LP Solution Polishing

Degenerate in MIP Solving

Youxuan Pan

March 13, 2025

Overview

1. Introduction
2. K-sample
3. Strengthen LP Integer
4. Lexicographic Dual Simplex Method
5. Solution Polishing

Dual Degeneracy

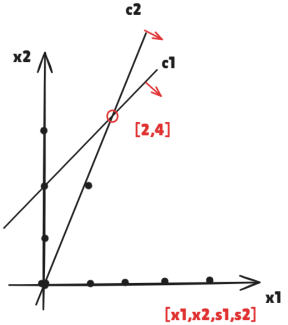
- Two types of degeneracy in LP
  - primal: multiple bases defining one vertex of the polyhedron
  - dual: facet of the polyhedron parallel to the objective function
- Most Problems are primal and dual degenerate
- Degeneracy is the most prominent cause of MIP performance variability

Example

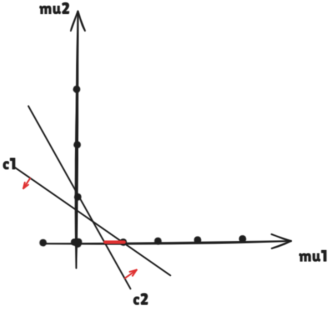
$$\begin{array}{ll}
\text{Max} & -3x_1 + 2x_2 \\
\text{s.t.} & -3x_1 + 3x_2 \leq 6 \\
 & -4x_1 + 2x_2 \leq 0 \\
 & x_1, x_2 \geq 0 \\
 & (x_1, x_2, {\color{red}s_1, s_2}) \rightarrow (x_1, {\color{red}x_2, s_1}, s_2) \\
 & (0, 0, 6, 0) \rightarrow (0, 0, 6, 0)
\end{array}$$

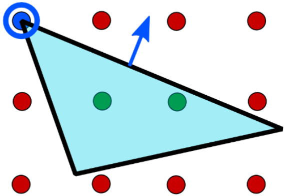
$$\begin{array}{ll}
\text{Min} & 6\mu_1 + 0\mu_2 \\
\text{s.t.} & 3\mu_1 + 4\mu_2 \leq 3 \\
 & 3\mu_1 + 2\mu_2 \geq 2 \\
 & \mu_1, \mu_2 \geq 0 \\
 & \text{Exit a optimal facet}
\end{array}$$

Dual Degeneracy (cont.)



[s1,s2]	[0,0]	[0,0,6,0]
[s1,x2]	[0,0]	[0,0,6,0]
[x1,x2]	[2,4]	[2,4,0,0]





K-sample

Problem:

- Performance Variability in MIP Solving
  - Significant variations in solving time and nodes
  - Unpredictable solver behavior

Motivation:

- Dual Degeneracy Effects
  - Cutting plane generation(Cut P_i)
  - Primal heuristics (Solution S_i)
  - Branching decisions(Solution S_i)

Solution:

- K-sample Algorithm
  - Sample K different optimal bases
  - Collect cutting planes and feasible solutions
  - Aggregate information for solving

K-sample (cont.)

Algorithm 1: ksample

Input: a MIP instance

Output: an optimal solution

preprocess the MIP instance and store it;

for $i = 1$ *to* $K - 1$ **do**

  sample an optimal basis B_i of the (initial) LP relaxation;

  while *executing the default root-node cut loop starting from* B_i **do**

    collect cutting planes in a local cut pool P_i;

    collect feasible solutions in a local solution pool S_i;

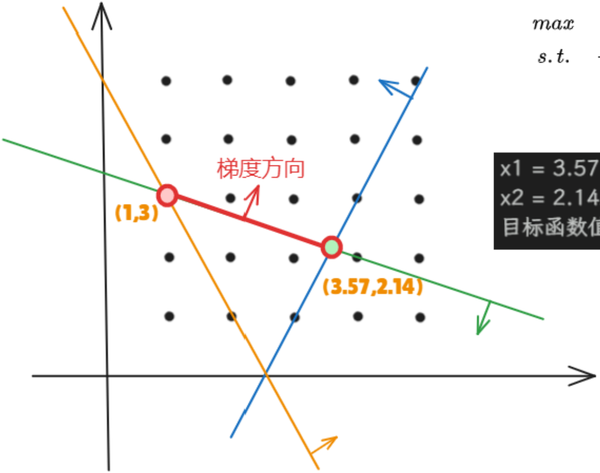
  end

end

solve the stored MIP instance (without any further preprocessing) by using the aggregated pools;

$P := \bigcup_{i=1}^{K-1} P_i$ and $S := \bigcup_{i=1}^{K-1} S_i$;

K-sample: A working example



$$
\begin{aligned}
max \quad & \tfrac{1}{3}x_1 + x_2 \\
s.t. \quad & -2x_1 + x_2 \geq -5 \\
& \tfrac{1}{3}x_1 + x_2 \leq \tfrac{10}{3} \\
& 2x_1 + x_2 \geq 5
\end{aligned}
$$

```
x1 = 3.5714285714285716
x2 = 2.1428571428571432
目标函数值 = 3.3333333333333334
```

Figure: K-sample Example

Strengthen LP Integer

Algorithm 2: Strengthen LP optimal

Input: Optimal LP solution x^*
Output: New optimal solution

```
while true do
    for each variable x_j do
        f_j ← x_j^* − ⌊x_j^*⌋;
        if 0 < f_j < 0.5 then
            c_j ← 1;
        end
        else if 0.5 ≤ f_j < 1 then
            c_j ← −1;
        end
    end
    Re-solve LP (fixed to optimal face) with new objective function and update;
    if no new integral interval bound found then
        return current solution;
        break;
    end
```

Strengthen LP optimal: A working example

Consider the following LP model:

$$\begin{aligned}
\max \quad & \frac{1}{3}x_1 + x_2 \\
\text{s.t.} \quad & -2x_1 + x_2 \geq -5 \\
& \frac{1}{3}x_1 + x_2 \leq \frac{10}{3} \\
& 2x_1 + x_2 \geq 5
\end{aligned}$$

The optimal solution is:

- $x_1 = 3.571428$
- $f_1 = 0.57 > 0.5 \Rightarrow c_1 = -1$
- $x_2 = 2.14285$
- $f_2 = 0.14 < 0.5 \Rightarrow c_2 = 1$

This leads to the new model:

$$\begin{aligned}
\min \quad & -x_1 + x_2 \\
\text{s.t.} \quad & -2x_1 + x_2 \geq -5 \\
& \frac{1}{3}x_1 + x_2 \leq \frac{10}{3} \\
& 2x_1 + x_2 \geq 5 \\
& \frac{1}{3}x_1 + x_2 = \frac{10}{3}
\end{aligned}$$

The optimal solution is:

- $x_1 = 1$
- $x_2 = 3$

Strengthen LP optimal: A working example (cont.)

$$
\begin{aligned}
max \quad & \tfrac{1}{3}x_1 + x_2 \\
s.t. \quad & -2x_1 + x_2 \geq -5 \\
& \tfrac{1}{3}x_1 + x_2 \leq \tfrac{10}{3} \\
& 2x_1 + x_2 \geq 5
\end{aligned}
$$

$$
\begin{aligned}
max \quad & -x_1 + x_2 \\
s.t. \quad & -2x_1 + x_2 \geq -5 \\
& \tfrac{1}{3}x_1 + x_2 \leq \tfrac{10}{3} \\
& 2x_1 + x_2 \geq 5 \\
& \tfrac{1}{3}x_1 + x_2 = \tfrac{10}{3}
\end{aligned}
$$

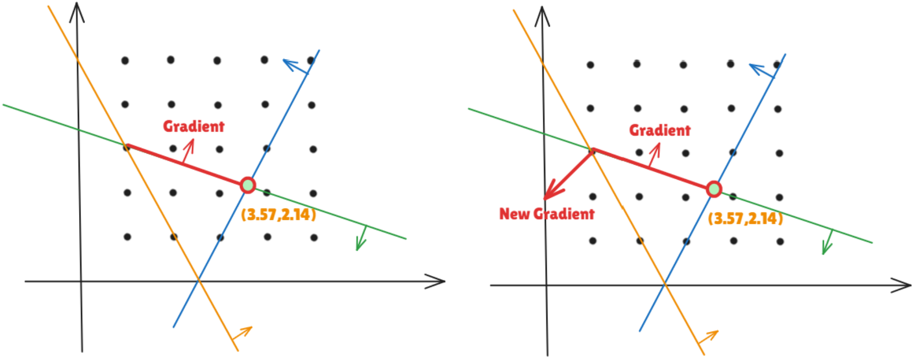


Figure: Strengthen LP optimal example

Lexicographic Dual Simplex Method

Problem:

- LP relaxation optimal solution has multiple equivalent optimal bases
- Multiple non-basic variables have zero reduced costs (dual degeneracy)
- Different bases may lead to very different numerical behaviors

Motivation:

- Need to systematically select a "good" basis from equivalent optimal ones
- Better basis could lead to:
  - More stable numerical computations
  - Better cut generation
  - Controlled growth of basis determinant

Solution:

- Lexicographic Dual Simplex Method

Lexicographic Dual Simplex Method

Algorithm 3: Lexicographic Dual Simplex Method

Input: Optimal LP tableau with possible dual degeneracy
Output: Lexicographically optimal basis

```
Identify nonbasic variables with nonzero reduced costs;
Fix these variables at their bounds;
Fix objective value at its optimal value;
for i = 1 to n do
    Set x_i as new objective ;
    Optimize LP with current fixed variables;
    if new nonbasic variables have nonzero reduced costs then
        Fix them at their bounds;
    end
end
Unfix all variables while maintaining final basis;
```

Lexicographic Dual Simplex Method: A working example

Initial LP Optimal Solution:

$$
\begin{aligned}
\max \quad & \frac{1}{3}x_1 + x_2 \\
\text{s.t.} \quad & -2x_1 + x_2 - s_1 = -5 \\
& \frac{1}{3}x_1 + x_2 + s_2 = \frac{10}{3} \\
& 2x_1 + x_2 - s_3 = 5
\end{aligned}
$$

- Basis: $x_1 = 3.571$, $x_2 = 2.143$, $s_3 = 4.286$ (reduced cost: 0.0)
- $s_1 = 0.0$(reduced cost: 0.0), $s_2 = 0.0$ (reduced cost: -1.0)

Lexicographic Dual Simplex Method: A working example (cont.)

Step 1: Fix s_2, optimize s_1

$$\begin{aligned} \max \quad & s_1 \\ \text{s.t.} \quad & -2x_1 + x_2 - s_1 = -5 \\ & \frac{1}{3}x_1 + x_2 + s_2 = \frac{10}{3} \\ & 2x_1 + x_2 - s_3 = 5 \\ & \frac{1}{3}x_1 + x_2 = \frac{10}{3} \\ & s_2 = 0 \end{aligned}$$

- $x_1 = 1.000$, $x_2 = 3.000$,$s_1 = 6.000$ (reduced cost: 0.0)
- $s_2 = 0.000$ (reduced cost: -2.400), $s_3 = 0.000$ (reduced cost: -1.400)

Solution Polishing Algorithm

Algorithm 5: LP solution polishing of SoPlex

Input: Optimal (dual) solution x (y) with basis $\mathcal{B}$ of LP (25)
Output: Optimal solution of LP (25) with less or equal number of basic problem variables

set of problem variable indices $\mathcal{C} = \{1, \ldots, n\}$
set of slack variable indices $\mathcal{R} = \{1, \ldots, m\}$
set of non-basic indices $\mathcal{N} = (\mathcal{R} \cup \mathcal{C}) \setminus \mathcal{B}$
set of integer variable indices $\mathcal{I} \subseteq \mathcal{C}$
foreach $i \in \mathcal{N}$ **do**
    \\\\ find entering candidate among non-basic indices
    if $i \in \mathcal{C} \wedge i \in \mathcal{I}$ **then**
        \\\\ integer problem variable x_i is non-basic, hence on its bound
        continue
    else
        if $(c - A^\top y)_i = 0$ **then**
            \\\\ x_i has zero reduced cost (pivoting preserves optimal solution value)
            $j \leftarrow$ non-basic index in $\mathcal{B}$ chosen by primal ratio test
            if $j \in \mathcal{C} \wedge j \in \mathcal{I}$ **then**
                \\\\ found an integer problem variable x_j to leave the basis
                $\mathcal{B} \leftarrow \mathcal{B} \setminus \{j\} \cup \{i\}$ \\\\ perform basis change
                update x, y and $\mathcal{N}$
            else
                \\\\ no suitable index found to leave the basis, reject candidate i
                continue
return solution x, y and basis $\mathcal{B}$

Solution Polishing Algorithm:A working example

$$
\begin{aligned}
\min \quad & -\frac{1}{3}x_1 - x_2 \\
\text{s.t.} \quad & 2x_1 - 2x_2 + s_1 = 5 \\
& \frac{1}{3}x_1 + x_2 + x_3 + s_2 = \frac{10}{3} \\
& -2x_1 - x_2 - x_3 + s_3 = -5 \\
& x_3 + s_4 = 1 \\
& x_1, x_2, s_1, s_2, s_3 \geq 0
\end{aligned}
$$

- Problem Varibale $\mathcal{C}$: x_1, x_2, x_3
- Integer Variable $\mathcal{I}$: x_1, x_2, x_3
- Slack Variable $\mathcal{R}$: s_1, s_2, s_3, s_4

LP optimal:

- $x_1 = 3.57$
- $x_2 = 2.14$
- $s_3 = 4.28$
- $s_4 = 1$

Non Basis:

- $x_3 = 0$ (reduced cost: -1)
- $s_1 = 0$ (reduced cost: 0)
- $s_2 = 0$ (reduced cost: -1)

Solution Polishing Algorithm: A working example (cont.)

$$
\begin{aligned}
min \quad & -\tfrac{1}{3}x_1 - x_2 \\
s.t. \quad & 2x_1 - 2x_2 + s_1 = 5 \\
& \tfrac{1}{3}x_1 + x_2 + x3 + s_2 = \tfrac{10}{3} \\
& -2x_1 - x_2 - x_3 + s_3 = -5 \\
& x_3 + s_4 = 1 \\
& x_1, x_2, s_1, s_2, s_3 \geq 0
\end{aligned}
$$

问题变量**C: x1 x2 x3**
整形变量**I: x1 x2 x3**
松弛变量**R: s1 s2 s3 s4**

LP optimal:
x1 = 3.57
x2 = 2.14
s3 = 4.28
s4 = 1

x3 = 0 , rc = -1 跳过，**x3**是问题变量同时也是整型变量，作为**NB**取到了**Bound**
s1 = 0 , rc =0 入基，满足检验数为**0**
s2 = 0 , rc =-1 不满足，检验数非**0**

LP optimal:
s1 = 8.3333
x2 = 3.3333
s3 = -1.6777
s4 = 1

x3 = 0
x1 = 0 取到了**Bound**
s2 = 0

Solution Polishing Algorithm: Numerical Experiments

Instance	Disabled	Only Root	Always	Speedup vs Disabled (Only Root)	Speedup vs Disabled (Always)
30n20b8	200.19	114.03	109.15	1.76	1.83
$CMS750_4$	NaN	18.44	41.55	NaN	NaN
air05	29.74	22.9	26.0	1.30	1.14
app1-1	2.34	3.72	3.47	0.63	0.68
app1-2	NaN	508.39	938.34	NaN	NaN

Table: Performance comparison of Solution Polishing strategies

- **Disabled**: Solution polishing is turned off
- **Only Root**: Solution polishing applied only at the root node
- **Always**: Solution polishing applied at every node
- **Speedup**: Ratio of solving time (higher is better)
- NaN indicates the solver did not find a solution within the time limit