage, values are chosen for a set of design variables; for example, the arcs upon which to interdict. The objective function of the first-stage problem requires us to evaluate the expected value of the solution to a second-stage linear program, some of whose parameters are stochastic. Furthermore, the design variables from the first stage appear in the constraints of the second-stage problem.

Instance	N	LB	UB	Avg. Sol Time (sec.)	Avg # Workers	Avg # LP Iterations	Avg # IP Solves
SNIP7x5	50	76.42 ± 4.24	80.87 ± 0.91	227.39	1.39	12.10	1.00
SNIP7x5	100	77.31 ± 3.14	80.87 ± 0.79	136.83	0.94	12.80	1.40
SNIP7x5	200	78.51 ± 3.12	80.85 ± 0.75	118.96	1.21	12.30	0.00
SNIP7x5	500	78.62 ± 1.37	80.50 ± 0.52	118.92	1.32	12.50	0.00
SNIP7x5	1000	80.10 ± 0.70	79.99 ± 0.21	149.78	2.05	13.20	0.00
SNIP7x5	2000	79.50 ± 1.09	80.42 ± 0.53	218.65	3.29	12.10	0.70
SNIP7x5	5000	80.34 ± 0.45	80.14 ± 0.31	630.75	7.17	10.10	0.00
SNIP4x9	50	9.96 ± 1.02	11.16 ± 0.26	328.22	0.66	9.20	2.00
SNIP4x9	100	9.47 ± 0.81	10.95 ± 0.12	273.13	1.17	9.90	2.00
SNIP4x9	200	10.92 ± 0.55	11.02 ± 0.17	129.03	0.99	10.40	4.90
SNIP4x9	500	10.45 ± 0.38	10.94 ± 0.05	580.33	1.42	9.60	6.70
SNIP4x9	1000	10.70 ± 0.24	10.91 ± 0.03	567.63	1.84	10.00	11.50
SNIP4x9	2000	10.55 ± 0.18	10.90 ± 0.05	787.59	2.35	10.10	11.60
SNIP4x9	5000	10.89 ± 0.15	10.82 ± 0.16	501.64	5.51	10.10	5.30
SNIP10x10	50	88.28 ± 1.86	88.75 ± 0.34	133.44	5.02	26.00	3.20
SNIP10x10	100	88.09 ± 2.92	88.52 ± 0.38	140.34	6.75	20.90	2.60
SNIP10x10	200	87.08 ± 1.83	88.55 ± 0.33	141.34	7.60	17.70	1.60
SNIP10x10	500	88.39 ± 0.85	88.20 ± 0.21	210.54	12.02	17.40	2.20
SNIP10x10	1000	87.90 ± 0.40	88.15 ± 0.12	362.42	21.47	17.80	1.80
SNIP10x10	2000	88.03 ± 0.41	88.09 ± 0.21	960.40	30.79	16.40	0.60
SNIP20x20	50	173.82 ± 3.88	192.03 ± 7.41	1098.74	24.03	86.50	2.30
SNIP20x20	100	180.36 ± 4.09	188.51 ± 3.25	2640.75	33.03	71.60	3.00
SNIP20x20	200	180.81 ± 2.87	187.00 ± 4.67	4767.96	36.47	66.60	3.50
SNIP20x20	500	182.38 ± 1.72	184.67 ± 3.28	21253.54	38.38	39.00	3.60

Table 1: Statistical bounds on optimal solution value and required computational effort.

Two-stage optimization problems arise in very general settings. Generally, we have

$$\text{MP}: \quad z^*_{\text{MP}} \overset{\text{def}}{=} \min_x \mathbb{E}_\omega\left[Q(x,\omega)\right] + g(x), \text{ s.t. } Ax = b, \; x \geq 0,$$

where $g(x)$ is a deterministic function of x, and $Q(x,\omega)$ represents the optimal objective function value of the second-stage problem:

$$\text{P}: \quad Q(x,\omega) \overset{\text{def}}{=} \min_y q(\omega)^T y, \text{ s.t } T(\omega)x + W(\omega)y = h(\omega), \; y \geq 0.$$

Here $q(\omega) \in \mathbb{R}^n, T(\omega) \in \mathbb{R}^{\ell \times m}, W(\omega) \in \mathbb{R}^{\ell \times n}$, and $h(\omega) \in \mathbb{R}^\ell$ may be random (functions of the realization ω). When $g(x) = c^T x$ and $W(\omega)$ is deterministic, we have a two-stage stochastic linear program with fixed recourse.

In many practical problems the number of possible scenarios K is prohibitively large, so a Monte Carlo approximation of MP is used. We will refer to this approach as sample path optimization, or sample average approximation. The idea is to draw N realizations (sample paths) and optimize over this representative sample. More specifically, let $MP_N(\omega_1, \ldots, \omega_N)$ denote a realization of the N-sample path problem. That is,

$$\text{MP}_N: \quad z^*_{\text{MP}_N(\omega_1,\ldots,\omega_N)} \overset{\text{def}}{=} \min_x N^{-1} \sum_{i=1}^N Q(x,\omega_i) + g(x). \text{ s.t. } Ax = b, \; x \geq 0.$$

The solution to MP_N is used to approximate the solution to the original problem MP.

Different mechanisms are available for selecting the sample used to generate the approximating problem MP_N For instance, we may use independent sampling (IS), antithetic variates (AV), and Latin Hypercube sampling (LH). Suppose X is a random element of the data $\{q, T, W, h\}$ having invertible cdf F, so $Q_i(x,\omega_i)$ is a function of $X(\omega_i)$. Under independent sampling we generate N independent values $\{U_1, \ldots, U_N\}$ uniformly distributed on [0,1], and we use $X(\omega_i) = F^{-1}(U_i)$ to compute $Q_i(x,\omega_i)$. Under AV, rather than drawing N independent numbers, we draw $N/2$ antithetic pairs $\{(U_i, 1-U_i), i = 1, 2, \ldots, N/2\}$ to obtain $\{U_1, \ldots, U_N\}$. Under LH, the interval [0,1] is divided into N segments, $[(i-1)/N, i/N]$, $i = 1\ldots, N$, and a sample is generated uniformly from each segment. These samples are shuffled to obtain $\{U_1, \ldots, U_N\}$.

Our work in this proposal dealt with the impact of the sampling procedure on the use of $z^*_{\text{MP}_N}$ as an estimator for z^*_{MP}. As with any statistical estimation problem, two important measures of performance are the estimator's variance and bias, which are combined as the mean squared error (MSE). (The MSE is equal to the bias squared plus the variance.) The relative contribution of variance and bias to MSE is problem- and parameter-specific.

The AV, and LH sampling procedures are usually prescribed for reducing variance. Other authors have investigated the use of AV and other techniques to reduce the variance of

$$N^{-1} \sum_{i=1}^N Q_i(x,\omega_i) + g(x),$$

which is an unbiased estimate of $\mathbb{E}_\omega\left[Q(x,\omega)\right] + g(x)$ for an arbitrarily chosen value of x. Here we are concerned with estimating z^*_{MP}, which is $\mathbb{E}_\omega\left[Q(x,\omega)\right] + g(x)$ evaluated at x^*_{MP}, the unknown optimal solution to MP. Our work was able to derive analytic expressions for the variance of the estimator $z^*_{\mathrm{MP}_N}$ in the context of the newsvendor problem. We show this variance is reduced under LH, but the effect of AV depends on the problem parameters. We use this example to provide insight to our computational results, where the variance reduction benefits of LH and AV are shown to be highly problem dependent.

In addition to variance reduction, we show the AV and LH sampling procedures may also reduce the bias of $z^*_{\mathrm{MP}_N}$. Under fairly general conditions, the solution to MP_N approaches that of MP with probability 1 as the number of realizations N increases. However, the solution to MP_N is biased in the sense that the expectation of the optimal objective function value of MP_N is less than that of MP:

$$\mathbb{E}_{(\omega_1,\ldots,\omega_N)}\left[z^*_{\mathrm{MP}_{N(\omega_1,\ldots,\omega_N)}}\right] \leq \mathbb{E}_{(\omega_1,\ldots,\omega_{N+1})}\left[z^*_{\mathrm{MP}_{N+1(\omega_1,\ldots,\omega_{N+1})}}\right] \leq z^*_{\mathrm{MP}}\ \forall N. \tag{1}$$

A related issue is that the optimal solution $x^*_N(\omega_1,\ldots,\omega_N)$ of MP_N may be suboptimal with respect to the objective function $\mathbb{E}_\omega\left[Q(x,\omega)\right] + g(x)$ of MP. We refer to:

$$\mathbb{E}_\omega\left[Q(x^*_{\mathrm{MP}_{N(\omega_1,\ldots,\omega_N)}},\omega)\right] + g\left(x^*_{\mathrm{MP}_{N(\omega_1,\ldots,\omega_N)}}\right) \tag{2}$$

as the *actual* cost of the sample path problem and $z^*_{\mathrm{MP}_N}$ as the *perceived* cost.

Our work was able to show that in the context of the newsvendor problem, AV and LH bring the both perceived and actual costs closer to z^*_{MP}. (The effect on the perceived cost is equivalent to reducing the bias of $z^*_{\mathrm{MP}_N}$.) To our knowledge, this was the first quantification of bias reduction via improved sampling for stochastic programs. This example is again used to motivate our computational results, where we examine bias reduction in a number of computational examples. This computational work extends a related paper **(author?)** [16], which examines the impact of LH on the bias of $z^*_{\mathrm{MP}_N}$ and on an upper bound for z^*_{MP} with a set of empirical examples.

3 Mixed Integer Nonlinear Programs

Many decision problems in scientific, engineering, and public sector applications involve both discrete decisions and nonlinear system dynamics that affect the quality of the final design or plan. Mixed-integer nonlinear programming (MINLP) problems combine the difficulty of optimizing over discrete variable sets with the challenges of handling nonlinear functions. MINLP is one of the most flexible modeling paradigms available for optimization problems. Key applications that MINLP can accurately model include optimal power flow problems; network design and expansion; reactor reloading models; and energy storage system design problems. Unfortunately, the wealth of applications that can be accurately modeled using MINLP is not yet matched by the capability of even the best solvers for this important class of problems. Algorithms and software often cannot provide acceptable solutions to practically sized instances in reasonable computing time. Our work in this proposal was to make progress towards transform MINLP into a paradigm that can not only model wide varieties of important problems, but also deliver practical solutions to them.

Our first fundamental contribution was the design and implementation of a software package FilMINT, an implementation of the LP/NLP-BB algorithm for convex MINLP. The software is among the most effective available solvers for convex instances **(author?)** [1].

A second main contribution is from strong reformulation techniques for MINLP. In **(author?)** [8], we gave explicit characterization of the convex hull of the union of a point and a bounded convex set defined by analytic functions and showed that for many classes of problems, the convex hull can be expressed via conic quadratic constraints, and thus relaxations can be solved via second-order cone programming. We applied our results to develop tight formulations of mixed integer nonlinear programs in which the nonlinear functions are separable and convex and in which indicator variables play an important role. In particular, we gave computational results for three applications – quadratic facility location, network design with congestion, and portfolio optimization with buy-in thresholds – that showed the power of the reformulation technique.

Continuing work in this domain includes the design of new primal heuristics for MINLP [3] and formulating and solving a MINLP for the design of a thermal insulation system [2].

3.1 Outer-Approximation Based Methods

Mixed-integer nonlinear programs are conveniently expressed as

$$z_{\text{MINLP}} = \min_{x \in X, y \in Y \cap \mathbb{Z}^p} \{f(x,y) \mid g(x,y) \leq 0\}, \tag{MINLP}$$

where $f : \mathbb{R}^{n+p} \rightarrow \mathbb{R}$ and $g : \mathbb{R}^{n+p} \rightarrow \mathbb{R}^m$ are twice continuously differentiable functions; x and y are continuous and discrete variables, respectively; and X and Y are compact polyhedral subsets of $\mathbb{R}^n$ and $\mathbb{R}^p$, respectively.

In this proposal, we created an implementation of the LP-NLP based branch-and-bound method of Quesada and Grossmann. In this method, an outerapproximation of the feasible region is formed such as

$$\min \eta \tag{MINLP-OA}$$

subject to

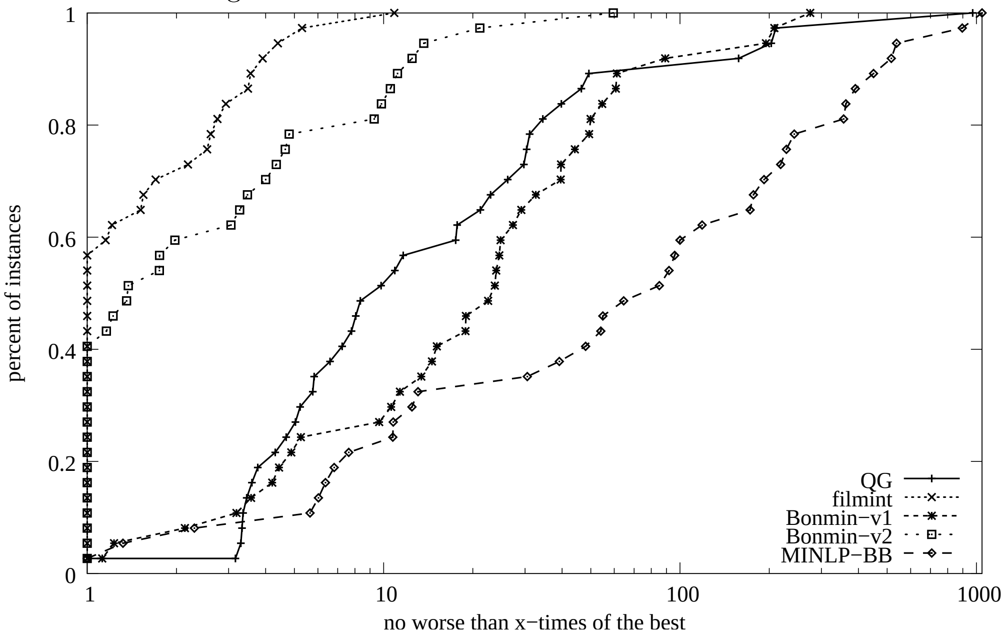
$$\begin{aligned}
f(x^k, y^k) + \nabla f(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} &\leq \eta \qquad \forall k = 1, \ldots K & (3)\\
g_i(x^k, y^k) + \nabla g_i(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} &\leq 0 \qquad \forall i \in M, \forall k = 1, 2, \ldots K & (4)\\
x &\in X \\
y &\in Y,
\end{aligned}$$

where the (x^k, y^k) are obtained in an iterative manner as described below. (MINLP-OA) is a MILP problem that is solved using a branch-and-bound method. At each node of the branch and bound tree, certain components of y may have their bounds fixed, and a relaxation (MINLP-OA-R) is formed by relaxing the integrality requirements on the variables y. If an integer feasible solution y^k is found when solving (MINLP-OA-R), the following (continuous) nonlinear program is solved:

$$\min\{f(x, y^k) \mid g_i(x, y^k) \leq 0 \ \forall i \in M, x \in X\}. \tag{NLP(y^k)}$$

The solution of NLP(y^k) yields a solution x^k, and the objective function $f(x,y)$ and constraints $g_i(x,y)$ are then linearized around (x^k, y^k), resulting in inequalities of the form (3) and (4). These inequalities are added to (MINLP-OA-R), and the branch and bound procedure continues.

Figure 1: Profile of MINLP Solver Performance



This proposal created a solver called FilMINT by customizing this algorithm using two existing software toolkits. FilMINT was created by combining the powerful mixed integer linear programming package MINTO with the state-of-the-art NLP software package filterSQP. The algorithm was further augmented with advanced IP features, such as cutting planes, preprocessing, primal heuristics, pseudo-cost-based branching, and adaptive node selection strategy. Additionally, improved point-selection schemes were developed for the linearizations required of the algorithm. The work has been extremely productive, as the performance of the FilMINT solver is competitive with all other software available for this class of problems. This is depicted graphically in Figure 1, a performance profile of the solution time on 37 moderately difficult instances taken from a variety of well-known test sites for instances of MINLP. Performance profiles were introduced to graphically depicting the relative performance of solvers. Lines in the graph that are high denote the "best" solvers, and so it is clear from the figure that the noncommercial solvers FilMINT and Bonmin-v2 are the most effective for these instances. The solver `QG` in the figure is the textbook implementation of LP-NLP based branch-and-bound method, without additional MIP enhancements, and `MINLP-BB` is an implementation of a traditional branch-and-bound method for MINLP.

We note that the performance of Bonmin (version 1) is superior to the commercial packages DICOPT and SBB. Moreover, a portion of the improvement in the solver Bonmin from (v1) to (v2) is due to the implementation of cut management ideas developed by the FilMINT team.

3.2 Strong Formulations

When solving mixed-integer linear programs (MILP), it is well-known that having a *strong formulation*, or a formulation whose relaxation is close to the convex hull of feasible solutions, is of the utmost importance. It stands to reason that having a strong formulation is equally important to solve MINLP. Since MINLP contains nonlinear constraints, and most algorithms for solving

MINLP have a mechanism for solving (continuous) NLPs as subproblems, it is natural to consider strong (re)-formulations of MINLP that introduce additional nonlinearities to the problem. In this research, we explored the use of nonlinear reformulations and nonlinear cuts for MINLP. As one example, we studied MINLPs that are driven by a collection of indicator variables where each indicator variable controls a subset of the decision variables.

A generic indicator-induced {0-1}-MINLP can be written as

$$z^* \stackrel{\text{def}}{=} \min_{(x,z)\in X\times(Z\cap\mathbb{B}^{|I|})} \{c^T x + d^T z \mid g_j(x,z) \leq 0 \ \forall j \in M,\ (x_{V_i}, z_i) \in S_i \ \forall i \in I\}, \tag{5}$$

where z are the indicator variables, x are the continuous variables and x_{V_i} denotes the collection of continuous variables (i.e. $x_j,\ j \in V_i$) controlled by the indicator variable z_i. Sets $X \subseteq \mathbb{R}^n$ and $Z \subseteq \mathbb{R}^{|I|}$ are polyhedral sets of appropriate dimension and S_i is the set of points that satisfy all constraints associated with the indicator variable z_i:

$$S_i \stackrel{\text{def}}{=} \left\{ (x_{V_i}, z_i) \in \mathbb{R}^{|V_i|} \times \mathbb{B} \ \middle|\ \begin{array}{ll} x_{V_i} = \hat{x}_{V_i} & \text{if } z_i = 0 \\ x_{V_i} \in \Gamma_i & \text{if } z_i = 1 \end{array} \right\},$$

where

$$\Gamma_i \stackrel{\text{def}}{=} \{x_{V_i} \in \mathbb{R}^{|V_i|} \mid f_j(x_{V_i}) \leq 0 \ \forall j \in C_i,\ u_k \geq x_k \geq \ell_k \ \forall k \in V_i\}$$

is bounded for all $i \in I$.

A first result obtained was to present a convex hull description of the following set:

$$Q = \Big\{ w \in \mathbb{R},\ x \in \mathbb{R}^n_+,\ z \in \mathbb{B}^n : w \geq \sum_{i=1}^n q_i x_i^2,\ u_i z_i \geq x_i \geq l_i z_i,\ i = 1, 2, \ldots, n \Big\}, \tag{6}$$

where $q_i \in \mathbb{R}_+$ and $u_i, l_i \in \mathbb{R}$ for all $i = 1, 2, \ldots, n$. The set Q appears in a number of non-linear mixed-integer programs as a substructure.

To understand the set Q, we first studied a simpler mixed-integer set with only 3 variables, which can be obtained by setting $n = 1$ and $q_1 = 1$ in (6). Let

$$S = \Big\{ (x, y, z) \in \mathbb{R}^2 \times \mathbb{B} : y \geq x^2,\ \ uz \geq x \geq lz,\ x \geq 0 \Big\}, \tag{7}$$

where $u, l \in \mathbb{R}$. A preliminary first result is that the convex hull of S is given by

$$\text{conv}(S) = S^c = \big\{ (x, y, z) \in \mathbb{R}^3 : yz \geq x^2,\ uz \geq x \geq lz,\ 1 \geq z \geq 0,\ x, y \geq 0 \big\}.$$

Using concepts prevalent in the mixed integer *linear* programming literature, this result can be lifted to a higher-dimensional space. Consider the following extended formulation of Q

$$\bar{Q} \stackrel{\text{def}}{=} \Big\{ w \in \mathbb{R},\ x \in \mathbb{R}^n, y \in \mathbb{R}^n, z \in \mathbb{R}^n : w \geq \sum_i q_i y_i, (x_i, y_i, z_i) \in S_i,\ \ i = 1, 2, \ldots, n \Big\},$$

where S_i has the same form as the set S in (7), except the bounds u and l are replaced with u_i and l_i. Note that if $(w, x, y, z) \in \bar{Q}$ then $(w, x, z) \in Q$, and therefore $\text{proj}_{(w,x,z)}(\bar{Q}) \subseteq Q$. On the other hand, for any $(w, x, z) \in Q$, letting let $y'_i = x_i^2$ gives a point $(w, x, y', z) \in \bar{Q}$. Therefore, $\bar{Q}$ is indeed an extended formulation of Q, or, in other words, $Q = \text{proj}_{(w,x,z)}(\bar{Q})$.

Using this insight, we were able to derive the convex hull of $\bar{Q}$.

$$\bar{Q}^c = \Big\{w \in \mathbb{R},\, x \in \mathbb{R}^n_;\, y \in \mathbb{R}^n,\, z \in \mathbb{R}^n \;:\; w \geq \sum_i q_i y_i, (x_i, y_i, z_i) \in S_i^c,\; i = 1, 2, \ldots, n\Big\}.$$

We also were able to give a convex hull description in the original space of variables:

$$Q^c = \Big\{(w, x, z) \in \mathbb{R}^{2n+1} : w \prod_{i \in S} z_i \geq \sum_{i \in S} q_i x_i^2 \prod_{l \in S \setminus \{i\}} z_l,\; S \subseteq \{1, 2, \ldots, n\} \qquad \text{(II)}$$
$$u_i z_i \geq x_i \geq l_i z_i, \quad x_i \geq 0, \quad i = 1, 2, \ldots, n\Big\}.$$

In is interesting that these (exponentially many) inequalities are both necessary and sufficient to describe the convex hull of Q in the space of the original variables: $\text{conv}(Q) = Q^c = \text{proj}_{(w,x,z)}(\bar{Q}^c)$.

An important generalization of the previous results is to the case of the convex hull of a point $\bar{x} \in \mathbb{R}^n$ and a bounded convex set defined by analytic functions. In other words, using an indicator variable $z \in \{0, 1\}$, define $W^0 = \big\{(x, z) \in \mathbb{R}^{n+1} \;:\; x = \bar{x},\; z = 0\big\}$, and

$$W^1 = \big\{(x, z) \in \mathbb{R}^{n+1} \;:\; f_i(x) \leq 0 \text{ for } i \in I,\; u \geq x - \bar{x} \geq l,\; z = 1\big\}$$

where $u, l \in \mathbb{R}^n_+$, and $I = \{1, \ldots, t\}$. We are interested in the convex hull of $W = W^1 \cup W^0$. Clearly, both W^0 and W^1 are bounded and W^0 is a convex set. Furthermore, if W^1 is also convex then

$$\text{conv}(W) = \{p \in \mathbb{R}^{n+1} \;:\; p = \alpha p^1 + (1 - \alpha)p^0,\; p^1 \in W^1,\; p^0 \in W^0,\; 1 \geq \alpha \geq 0\}.$$

The main result is a description of this set in the space of original variables:

Theorem 1 *If* W^1 *is convex, then* $\text{conv}(W) = W^- \cup W^0$, *where*

$$W^- \;=\; \Big\{(x, z) \in \mathbb{R}^{n+1} :\; f_i(x/z) \leq 0 \textit{ for } i \in I, \quad uz \geq x \geq lz, \quad 1 \geq z > 0\Big\}.$$

Corollary 2 $\text{conv}(W) \;=\; \text{closure}(W^-)$.

This work has already borne great fruit. As an example, the perspective reformulation was applied on a quadratic-cost uncapacitated facility location problem introduced by Günlük, Lee, and Weismantel in 2007. The largest of their test instances was solved by Lee, requiring 21,206 CPU-seconds and 29,277 search nodes. The same instance was solved by using the strong-reformulation in only 44 branch-and-bound nodes and 23 CPU-seconds, a speedup factor of more than 900. This improvement is impressive, but not surprising to MIP researchers, who for decades have understood the importance of strong formulations.

4 Mixed Integer Linear Programming

Our primary work in the area of mixed integer linear programming (MILP) has been on the novel combination of isomorphism-pruning and enumeration to improve bounds on an important problem in coding theory [10], and on the development of a new branching method called *orbital branching* in [18]. The generalization of this method to branching to constraints [19] has solved to optimality instances asking for the smallest incidence width of Steiner Triple Systems that have been unsolved for decades.

To concisely present the results of work, consider a covering MILP of the form

$$\min_{x \in \mathcal{F}} \{e^T x\}, \text{ with } \mathcal{F} \overset{\text{def}}{=} \{x \in \{0,1\}^n \mid Ax \geq e\}, \tag{8}$$

where $A \in \{0,1\}^{m \times n}$ and e is a vector of ones of conformal size. The restriction of our work to set covering problems is mainly for notation convenience, but also of practical significance, since many important set covering problems contain a great deal of symmetry.

The focus of our MILP research is on cases when (8) is highly-symmetric, a concept that can be formalized as follows. Let Π^n be the set of all permutations of $I^n = \{1, \ldots, n\}$, so that Π^n is the symmetric group of I^n. For a vector $\lambda \in \mathbb{R}^n$, the permutation $\pi \in \Pi^n$ acts on λ by permuting its coordinates, a transformation that we denote as

$$\pi(\lambda) = (\lambda_{\pi_1}, \lambda_{\pi_2}, \ldots \lambda_{\pi_n}).$$

Since all objective function coefficients in (8) are identical, permuting the coordinates of a solution does not change its objective value, i.e. $e^T x = e^T(\pi(x)) \forall x \in \mathcal{F}$. The *symmetry group* $\mathcal{G}$ of (8) is the set of permutations of the variables that maps each feasible solution onto a feasible solution of the same value. In this case,

$$\mathcal{G} \overset{\text{def}}{=} \{\pi \in \Pi^n \mid \pi(x) \in \mathcal{F} \quad \forall x \in \mathcal{F}\}.$$

Typically, the symmetry group $\mathcal{G}$ of feasible solutions is not known. However, by closely examining the structure of the problem, many of the permutations making up the group can be found, and this subgroup of the original group $\mathcal{G}$ can be employed in its place. Specifically, given a permutation $\pi \in \Pi^n$ and a permutation $\sigma \in \Pi^m$, let $A(\pi, \sigma)$ be the matrix obtained by permuting the columns of A by π and the rows of A by σ, i.e. $A(\pi, \sigma) = P_\sigma A P_\pi$, where P_σ and P_π are permutation matrices. Consider the set of permutations

$$\mathcal{G}(A) \overset{\text{def}}{=} \{\pi \in \Pi^n \mid \exists \sigma \in \Pi^m \text{ such that } A(\pi, \sigma) = A\}.$$

For any $\pi \in \mathcal{G}(A)$, if $\hat{x} \in \mathcal{F}$, then $\pi(\hat{x}) \in \mathcal{F}$, so $\mathcal{G}(A)$ forms a subgroup of $\mathcal{G}$, and the group $\mathcal{G}(A)$ is the group used in our computations. The group $\mathcal{G}(A)$ can act on an arbitrary set of points $\mathcal{Z}$, but in our work, it acts on either $\mathbb{R}^n$ or $\{0,1\}^n$.

A powerful idea called *isomorphism pruning* examines the symmetry group of the problem in order to prune isomorphic subproblems of the enumeration tree. The branching method introduced as a result of our work, *orbital branching*, also uses the symmetry group of the problem. However, instead of examining this group to ensure that an isomorphic node will never be evaluated, the group is used to guide the branching decision. At the cost of potentially evaluating isomorphic subproblems, orbital branching allows for considerably more flexibility in the choice of branching entity than isomorphism pruning. Furthermore, orbital branching can be easily incorporated within a standard MIP solver and even exploit problem symmetry that may only be locally present at a nodal subproblem.

For a point $z \in \mathcal{Z}$, the *orbit* of z under the action of the group Γ is the set of all elements of $\mathcal{Z}$ to which z can be sent by permutations in Γ, i.e.,

$$\text{orb}(\Gamma, z) \stackrel{\text{def}}{=} \{z' \in \mathcal{Z} \mid \exists \pi \in \Gamma \text{ such that } z' = \pi(z)\} = \{\pi(z) \mid \pi \in \Gamma\}.$$

Orbital branching is based on the following simple observations. If $\lambda^T x \leq \lambda_0$ is a valid inequality for (8) and $\pi \in \mathcal{G}$, then $\pi(\lambda)^T x \leq \lambda_0$ is also a valid inequality for (8). In constraint orbital branching, given an integer vector $(\lambda, \lambda_0) \in \mathbb{Z}^{n+1}$, a base branching disjunction of the form

$$(\lambda^T x \leq \lambda_0) \vee (\lambda^T x \geq \lambda_0 + 1)$$

is used, simultaneously considering all symmetrically equivalent forms of $\lambda x \leq \lambda_0$. Specifically, the branching disjunction is

$$\left(\bigvee_{\mu \in \text{orb}(\mathcal{G}, \lambda)} \mu^T x \leq \lambda_0 \right) \vee \left(\bigwedge_{\mu \in \text{orb}(\mathcal{G}, \lambda)} \mu^T x \geq \lambda_0 + 1 \right).$$

Further, by exploiting the symmetry in the problem, one need only consider one representative problem for the left portion of this disjunction. That is, either the "equivalent" form of $\lambda x \leq \lambda_0$ holds for one of the members of $\text{orb}(\mathcal{G}, \lambda)$, or the inequality $\lambda x \geq \lambda_0 + 1$ holds for all of them. This is obviously a feasible division of the search space. We have demonstrated that for any vectors $\mu_i, \mu_j \in \text{orb}(\mathcal{G}, \lambda)$, the subproblem formed by adding the inequality $\mu_i^T x \leq \mu_0$ is equivalent to the subproblem formed by adding the inequality $\mu_j^T x \leq \mu_0$. Therefore, we need to keep only one of these representative subproblems, pruning the $|\text{orb}(\mathcal{G}, \lambda)| - 1$ equivalent subproblems. The orbital branching method introduced in our first work [18], is a special case where $(\lambda, \lambda_0) = (e_k, 0)$.

To demonstrate the effectiveness of orbital branching, a series of computational experiments were done. Table 2 shows the result of this computational experiment. The results show that the number of subproblems evaluated by orbital branching and CPU times required to solve the instances are quite comparable. Orbital branching proves to be faster than the powerful commercial solver CPLEX in all but two cases, while in all cases the number of evaluated nodes is remarkably smaller.

5 Conclusions

This report described the progress made on optimization algorithms and software for solving problems containing uncertainty and nonconvexity. A complete list of publications is given in the bibliography section. Support from the Department of Energy under grant agreement DE-FG02-05ER25694 is gratefully acknowledged.

References

[1] K. Abhishek, S. Leyffer, and J. T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. Preprint ANL/MCS-P1374-0906, Mathematics and Computer Science Division, Argonne National Lab, 2006.

	Orbital Branching		**Isomorphism Pruning**		**CPLEX v10.1**	
Instance	Time	Nodes	Time	Nodes	Time	Nodes
cod83	1	25	19	33	391	32077
cod93	167	1361	651	103	**fail**	488136
cod105	242	11	2000	15	1245	1584
cov954	6	249	24	126	9	1514
cov1053	240	9775	35	111	937	99145
cov1054	179	1249	130	108	**fail**	239266
cov1075	152	381	118	169	141	10278
cov1076	1415	31943	3634	5121	**fail**	1179890
f5	45	1125	-	-	1150	54018
sts45	8	4709	31	513	24	51078
sts63	20	5533	120	1247	3414	4974655
sts81	39	6293	68	199	**fail**	12572533

Table 2: Comparison of Orbital Branching, Isomorphism Pruning, and CPLEX v10.1

[2] K. Abhishek, S. Leyffer, and J. T. Linderoth. Modeling without categorical variables: A mixed integer nonlinear program for the optimization of thermal insulation systems. Preprint ANL/MCS-P1434-0607, Mathematics and Computer Science Division, Argonne National Lab, 2007.

[3] K. Abhishek, S. Leyffer, and J. T. Linderoth. Feasibility pump heuristics for mixed integer nonlinear programs. 2008.

[4] M. Freimer, D. Thomas, and J. Linderoth. Reducing bias in stochastic linear programming with sampling methods. Technical Report 05T-002, Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, Pennsylvania, 2005.

[5] B. Gemici, J. T. Linderoth, S. D. Wu, and J. Moore. R&d project portfolio analysis for the semiconductor industry. Technical Report 06T-005, Department of Industrial and Systems Engineering, Lehigh University, 2006.

[6] W. Glankwamdee and J. T. Linderoth. Lookahead branching for mixed integer programming. Technical Report 06T-004, Department of Industrial and Systems Engineering, Lehigh University, 2006.

[7] W. Glankwamdee and J. T. Linderoth. MW: A software framework for combinatorial optimization on computational grids. In E. Talbi, editor, *Parallel Combinatorial Optimization*, pages 239–262. Wiley-Interscience, Hoboken, New Jersey, 2006.

[8] O. Günlük and J. Linderoth. Perspective relaxation of mixed integer nonlinear programs with indicator variables. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *IPCO 2008:*

The Thirteenth Conference on Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science, volume 5035, pages 1–16, 2008.

[9] U. Janjarassuk and J. Linderoth. Reformulation and sampling to solve a stochastic network interdiction problem. *Networks*, 2008. To appear.

[10] J. Linderoth, F. Margot, and G. Thain. Improving bounds for covering designs. Unpublished Working Paper, 2007.

[11] J. Linderoth, F. Margot, and G. Thain. Improving bounds on the football pool problem via symmetry reduction and high-throughput computing. Submitted, 2007.

[12] J. Linderoth, F. Margot, and G. Thain. The teragrid-iron: A natural turf for high-throughput computing. 2007.

[13] J. Linderoth and R. Musmanno. Optimization on grids—optimization for grids. *Parallel Computing*, 32:627–628, 2006.

[14] J. T. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Mathematical Programming, Series B*, 103:251–282, 2005.

[15] J. T. Linderoth and T. K. Ralphs. Noncommercial software for mixed-integer linear programming. In *Integer Programming: Theory and Practice*, pages 253–303. CRC Press Operations Research Series, 2005.

[16] J. T. Linderoth, A. Shapiro, and S. J. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142:219–245, 2006.

[17] C. Novoa, R. Berger, J. T. Linderoth, and R. Storer. A set-partitioning-based model and solution procedures for the stochastic vehicle routing problem. Technical Report 06T-008, Department of Industrial and Systems Engineering, Lehigh University, 2006.

[18] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. In M. Fischetti and D. Williamson, editors, *IPCO 2007: The Twelfth Conference on Integer Programming and Combinatorial Optimization*, pages 104–118. Springer, 2007.

[19] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Constraint orbital branching. In *IPCO 2008: The Thirteenth Conference on Integer Programming and Combinatorial Optimization*, 2008. To appear.