Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

WILLIAM COOK[1]
THORSTEN KOCH[2]
DANIEL E. STEFFY[1]
KATI WOLTER[3]

An Exact Rational Mixed-Integer Programming Solver

[1] School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA. Supported by NSF Grant CMMI-0726370, ONR Grant N00014-08-1-1104.
[2] Zuse Institute Berlin, Germany.
[3] Zuse Institute Berlin, Germany. Research funded by DFG Priority Program 1307 “Algorithm Engineering”.

ZIB-Report 11-07 (March 2011)

An Exact Rational Mixed-Integer Programming Solver

William Cook*[1], Thorsten Koch[2], Daniel E. Steffy*[1], and Kati Wolter**[2]

[1] School of Industrial and Systems Engineering,
Georgia Institute of Technology, Atlanta, GA,
bico@isye.gatech.edu,desteffy@gatech.edu
[2] Zuse Institute Berlin, Germany, {koch,wolter}@zib.de

Abstract We present an exact rational solver for mixed-integer linear programming that avoids the numerical inaccuracies inherent in the floating-point computations used by existing software. This allows the solver to be used for establishing theoretical results and in applications where correct solutions are critical due to legal and financial consequences. Our solver is a hybrid symbolic/numeric implementation of LP-based branch-and-bound, using numerically-safe methods for all binding computations in the search tree. Computing provably accurate solutions by dynamically choosing the fastest of several safe dual bounding methods depending on the structure of the instance, our exact solver is only moderately slower than an inexact floating-point branch-and-bound solver. The software is incorporated into the SCIP optimization framework, using the exact LP solver QSopt_ex and the GMP arithmetic library. Computational results are presented for a suite of test instances taken from the Miplib and Mittelmann collections.

1 Introduction

Mixed-integer programming (MIP) is a powerful and flexible tool for modeling and solving decision problems. Software based on these ideas is utilized in many application areas. Despite their widespread use, few available software packages provide any guarantee of correct answers or certification of results. Possible inaccuracy is caused by the use of floating-point (FP) numbers [14]. FP calculations necessitate the use of built-in tolerances for testing feasibility and optimality, and can lead to calculation errors in the solution of linear-programming (LP) relaxations and in the methods used for creating cutting planes to improve these relaxations.

Due to a number of reasons, for many industrial MIP applications near optimal solutions are sufficient. Cplex, for example, defaults to a relative MIP optimality tolerance of 0.001. Moreover, when data describing a problem arises from imprecise sources, exact feasibility is usually not necessary. Nonetheless,

* Research supported by NSF Grant CMMI-0726370, ONR Grant N00014-08-1-1104.
** Research funded by DFG Priority Program 1307 "Algorithm Engineering".

accuracy is important in many settings. Direct examples arise in the use of MIP models to establish fundamental theoretical results and in subroutines for the construction of provably accurate cutting planes. Furthermore, industrial customers of MIP software request modules for exact solutions in critical applications. Such settings include the following.

- Feasibility problems, e.g., chip verification in the VLSI design process [1].
- Compiler optimization, including instruction scheduling [22].
- Combinatorial auctions [21], where serious legal and financial consequences can result from incorrect solutions.

Optimization software relying exclusively on exact rational arithmetic has been observed to be prohibitively slow, motivating the development of more sophisticated techniques to compute exact solutions. Significant progress has been made recently toward computationally solving LP models exactly over the rational numbers using hybrid symbolic/numeric methods [7,10,12,16,17], including the release of the software package QSopt_ex [6]. Exact MIP has seen less computational progress than exact LP, but significant first steps have been taken. An article by Neumaier and Shcherbina [19] describes methods for safe MIP computation, including strategies for generating safe LP bounds, infeasibility certificates, and cutting planes. The methods they describe involve directed rounding and interval arithmetic with FP numbers to avoid incorrect results.

The focus of this article is to introduce a hybrid branch-and-bound approach for exactly solving MIPs over the rational numbers. Section 2 describes how rational and safe FP computation can be coupled together, providing a fast and general framework for exact computation. Section 3 describes several methods for computing valid LP bounds, which is a critical component of the hybrid approach. Section 4 describes an exact branch-and-bound implementation within SCIP [1,2] and includes detailed computational results on a range of test libraries comparing different dual bounding strategies. The exact solver is compared with an inexact branch-and-bound solver and observed to be only moderately slower.

2 Hybrid Rational/Safe Floating-Point Approach

Two ideas for exact MIP proposed in the literature, and tested to some extent, are the *pure rational approach* [7] and the *safe-FP approach* [9,19]. Both utilize LP-based branch-and-bound. The difference lies in how they ensure the computed results are correct.

In the pure rational approach, correctness is achieved by storing the input data as rational numbers, by performing all arithmetic operations over the rationals, and by applying an exact LP solver [12] in the dual bounding step. This approach is especially interesting because it can handle a broad class of problems: MIP instances described by rational data. However, replacing all FP operations by rational computation will increase running times noticeably. For example, while the exact LP solver QSopt_ex avoids many unnecessary rational computations and is efficient on average, Applegate et al. [7] observed a greater

slowdown when testing an exact MIP solver that relied on rational arithmetic and called QSopt_ex for each node LP computation.

In order to limit the degradation in running time, the idea of the safe-FP approach is to continue to use FP-numbers as much as possible, particularly within the LP solver. However, extra work is necessary to ensure correct decisions in the branch-and-bound algorithm. Correctness of certain computations can be ensured by controlling the rounding mode for FP operations. Valid dual bounds can often be obtained by post-processing approximate LP solutions; this type of safe dual bounding technique has been successfully implemented in Concorde [5] for the traveling salesman problem. A generalization of the method for MIPs is described in [19]. Furthermore, the idea of manipulating the rounding mode can be applied to cutting-plane separation. In [9], this idea was used to generate numerically safe Gomory mixed-integer cuts. Nevertheless, whether the safe-FP approach leads to acceptable running times for general MIPs has not been investigated. Although the safe-FP version of branch-and-bound has great advantages in speed over the pure rational approach, it has several disadvantages. Everything, including input data and primal solutions, is stored as FP numbers. Therefore, correct results can only be ensured for MIP instances that are given by FP-representable data and that have a FP-representable optimal solution if they are feasible. Some rationally defined problems can be scaled to have FP-representable data. However, this is not always possible due to the limited representation of floating-point numbers, and the resulting large coefficients can lead to numerical difficulties. The applicability is even further limited as the safe dual bounding method discussed in [19] requires, in general, lower and upper bounds on all variables. Weakness in the safely generated bound values may also increase the number of nodes processed by the branch-and-bound solver. Additionally, due to numerical difficulties, some branch-and-bound nodes may only be processable by an exact LP solver.

To summarize, the pure rational approach is always applicable but introduces a large overhead in running time while the safe-FP approach is more efficient but of limited applicability.

Since we want to solve MIPs that are given by rational data efficiently and exactly we have developed a version of branch-and-bound that attempts to combine the advantages of the pure rational and safe-FP approaches, and to compensate for their individual weaknesses. The idea is to work with two branch-and-bound processes. The *main process* implements the rational approach. Its result is surely correct and will be issued to the user. The other one serves as a *slave process*, where the faster safe-FP approach is applied. To achieve reasonable running time, whenever possible the expensive rational computation of the main process will be skipped and certain decisions from the faster safe-FP process will be substituted. In particular, safe dual bound computations in the slave process can often replace exact LP solves in the main process. The rational process provides the exact problem data, allows to correctly store primal solutions, and makes exact LP solves possible whenever needed.

Algorithm 1 Branch-and-bound for exactly solving MIPs

Input: (MIP) $\max\{c^T x : x \in P\}$ with $P := \{x \in \mathbb{R}^n : Ax \leq b, x_i \in \mathbb{Z} \text{ for all } i \in I\}$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, and $I \subseteq \{1, \ldots, n\}$.

Output: *Exact* optimal solution $x^\star$ of MIP with objective value $c^\star$ or conclusion that MIP is infeasible ($c^\star = -\infty$).

1. *FP-problem:* Store (FP-MIP) $\max\{\widetilde{c}^T x : x \in \widetilde{P}\}$ with $\widetilde{P} := \{x \in \mathbb{R}^n : \widetilde{A}x \leq \widetilde{b}, x_i \in \mathbb{Z} \text{ for all } i \in I\}$, $\widetilde{A} \in \mathbb{M}^{m \times n}$, $\widetilde{b} \in \mathbb{M}^m$, and $\widetilde{c} \in \mathbb{M}^n$.
2. *Init:* Set $\mathcal{L} := \{(P, \widetilde{P})\}$, $L := -\infty$, x^{MIP} to be empty, and $c^{\text{MIP}} := -\infty$.
3. *Abort:* If $\mathcal{L} = \emptyset$, stop and return x^{MIP} and c^{MIP}.
4. *Node selection:* Choose $(P_j, \widetilde{P}_j) \in \mathcal{L}$ and set $\mathcal{L} := \mathcal{L} \setminus \{(P_j, \widetilde{P}_j)\}$.
5. *Dual bound:* Solve LP-relaxation $\max\{\widetilde{c}^T x : x \in \widetilde{LP}_j\}$ *approximately.*
   (a) If $\widetilde{LP}_j$ is *claimed* to be empty, *safely* check if LP_j is empty.
      i. If LP_j is empty, set $c^\star := -\infty$.
      ii. If LP_j is not empty, solve LP-relaxation $\max\{c^T x : x \in LP_j\}$ *exactly.* Let $x^\star$ be an *exact* optimal LP solution and $c^\star$ its objective value.
   (b) If $\widetilde{LP}_j$ is *claimed* not to be empty, let $x^\star$ be an *approximate* optimal LP solution and compute a *safe* dual bound $c^\star$ with $\max\{c^T x : x \in LP_j\} \leq c^\star$.
6. *Bounding:* If $\overline{c^\star} \leq L$, goto Step 3.
7. *Primal bound:*
   (a) If $x^\star$ is *approximate* LP solution and claimed to be feasible for FP-MIP, solve LP-relaxation $\max\{c^T x : x \in LP_j\}$ *exactly.* If LP_j is *in fact* empty, goto Step 3. Otherwise, let $x^\star$ be an *exact* optimal LP solution and $c^\star$ its objective value.
   (b) If $x^\star$ is *exact* LP solution and *truly* feasible for MIP:
      i. If $c^\star > c^{\text{MIP}}$, set $x^{\text{MIP}} := x^\star$, $c^{\text{MIP}} := c^\star$, and $L := \underline{c^\star}$.
      ii. Goto Step 3.
8. *Branching:* Choose index $i \in I$ with $x_i^\star \notin \mathbb{Z}$.
   (a) Split P_j in $Q_1 := P_j \cap \{x : x_i \leq \lfloor x_i^\star \rfloor\}$, $Q_2 := P_j \cap \{x : x_i \geq \lceil x_i^\star \rceil\}$.
   (b) Split $\widetilde{P}_j$ in $\widetilde{Q}_1 := \widetilde{P}_j \cap \{x : x_i \leq \lfloor x_i^\star \rfloor\}$, $\widetilde{Q}_2 := \widetilde{P}_j \cap \{x : x_i \geq \lceil x_i^\star \rceil\}$.
   Set $\mathcal{L} := \mathcal{L} \cup \{(Q_1, \widetilde{Q}_1), (Q_2, \widetilde{Q}_2)\}$ and goto Step 3.

The complete procedure is given in Alg. 1. The set of FP-representable numbers is denoted by $\mathbb{M}$; lower and upper approximations of $x \in \mathbb{Q}$ are denoted $\underline{x} \in \mathbb{M}$ and $\overline{x} \in \mathbb{M}$, respectively. The slave process, which utilizes the safe-FP approach, works on a MIP instance with FP-representable data. It is set up in Step 1 of the algorithm. If the input data are already FP-representable, both processes solve the same MIP instance, i.e., $\widetilde{P} := P$ and $\widetilde{c} := c$ in Step 1. In principle, this results in employing only the safe-FP approach except for some necessary exact LP solves. Otherwise, an approximation of the MIP with $P \approx \widetilde{P}$, $c \approx \widetilde{c}$ or a relaxation with $P \subseteq \widetilde{P}$, $c = \widetilde{c}$ is constructed; called *FP-approximation* and *FP-relaxation*, respectively. The choice depends on the dual bounding method applied in the slave process (see Sect. 3).