DANIEL E. STEFFY[1]
KATI WOLTER[2]

Valid Linear Programming Bounds for Exact Mixed-Integer Programming

[1] Zuse Institute Berlin, Germany. Research supported by NSF Grant CMMI-0726370, ONR Grant N00014-08-1-1104.
[2] Zuse Institute Berlin, Germany. Research funded by DFG Priority Program 1307 “Algorithm Engineering”.

ZIB-Report 11-08 (March 2011)

Valid Linear Programming Bounds for Exact Mixed-Integer Programming

Daniel E. Steffy, Kati Wolter
Department of Optimization, Zuse Institute Berlin,
Takustr. 7, 14195 Berlin, Germany, {steffy@zib.de, wolter@zib.de}

Fast computation of valid linear programming (LP) bounds serves as an important subroutine for solving mixed-integer programming problems exactly. We introduce a new method for computing valid LP bounds designed for this application. The algorithm corrects approximate LP dual solutions to be exactly feasible, giving a valid bound. Solutions are repaired by performing a projection and a shift to ensure all constraints are satisfied; bound computations are accelerated by reusing structural information through the branch-and-bound tree. We demonstrate this method to be widely applicable and faster than solving a sequence of exact LPs. Several variations of the algorithm are described and computationally evaluated in an exact branch-and-bound algorithm within the mixed-integer programming framework SCIP.

Key words: mixed-integer programming; exact computation
History:

1. Introduction

Software to solve mixed-integer programming (MIP) problems is widely used in both industrial and academic settings. However, due to the use of floating-point arithmetic, most software packages are susceptible to numerical mistakes which can lead to incorrect results. While a degree of numerical error is often tolerated by users in some settings, there are a number of applications where truly correct and exact solutions are desirable or necessary. Such areas include the use of MIP models to establish theoretical results, to verify the correctness of VLSI chip designs, or to determine winners for combinatorial auctions; additional examples are given in (Cook et al., 2011; Steffy, 2011).

Implementing software to solve LPs and MIPs entirely in exact arithmetic can result in a considerable slowdown. Recent work has focused on developing efficient methods to solve LPs problems exactly over the rational numbers using a mix of floating-point and exact computation (Applegate et al., 2007a; Dhiflaoui et al., 2003; Espinoza, 2006; Koch, 2004; Kwappik, 1998). A solver based on these ideas is implemented and studied by Applegate et al. (2007a) as QSopt_ex (Applegate et al., 2007b). These methods exploit the fact that even in the presence of some numerical errors, floating-point LP solvers are often able to find an optimal or near optimal LP basis. Once an optimal LP basis is identified, the exact rational solution can be computed and verified without requiring that the earlier steps of the algorithm were performed exactly.

In (Applegate et al., 2007a) an exact rational MIP solver based on QSopt_ex was tested in which an exact branch-and-bound tree was maintained and each LP encountered was solved exactly. While QSopt_ex was only moderately slower than the floating-point LP solvers, the exact MIP code experienced a more significant slowdown when compared to commercial solvers. In order to improve running times for exact MIP it has been observed that solving the LP relaxation at each node exactly is not always necessary, as long as valid LP bounds can be computed. This idea is discussed by Applegate et al. (2006) and Neumaier and Shcherbina (2004). A recently developed exact rational MIP solver, described by Cook et al. (2011), uses a combination of exact rational arithmetic and safe floating-point computation.

In Section 2, we describe some known methods for generating valid bounds for LP problems. In Section 3, we describe a new algorithm for generating valid LP dual bounds, which we will refer to as the project-and-shift method. A description of our computational experiments is presented in Section 4 and conclusions and future work are discussed in Section 5.

2. Previous Work

The most straightforward way of computing valid LP bounds at nodes of a branch-and-bound tree is to solve each LP relaxation exactly. The node LPs in a branch-and-bound tree are typically solved by the dual simplex algorithm which can be warm started with an optimal basis from the parent node; reoptimization can often be accomplished with a small number of simplex pivots. This type of warm start can also be used when solving node LPs exactly. However, computing exact LP solutions in this way may still be much slower than the floating-point LP solver. Even in the case when the exact LP solver quickly determines the optimal basis by performing additional pivots in floating-point arithmetic, it would still compute an exact solution and verify its optimality at that node, which can be time consuming. The cost associated with computing numerous node LP solutions exactly is an explanation for why the exact MIP solver tested in (Applegate et al., 2007a) experienced a greater relative slowdown than their exact LP solver, when compared to floating-point codes.

Despite the possible disadvantages of solving an exact LP at every node, it is important to recognize that an exact LP solver will be a necessary component of an exact MIP solver and is used to compute exact primal solutions. An exact LP solver also has the advantage that it will provide the tightest valid LP bound at any node of the branch-and-bound tree.

Any feasible dual solution gives a valid bound on the primal LP objective value. In many cases an approximate dual solution can be corrected to generate a valid bound in this manner. If all primal variables have finite upper and lower bounds then this structure allows any approximate dual solution to be corrected by adjusting the dual variables corresponding to the primal variable bounds. This idea was used to compute valid dual LP bounds within the Concorde software package which is designed to solve Traveling Salesman Problem (TSP) instances by branch-and-cut where each variable is bounded by zero and one (Applegate et al., 2006). Neumaier and Shcherbina (2004) described this procedure more generally for MIPs having finite upper and lower bounds on all primal variables. Consider the following primal dual pair of LPs:

$$\begin{array}{llll} \textbf{Primal:} & & \textbf{Dual:} & \\ \max & c^T x & \min & b^T y - l^T z_l + u^T z_u \\ \text{s.t.} & Ax \le b & \text{s.t.} & A^T y \; - I z_l \; + I z_u \; = c \\ & l \le x \le u & & y, z_l, z_u \ge 0 \end{array}$$

Any approximate dual solution $\tilde{y}, \tilde{z}_l, \tilde{z}_u \ge 0$ can be corrected to be exactly dual feasible by increasing z_l, z_u. If $r = c - A^T\tilde{y} + I\tilde{z}_l - I\tilde{z}_u$ is the error of the approximate solution, then a feasible solution is given by: $(y, z_l, z_u) = (\tilde{y}, \tilde{z}_l + r^+, \tilde{z}_u + r^-)$. Where $r_i^+ = \max(r_i, 0)$ and $r_i^- = \max(-r_i, 0)$. This gives $b^T y - l^T z_l + u^T z_u$ as a valid upper bound on the primal objective. Since this dual bounding method corrects approximate dual solutions using the dual variables coming from primal bound constraints we will call it the *primal-bound-shift method.* The difference between the bound and the objective value of the approximate dual solution will be small if the approximate dual solution does not violate the constraints by a large amount and the bounds l, u on the primal variables are not large.

Proposition 2.1. *Let $\tilde{y}, \tilde{z}_l, \tilde{z}_u \geq 0$ be an approximate dual solution, with cost $b^T\tilde{y} - l^T\tilde{z}_l + u^T\tilde{z}_u$, then the bound computed by the primal-bound-shift method described above will be $-l^Tr^+ + u^Tr^-$ larger than the cost of the approximate dual solution (if computed exactly).*

Proof. Subtracting the objective value of the approximate solution from the objective value of the corrected solution gives: $(b^Ty - l^Tz_l + u^Tz_u) - (b^T\tilde{y} - l^T\tilde{z}_l + u^T\tilde{z}_u) = -l^Tr^+ + u^Tr^-$. □

Neumaier and Shcherbina observed that exact precision arithmetic can be entirely avoided when computing r and the bound by using floating-point computation and interval arithmetic (or directed rounding if the problem is described in floating-point representable numbers). The strength and simplicity of computing this bound suggests that it will be an excellent choice when tight primal variable bounds are available. The drawback is that if some variable bounds are very large or missing then it could produce weak or infinite bounds. We found that in our test set, 31 out of 59 problems were missing at least some variable bounds, which could lead to failure of this method.

Some recent studies (Althaus and Dumitriu, 2009; Jansson, 2004; Keil and Jansson, 2006) have looked at solving or detecting feasibility of LPs using interval methods. The methods presented in these articles are more general and sophisticated than the primal-bound-shift method. Althaus and Dumitriu (2009) describe an algorithm to certify feasibility and produce valid bounds for LPs. Their algorithm identifies the implied equalities of an LP and then, using safe interval methods, corrects the interval solution to satisfy all of the constraints by shifting it toward the relative interior of the polyhedron. They implemented a version of the algorithm to certify feasibility of problems and experienced a high rate of success. A variant of their algorithm for computing valid LP bounds is also described. Their method does not require special assumptions on the problem structure and most computations can be performed using fast interval methods. However, it requires the solution of an auxiliary problem to identify the implied equalities each time a bound is computed and can potentially fail when numerical problems are encountered.

3. Project-and-Shift

The methods described in the previous section determine a valid dual solution, or an interval containing a valid dual solution by correcting an approximate dual solution. Similarly, the method presented in this section will generate valid bounds by repairing approximate dual