An exponential lower bound for Cunningham’s rule

Oliver Friedmann

Revised March 20, 2015 (DA)
Linear Programming and Simplex
The simplex method, Dantzig (1947)

maximize \( c^T x \)

subject to \( Ax \leq b \)
The first step is to add slack variables making a system of equations called a **dictionary**
The vertices of the polytope are represented by basic feasible solutions formed by solving for 3 basic variables.
Linear Programming and Simplex

Basic feasible solutions and pivoting

\[
\begin{align*}
\text{max} & \quad -1 + 2x_1 - 2x_3 - x_5 \\
\text{s.t.} & \quad x_2 = 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\
& \quad x_4 = 2 - x_3 - x_5 \\
& \quad x_6 = 1 - x_5 \\
& \quad x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}
\]

- The vertices of the polytope are represented by basic feasible solutions formed by solving for 3 basic variables.
- Moving along an edge corresponds to pivoting: exchange a basic variable (LHS) with a non-basic variable (RHS).
The vertices of the polytope are represented by basic feasible solutions formed by solving for 3 basic variables.

Moving along an edge corresponds to pivoting: exchange a basic variable (LHS) with a non-basic variable (RHS).

A non-basic variable with positive coefficient in the objective is chosen to enter the basis.

A pivot selection rule chooses which variable enters.
Basic feasible solutions and pivoting

max $\ 9 - 6x_2 - 2x_4 + x_5$

s.t. $\ x_1 = 7 - 3x_2 - 2x_4$
     $\ x_3 = 2 - x_4 - x_5$
     $\ x_6 = 1 - x_5$

$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$

- The vertices of the polytope are represented by basic feasible solutions formed by solving for 3 basic variables.
- Moving along an edge corresponds to pivoting: exchange a basic variable (LHS) with a non-basic variable (RHS).
- A non-basic variable with positive coefficient in the objective is chosen to enter the basis.
- A pivot selection rule chooses which variable enters.
max $10 - 6x_2 - 2x_4 - x_6$

s.t. $x_1 = 7 - 3x_2 - 2x_4$
$x_3 = 1 - x_4 + x_6$
$x_5 = 1 - x_6$
$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$

- The vertices of the polytope are represented by basic feasible solutions formed by solving for 3 basic variables.
- Moving along an edge corresponds to pivoting: exchange a basic variable (LHS) with a non-basic variable (RHS).
- A non-basic variable with positive coefficient in the objective is chosen to enter the basis.
- A pivot selection rule chooses which variable enters.
Simplex and Pivots
A **pivoting rule** chooses the entering and leaving variable at each iteration

- **Deterministic, oblivious** rules: eg. Largest coefficient (Dantzig), Least subscript (Bland), Greatest improvement are known to be exponential.
Simplex Method and Pivoting Rules

A pivoting rule chooses the entering and leaving variable at each iteration

- **Deterministic, oblivious** rules: eg. Largest coefficient (Dantzig), Least subscript (Bland), Greatest improvement are known to be exponential.

- **Randomized** rules: eg. Random edge and Random facet: Random facet is subexponential (Kalai), (Sharir-Welzl)
A **pivoting rule** chooses the entering and leaving variable at each iteration.

- **Deterministic, oblivious** rules: eg. Largest coefficient (Dantzig), Least subscript (Bland), Greatest improvement are are known to be exponential.

- **Randomized** rules: eg. Random edge and Random facet: Random facet is subexponential (Kalai), (Sharir-Welzl)

- **History-based** rules: eg. Least entered(Zadeh), Round robin(Cunningham) are exponential (Friedmann et al.)
A **pivoting rule** chooses the entering and leaving variable at each iteration

- **Deterministic, oblivious** rules: eg. Largest coefficient (Dantzig), Least subscript (Bland), Greatest improvement are known to be exponential.

- **Randomized** rules: eg. Random edge and Random facet: Random facet is subexponential (Kalai), (Sharir-Welzl)

- **History-based** rules: eg. Least entered (Zadeh), Round robin (Cunningham) are exponential (Friedmann et al.)

- We show an exponential lower bound for Cunningham’s rule
Cunningham’s **ROUND-ROBIN** rule

Order the variables cyclically in any way. Starting from the last entered variable, consider the vertices in circular order choosing the first candidate with positive objective coefficient.
History-based pivoting rules

Cunningham’s **ROUND-ROBIN** rule

Order the variables cyclically in any way. Starting from the last entered variable, consider the vertices in circular order choosing the first candidate with positive objective coefficient.

- No analysis of history based rules for over 39 years
History-based pivoting rules

Cunningham’s ROUND-ROBIN rule
Order the variables cyclically in any way. Starting from the last entered variable, consider the vertices in circular order choosing the first candidate with positive objective coefficient.

- No analysis of history based rules for over 39 years
- They have some features in common with randomized rules (eg. fairness)
History-based pivoting rules

Cunningham’s **ROUND-ROBIN** rule

Order the variables cyclically in any way. Starting from the last entered variable, consider the vertices in circular order choosing the first candidate with positive objective coefficient.

- No analysis of history based rules for over 39 years
- They have some features in common with randomized rules (eg. fairness)
- Have some hope of being at least subexponential (?)
Abstract

Concrete

LP-type problems

Turn-based stochastic games
2 1/2 players

Mean payoff games
2 players

Parity games
2 players

Linear programming

Markov decision problems
1 1/2 players

Deterministic MDPs
1 player
From Policy Iteration to Simplex

Abstract

Concrete

LP-type problems

Turn-based stochastic games

Mean payoff games

Parity games

\[ \in \text{NP} \cap \text{coNP} \]

\[ \in \text{P} \]

Linear programming

Markov decision problems

Deterministic MDPs

\[ 1^{1/2} \text{ players} \]

\[ 2 \text{ players} \]

\[ 2^{1/2} \text{ players} \]

\[ 2 \text{ players} \]

\[ 1 \text{ player} \]
Games and Policy Iteration
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- **Blue edge**: action taken
- **Green circle**: current node
- **Goal**: maximize expected reward before reaching $t$ from current node

Reward: $-1$
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- **Blue edge:** action taken
- **Green circle:** current node
- **Goal:** maximize expected reward before reaching $t$ from current node

**Reward:** $-1$
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1$
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1 - 4$
Large circle: player 0 (us)
Square: player 1 (random)
Small circle: our reward
Edge: possible actions (with probability)
Blue edge: action taken
Green circle: current node
Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1 - 4$
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1 - 4$
Games and Policy Iteration

Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- **Blue edge**: action taken
- **Green circle**: current node
- **Goal**: maximize expected reward before reaching \( t \) from current node

**Reward**: \(-1 - 4\)
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1 - 4$
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1 - 4$
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Shapley (1953), Bellman (1957):
There exists an optimal history-independent choice from each state.

Reward: $-1 - 4$
Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1 - 4$
Games and Policy Iteration

Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

Reward: $-1 - 4 + 6$
Games and Policy Iteration

Markov decision processes (MDPs)

- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching $t$ from current node

\[
\text{Reward: } -1 - 4 + 6 = 1
\]
A policy $\pi$ is a choice of an action each of our nodes. (blue edges)
A policy $\pi$ is a choice of an action each of our nodes. (blue edges)

The value $\text{VAL}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected reward moving from $i$ to $t$ (blue numbers)

---

Diagram:

- States: 2, 0, -4, -1, t
- Edges with blue numbers:
  - From 2 to 0: 2/3
  - From 0 to -4: 1/3
  - From -4 to -1: 1/3
  - From -1 to 0: 1/3
  - From 0 to t: 2/3
  - From t to 0: 1/3
  - From 0 to 2: 2/3
  - From 2 to t: 2/3

---
A policy $\pi$ is a choice of an action each of our nodes. (blue edges)

The value $\text{VAL}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected reward moving from $i$ to $t$ (blue numbers).

A policy change (pivot) is an improving switch w.r.t. $\pi$ if it improves the value.
A policy $\pi$ is a choice of an action each of our nodes. (blue edges)

The value $\text{val}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected reward moving from $i$ to $t$ (blue numbers)

A policy change (pivot) is an improving switch w.r.t. $\pi$ if it improves the value.

Policy improvement: make any improving switch
A policy $\pi$ is a choice of an action each of our nodes. (blue edges)

The value $\text{VAL}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected reward moving from $i$ to $t$ (blue numbers)

A policy change (pivot) is an improving switch w.r.t. $\pi$ if it improves the value.

Policy improvement: make any improving switch
A policy $\pi$ is a choice of an action each of our nodes. (blue edges)

The value $\text{VAL}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected reward moving from $i$ to $t$ (blue numbers)

A policy change (pivot) is an improving switch w.r.t. $\pi$ if it improves the value.

Policy improvement: make any improving switch
A policy $\pi$ is a choice of an action each of our nodes. (blue edges)

The value $\text{VAL}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected reward moving from $i$ to $t$ (blue numbers)

A policy change (pivot) is an improving switch w.r.t. $\pi$ if it improves the value.

Policy improvement: make any improving switch

A policy $\pi^*$ is optimal iff there are no improving switches.
Optimality conditions for each vertex:

\[ v_i = \text{VAL}_\pi(i) \]
Optimality conditions for each vertex:
- \( v_i = \text{VAL}_\pi(i) \)
- \( v_A = \max\{v_D, v_t\} \)
Optimality conditions for each vertex:

- \( v_i = \text{VAL}_\pi(i) \)
- \( v_A = \max\{v_D, v_t\} \)
- \( v_B = \max\{-4 + v_D, v_t\} \)
Formulating an LP - I

- **Optimality conditions** for each vertex:
  - \( v_i = \text{VAL}_\pi(i) \)
  - \( v_A = \max\{v_D, v_t\} \)
  - \( v_B = \max\{-4 + v_D, v_t\} \)
  - \( v_C = \max\{-4 + v_D, -1 + v_B\} \)
■ **Optimality conditions** for each vertex:
  
  \[
  v_i = \text{VAL}_\pi(i)
  \]
  
  \[
  v_A = \max\{v_D, v_t\}
  \]
  
  \[
  v_B = \max\{-4 + v_D, v_t\}
  \]
  
  \[
  v_C = \max\{-4 + v_D, -1 + v_B\}
  \]
  
  \[
  v_D = \frac{1}{3}(6 + v_t) + \frac{2}{3}v_A
  \]
Optimality conditions for each vertex:

- $v_i = \text{VAL}_\pi(i)$
- $v_A = \max\{v_D, v_t\}$
- $v_B = \max\{-4 + v_D, v_t\}$
- $v_C = \max\{-4 + v_D, -1 + v_B\}$
- $v_D = \frac{1}{3}(6 + v_t) + \frac{2}{3}v_A$
- Note: $v_t = 0$
Formulating an LP - II

- Optimality conditions

\[ v_A = \max\{v_D, 0\} \]
\[ v_B = \max\{-4 + v_D, 0\} \]
\[ v_C = \max\{-4 + v_D, -1 + v_B\} \]
\[ v_D = 2 + \frac{2}{3} v_A \]
Formulating an LP - II

- Optimality conditions

\[ v_A = \max\{v_D, 0\} \]
\[ v_B = \max\{-4 + v_D, 0\} \]
\[ v_C = \max\{-4 + v_D, -1 + v_B\} \]
\[ v_D = 2 + \frac{2}{3} v_A \]

- Linear Program

\[
\begin{align*}
\min w &= v_A + v_B + v_C \\
v_A &\geq v_D \\
v_A &\geq 0 \\
v_B &\geq -4 + v_D \\
v_B &\geq 0 \\
v_C &\geq -4 + v_D \\
v_C &\geq -1 + v_B \\
v_D &= 2 + \frac{2}{3} v_A
\end{align*}
\]
Optimality conditions

\[ v_{A} = \max\{v_{D}, 0\} \]
\[ v_{B} = \max\{-4 + v_{D}, 0\} \]
\[ v_{C} = \max\{-4 + v_{D}, -1 + v_{B}\} \]
\[ v_{D} = 2 + \frac{2}{3} v_{A} \]
■ Optimality conditions

\[ v_A = \max\{v_D, 0\} \]
\[ v_B = \max\{-4 + v_D, 0\} \]
\[ v_C = \max\{-4 + v_D, -1 + v_B\} \]
\[ v_D = 2 + \frac{2}{3} v_A \]

■ Linear Program \((v_D\) eliminated\)

\[
\begin{align*}
\min w &= v_A + v_B + v_C \\
v_A &\geq 2 + \frac{2}{3} v_A \\
v_A &\geq 0 \\
v_B &\geq -2 + \frac{2}{3} v_A \\
v_B &\geq 0 \\
v_C &\geq -2 + \frac{2}{3} v_A \\
v_C &\geq -1 + v_B
\end{align*}
\]
Optimality conditions

\[ v_A = \max\{v_D, 0\} \]
\[ v_B = \max\{-4 + v_D, 0\} \]
\[ v_C = \max\{-4 + v_D, -1 + v_B\} \]
\[ v_D = 2 + \frac{2}{3} v_A \]
Formulating an LP - IV

- **Optimality conditions**

  \[
  v_A = \max\{v_D, 0\}
  \]

  \[
  v_B = \max\{-4 + v_D, 0\}
  \]

  \[
  v_C = \max\{-4 + v_D, -1 + v_B\}
  \]

  \[
  v_D = 2 + \frac{2}{3}v_A
  \]

- **Linear Program (Dual standard form)**

  \[
  \begin{align*}
  \min w &= v_A + v_B + v_C \\
  \frac{1}{3}v_A &\geq 2 \\
  v_A &\geq 0 \\
  -\frac{2}{3}v_A + v_B &\geq -2 \\
  v_B &\geq 0 \\
  -\frac{2}{3}v_A + v_C &\geq -2 \\
  -v_B + v_C &\geq -1
  \end{align*}
  \]
From dual to primal

\[
\begin{align*}
\text{min } w &= v_A + v_B + v_C \\ 
\frac{1}{3}v_A &\geq 2 (x_1) \\
v_A &\geq 0 (x_2) \\
-\frac{2}{3}v_A + v_B &\geq -2 (x_3) \\
v_B &\geq 0 (x_4) \\
-\frac{2}{3}v_A + v_C &\geq -2 (x_5) \\
-v_B + v_C &\geq -1 (x_6)
\end{align*}
\]
Games and Policy Iteration

From dual to primal

\[
\begin{align*}
\min w &= v_A + v_B + v_C \quad \text{Dual} \\
\frac{1}{3}v_A &\geq 2 \quad (x_1) \\
v_A &\geq 0 \quad (x_2) \\
-\frac{2}{3}v_A + v_B &\geq -2 \quad (x_3) \\
v_B &\geq 0 \quad (x_4) \\
-\frac{2}{3}v_A + v_C &\geq -2 \quad (x_5) \\
-v_B + v_C &\geq -1 \quad (x_6)
\end{align*}
\]

\[
\begin{align*}
\max z &= 2x_1 - 2x_3 - 2x_5 \quad \text{Primal} \\
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 - \frac{2}{3}x_5 &= 1 \\
x_3 + x_4 &= 1 \\
x_5 + x_6 &= 1 \\
x_i &\geq 0
\end{align*}
\]
Games and Policy Iteration

From dual to primal

\[
\begin{align*}
\min w &= v_A + v_B + v_C & \text{Dual} \\
\frac{1}{3} v_A &\geq 2 \quad (x_1) \\
v_A &\geq 0 \quad (x_2) \\
-\frac{2}{3} v_A + v_B &\geq -2 \quad (x_3) \\
v_B &\geq 0 \quad (x_4) \\
-\frac{2}{3} v_A + v_C &\geq -2 \quad (x_5) \\
-v_B + v_C &\geq -1 \quad (x_6)
\end{align*}
\]

\[
\begin{align*}
\max z &= 2x_1 - 2x_3 - 2x_5 & \text{Primal} \\
\frac{1}{3} x_1 + x_2 - \frac{2}{3} x_3 - \frac{2}{3} x_5 &= 1 \\
x_3 + x_4 &= 1 \\
x_5 + x_6 &= 1 \\
x_i &\geq 0
\end{align*}
\]

- What are the variables \(x_i\)?
From dual to primal

\[ \min w = v_A + v_B + v_C \]

\[ \begin{align*}
\frac{1}{3}v_A & \geq 2 \quad (x_1) \\
v_A & \geq 0 \quad (x_2) \\
-\frac{2}{3}v_A + v_B & \geq -2 \quad (x_3) \\
v_B & \geq 0 \quad (x_4) \\
-\frac{2}{3}v_A + v_C & \geq -2 \quad (x_5) \\
-v_B + v_C & \geq -1 \quad (x_6)
\end{align*} \]

\[ \max z = 2x_1 - 2x_3 - 2x_5 \]

\[ \begin{align*}
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 - \frac{2}{3}x_5 & = 1 \\
x_3 + x_4 & = 1 \\
x_5 + x_6 & = 1 \\
x_i & \geq 0
\end{align*} \]

- The \( x_i \) are action variables!

Oliver Friedmann
LRC Lower Bound
Revised March 20, 2015 (DA)
**Games and Policy Iteration**

### From dual to primal

\[ \min w = v_A + v_B + v_C \]

**Dual**

\[ \begin{align*}
\frac{1}{3} v_A & \geq 2 \ (x_1) \\
v_A & \geq 0 \ (x_2) \\
-\frac{2}{3} v_A + v_B & \geq -2 \ (x_3) \\
v_B & \geq 0 \ (x_4) \\
-\frac{2}{3} v_A + v_C & \geq -2 \ (x_5) \\
-v_B + v_C & \geq -1 \ (x_6)
\end{align*} \]

\[ \max z = 2x_1 - 2x_3 - 2x_5 \]

**Primal**

\[ \begin{align*}
\frac{1}{3} x_1 + x_2 - \frac{2}{3} x_3 & - \frac{2}{3} x_5 = 1 \\
x_3 + x_4 & = 1 \\
x_5 + x_6 & = 1 \\
x_i & \geq 0
\end{align*} \]

- The \( x_i \) are action variables!
From dual to primal

\[ \min w = v_A + v_B + v_C \]

\[ \begin{align*}
\frac{1}{3}v_A & \geq 2 \quad (x_1) \\
v_A & \geq 0 \quad (x_2) \\
\frac{2}{3}v_A + v_B & \geq -2 \quad (x_3) \\
v_B & \geq 0 \quad (x_4) \\
\frac{2}{3}v_A + v_C & \geq -2 \quad (x_5) \\
-v_B + v_C & \geq -1 \quad (x_6)
\end{align*} \]

\[ \max z = 2x_1 - 2x_3 - 2x_5 \]

\[ \begin{align*}
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 - \frac{2}{3}x_5 &= 1 \\
x_3 + x_4 &= 1 \\
x_5 + x_6 &= 1 \\
x_i &\geq 0
\end{align*} \]

- The \( x_i \) are action variables!
Variables of the primal LP

- $x_i$ is the expected number of times action $i$ is used, summed over all starting states.

\[
\begin{align*}
\text{max} & \quad -1 + 2x_1 - 2x_3 - x_5 \\
\text{s.t.} & \quad x_2 = 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\
& \quad x_4 = 2 - x_3 - x_5 \\
& \quad x_6 = 1 \geq 0 \\
x_1 = 0 & \\
x_2 = 1 & \\
x_3 = 0 & \\
x_5 = 0 & \\
x_6 = 1 &
\end{align*}
\]
Variables of the primal LP

- $x_i$ is the expected number of times action $i$ is used, summed over all starting states.
- The actions taken form the basic variables of the LP.

\[
\begin{align*}
\max & -1 + 2x_1 - 2x_3 - x_5 \\
\text{s.t.} & \\
& x_2 = 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\
& x_4 = 2 - x_3 - x_5 \\
& x_6 = 1 - x_5 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}
\]
- $x_i$ is the expected number of times action $i$ is used, summed over all starting states.
- The actions taken form the basic variables of the LP
- Solving for $\{x_2, x_4, x_6\}$ we get:

$$\begin{align*}
\text{max } & \quad -1 + 2x_1 - 2x_3 - x_5 \\
\text{s.t. } & \quad x_2 = 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\
& \quad x_4 = 2 - x_3 - x_5 \\
& \quad x_6 = 1 - x_5 \\
& \quad x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}$$
Policy improvements are LP pivots

\[
\begin{align*}
\text{max} & \quad -1 + 2x_1 - 2x_3 - x_5 \\
\text{s.t.} & \quad x_2 = 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\
& \quad x_4 = 2 - x_3 - x_5 \\
& \quad x_6 = 1 - x_5 \\
& \quad x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}
\]
Policy improvements are LP pivots

\[
\begin{align*}
\text{max} & \quad 5 - 6x_2 + 2x_3 + 3x_5 \\
\text{s.t.} & \quad x_1 = 3 - 3x_2 + 2x_3 + 2x_5 \\
& \quad x_4 = 2 - x_3 - x_5 \\
& \quad x_6 = 1 - x_5 \\
& \quad x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}
\]
Policy improvements are LP pivots

\[
\begin{align*}
\text{max} & \quad 9 - 6x_2 - 2x_4 + x_5 \\
\text{s.t.} & \quad x_1 = 7 - 3x_2 - 2x_4 \\
& \quad x_3 = 2 - x_4 - x_5 \\
& \quad x_6 = 1 - x_5 \\
& \quad x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}
\]
Policy improvements are LP pivots

\[
\begin{align*}
\text{max} & \quad 10 - 6x_2 - 2x_4 - x_6 \\
\text{s.t.} & \quad x_1 = 7 - 3x_2 - 2x_4 \\
& \quad x_3 = 1 - x_4 + x_6 \\
& \quad x_5 = 1 - x_6 \\
& \quad x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{align*}
\]
Policy improvements are LP pivots

- Out degree of player 0 nodes = 2
- Polyhedron is a deformed cube
- Oriented by objective function
- Polyhedron gives an acyclic USO
- Degree 2 MDPs give lower bounds for LPs and USOs!
Optimality Theorem for MDP and LP

- Optimal policy $\pi^*$:

  $$\forall i \in S : \text{VAL}_{\pi^*}(i) = \max_{a \in A_i} r_a + \sum_{j \in S} p_{a,j} \text{VAL}_{\pi^*}(j)$$

  $A_i$ is the set of actions from $i$
  $r_a$ is the expected reward of using action $a$
  $p_{a,j}$ is the probability of moving to $j$ when using action $a$. 

Optimality Theorem for MDP and LP

- Optimal policy $\pi^*$:

$$\forall i \in S : \text{VAL}_{\pi^*}(i) = \max_{a \in A_i} r_a + \sum_{j \in S} p_{a,j} \text{VAL}_{\pi^*}(j)$$

$A_i$ is the set of actions from $i$
$r_a$ is the expected reward of using action $a$
$p_{a,j}$ is the probability of moving to $j$ when using action $a$.

- LP formulation:

$$\text{minimize} \sum_{i \in S} v_i$$

$$s.t. \ \forall i \in S \ \forall a \in A_i : v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j$$
Primal and dual LPs for MDPs

minimize \( \sum_{i \in S} v_i \)

s.t. \( \forall i \in S \ \forall a \in A_i : v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j \)

maximize \( \sum_{i \in S} \sum_{a \in A_i} r_a x_a \)

s.t. \( \forall i \in S : \sum_{a \in A_i} x_a = 1 + \sum_{j \in S} \sum_{a \in A_j} p_{a,i} x_a \)

\( x_a \geq 0, \ \forall a \)
Primal and dual LPs for MDPs

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in S} v_i \\
\text{s.t.} & \quad \forall i \in S \forall a \in A_i : v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j
\end{align*}
\]

\[
\begin{align*}
\text{maximize} & \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a \\
\text{s.t.} & \quad \forall i \in S : \sum_{a \in A_i} x_a = 1 + \sum_{j \in S} \sum_{a \in A_j} p_{a,i} x_a \\
x_a & \geq 0, \quad \forall a
\end{align*}
\]

- policy improvement algorithm \(\equiv\) simplex method on primal

Finiteness and correctness of policy improvement follow from finiteness and correctness of simplex method! Lower bounds for policy improvement give lower bounds for simplex method!
Primal and dual LPs for MDPs

minimize \[ \sum_{i \in S} v_i \]

s.t. \( \forall i \in S \forall a \in A_i : v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j \)

maximize \[ \sum_{i \in S} \sum_{a \in A_i} r_a x_a \]

s.t. \( \forall i \in S : \sum_{a \in A_i} x_a = 1 + \sum_{j \in S} \sum_{a \in A_j} p_{a,i} x_a \)

\( x_a \geq 0, \ \forall a \)

- policy improvement algorithm \( \equiv \) simplex method on primal
- Finiteness and correctness of policy improvement follow from finiteness and correctness of simplex method
Primal and dual LPs for MDPs

minimize \( \sum_{i \in S} v_i \)

s.t. \( \forall i \in S \forall a \in A_i : v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j \)

maximize \( \sum_{i \in S} \sum_{a \in A_i} r_a x_a \)

s.t. \( \forall i \in S : \sum_{a \in A_i} x_a = 1 + \sum_{j \in S} \sum_{a \in A_j} p_{a,i} x_a \)

\( x_a \geq 0, \ \forall a \)

- policy improvement algorithm \( \equiv \) simplex method on primal
- Finiteness and correctness of policy improvement follow from finiteness and correctness of simplex method
- Lower bounds for policy improvement give lower bounds for simplex method!
Lower Bound for Least Recently Considered Rule
We define a family of lower bound MDPs $G_n$ such that the ROUND-ROBIN pivoting rule will simulate an $n$-bit binary counter.
Lower bound construction

- We define a family of lower bound MDPs $G_n$ such that the ROUND-ROBIN pivoting rule will simulate an $n$-bit binary counter.
- We make use of exponentially growing rewards (and penalties): To get a higher reward the MDP is willing to sacrifice everything that has been built up so far.
We define a family of lower bound MDPs $G_n$ such that the ROUND-ROBIN pivoting rule will simulate an $n$-bit binary counter.

We make use of exponentially growing rewards (and penalties): To get a higher reward the MDP is willing to sacrifice everything that has been built up so far.

Notation: Integer priority $p$ corresponds to reward $(-N)^p$, where $N = 7n + 1$.

\[\ldots < 5 < 3 < 1 < 2 < 4 < 6 < \ldots\]

The use of priorities is inspired by parity games.
Selection Ordering

Cunningham’s **ROUND-ROBIN** rule

Perform improving switches in a round-robin fashion.

**Selection Ordering** = linear ordering on the edges

Proof of **Small Diameter Theorem** implies:

**Corollary**

There is a selection ordering s.t. Cunningham’s rule requires linearly many iterations in the worst-case.

**Consequence:** lower bound construction is equipped with particular selection ordering
Lower Bound Design Principles

- Simulate binary counter
Simulate binary counter
End in a single sink loop with no cost
Lower Bound for Least Recently Considered Rule

Lower Bound Design Principles

- Simulate binary counter
- End in a single sink loop with no cost
- Incrementation of binary counter consists of intermediate phases
Simulate binary counter

End in a single sink loop with no cost

Incrementation of binary counter consists of intermediate phases

Structure relating to single bit as profitable pass-through structure if bit is set
Lower Bound Design Principles

- Simulate binary counter
- End in a single sink loop with no cost
- Incrementation of binary counter consists of intermediate phases
- Structure relating to single bit as profitable pass-through structure if bit is set
- Implementation of bit structure by zero-cost cycles with exponentially small exit-probability
Setting a bit $b$

- Choose starting policy

\[ \begin{align*}
\text{Choose starting policy} \\
\end{align*} \]
Setting a bit $b$

- Choose starting policy
- $v_w = 0, v_b = 0$  \quad \text{$b$ unset}
Setting a bit $b$

- Choose starting policy
- $v_w = 0, v_b = 0 \quad b \text{ unset}$
- Improving switch found
Setting a bit $b$

- Choose starting policy
- $v_w = 0, v_b = 0$ \(b\) unset
- Improving switch found
- $v_w = 0, v_b = r_{10} > 0$
Setting a bit $b$

- Choose starting policy
- $v_w = 0, v_b = 0$  \( b \) unset
- Improving switch found
- $v_w = 0, v_b = r_{10} > 0$
- Improving switch found
Setting a bit $b$

- Choose starting policy
- $v_w = 0, v_b = 0 \quad b$ unset
- Improving switch found
- $v_w = 0, v_b = r_{10} > 0$
- Improving switch found
- $v_w = r_9 + r_{10} > 0, v_b = r_{10} > 0 \quad b$ set
Setting a bit $b$

- Choose starting policy
  - $v_w = 0, v_b = 0$  \( b \) unset

- Improving switch found
  - $v_w = 0, v_b = r_{10} > 0$

- Improving switch found
  - $v_w = r_9 + r_{10} > 0, v_b = r_{10} > 0$  \( b \) set

- No improving switches for any valid choice of $r_9, r_{10}$
Basic Construction of Counter

Start with a single sink node.
Add three bits.

Design principles:

- Every bit is represented by a simple cycle.
- Going through a set bit is profitable.
- Going through an unset bit is unprofitable.
- Going through higher set bits is more profitable than going through lower set bits.
Connect all bits with the sink.
Start with unset bits.
Add uplink structure that allows to go through set bits or bypass them.

\( v_i = 0 \) for all \( i \)
Small exit-probability almost hides the value of the exit nodes completely.
It is improving to set any of the bits. This will be a general principle: All unset bits can be set at the same time.
First bit has been set.

\[ z = 4 \]

\[ v_{b1} = 4 \]

- The exit node will be eventually taken with probability 1.
- It is now profitable to go through the set bit.
Update uplink of first bit: go through.

\[ z = 6 \]

\[ v_{w_1} = 2, v_{b_1} = 4 \]
Second bit has been set.

\[ z = 22 \]

\[ v_{w_1} = 2, v_{b_1} = 4, v_{b_2} = 16 \]
Update uplink of second bit:
go through.
\[ z = 46 \]
\[ v_{w_1} = 10, v_{b_1} = 12 \]
\[ v_{w_2} = 8, v_{b_2} = 16 \]
Problem: we cannot reuse the first bit again.
Start over again.

- Setting a higher bit has to lead to resetting the lower bits.
- There have to be outgoing edges of the cycles that have immediate access to the next set bit.
Basic Construction of Counter

Remove direct link to the sink.
Add second uplink structure, called selector structure.
Give all cycles immediate access to the selector structure.
Small exit-probability still hides the value of the exit nodes completely.
It is still improving to set any of the bits.
Step 1: First bit has been set.

\[ z = 4 \]
\[ v_{b_1} = 4 \]
Step 2: Update selector of first bit.

\[ z = 10 \]

\[ v_{b_2} = v_{b_3} = v_{u_1} = 2, \quad v_{b_1} = 4 \]
Step 4: Update uplink of first bit. (Step 3 missing)

\[ z = 12 \]

\[ v_{w_1} = v_{b_2} = v_{b_3} = v_{u_1} = 2 \]

\[ v_{b_1} = 4 \]
Step 1: Second bit has been set.

\[ z = 26 \]
\[ \nu_{w_1} = \nu_{b_3} = \nu_{u_1} = 2 \]
\[ \nu_{b_1} = 4, \nu_{b_2} = 16 \]
Step 2: Update selector of second bit.

\[ z = 34 \]

\[ v_{w_1} = v_{b_3} = v_{u_1} = 2 \]
\[ v_{b_1} = 4, v_{b_2} = 16, v_{u_2} = 8 \]
Step 2: Update selector of first bit.

\[ z = 46 \]

\[ v_{b3} = v_{u1} = v_{u2} = 8 \]

\[ v_{b1} = 4, v_{b2} = 16, v_{w1} = 2 \]
Step 3: Reset first bit.

\[ z = 54 - 4\epsilon \]

\[ v_{b_1} = v_{b_3} = v_{u_1} = v_{u_2} = 8 \]

\[ v_{b_2} = 16 \]

\[ v_{w_1} = -2 + 4\epsilon + 8(1 - \epsilon) = 6 - 4\epsilon \]
Step 4: Update uplink of second bit.

\[ z = 62 + 4\epsilon \]

\[ v_{b_1} = v_{b_3} = v_{u_1} = v_{u_2} = 8 \]

\[ v_{b_2} = 16, v_{w_2} = 8 \]

\[ v_{w_1} = -2 + 12\epsilon + 8(1 - \epsilon) = 6 + 4\epsilon \]
Step 4: Update uplink of first bit.

\[ z = 64 \]

(Setting \( \epsilon = 1/4 \), \( 62 + 4\epsilon < 64 \))

\[ v_{b_1} = v_{b_3} = v_{u_1} = v_{u_2} = 8 \]

\[ v_{b_2} = 16, \; v_{w_1} = v_{w_2} = 8 \]
Step 1: First bit has been set.

\[ z = 68 \]

\[ v_{b_1} = 12, \ v_{b_3} = v_{u_1} = v_{u_2} = 8 \]

\[ v_{b_2} = 16, \ v_{w_1} = v_{w_2} = 8 \]
Step 2: Update selector of first bit.

\[ z = 72 \]

\[ v_{b_1} = 12, v_{b_3} = v_{u_1} = 10 \]

\[ v_{b_2} = 16, v_{u_2} = v_{w_1} = v_{w_2} = 8 \]
Step 4: Update uplink of first bit. (Step 3 missing)

\[ z = 74 \]

\[ v_{b_1} = 12, \quad v_{w_1} = v_{b_3} = v_{u_1} = 10 \]

\[ v_{b_2} = 16, \quad v_{u_2} = v_{w_2} = 8 \]
Intermediate Steps:

1. Set least unset bit by moving into cycle.
2. Update selector lane.
3. Reset lower set bits by using selector lane.
4. Update uplink lane.
Summary:

1. Generalizes to give $n$-bit binary counter
2. Exponential number of policy improvements
3. Corresponding LP has exponential number of pivots
4. Tune for particular pivot selection rule
We now give a lower bound on:

Least Recently Considered

Select improving edge in a round robin fashion.

An ordering on the edges is fixed.

**Theorem**: there is an ordering on the edges s.t. the Round Robin Rule solves the game in linearly many iterations.

**Hence**: we, as designers, specify the ordering
We now give a lower bound on:

**Least Recently Considered**

Select improving edge in a round robin fashion.

**Ordering**: order edges s.t. steps 1–4 are performed one after the other

**Problem**: all unset bits can be set at the same time.
We now give a lower bound on:

**Least Recently Considered**

Select improving edge in a round robin fashion.

**Ordering**: order edges s.t. steps 1–4 are performed one after the other

**Problem**: all unset bits can be set at the same time.

**Solution**: replace simple cycles of higher bits by longer cycles
We now give a lower bound on:

Least Recently Considered

Select improving edge in a round robin fashion.

Start with unset bit again.
We now give a lower bound on:

Least Recently Considered

Select improving edge in a round robin fashion.

Start with unset bit again.
We now give a lower bound on:

Least Recently Considered

Select improving edge in a round robin fashion.

Closing the cycle happens one at edge at a time.
We now give a lower bound on:

**Least Recently Considered**

Select improving edge in a round robin fashion.

Closing the cycle happens one at a time.
We now give a lower bound on:

Least Recently Considered

Select improving edge in a round robin fashion.

Closing the cycle happens one at edge at a time.
We now give a lower bound on:

Least Recently Considered
Select improving edge in a round robin fashion.

Closing the cycle happens one at edge at a time.
We now give a lower bound on:

**Least Recently Considered**

Select improving edge in a round robin fashion.

Closing the cycle happens one at edge at a time.
Round Robin Lower Bound Construction

Set first bit.
Round Robin Lower Bound Construction

Set single edge of second bit.
Round Robin Lower Bound Construction

Other edge of second bit now improving, but smaller. Set single edge of third bit.
Round Robin Lower Bound Construction

Second edge of third bit now improving, but smaller. Update selector of first bit.
Reset single edge of third bit.
Round Robin Lower Bound Construction

Reset second edge of third bit.
Reset last edge of third bit.
Round Robin Lower Bound Construction

Reset single edge of second bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

Reset last edge of second bit.
Update uplink of first bit.
Set single edge of second bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

Other edge of second bit now improving, but smaller. Set single edge of third bit.
Round Robin Lower Bound Construction

Set other edge of second bit, i.e. set second bit.
Round Robin Lower Bound Construction

Set second edge of third bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

Last edge of third bit now improving, but smaller.
Update selector of second bit.
Round Robin Lower Bound Construction

Update selector of first bit.
Reset single edge of third bit.
Round Robin Lower Bound Construction

Reset second edge of third bit.
Round Robin Lower Bound Construction

Reset last edge of third bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

0

1

R

Reset first bit.
Update uplink of second bit.
Update uplink of first bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

Set first bit.

Oliver Friedmann

LRC Lower Bound

Revised March 20, 2015 (DA)
Set single edge of third bit.
Round Robin Lower Bound Construction

Update selector of first bit.
Round Robin Lower Bound Construction

Reset single edge of third bit.
Round Robin Lower Bound Construction

Reset second edge of third bit.
Round Robin Lower Bound Construction

Reset last edge of third bit.
Update uplink of first bit.
Round Robin Lower Bound Construction

Set single edge of third bit.
Set second edge of third bit.
Set last edge of third bit, i.e. set third bit.
Update selector of third bit.
Round Robin Lower Bound Construction

Update selector of second bit.
Round Robin Lower Bound Construction

Update selector of first bit.
Reset first edge of second bit, i.e. reset second bit.
Round Robin Lower Bound Construction

Reset other edge of second bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

\[ \text{Reset first bit.} \]
Update uplink of third bit.
Round Robin Lower Bound Construction

Update uplink of second bit.
Round Robin Lower Bound Construction

Update uplink of first bit.
Round Robin Lower Bound Construction

Set first bit.
Set single edge of second bit.
Round Robin Lower Bound Construction

Other edge of second bit now improving, but smaller. Update selector of first bit.
Reset single edge of second bit.
Reset last edge of second bit.
Round Robin Lower Bound Construction

Update uplink of first bit.
Set single edge of second bit.
Round Robin Lower Bound Construction

Set other edge of second bit, i.e. set second bit.
Update selector of second bit.
Update selector of first bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

Reset first bit.
Round Robin Lower Bound Construction

Update uplink of second bit.
Update uplink of first bit.
Lower Bound for Least Recently Considered Rule

Round Robin Lower Bound Construction

Set first bit.
Round Robin Lower Bound Construction

Update selector of first bit.
Round Robin Lower Bound Construction

Update uplink of first bit.
Round Robin Lower Bound Construction

Finished.
Subexponential lower bound
Some out degrees are 3
New construction

- Lower bound is exponential in MDP size
New construction

- Lower bound is exponential in MDP size
- Binary node-out-degree
New construction

- Lower bound is exponential in MDP size
- Binary node-out-degree
- Applies to LP and acyclic USO
New construction

- Lower bound is exponential in MDP size
- Binary node-out-degree
- Applies to LP and acyclic USO
- Paper and animation available at www.oliverfriedmann.com
(LHS variables show binary node-out-degree)

\[
\begin{align*}
\text{max} & \quad \sum_{i=1}^{n} \left( (a_1^i + b_1^i + c_1^i) \cdot (\Omega(g_i) + \varepsilon \cdot \Omega(h_i)) + (d_1^i + e_1^i) \cdot \varepsilon \cdot \Omega(h_i) + e_0^i \cdot \Omega(s) \right) \\
\text{s.t.} & \quad (a_i) \quad a_1^i + a_0^i = 1 + a_{i-1}^0 + \varepsilon \cdot (a_{i-1} - b_{i-1} + c_{i-1} + d_{i-1} + e_{i-1}) \\
& \quad (b_i) \quad b_1^i + b_0^i = 1 + b_{i+1}^0 + d_{i+1}^0 \\
& \quad (c_i) \quad c_1^i + c_0^i = 1 + \begin{cases} c_{i+1}^0 & \text{if } i < n \\ \sum_{j=1}^{n} e_j^0 & \text{if } i = n \end{cases} \\
& \quad (d_i) \quad d_1^i + d_0^i = 1 + (a_1^i + b_1^i + c_1^i + d_1^i + e_1^i) \cdot \begin{cases} \frac{1-\varepsilon}{2} & \text{if } i > 1 \\ 1 - \varepsilon & \text{if } i = 1 \end{cases} \\
& \quad (e_i) \quad e_1^i + e_0^i = 1 + (a_1^i + b_1^i + c_1^i + d_1^i + e_1^i) \cdot \begin{cases} \frac{1-\varepsilon}{2} & \text{if } i > 1 \\ 1 - \varepsilon & \text{if } i = 1 \end{cases}
\end{align*}
\]
Concluding Remarks
Open problems

- Obtain lower bounds for related history-based pivoting rules
  - Least-recently basic
  - Least-recently entered
  - Least basic iterations
Open problems

- Obtain lower bounds for related history-based pivoting rules
  - Least-recently basic
  - Least-recently entered
  - Least basic iterations

- Get lower bounds for the Network simplex method
Open problems

- Obtain lower bounds for related history-based pivoting rules
  - Least-recently basic
  - Least-recently entered
  - Least basic iterations

- Get lower bounds for the Network simplex method

- Is there a strongly polytime algorithm for LP?
The slide usually called “the end”.

Thank you for listening!