CS389L: Automated Logical Reasoning

Relaxation Based Techniques for ILP: Cuts-from-Proofs

Işıl Dillig

New Topic: Relaxation-based methods

- Recall: There are two classes of techniques for determining satisfiability over integers
  1. Elimination-based techniques: Omega Test, Cooper's method
  2. Relaxation-based techniques: Branch-and-bound, Gomory cuts, Cuts-from-Proofs
- Today's topic: Relaxation-based methods

Relaxation-Based Techniques Overview

Main idea behind relaxation-based techniques for solving $A\vec{x} \leq \vec{b}$

1. First, ignore integrality requirement
2. Use LP algorithm (e.g., Simplex) to obtain rational solution
3. No rational solution $\Rightarrow$ unsatisfiable
4. Rational solution also integer $\Rightarrow$ satisfiable
5. Add additional constraint to guide search for integer solution
6. Repeat until (3) or (4) holds

Popular Relaxation-Based Techniques

Existing relaxation-based techniques:

- Branch-and-bound (B&B): simplest relaxation-based technique
- Gomory cuts: uses reasoning performed by Simplex to infer valid inequalities
- Branch-and-cut: combines B&B and Gomory cuts
- Cuts-from-proofs: technique developed during my PhD

We'll only talk about B&B and Cuts-from-Proofs

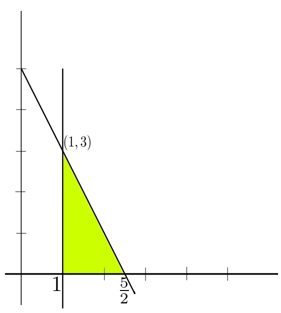
Branch and Bound

- B&B is simplest relaxation-based technique
- Suppose Simplex yields fractional solution
- Means there exists some variable x_i assigned to non-integer f_i
- B&B constructs two subproblems:

$$A\vec{x} \leq \vec{b} \wedge x_i \leq \lfloor f_i \rfloor \quad A\vec{x} \leq \vec{b} \wedge x_i \geq \lceil f_i \rceil$$

- Original problem has a solution iff either subproblem has integer solution.

An Example



- Consider the system:

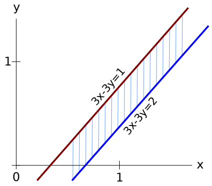
$$\begin{array}{rcr} 2x + y & \leq & 5 \\ -x & \leq & -1 \\ -y & \leq & 0 \end{array}$$

- Suppose Simplex yields $(\frac{5}{2}, 0)$
- B&B constructs subproblem with $x \geq 3$
- This problem has no rational solution
- B&B constructs subproblem with $x \leq 2$
- All vertices integers, thus Simplex yields integer solution $\Rightarrow$ satisfiable

Problem with Branch and Bound

- B&B very simple, but in many cases, it does not terminate!
- Consider the following set of inequalities:

$$\begin{array}{rcl} -3x+3y+z & \leq & -1 \\ 3x-3y+z & \leq & 2 \\ z & = & 0 \end{array}$$



- No integer points, but infinitely many rational solutions

Limitation of Branch and Bound

- As example illustrates, B&B does not have termination guarantees
- It always terminates if feasible region is bounded
- If feasible region is unbounded and does not contain integer points, B&B does not terminate
- In some settings, such as operations research, finite feasible region assumption may be sensible
- But if we want to use B&B to determine satisfiability, this assumption is unrealistic

Limitation of Branch and Bound, cont

- Good news: If there is integer solution to system $A\vec{x} \leq \vec{b}$, it must lie within a pre-computable finite bound
- Bad news: This bound is finite but very large
- If A is $m \times n$ with max coefficient c, solution must satisfy:

$$x_j \leq ((m+n) \cdot n \cdot c)^n$$

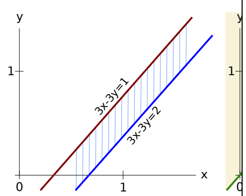
- For 5×5 matrix with max coefficient 5, this bound is $976,562,500,000$
- This only gives a termination guarantee "in theory"

Cuts-from-Proofs: Motivating Example

- Cuts-from-Proofs generalizes B&B
- Instead of branching around coordinate planes, it computes a custom plane with no integer points

$$-3x+3y+3z=-1$$

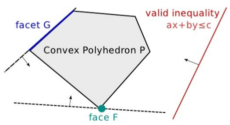
- Then, we create two subproblems by branching around this plane
- Neither subproblem has integer solution, so cuts-from-proof immediately terminates



Some Polyhedral Terminology

Before talking about cuts-from-proofs, some new terminology

- An inequality $\pi\vec{x} \leq c$ called valid inequality if it is satisfied by all points in polytope
- A face F is the set of points in the polytope satisfying $\pi\vec{x} = c$ for some valid inequality $\pi\vec{x} \leq c$



- A vertex of polytope is a face of dimension 0
- A facet of polytope P is a face of dimension $dim(P)-1$

Overview of Cuts-from-Proofs

1. Run Simplex, obtain a point v on polytope
2. Compute defining constraints of this point
   - Intersection I of defining constraints contain face of polytope
   - I does not have to be point; can be line, plane, hyperplane ...
3. Determine if I contains integer solutions
   - can be done in polynomial time using Hermite normal forms
4. If I contains integer points, perform regular B&B
   - there might be an integer that lies on I and is inside polytope

Overview of Cuts-from-Proofs, cont

5. If I has no integers, compute proof of unsatisfiability
   - Proof of unsat is implied by I and does not contain integers
   - Can also be computed in poly-time using Hermite normal forms
6. Compute planes closest to and on either side proof of unsat containing integer points
7. Solve two new subproblems stipulating solution must lie on one side of these planes
8. Problem has integer solution if either of these subproblems has solution

Defining Constraints of Vertex

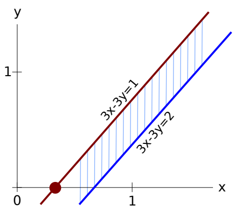
- Suppose we run Simplex on $A\vec{x} \leq \vec{b}$ and obtain point v
- $A_i\vec{x} = b_i$ is defining constraint of v if:
  1. $A_i\vec{x} \leq b_i$ is a row of $A\vec{x} \leq \vec{b}$
  2. v satisfies $A_i\vec{x_i} = b_i$
- Simplex always returns a point, but intersection of defining constraints does not have to be a point
- It can be line, plane, or k-dimensional hyperplane

Example

- Consider again the system:

$$\begin{array}{rcr} -3x + 3y + z & \leq & -1 \\ 3x - 3y + z & \leq & 2 \\ z & = & 0 \end{array}$$

- Suppose Simplex yields $(\frac{1}{3}, 0, 0)$
- What are the defining constraints? $-3x + 3y + z = -1$ and $z = 0$
- Intersection of defining constraints is the face shown in red line



Checking Integer Solutions on Face

Next step in algorithm: Decide if intersection of defining constraints contains integer points

- To do this, we need matrix representation called Hermite Normal Form (HNF)
- An $m \times m$ matrix H is in HNF if:
  1. H is lower-triangular
  2. Diagonal entries are positive ($h_{ii} > 0$)
  3. $h_{ij} \leq 0$ and $|h_{ij}| < h_{ii}$ for $i > j$

$$\begin{bmatrix} 1 & 0 & 0 \\ -3 & 5 & 0 \\ -1 & 0 & 2 \end{bmatrix}$$

Unimodular Matrices

- We can convert every matrix A to its Hermite Normal Form representation
- For this, we need to define unimodular matrix
- A matrix U is unimodular if it has only integer entries and $|\det(U)| = 1$
- Example:

$$\begin{bmatrix} 1 & 3 & -7 \\ 0 & -1 & 2 \\ -1 & 0 & 2 \end{bmatrix}$$

HNF of Matrix

- Theorem: For every $m \times n$ matrix A with $rank(A) = m$, there exists $n \times n$ unimodular matrix U such that:

$$AU = [H \mid 0]$$

- H is called the Hermite Normal Form of A and is unique
- Furthermore, the HNF of any matrix A can be computed in polynomial time!
- Won't talk about HNF conversion algorithm in class

Why is HNF Useful?

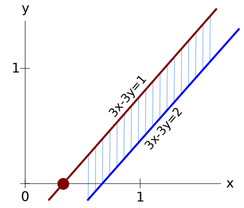
- Theorem: The linear equality system $A\vec{x} = \vec{b}$ has solutions if and only if $H^{-1}\vec{b}$ has integer entries (where H is $HNF(A)$)
- Use theorem to determine if intersection of defining constraints has integer solution
- Let $A'\vec{x} = \vec{b'}$ be defining constraints
- Compute the Hermite Normal Form H of A' and check if $H^{-1}\vec{b'}$ has only integers

Example

- For the point $(\frac{1}{3}, 0, 0)$, defining constraints were $z = 0$ and $-3x + 3y + z = -1$
- Written as a matrix $A'\vec{x} = \vec{b'}$:

$$\begin{bmatrix} 0 & 0 & 1 \\ -3 & 3 & 1 \end{bmatrix} \vec{x} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

- Hermite normal form H of A': $\begin{bmatrix} 1 & 0 \\ -2 & 3 \end{bmatrix}$
- $H^{-1}\vec{b}$ is $\begin{bmatrix} 0 \\ -\frac{1}{3} \end{bmatrix}$
- Does intersection of defining constraints have integer solution? No



When Face Doesn't Contain Integers

- If we determine intersection of defining constraints has no integers, want to exclude this entire face from search space.
- Safe because there are no integer points on this face, we won't be missing any integer points by excluding this face
- To exclude this face from search space, we compute a proof of unsatisfiability for defining constraints

Proof of Unsatisfiability

- A proof of unsatisfiability of $A'\vec{x} = \vec{b'}$ is a single linear equality E such that:
  1. E is implied by $A'\vec{x} = \vec{b'}$
  2. E does not have integer solutions
- In what way does E prove $A'\vec{x} = \vec{b'}$ is unsatisfiable?
- From condition (1), we have $A'\vec{x} = \vec{b'} \Rightarrow E$
- Contrapositive of this is: $\neg E \Rightarrow A'\vec{x} \neq \vec{b'}$
- Since E is unsat, $\neg E$ is true; thus, $A'\vec{x} \neq \vec{b'}$ always true
- Hence, E establishes unsatisfiability of $A'\vec{x} = \vec{b'}$

Computing Proof of Unsatisfiability

- We can compute proofs of unsatisfiability using HNF
- Let $A'\vec{x} = \vec{b'}$ be a system with no integer solutions.
- Compute Hermite Normal Form H of A' and H^{-1}
- Now, consider $H^{-1}A\vec{x} = H^{-1}\vec{b}$:

$$\underbrace{\begin{bmatrix} \text{———} & r_1 & \text{———} \\ & \cdots & \\ a_1 & \cdots & a_n \\ & \cdots & \\ \text{———} & r_m & \text{———} \end{bmatrix}}_{H^{-1}A'} \vec{x} = \underbrace{\begin{bmatrix} c_1 \\ \cdots \\ \frac{n}{d} \\ \cdots \\ c_m \end{bmatrix}}_{H^{-1}\vec{b'}}$$

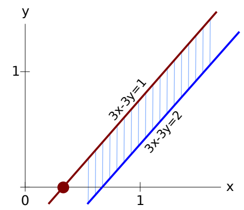
- Proof of unsatisfiability of $A'\vec{x} = \vec{b'}$:

$$a_1 dx_1 + a_2 dx_2 + \ldots a_n dx_n = n_i$$

Example

- Defining constraints for $(\frac{1}{3}, 0, 0)$:

$$\begin{bmatrix} 0 & 0 & 1 \\ -3 & 3 & 1 \end{bmatrix} \vec{x} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$



- Multiply both sides by H^{-1}:

$$\frac{1}{3}\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ -3 & 3 & 1 \end{bmatrix} \vec{x} = \frac{1}{3}\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

Example, cont

- Result of multiplication:

$$\begin{bmatrix} 0 & 0 & 1 \\ -1 & 1 & 1 \end{bmatrix} \vec{x} = \begin{bmatrix} 0 \\ -\frac{1}{3} \end{bmatrix}$$

- What is proof of unsatisfiability?

$$-3x + 3y + 3z = -1$$

- This equality has no integer solutions because the GCD of coefficients does not evenly divide constant on RHS.

- Furthermore, it is implied by the defining constraints $-3x + 3y + z = -1$ and $z = 0$

Branching around the Proof

- Consider proof of unsatisfiability of the system $A'\vec{x} = \vec{b'}$

$$a_1x_1 + a_2x_2 + \ldots a_nx_n = c$$

- To "branch around" this plane, need to compute the closest planes parallel to the proof of unsatisfiability containing integer points

- Let g be the GCD of $a_1, a_2, \ldots, a_n$

- The closest parallel planes containing integers given by:

$$\frac{a_1}{g}x_1 + \frac{a_2}{g}x_2 + \ldots \frac{a_n}{g}x_n = \left\lfloor \frac{c}{g} \right\rfloor$$

$$\frac{a_1}{g}x_1 + \frac{a_2}{g}x_2 + \ldots \frac{a_n}{g}x_n = \left\lceil \frac{c}{g} \right\rceil$$

Branching Around the Proof, cont.

- Thus, if solution is to the "left" of proof plane, must satisfy:

$$\frac{a_1}{g}x_1 + \frac{a_2}{g}x_2 + \ldots \frac{a_n}{g}x_n \leq \left\lfloor \frac{c}{g} \right\rfloor \quad (1)$$

- And if it is to the "right", it must satisfy:

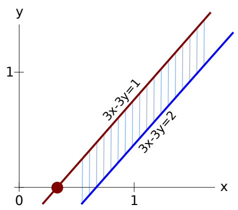
$$\frac{a_1}{g}x_1 + \frac{a_2}{g}x_2 + \ldots \frac{a_n}{g}x_n \geq \left\lceil \frac{c}{g} \right\rceil \quad (2)$$

- Thus, we construct two subproblems one where we conjoin $A\vec{x} \leq \vec{b'}$ with (1), and another where we conjoin it with (2)

Example

- Defining constraints for $(\frac{1}{3}, 0, 0)$:

$$\begin{bmatrix} -3 & 3 & 1 \\ 0 & 0 & 1 \end{bmatrix} \vec{x} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$



- Earlier, we computed proof of unsatisfiability as:

$$-3x + 3y + 3z = -1$$

- What are the closest parallel planes containing integers?

$$-x + y + z = -1$$
$$-x + y + z = 0$$

Example, cont

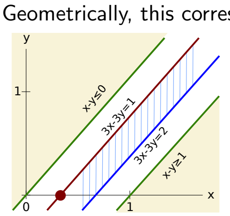
- Thus, we solve two subproblems, one with additional constraint:

$$-x + y + z \leq -1$$

- The other subproblem has additional constraint:

$$-x + y + z \geq 0$$

- Geometrically, this corresponds to solving these two problems:



- Neither subproblem has solution
- Algorithm terminates after one step

What Have We Achieved?

- Guarantee: By branching around proof of unsat, Simplex will never yield point with same defining constraints again!

- Proof: Suppose defining constraints in current iteration were $A'\vec{x} = \vec{b'}$ with proof of unsatisfiability E

- Suppose we get a vertex v with same defining constraints next

- Observation 1: By definition, v must satisfy $A'\vec{x} = \vec{b'}$

- Observation 2: v cannot satisfy E because it would violate the new inequalities we added

- But since $A'\vec{x} = \vec{b'} \Rightarrow E$, this implies v can't satisfy $A'\vec{x} = \vec{b'}$

- Thus, we will never get a solution with same defining constraints again

What About an Unsatisfiable Subset?

- Ok, we excluded intersection of defining constraints from search space
- But what if there is a subset of these defining constraints whose intersection has no integers?
- For instance, suppose we have 3 defining constraints, but the intersection of 2 of them has no integers
- If we are in 3D space, intersection of 3 constraints a point, but intersection of 2 constraints is a whole line
- Thus, we want to exclude the higher-dimensional face with no integers

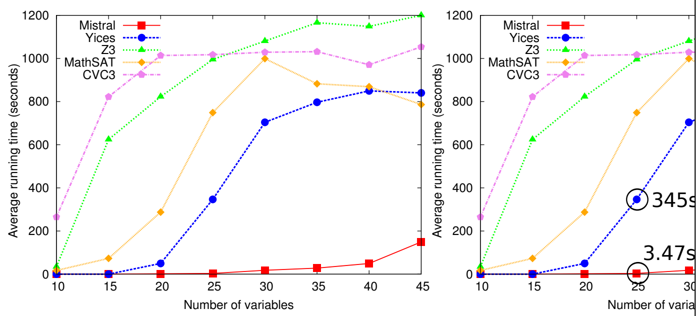
Another Guarantee

- If we add new inequalities in a certain order, we are also guaranteed to eventually find proof of unsatisfiability for this higher dimensional subspace!
- Suppose again we have 3 defining constraints, but the intersection of 2 of them has no integers
- If Simplex keeps yields another point on this line, we will obtain a proof of unsatisfiability of this line in the next step

Experiments

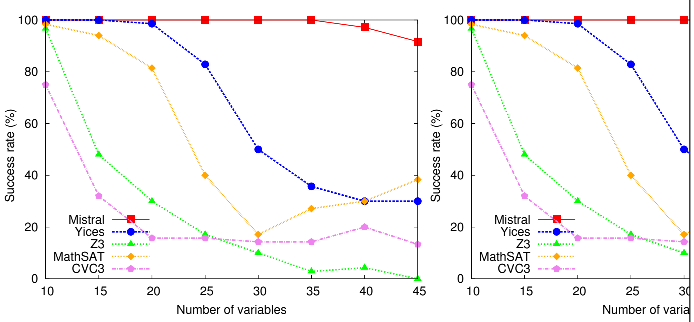
- We compared performance of this new algorithm against four other tools solving qff linear inequalities
    1. CVC3: uses Omega Test
    2. Yices: uses branch-and-cut (B&B + Gomory cuts)
    3. Z3: uses branch-and-cut (B&B + Gomory cuts)
    4. MathSAT: uses Omega Test + branch-and-cut

Experiments



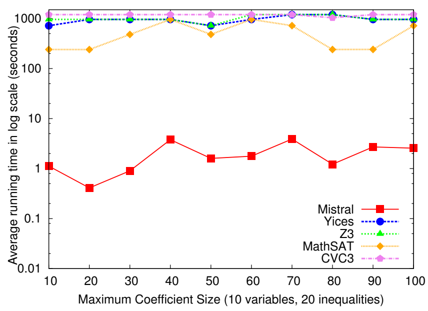
Number of variables vs. average running time.

Experiments



Number of variables vs. percent of successful runs with a 1200 second time-out.

Experiments



Maximum coefficient vs. average running time for 10×20 system

Cuts-from-Proofs Summary

- New relaxation-based technique for solving qff linear integer inequalities
- Computes custom planes to branch around for each problem
- These custom planes are generated by computing proof of unsatisfiability for defining constraints
- At least for some problems, this algorithm does significantly better than existing ones