An exponential lower bound for Cunningham's rule

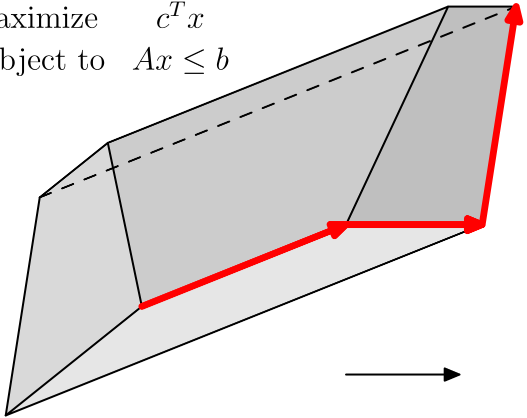
Oliver Friedmann

Revised March 20, 2015 (DA)

Linear Programming and Simplex

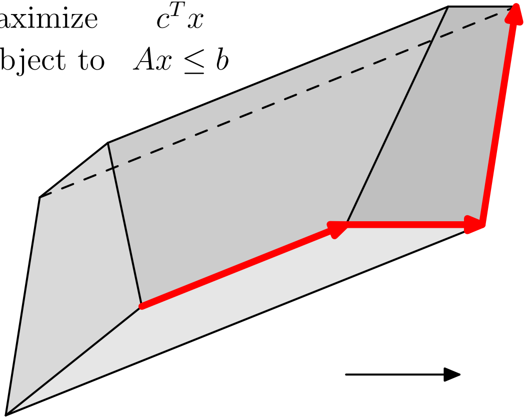
The simplex method, Dantzig (1947)

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \end{array}$$



The simplex method, Dantzig (1947)

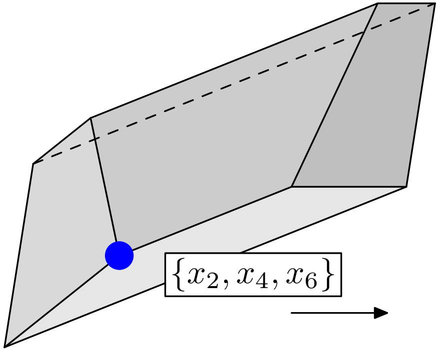
$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \le b \end{array}$$



- The first step is to add slack variables making a system of equations called a dictionary

Basic feasible solutions and pivoting

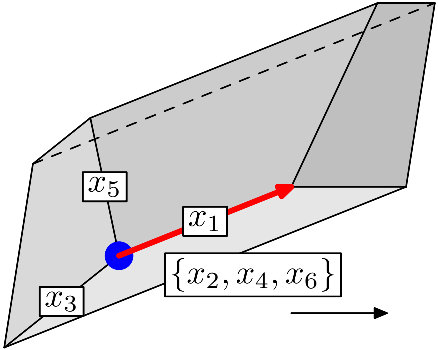
$$
\begin{array}{llcl}
\max & 2x_1 - 2x_3 - 2x_5 - x_6 & & \\
\text{s.t.} & \frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 - \frac{2}{3}x_5 & = & 1 \\
& x_3 + x_4 - x_6 & = & 1 \\
& x_5 + x_6 & = & 1 \\
& x_1, x_2, x_3, x_4, x_5, x_6 & \geq & 0
\end{array}
$$



- The vertices of the polytope are represented by basic feasible solutions formed by solving for 3 basic variables

Basic feasible solutions and pivoting

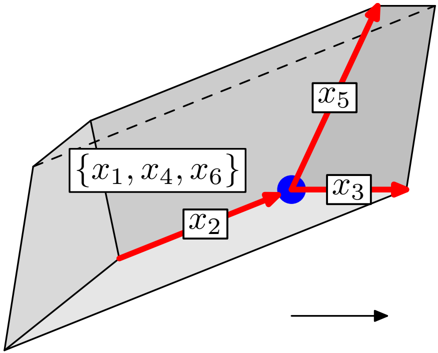
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{-1 + 2x_1 - 2x_3 - x_5} \\ \text{s.t.} & x_2 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\ & x_4 & = & 2 - x_3 - x_5 \\ & x_6 & = & 1 - x_5 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0} \end{array}$$



- 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).

Basic feasible solutions and pivoting

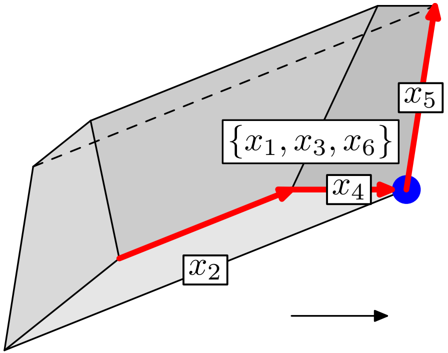
$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{5 - 6x_2 + 2x_3 + 3x_5} \\
\text{s.t.} & x_1 & = & 3 - 3x_2 + 2x_3 + 2x_5 \\
 & x_4 & = & 2 - x_3 - x_5 \\
 & x_6 & = & 1 - x_5 \\
 & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0}
\end{array}
$$



- 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

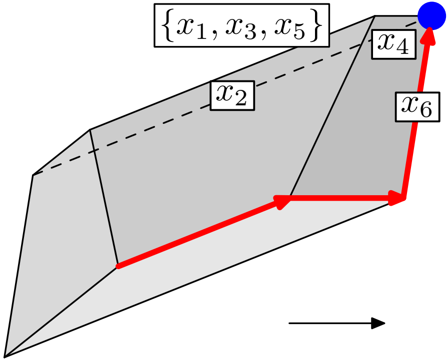
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{9 - 6x_2 - 2x_4 + x_5} \\ \text{s.t.} & x_1 & = & 7 - 3x_2 - 2x_4 \\ & x_3 & = & 2 - x_4 - x_5 \\ & x_6 & = & 1 - x_5 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0} \end{array}$$



- 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

$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{10 - 6x_2 - 2x_4 - x_6} \\
\text{s.t.} & x_1 & = & 7 - 3x_2 - 2x_4 \\
 & x_3 & = & 1 - x_4 + x_6 \\
 & x_5 & = & 1 - x_6 \\
 & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0}
\end{array}
$$



- 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

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 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 are known to be exponential.
- Randomized rules: eg. Random edge and Random facet: Random facet is subexponential (Kalai), (Sharir-Welzl)

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 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.)

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 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

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.

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)

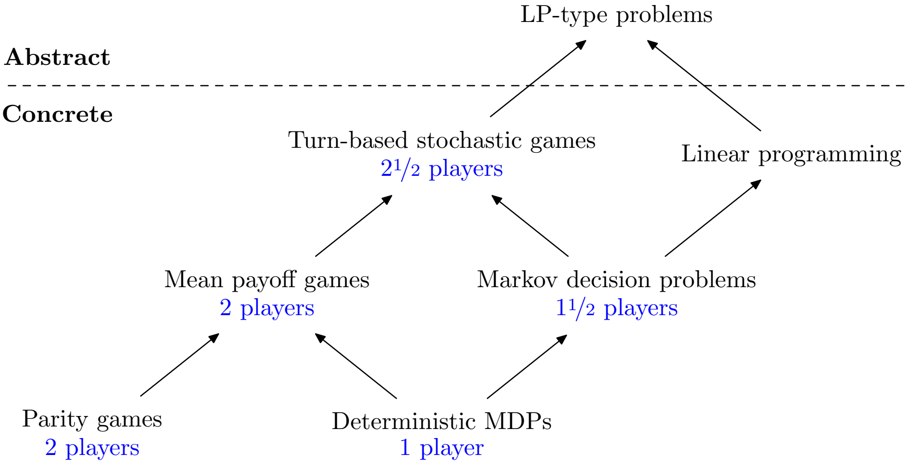
History-based pivoting rules

Cunningham's ROUND-ROBIN rule

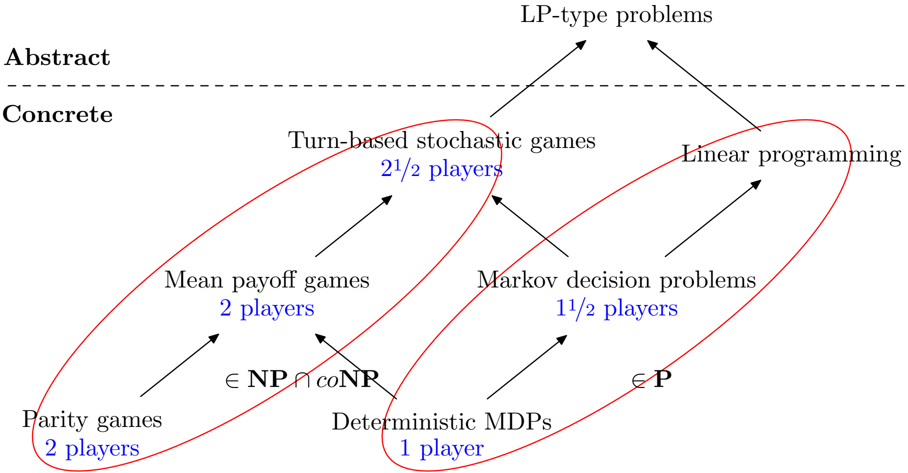
Order the variables cyclically in any way. Starting from the last entered variable, consider the vertices in circular order choosing the first candidate with positive objective coefficient.

- 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 (?)

From Policy Iteration to Simplex



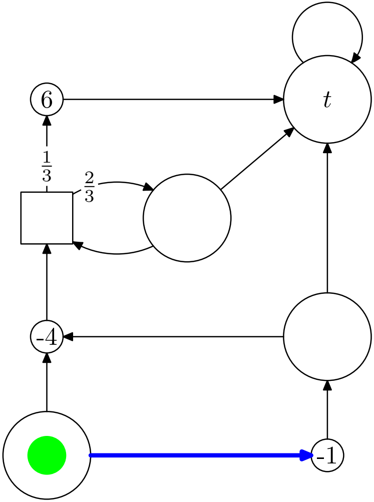
From Policy Iteration to Simplex



Games and Policy Iteration

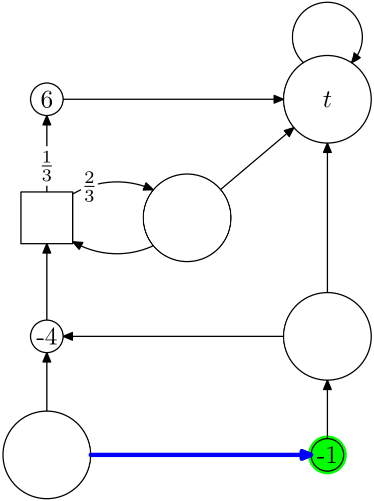
Markov decision processes (MDPs)

- 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)

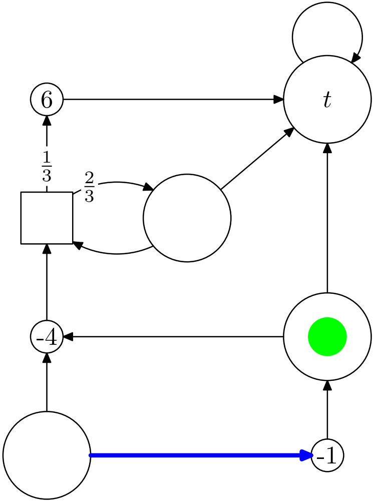
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: -1

Markov decision processes (MDPs)

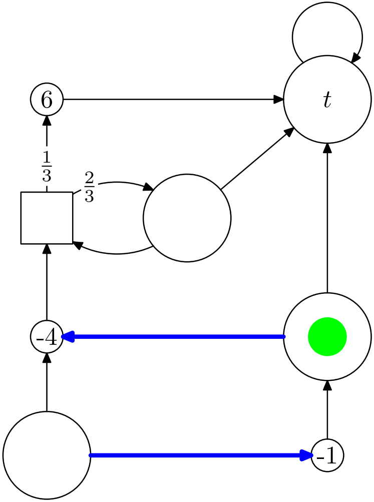
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: -1

Markov decision processes (MDPs)

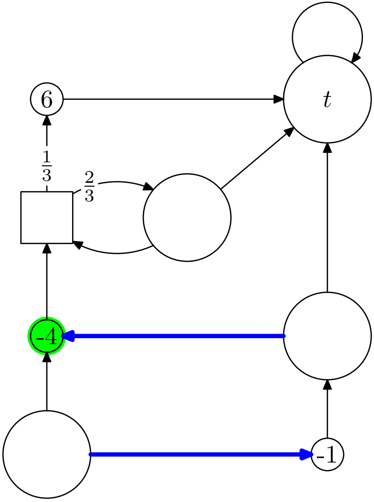
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: -1

Markov decision processes (MDPs)

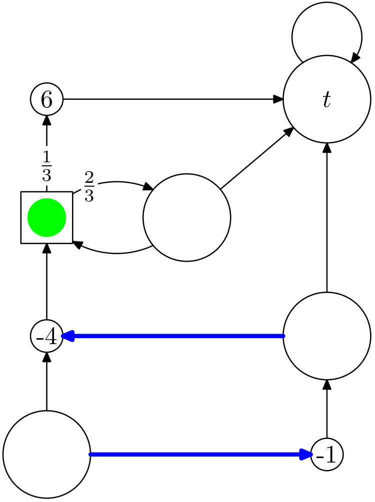
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1 - 4$

Markov decision processes (MDPs)

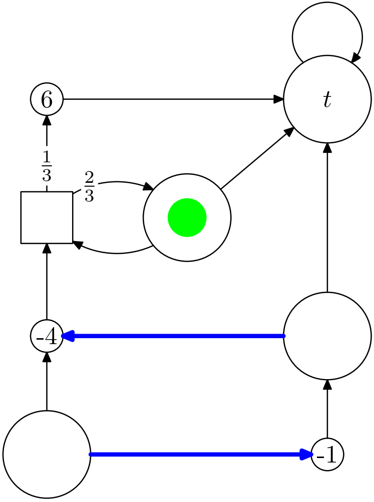
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1-4$

Markov decision processes (MDPs)

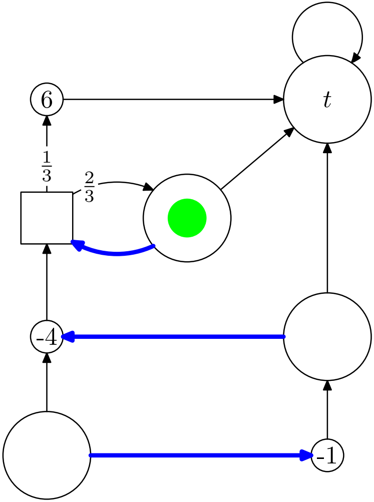
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1 - 4$

Markov decision processes (MDPs)

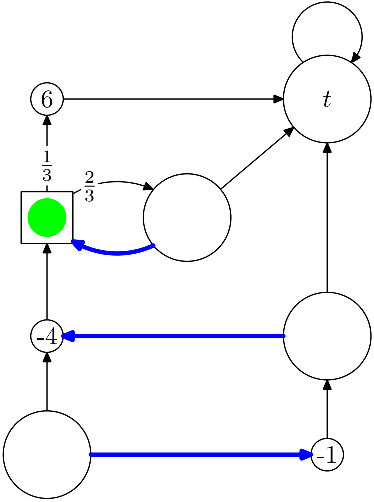
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1 - 4$

Markov decision processes (MDPs)

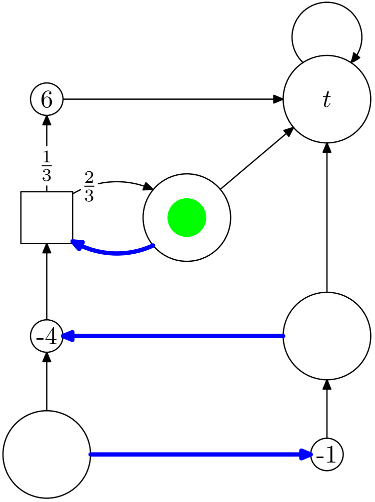
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1 - 4$

Markov decision processes (MDPs)

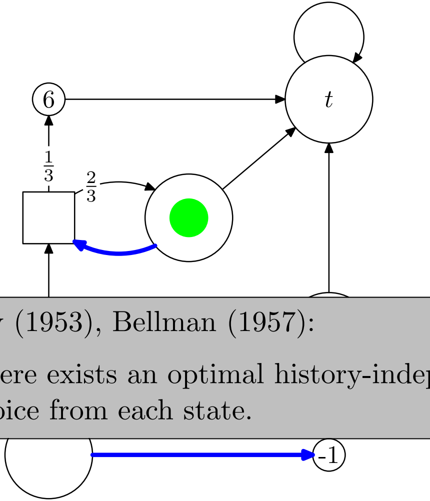
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



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



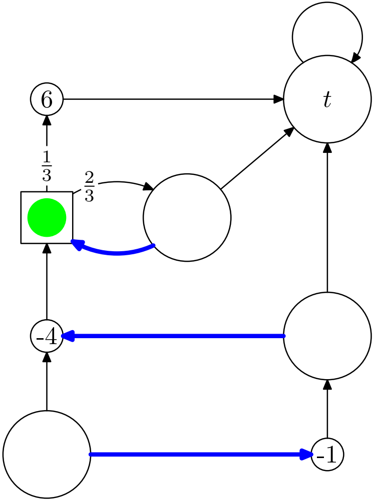
Shapley (1953), Bellman (1957):

There exists an optimal history-indep
choice from each state.

Reward: $-1 - 4$

Markov decision processes (MDPs)

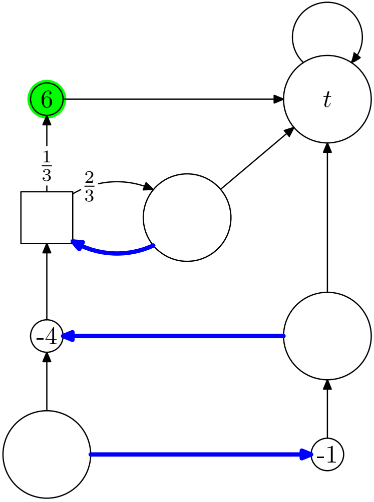
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1-4$

Markov decision processes (MDPs)

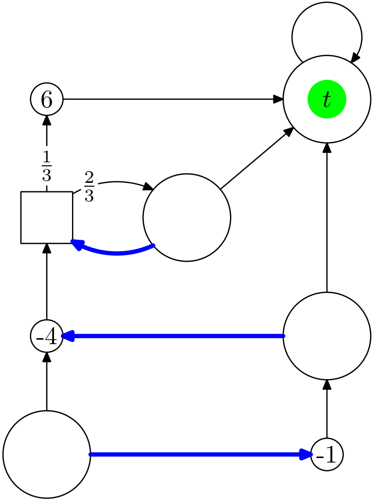
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1 - 4 + 6$

Markov decision processes (MDPs)

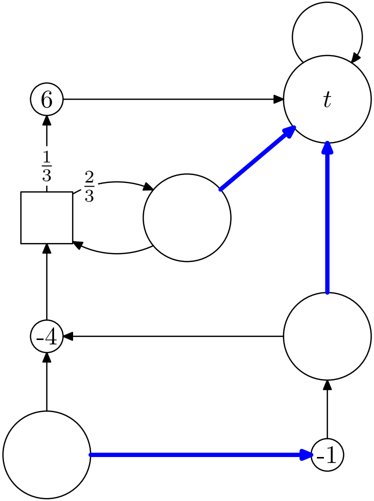
- Large circle: player 0 (us)
- Square: player 1 (random)
- Small circle: our reward
- Edge: possible actions (with probability)
- Blue edge: action taken
- Green circle: current node
- Goal: maximize expected reward before reaching t from current node



Reward: $-1 - 4 + 6 \ = \ 1$

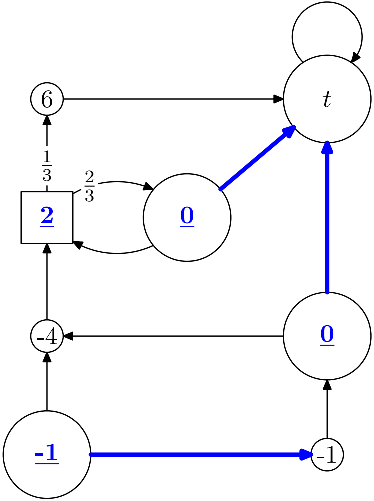
Policies and corresponding values

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



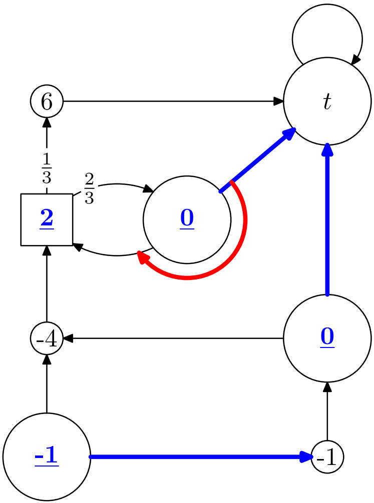
Policies and corresponding values

- A policy π 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 π, is the expected reward moving from i to t (blue numbers)



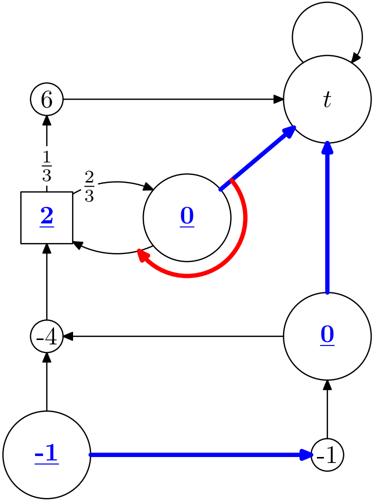
Policies and corresponding values

- A policy π 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 π, is the expected reward moving from i to t (blue numbers)
- A policy change (pivot) is an improving switch w.r.t. π if it improves the value.



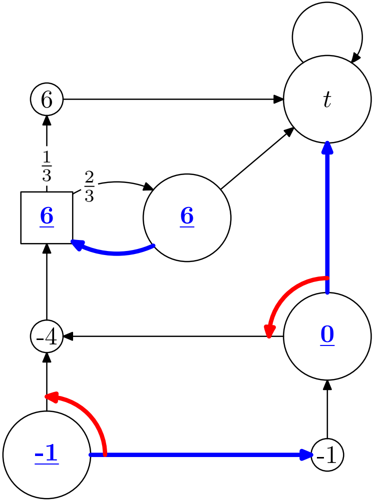
Policies and corresponding values

- A policy π 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 π, is the expected reward moving from i to t (blue numbers)
- A policy change (pivot) is an improving switch w.r.t. π if it improves the value.
- Policy improvement: make any improving switch



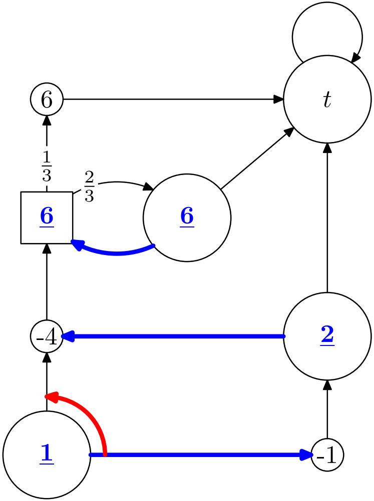
Policies and corresponding values

- A policy π 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 π, is the expected reward moving from i to t (blue numbers)
- A policy change (pivot) is an improving switch w.r.t. π if it improves the value.
- Policy improvement: make any improving switch



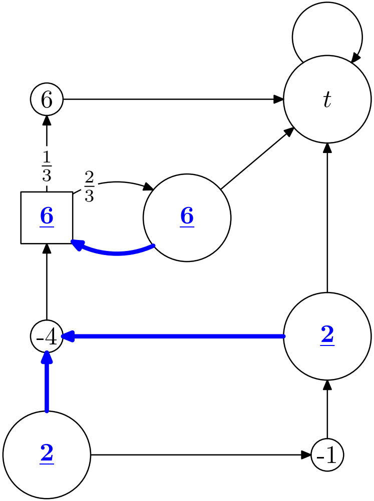
Policies and corresponding values

- A policy π 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 π, is the expected reward moving from i to t (blue numbers)
- A policy change (pivot) is an improving switch w.r.t. π if it improves the value.
- Policy improvement: make any improving switch



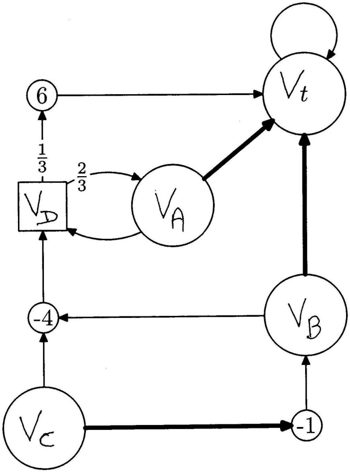
Policies and corresponding values

- A policy π 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 π, is the expected reward moving from i to t (blue numbers)
- A policy change (pivot) is an improving switch w.r.t. π if it improves the value.
- Policy improvement: make any improving switch
- A policy π^* is optimal iff there are no improving switches.



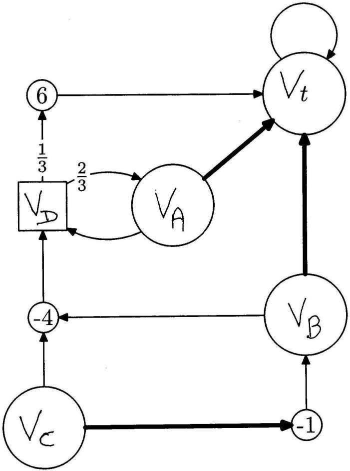
Formulating an LP - I

- Optimality conditions for each vertex: $v_i = \mathrm{VAL}_\pi(i)$



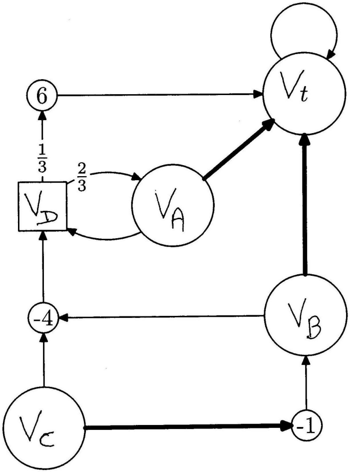
Formulating an LP - I

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



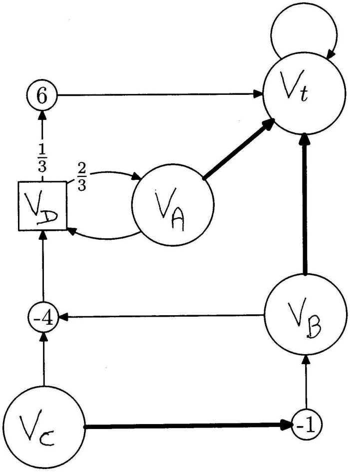
Formulating an LP - I

- Optimality conditions for each vertex: $v_i = \mathrm{VAL}_\pi(i)$
- $v_A = \max\{v_D, v_t\}$
- $v_B = \max\{-4 + v_D, v_t\}$



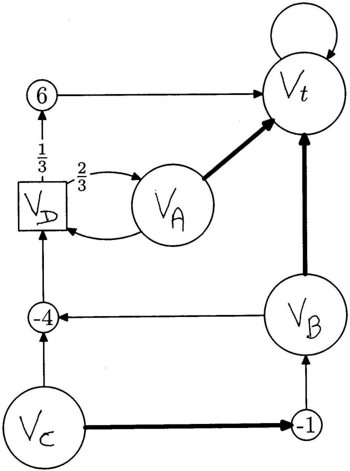
Formulating an LP - I

- Optimality conditions for each vertex: $v_i = \mathrm{VAL}_\pi(i)$
- $v_A = \max\{v_D, v_t\}$
- $v_B = \max\{-4 + v_D, v_t\}$
- $v_C = \max\{-4 + v_D, -1 + v_B\}$



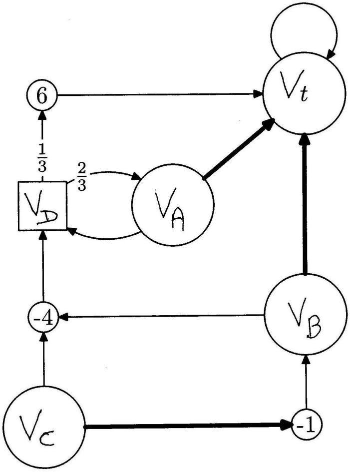
Formulating an LP - I

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



Formulating an LP - I

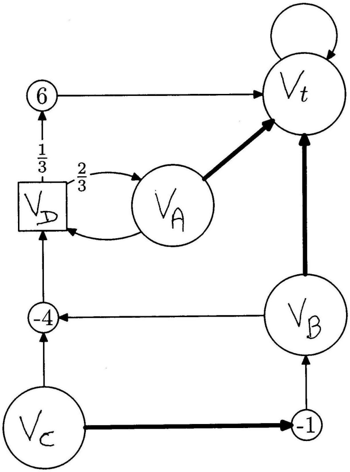
- Optimality conditions for each vertex: $v_i = \mathrm{VAL}_\pi(i)$
- $v_A = \max\{v_D, v_t\}$
- $v_B = \max\{-4 + v_D, v_t\}$
- $v_C = \max\{-4 + v_D, -1 + v_B\}$
- $v_D = \frac{1}{3}(6 + v_t) + \frac{2}{3} v_A$
- Note: $v_t = 0$



Formulating an LP - II

- Optimality conditions

$$\begin{aligned} v_A &= \max\{v_D, 0\} \\ v_B &= \max\{-4 + v_D, 0\} \\ v_C &= \max\{-4 + v_D, -1 + v_B\} \\ v_D &= 2 + \frac{2}{3} v_A \end{aligned}$$



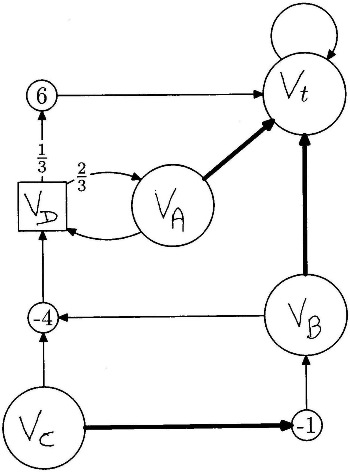
Formulating an LP - II

- Optimality conditions

$$\begin{aligned} v_A &= \max\{v_D, 0\} \\ v_B &= \max\{-4 + v_D, 0\} \\ v_C &= \max\{-4 + v_D, -1 + v_B\} \\ v_D &= 2 + \frac{2}{3} v_A \end{aligned}$$

- Linear Program

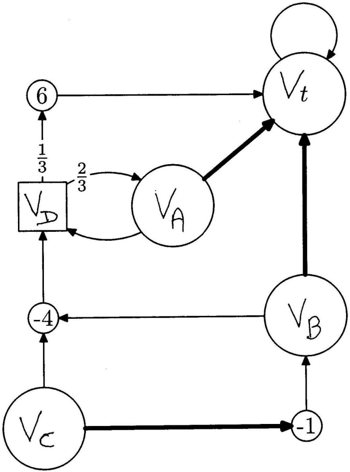
$$\begin{aligned} \min w &= v_A + v_B + v_C \\ v_A &\geq v_D \\ v_A &\geq 0 \\ v_B &\geq -4 + v_D \\ v_B &\geq 0 \\ v_C &\geq -4 + v_D \\ v_C &\geq -1 + v_B \\ v_D &= 2 + \frac{2}{3} v_A \end{aligned}$$



Formulating an LP - III

- Optimality conditions

$$
\begin{aligned}
v_A &= \max\{v_D, 0\} \\
v_B &= \max\{-4 + v_D, 0\} \\
v_C &= \max\{-4 + v_D, -1 + v_B\} \\
v_D &= 2 + \frac{2}{3} v_A
\end{aligned}
$$



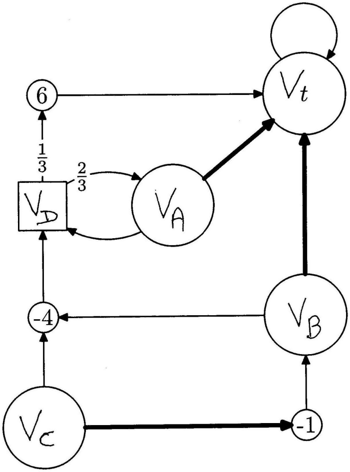
Formulating an LP - III

- Optimality conditions

$$\begin{aligned} v_A &= \max\{v_D, 0\} \\ v_B &= \max\{-4+v_D, 0\} \\ v_C &= \max\{-4+v_D, -1+v_B\} \\ v_D &= 2+\frac{2}{3}v_A \end{aligned}$$

- Linear Program (v_D eliminated)

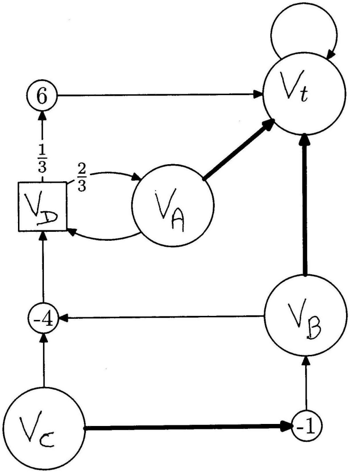
$$\begin{aligned} \min w &= v_A + v_B + v_C \\ v_A &\geq 2+\frac{2}{3}v_A \\ v_A &\geq 0 \\ v_B &\geq -2+\frac{2}{3}v_A \\ v_B &\geq 0 \\ v_C &\geq -2+\frac{2}{3}v_A \\ v_C &\geq -1+v_B \end{aligned}$$



Formulating an LP - IV

- Optimality conditions

$$
\begin{aligned}
v_A &= \max\{v_D, 0\} \\
v_B &= \max\{-4 + v_D, 0\} \\
v_C &= \max\{-4 + v_D, -1 + v_B\} \\
v_D &= 2 + \frac{2}{3} v_A
\end{aligned}
$$



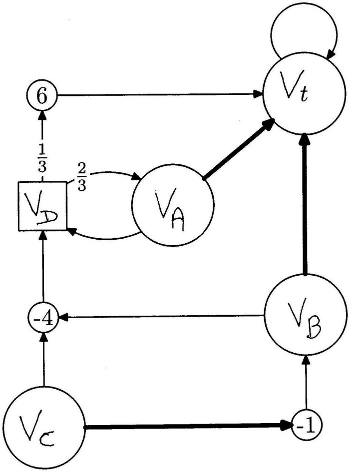
Formulating an LP - IV

- Optimality conditions

$$
\begin{aligned}
v_A &= \max\{v_D, 0\} \\
v_B &= \max\{-4 + v_D, 0\} \\
v_C &= \max\{-4 + v_D, -1 + v_B\} \\
v_D &= 2 + \frac{2}{3} v_A
\end{aligned}
$$

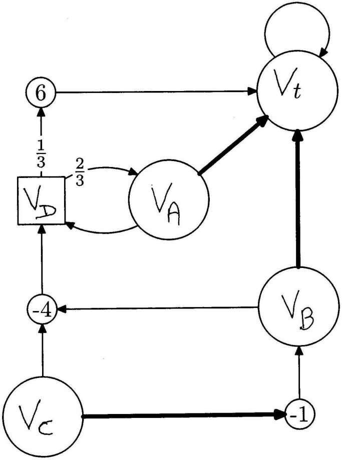
- Linear Program (Dual standard form)

$$
\begin{array}{rcr}
\min w = v_A + v_B + v_C & & \\
\frac{1}{3} v_A & \geq & 2 \\
v_A & \geq & 0 \\
-\frac{2}{3} v_A + v_B & \geq & -2 \\
v_B & \geq & 0 \\
-\frac{2}{3} v_A + v_C & \geq & -2 \\
-v_B + v_C & \geq & -1
\end{array}
$$



From dual to primal

$$
\begin{array}{rcll}
\min w = v_A + v_B + v_C & & \textcolor{red}{\textit{Dual}} & \\
\frac{1}{3}v_A & \geq & 2 & (\textcolor{blue}{x_1}) \\
v_A & \geq & 0 & (\textcolor{blue}{x_2}) \\
-\frac{2}{3}v_A + v_B & \geq & -2 & (\textcolor{blue}{x_3}) \\
v_B & \geq & 0 & (\textcolor{blue}{x_4}) \\
-\frac{2}{3}v_A \quad + v_C & \geq & -2 & (\textcolor{blue}{x_5}) \\
-v_B + v_C & \geq & -1 & (\textcolor{blue}{x_6})
\end{array}
$$

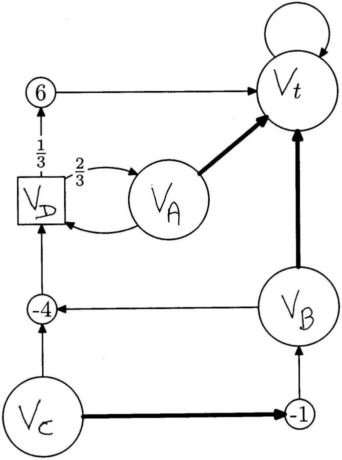


From dual to primal

$$
\begin{array}{lrrl}
\min w = v_A + v_B + v_C & & & \textit{Dual} \\
\frac{1}{3}v_A & \geq & 2 & (x_1) \\
v_A & \geq & 0 & (x_2) \\
-\frac{2}{3}v_A + v_B & \geq & -2 & (x_3) \\
v_B & \geq & 0 & (x_4) \\
-\frac{2}{3}v_A \qquad + v_C & \geq & -2 & (x_5) \\
-v_B + v_C & \geq & -1 & (x_6)
\end{array}
$$

-

$$
\begin{array}{lrrl}
\max z = 2x_1 \quad - 2x_3 \quad - 2x_5 & & & \textit{Primal} \\
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 \quad - \frac{2}{3}x_5 & = & 1 & \\
x_3 + x_4 & = & 1 & \\
x_5 + x_6 & = & 1 & \\
x_i & \geq & 0 &
\end{array}
$$

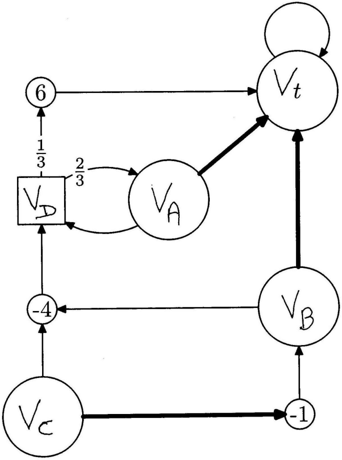


From dual to primal

$$
\begin{array}{llll}
\min w = v_A + v_B + v_C & & & \textit{Dual} \\
\frac{1}{3}v_A & \geq & 2 & (x_1) \\
v_A & \geq & 0 & (x_2) \\
-\frac{2}{3}v_A + v_B & \geq & -2 & (x_3) \\
v_B & \geq & 0 & (x_4) \\
-\frac{2}{3}v_A + v_C & \geq & -2 & (x_5) \\
-v_B + v_C & \geq & -1 & (x_6)
\end{array}
$$

■

$$
\begin{array}{llll}
\max z = 2x_1 - 2x_3 - 2x_5 & & & \textit{Primal} \\
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 - \frac{2}{3}x_5 & = & 1 \\
x_3 + x_4 & = & 1 \\
x_5 + x_6 & = & 1 \\
x_i & \geq & 0
\end{array}
$$



- What are the variables x_i ?

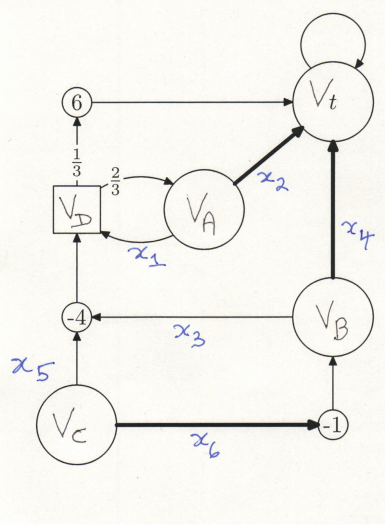
From dual to primal

$$
\begin{array}{lcrl}
\min w = v_A + v_B + v_C & & \textit{Dual} & \\
\frac{1}{3}v_A & \geq & 2 & (x_1) \\
v_A & \geq & 0 & (x_2) \\
-\frac{2}{3}v_A + v_B & \geq & -2 & (x_3) \\
v_B & \geq & 0 & (x_4) \\
-\frac{2}{3}v_A \quad + v_C & \geq & -2 & (x_5) \\
-v_B + v_C & \geq & -1 & (x_6)
\end{array}
$$

■

$$
\begin{array}{rcl}
\max z = 2x_1 \quad - 2x_3 \quad - 2x_5 & & \textit{Primal} \\
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 \quad - \frac{2}{3}x_5 & = & 1 \\
x_3 + x_4 & = & 1 \\
x_5 + x_6 & = & 1 \\
x_i & \geq & 0
\end{array}
$$

- The x_i are action variables !



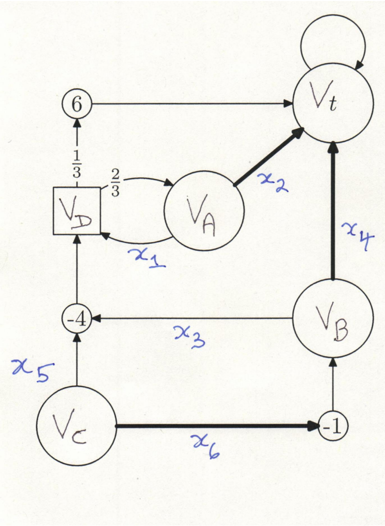
From dual to primal

$$
\begin{array}{lcrl}
\min w = v_A + v_B + v_C & & \textit{Dual} & \\
\frac{1}{3}v_A & \geq & 2 & (x_1) \\
v_A & \geq & 0 & (x_2) \\
-\frac{2}{3}v_A + v_B & \geq & -2 & (x_3) \\
v_B & \geq & 0 & (x_4) \\
-\frac{2}{3}v_A \quad + v_C & \geq & -2 & (x_5) \\
-v_B + v_C & \geq & -1 & (x_6)
\end{array}
$$

-

$$
\begin{array}{lcl}
\max z = 2x_1 \quad - 2x_3 \quad - 2x_5 & & \textit{Primal} \\
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 \quad - \frac{2}{3}x_5 & = & 1 \\
x_3 + x_4 & = & 1 \\
x_5 + x_6 & = & 1 \\
x_i & \geq & 0
\end{array}
$$

- The x_i are action variables !



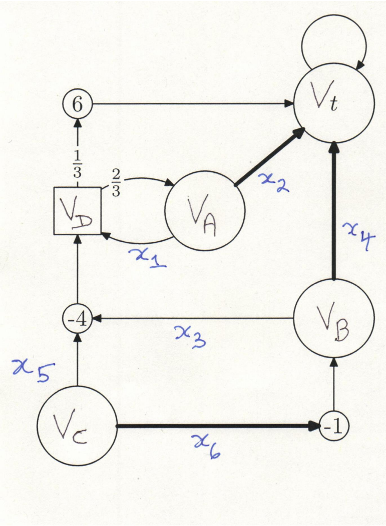
From dual to primal

$$
\begin{array}{lcrl}
\min w = v_A + v_B + v_C & & \textit{Dual} & \\
\frac{1}{3}v_A & \geq & 2 & (x_1) \\
v_A & \geq & 0 & (x_2) \\
-\frac{2}{3}v_A + v_B & \geq & -2 & (x_3) \\
v_B & \geq & 0 & (x_4) \\
-\frac{2}{3}v_A + v_C & \geq & -2 & (x_5) \\
-v_B + v_C & \geq & -1 & (x_6)
\end{array}
$$

■

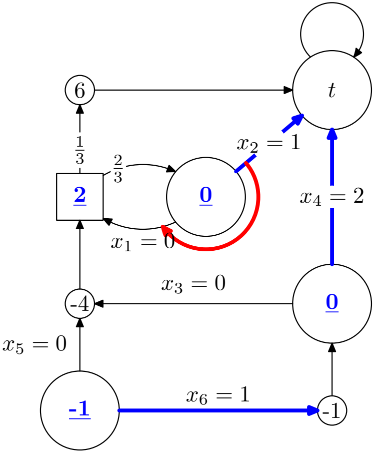
$$
\begin{array}{lcl}
\max z = 2x_1 - 2x_3 - 2x_5 & & \textit{Primal} \\
\frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 - \frac{2}{3}x_5 & = & 1 \\
x_3 + x_4 & = & 1 \\
x_5 + x_6 & = & 1 \\
x_i & \geq & 0
\end{array}
$$

- The x_i are action variables !



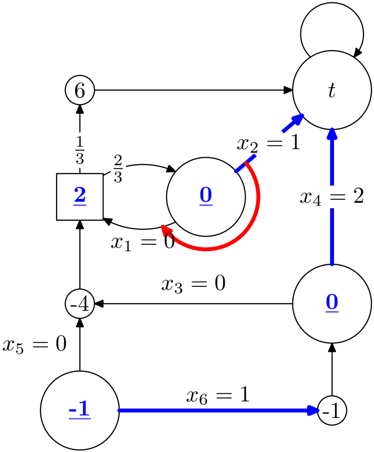
Variables of the primal LP

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



Variables of the primal LP

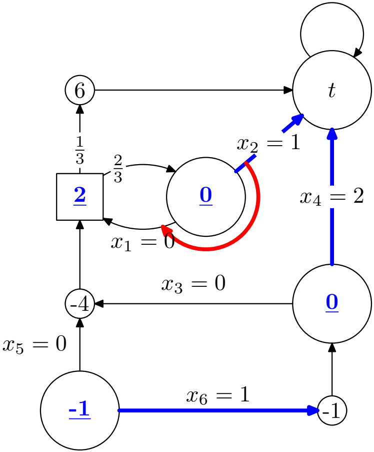
- x_i is the expected number of times action i is used, summed over all starting states.
- The actions taken form the basic variables of the LP



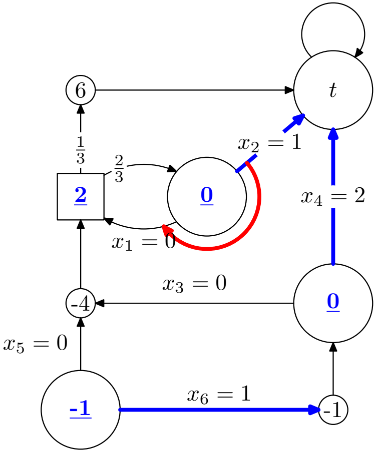
Variables of the primal LP

- 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
- Solving for $\{x_2, x_4, x_6\}$ we get:

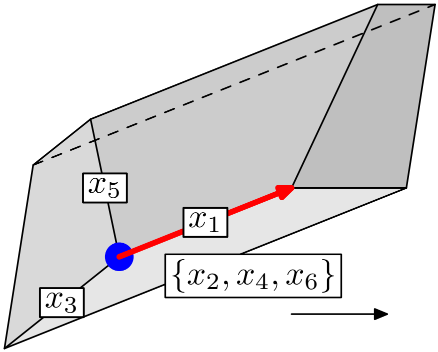
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{-1 + 2x_1 - 2x_3 - x_5} \\ \text{s.t.} & x_2 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\ & x_4 & = & 2 - x_3 - x_5 \\ & x_6 & = & 1 - x_5 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0} \end{array}$$



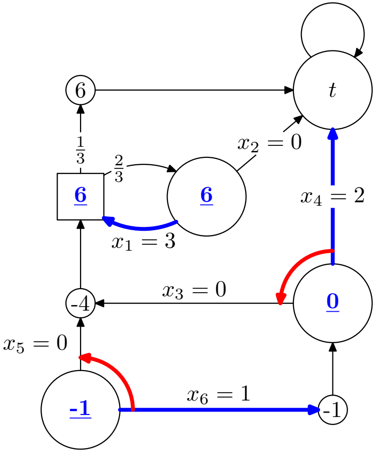
Policy improvements are LP pivots



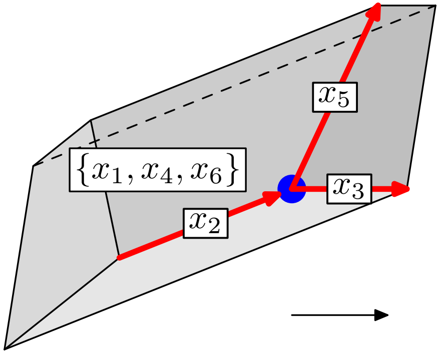
$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{-1 + 2x_1 - 2x_3 - x_5} \\
\text{s.t.} & x_2 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\
& x_4 & = & 2 - x_3 - x_5 \\
& x_6 & = & 1 - x_5 \\
& \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0}
\end{array}
$$



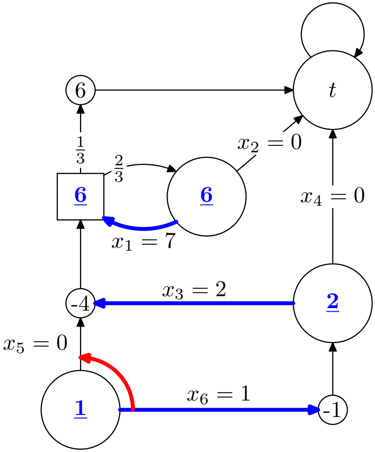
Policy improvements are LP pivots



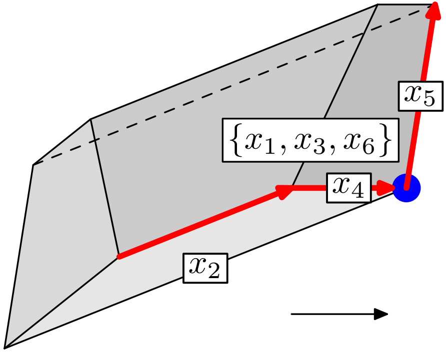
$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{5 - 6x_2 + 2x_3 + 3x_5} \\
\text{s.t.} & x_1 & = & 3 - 3x_2 + 2x_3 + 2x_5 \\
 & x_4 & = & 2 - x_3 - x_5 \\
 & x_6 & = & 1 - x_5 \\
 & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0}
\end{array}
$$



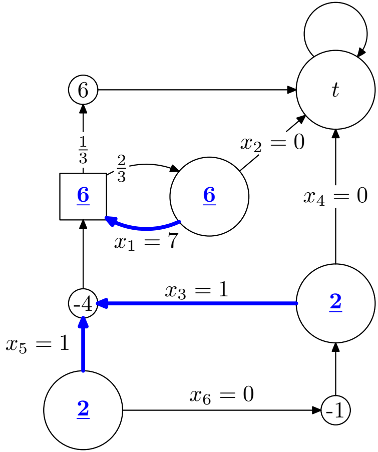
Policy improvements are LP pivots



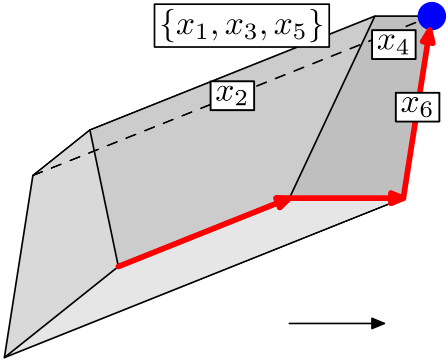
$$\begin{aligned} \max \quad & 9 - 6x_2 - 2x_4 + x_5 \\ \text{s.t.} \quad & x_1 = 7 - 3x_2 - 2x_4 \\ & x_3 = 2 - x_4 - x_5 \\ & x_6 = 1 - x_5 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned}$$



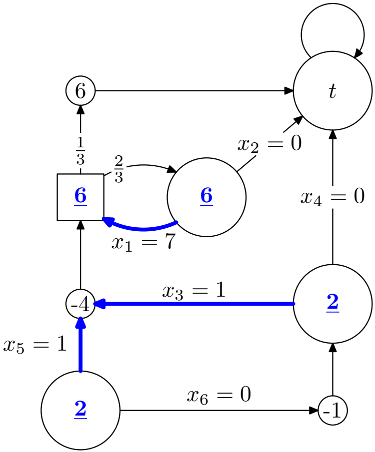
Policy improvements are LP pivots



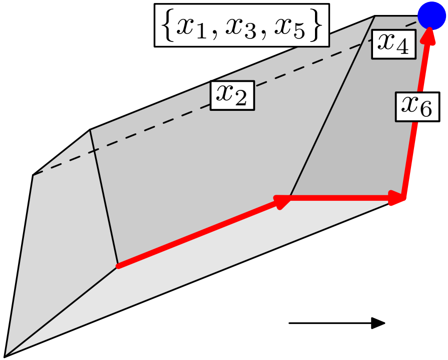
$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{10 - 6x_2 - 2x_4 - x_6} \\
\text{s.t.} & x_1 & = & 7 - 3x_2 - 2x_4 \\
& x_3 & = & 1 - x_4 + x_6 \\
& x_5 & = & 1 - x_6 \\
& \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0}
\end{array}
$$



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 π^*:

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

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 π^*:

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

  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:

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

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

Primal and dual LPs for MDPs

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

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

$$\text{maximize} \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a$$

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

$$x_a \geq 0, \quad \forall a$$

Primal and dual LPs for MDPs

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

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

$$\text{maximize} \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a$$

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

$$x_a \geq 0, \quad \forall a$$

- policy improvement algorithm $\equiv$ simplex method on primal

Primal and dual LPs for MDPs

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

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

$$\text{maximize} \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a$$

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

$$x_a \geq 0, \quad \forall a$$

- 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

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

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

$$\text{maximize} \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a$$

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

$$x_a \geq 0, \quad \forall a$$

- 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

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.

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.

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.
- 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

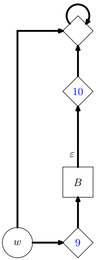
Lower Bound Design Principles

- Simulate binary counter
- End in a single sink loop with no cost

Lower Bound Design Principles

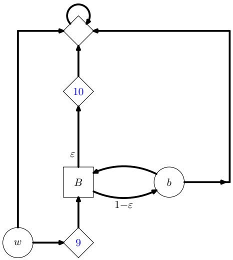
- Simulate binary counter
- End in a single sink loop with no cost
- Incrementation of binary counter consists of intermediate phases

Lower Bound Design Principles



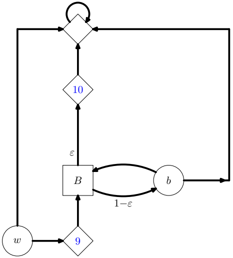
- Simulate binary counter
- End in a single sink loop with no cost
- Incrementation of binary counter consists of intermediate phases
- Structure relating to single bit as profitable pass-through structure if bit is set

Lower Bound Design Principles



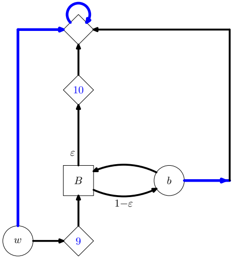
- Simulate binary counter
- End in a single sink loop with no cost
- Incrementation of binary counter consists of intermediate phases
- Structure relating to single bit as profitable pass-through structure if bit is set
- Implementation of bit structure by zero-cost cycles with exponentially small exit-probability

Setting a bit b



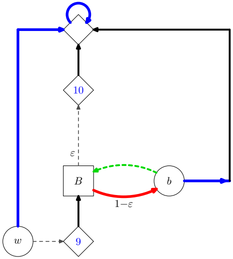
- Choose starting policy

Setting a bit b



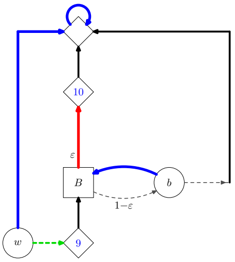
- Choose starting policy
- $v_w = 0, v_b = 0$ $\quad$ b unset

Setting a bit b



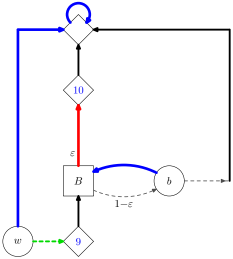
- Choose starting policy
- $v_w = 0, v_b = 0$ $\quad$ b unset
- Improving switch found

Setting a bit b



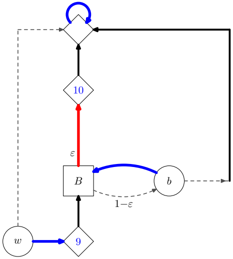
- Choose starting policy
- $v_w = 0, v_b = 0$ $\quad$ b unset
- Improving switch found
- $v_w = 0, v_b = r_{10} > 0$

Setting a bit b



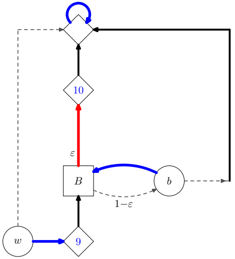
- Choose starting policy
- $v_w = 0, v_b = 0$ b unset
- Improving switch found
- $v_w = 0, v_b = r_{10} > 0$
- Improving switch found

Setting a bit b



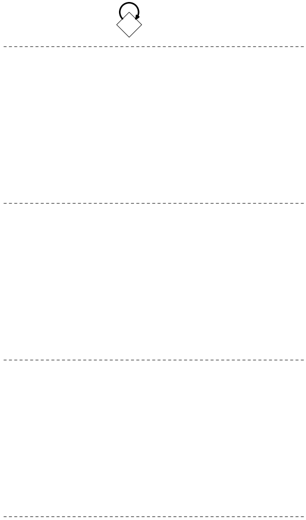
- Choose starting policy
- $v_w = 0, v_b = 0$ b unset
- Improving switch found
- $v_w = 0, v_b = r_{10} > 0$
- Improving switch found
- $v_w = r_9 + r_{10} > 0, v_b = r_{10} > 0$ b set

Setting a bit b



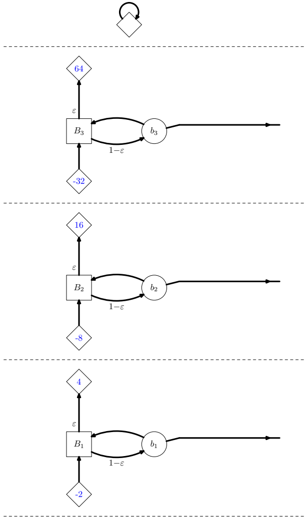
- Choose starting policy
- $v_w = 0, v_b = 0$ b unset
- Improving switch found
- $v_w = 0, v_b = r_{10} > 0$
- Improving switch found
- $v_w = r_9 + r_{10} > 0, v_b = r_{10} > 0$ b set
- No improving switches for any valid choice of r_9, r_{10}

Basic Construction of Counter



Start with a single sink node.

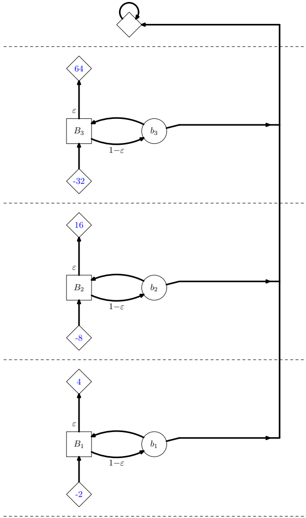
Basic Construction of Counter



Add three bits.
Design principles:

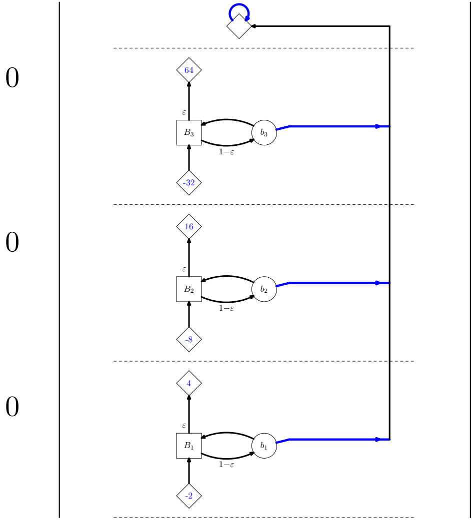
- Every bit is represented by a simple cycle.
- Going through a set bit is profitable.
- Going through an unset bit is unprofitable.
- Going through higher set bits is more profitable than going through lower set bits.

Basic Construction of Counter



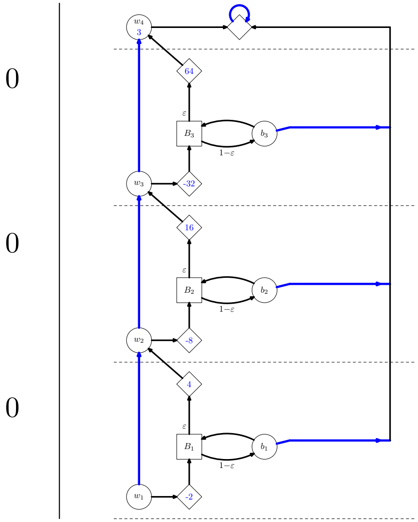
Connect all bits with the sink.

Basic Construction of Counter



Start with unset bits.

Basic Construction of Counter



Add uplink structure that allows to go through set bits or bypass them.

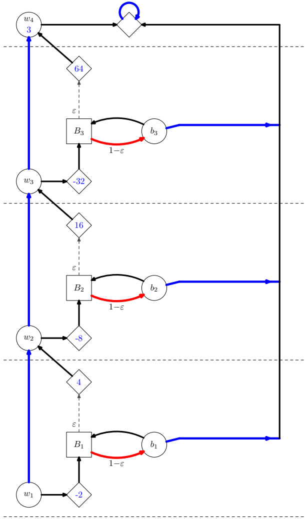
$v_i = 0$ for all i

Basic Construction of Counter

0

0

0



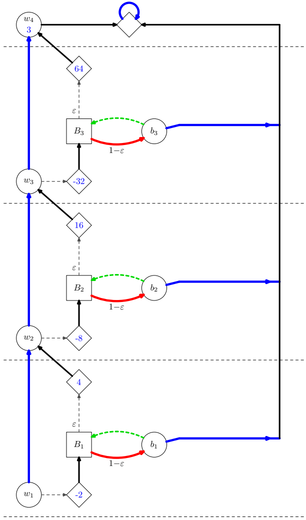
Small exit-probability almost hides the value of the exit nodes completely.

Basic Construction of Counter

0

0

0



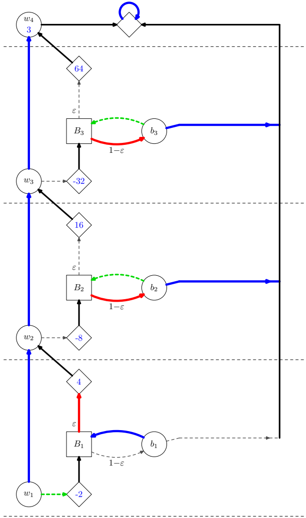
It is improving to set any of the bits.
This will be a general principle: All unset bits can be set at the same time.

Basic Construction of Counter

0

0

1



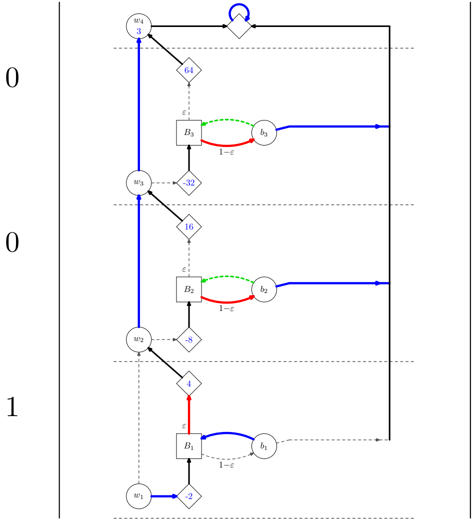
First bit has been set.

$$z = 4$$

$$v_{b_1} = 4$$

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

Basic Construction of Counter

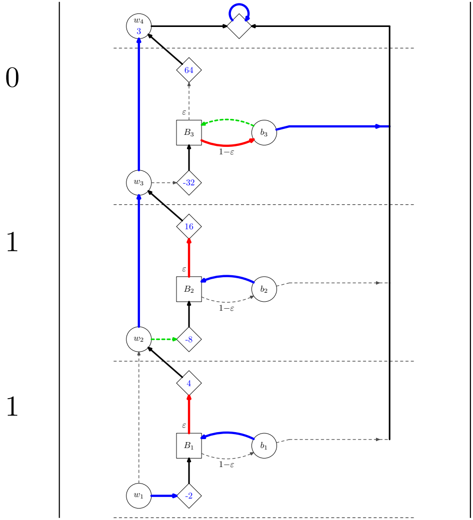


Update uplink of first bit: go through.

$z = 6$

$v_{w_1} = 2, v_{b_1} = 4$

Basic Construction of Counter

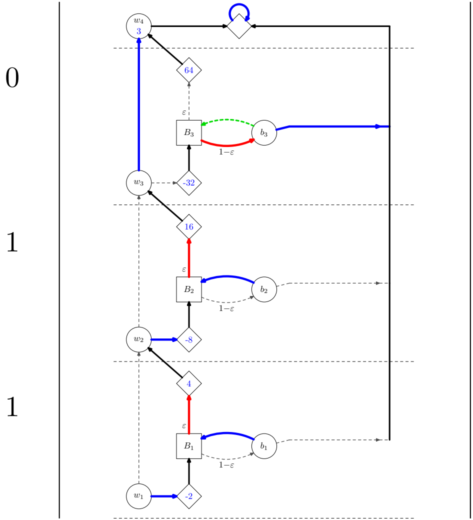


Second bit has been set.

$z = 22$

$v_{w_1} = 2, v_{b_1} = 4, v_{b_2} = 16$

Basic Construction of Counter



Update uplink of second bit: go through.

$z = 46$

$v_{w_1} = 10, v_{b_1} = 12$

$v_{w_2} = 8, v_{b_2} = 16$

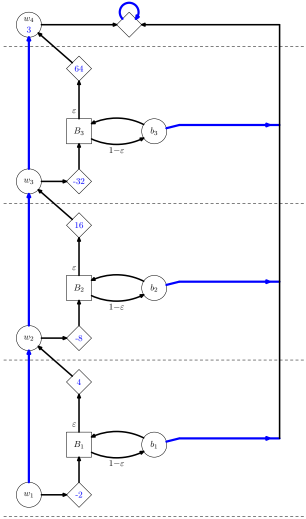
Problem: we cannot reuse the first bit again.

Basic Construction of Counter

0

0

0



Start over again.

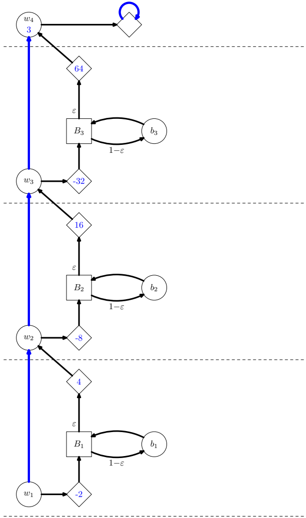
- Setting a higher bit has to lead to resetting the lower bits.
- There have to be outgoing edges of the cycles that have immediate access to the next set bit.

Basic Construction of Counter

0

0

0



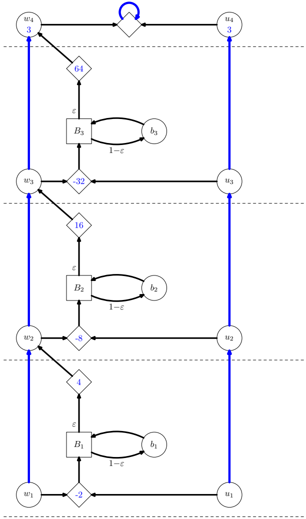
Remove direct link to the sink.

Basic Construction of Counter

0

0

0



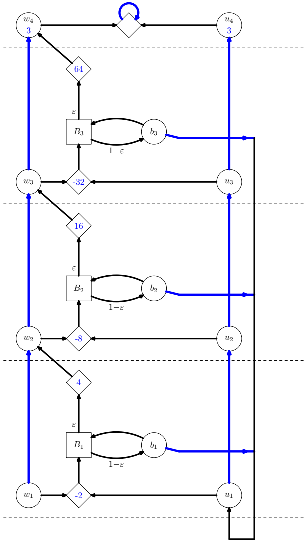
Add second uplink structure, called selector structure.

Basic Construction of Counter

0

0

0



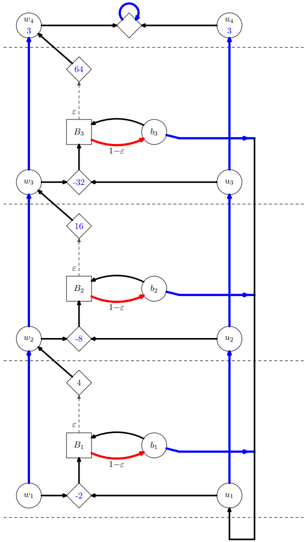
Give all cycles immediate access to the selector structure.

Basic Construction of Counter

0

0

0



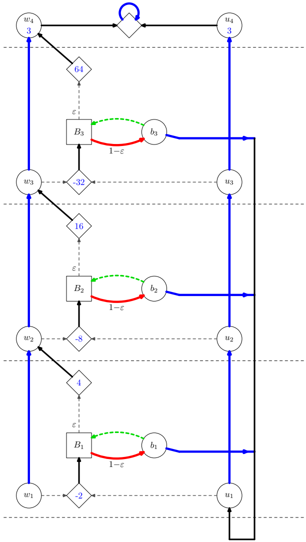
Small exit-probability still hides the value of the exit nodes completely.

Basic Construction of Counter

0

0

0



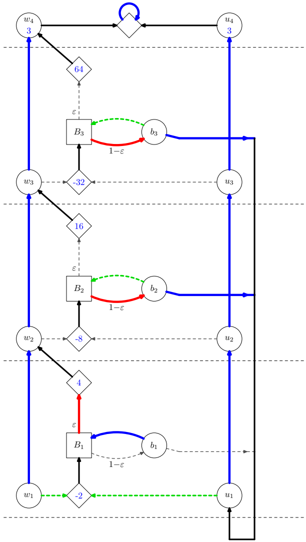
It is still improving to set any of the bits.

Basic Construction of Counter

0

0

1

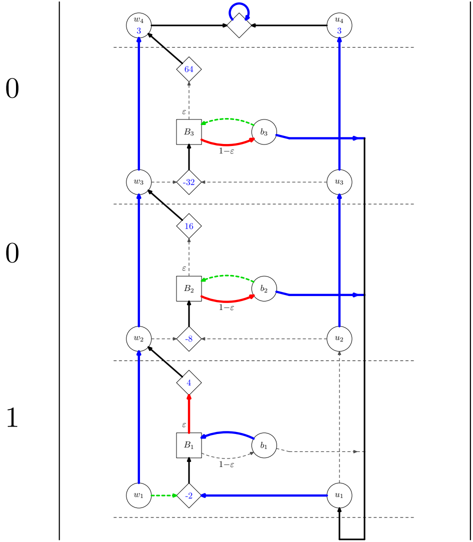


Step 1: First bit has been set.

$z = 4$

$v_{b_1} = 4$

Basic Construction of Counter



Step 2: Update selector of first bit.

$z = 10$

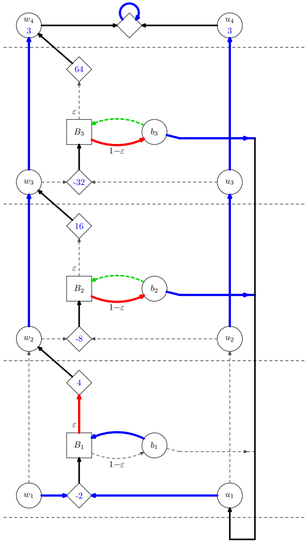
$v_{b_2} = v_{b_3} = v_{u_1} = 2, v_{b_1} = 4$

Basic Construction of Counter

0

0

1



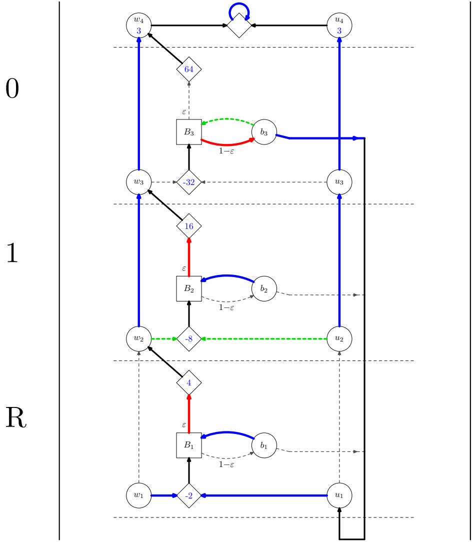
Step 4: Update uplink of first bit. (Step 3 missing)

$z = 12$

$v_{w_1} = v_{b_2} = v_{b_3} = v_{u_1} = 2$

$v_{b_1} = 4$

Basic Construction of Counter



Step 1: Second bit has been set.

$z = 26$

$v_{w_1} = v_{b_3} = v_{u_1} = 2$

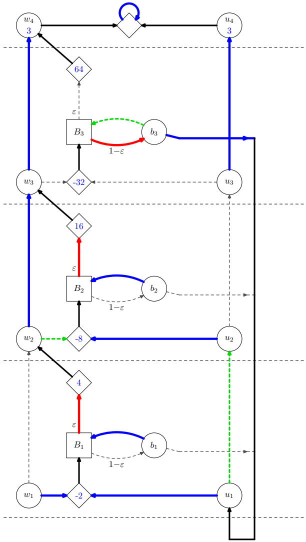
$v_{b_1} = 4, v_{b_2} = 16$

Basic Construction of Counter

0

1

R



Step 2: Update selector of second bit.

$z = 34$

$v_{w_1} = v_{b_3} = v_{u_1} = 2$

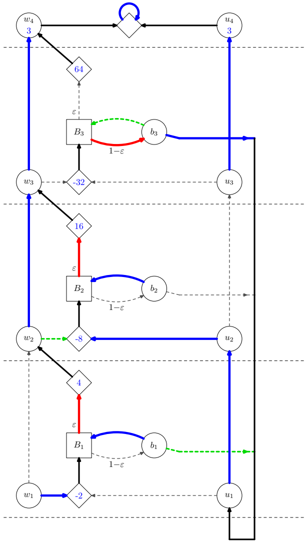
$v_{b_1} = 4, v_{b_2} = 16, v_{u_2} = 8$

Basic Construction of Counter

0

1

R



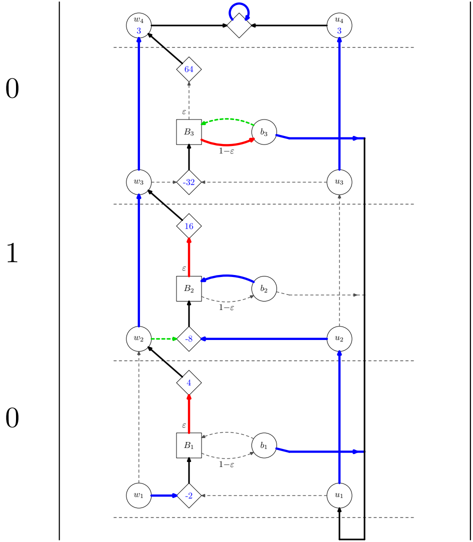
Step 2: Update selector of first bit.

$z = 46$

$v_{b_3} = v_{u_1} = v_{u_2} = 8$

$v_{b_1} = 4, v_{b_2} = 16, v_{w_1} = 2$

Basic Construction of Counter



Step 3: Reset first bit.

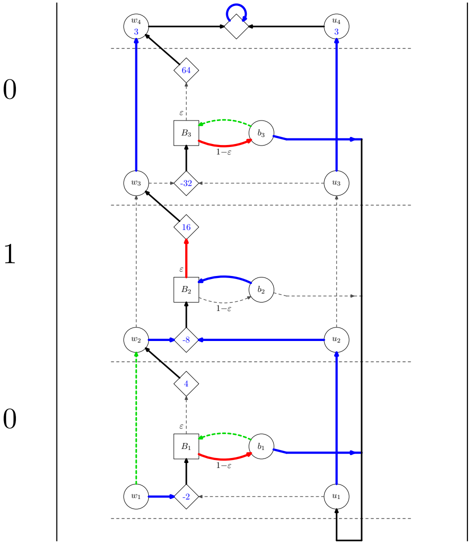
$z = 54 - 4\epsilon$

$v_{b_1} = v_{b_3} = v_{u_1} = v_{u_2} = 8$

$v_{b_2} = 16$

$v_{w_1} = -2 + 4\epsilon + 8(1-\epsilon) = 6 - 4\epsilon$

Basic Construction of Counter



Step 4: Update uplink of second bit.

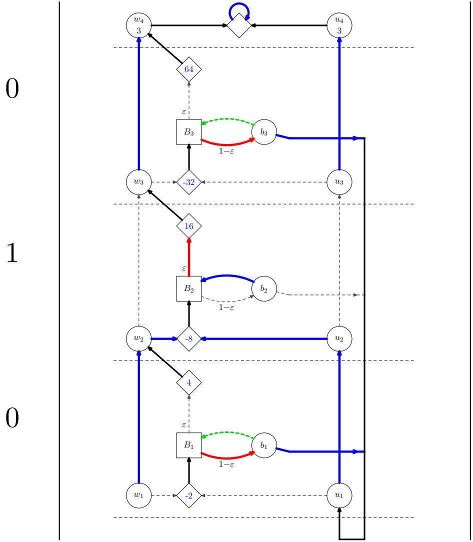
$z = 62 + 4\epsilon$

$v_{b_1} = v_{b_3} = v_{u_1} = v_{u_2} = 8$

$v_{b_2} = 16, v_{w_2} = 8$

$v_{w_1} = -2 + 12\epsilon + 8(1-\epsilon) = 6 + 4\epsilon$

Basic Construction of Counter



Step 4: Update uplink of first bit.

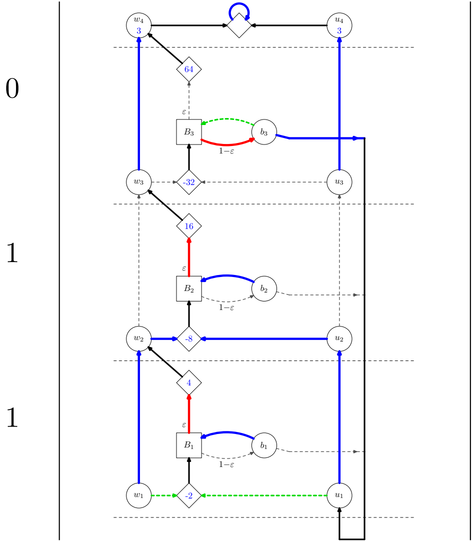
$z = 64$

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

$v_{b_1} = v_{b_3} = v_{u_1} = v_{u_2} = 8$

$v_{b_2} = 16, v_{w_1} = v_{w_2} = 8$

Basic Construction of Counter



Step 1: First bit has been set.

$$z = 68$$

$$v_{b_1} = 12, v_{b_3} = v_{u_1} = v_{u_2} = 8$$

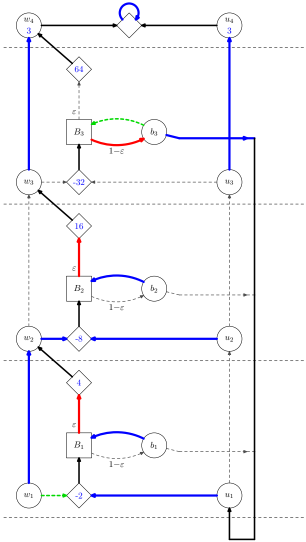
$$v_{b_2} = 16, v_{w_1} = v_{w_2} = 8$$

Basic Construction of Counter

0

1

1



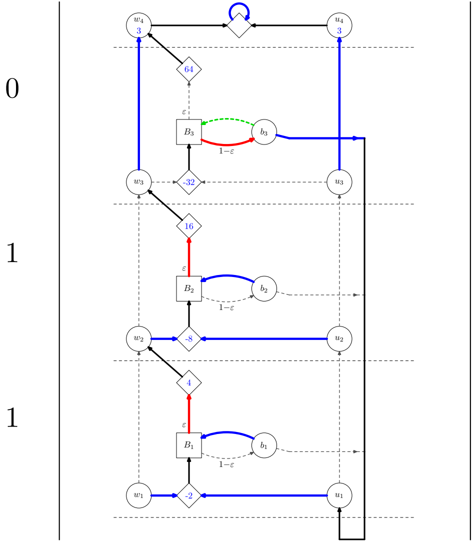
Step 2: Update selector of first bit.

$z = 72$

$v_{b_1} = 12, v_{b_3} = v_{u_1} = 10$

$v_{b_2} = 16, v_{u_2} = v_{w_1} = v_{w_2} = 8$

Basic Construction of Counter



Step 4: Update uplink of first bit. (Step 3 missing)

$z = 74$

$v_{b_1} = 12, v_{w_1} = v_{b_3} = v_{u_1} = 10$

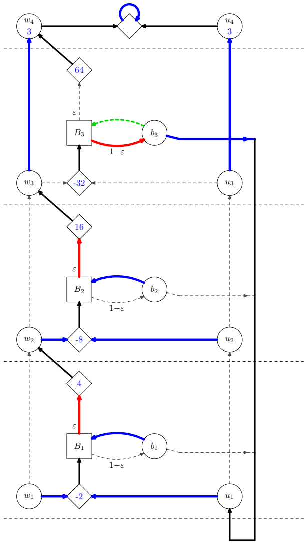
$v_{b_2} = 16, v_{u_2} = v_{w_2} = 8$

Basic Construction of Counter

0

1

1



Intermediate Steps:

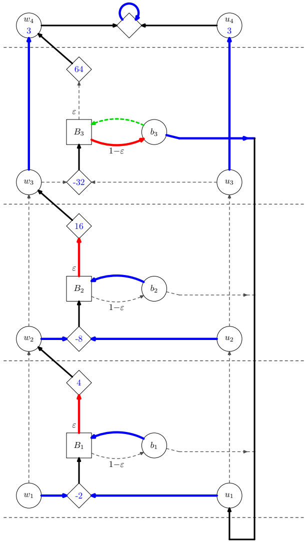
1. Set least unset bit by moving into cycle.
2. Update selector lane.
3. Reset lower set bits by using selector lane.
4. Update uplink lane.

Basic Construction of Counter

0

1

1



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

Least Recently Considered

We now give a lower bound on:

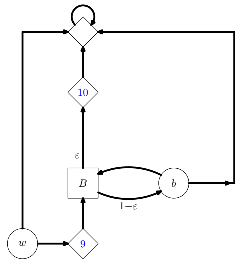
Least Recently Considered

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



Least Recently Considered

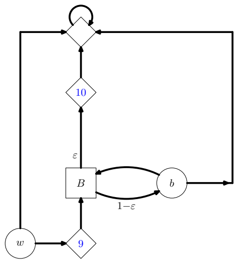
We now give a lower bound on:

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.



Least Recently Considered

We now give a lower bound on:

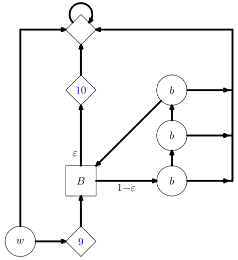
Least Recently Considered

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



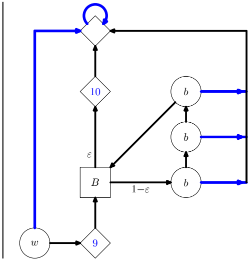
Least Recently Considered

We now give a lower bound on:

Least Recently Considered

Select improving edge in a round robin fashion.

Start with unset bit again.



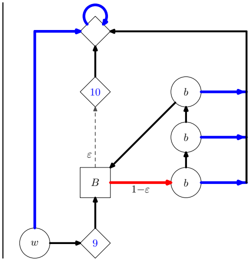
Least Recently Considered

We now give a lower bound on:

Least Recently Considered

Select improving edge in a round robin fashion.

Start with unset bit again.



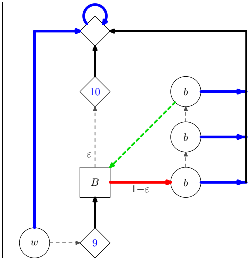
Least Recently Considered

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.



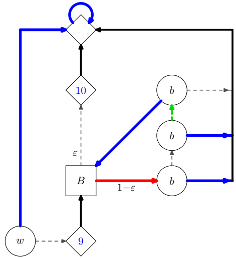
Least Recently Considered

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.



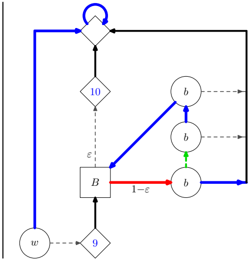
Least Recently Considered

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.



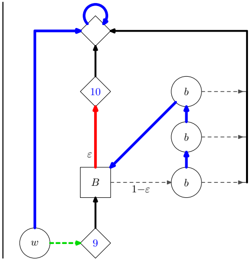
Least Recently Considered

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.



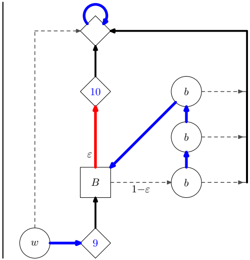
Least Recently Considered

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.

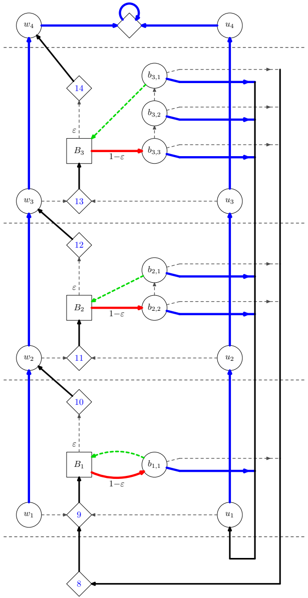


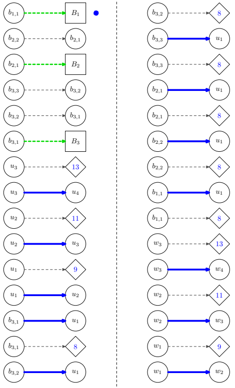
Round Robin Lower Bound Construction

0

0

0





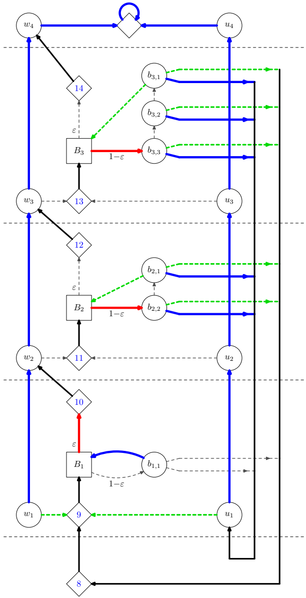
Set first bit.

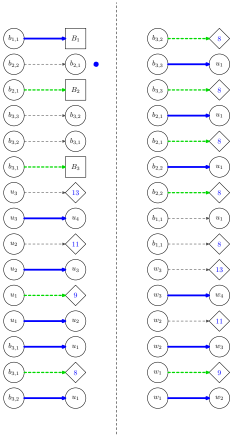
Round Robin Lower Bound Construction

0

0

1





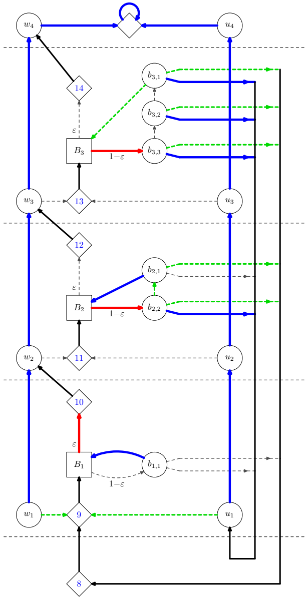
Set single edge of second bit.

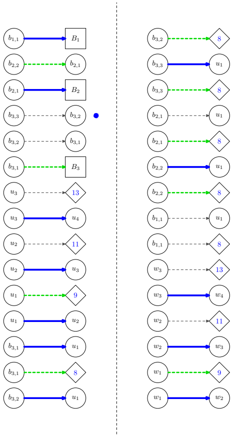
Round Robin Lower Bound Construction

0

0

1





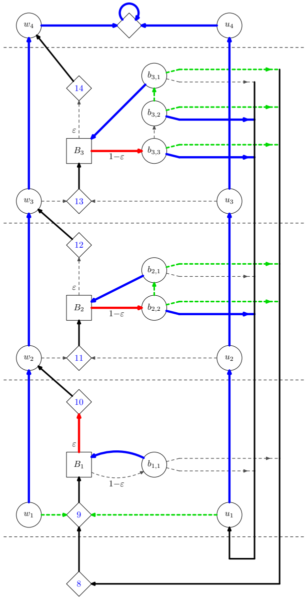
Other edge of second bit now improving, but smaller. Set single edge of third bit.

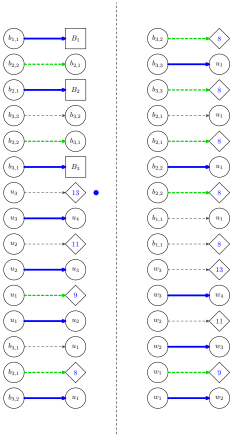
Round Robin Lower Bound Construction

0

0

1





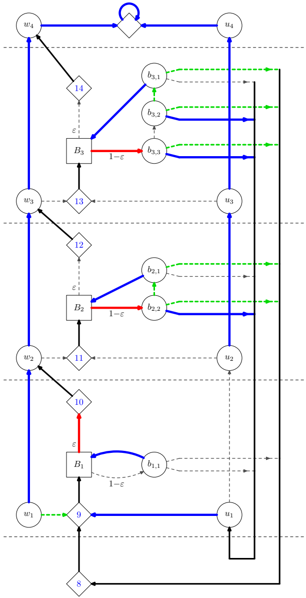
Second edge of third bit now improving, but smaller. Update selector of first bit.

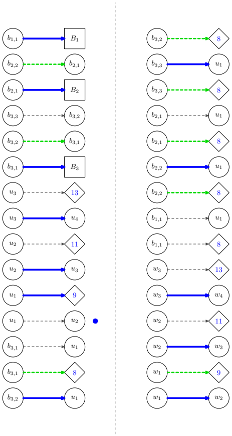
Round Robin Lower Bound Construction

0

0

1





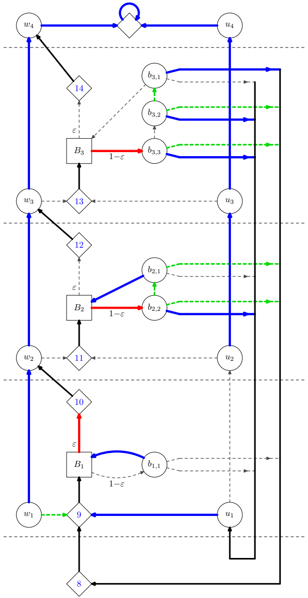
Reset single edge of third bit.

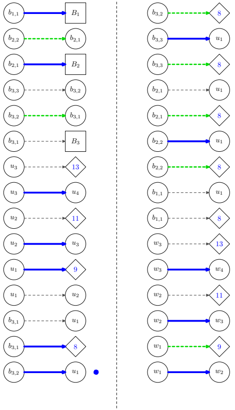
Round Robin Lower Bound Construction

0

0

1





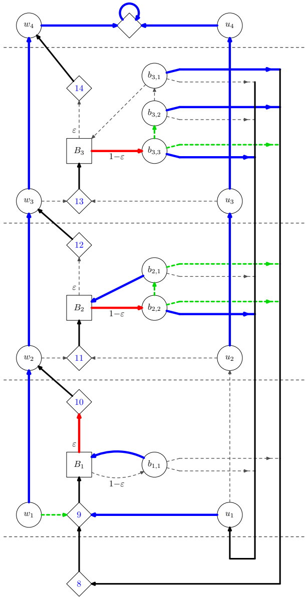
Reset second edge of third bit.

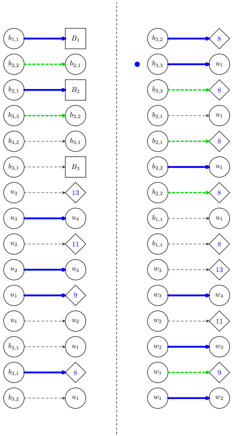
Round Robin Lower Bound Construction

0

0

1





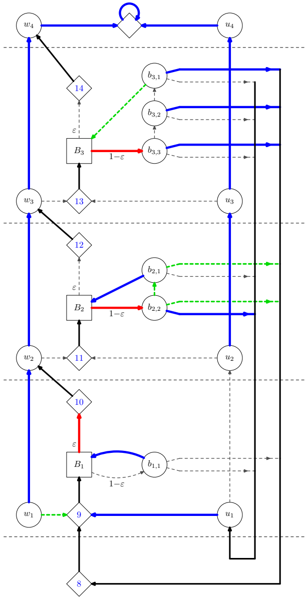
Reset last edge of third bit.

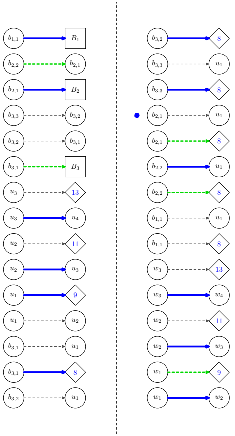
Round Robin Lower Bound Construction

0

0

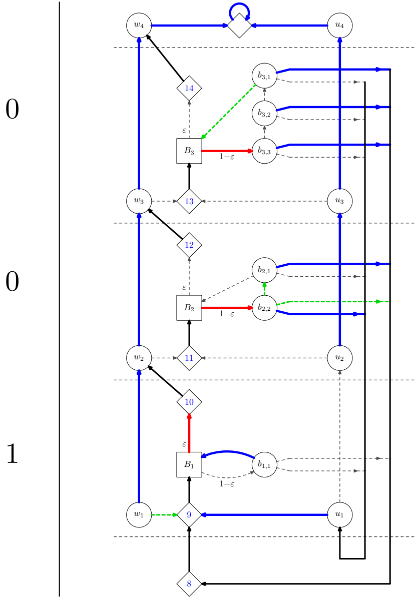
1

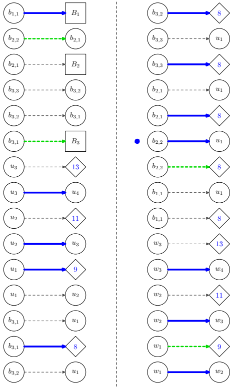




Reset single edge of second bit.

Round Robin Lower Bound Construction





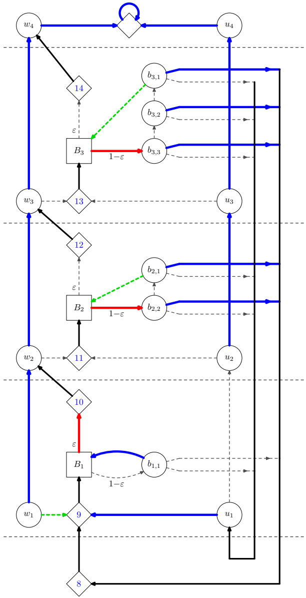
Reset last edge of second bit.

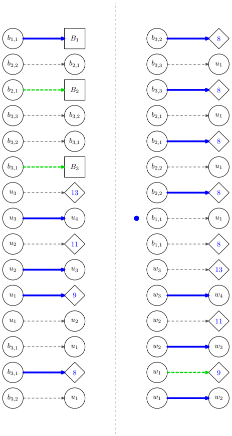
Round Robin Lower Bound Construction

0

0

1





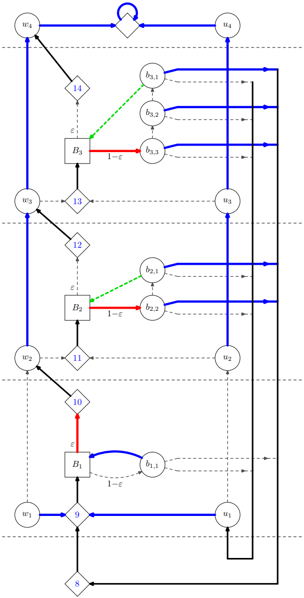
Update uplink of first bit.

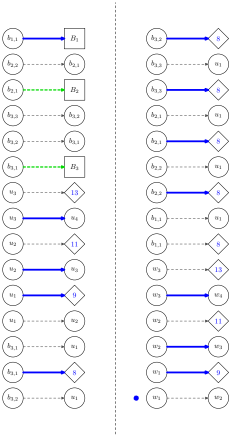
Round Robin Lower Bound Construction

0

0

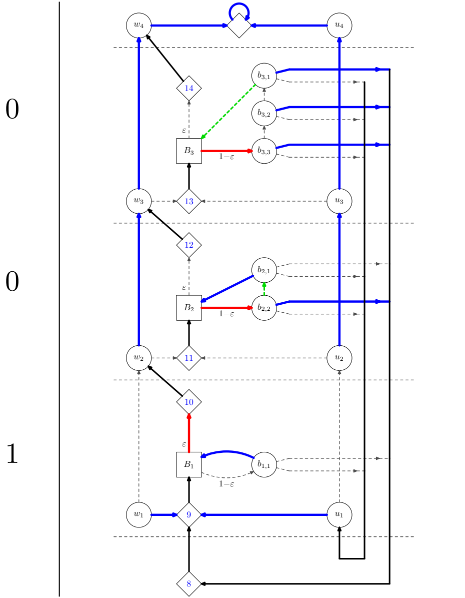
1

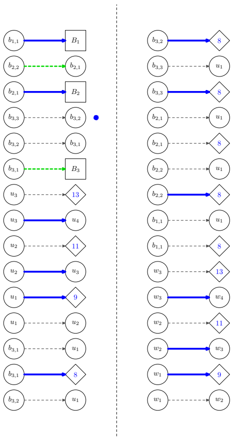




Set single edge of second bit.

Round Robin Lower Bound Construction





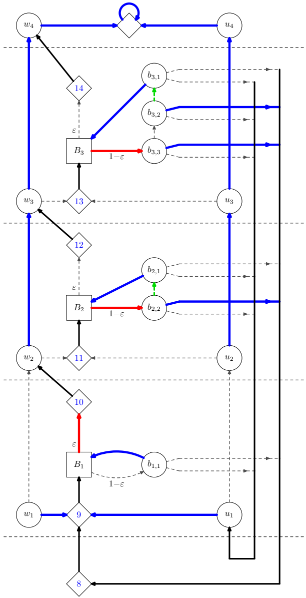
Other edge of second bit now improving, but smaller. Set single edge of third bit.

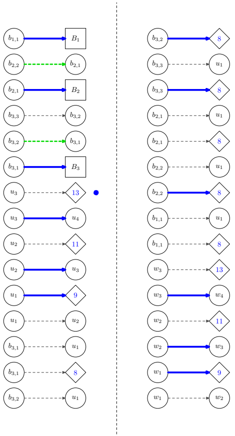
Round Robin Lower Bound Construction

0

0

1





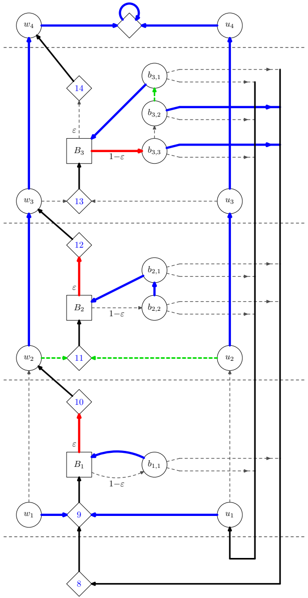
Set other edge of second bit, i.e. set second bit.

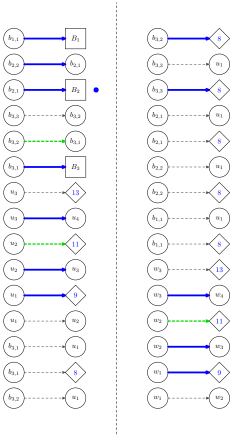
Round Robin Lower Bound Construction

0

1

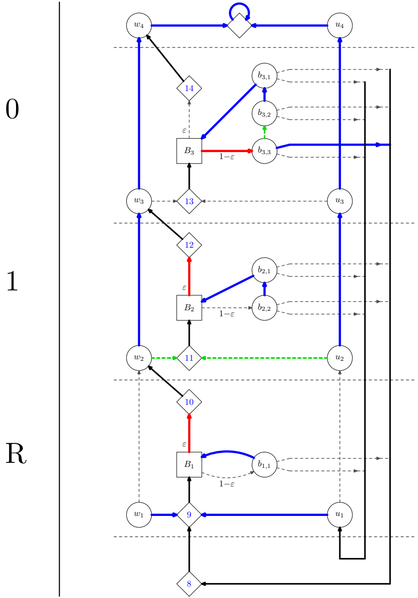
R

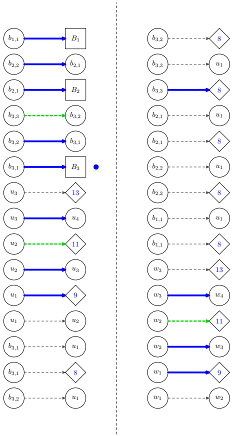




Set second edge of third bit.

Round Robin Lower Bound Construction





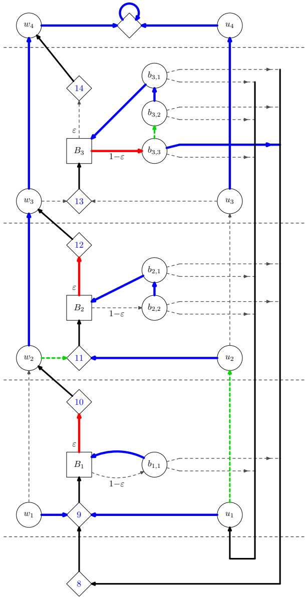
Last edge of third bit now improving, but smaller.
Update selector of second bit.

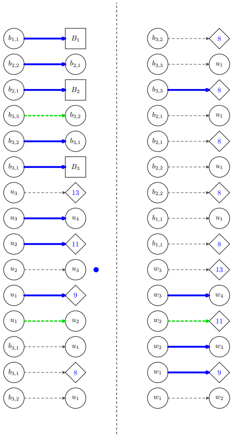
Round Robin Lower Bound Construction

0

1

R





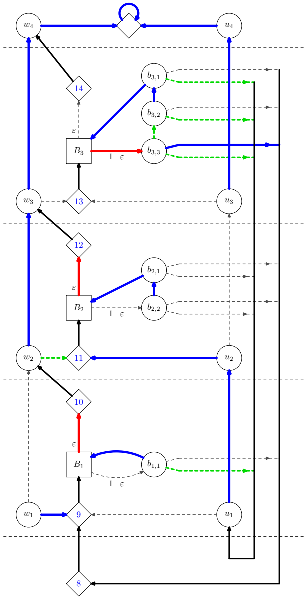
Update selector of first bit.

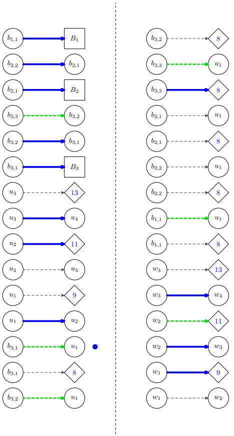
Round Robin Lower Bound Construction

0

1

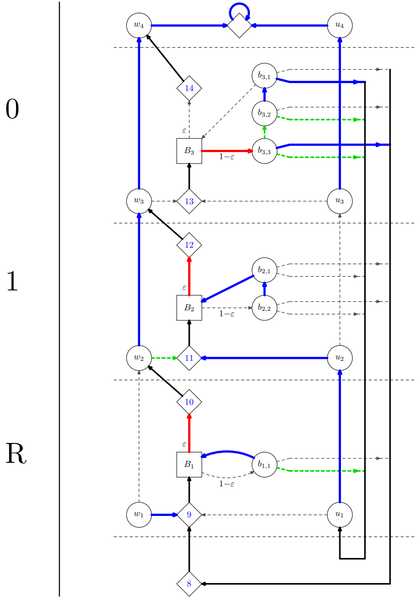
R

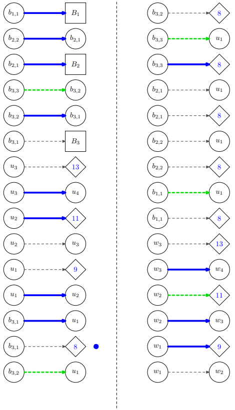




Reset single edge of third bit.

Round Robin Lower Bound Construction





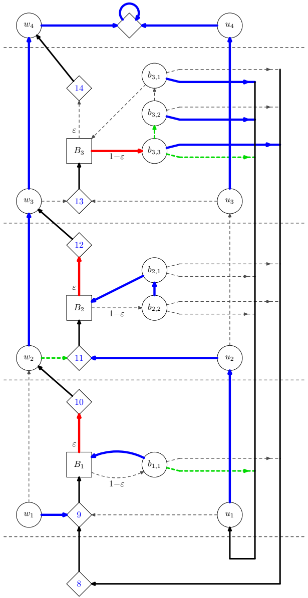
Reset second edge of third bit.

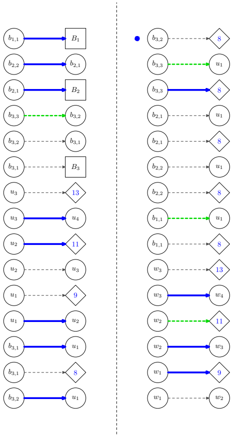
Round Robin Lower Bound Construction

0

1

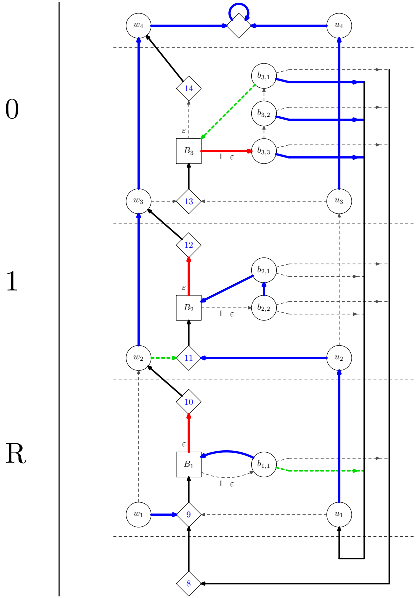
R

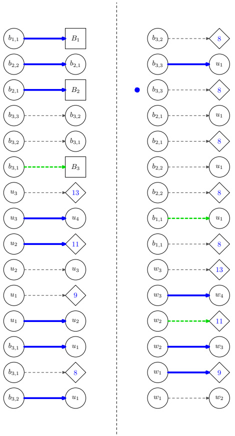




Reset last edge of third bit.

Round Robin Lower Bound Construction





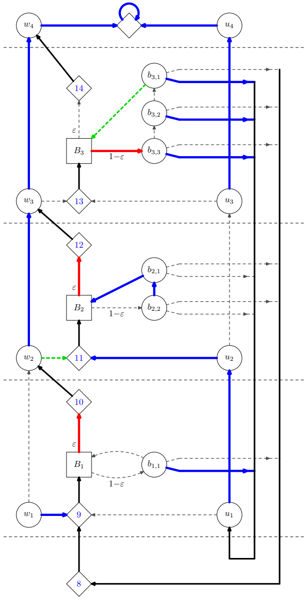
Reset first bit.

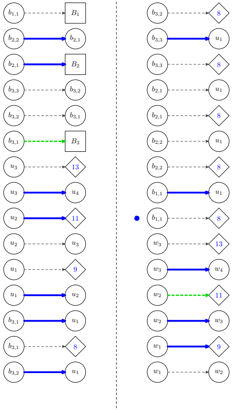
Round Robin Lower Bound Construction

0

1

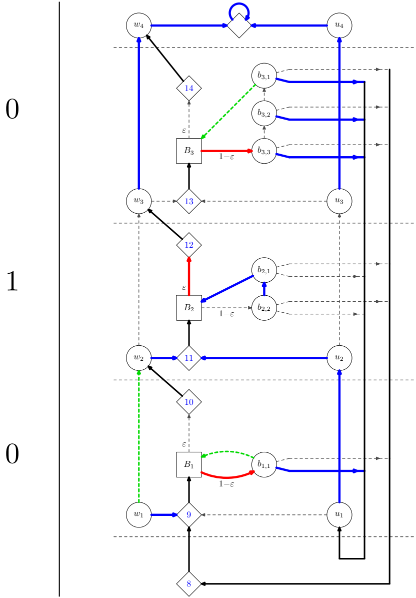
0

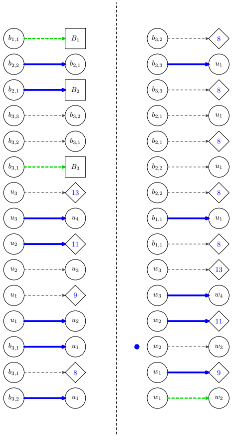




Update uplink of second bit.

Round Robin Lower Bound Construction





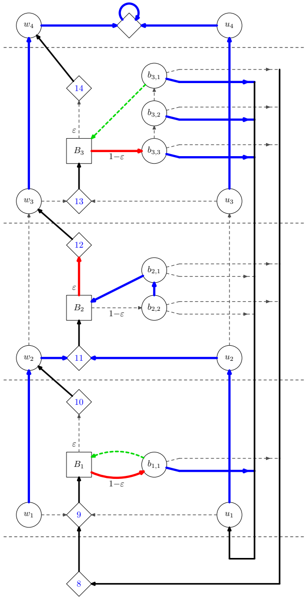
Update uplink of first bit.

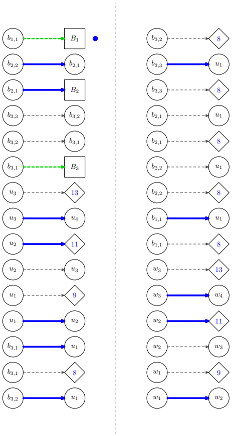
Round Robin Lower Bound Construction

0

1

0





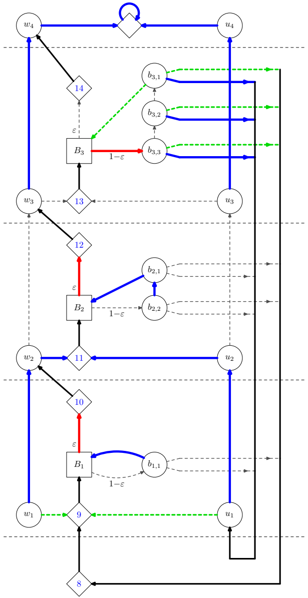
Set first bit.

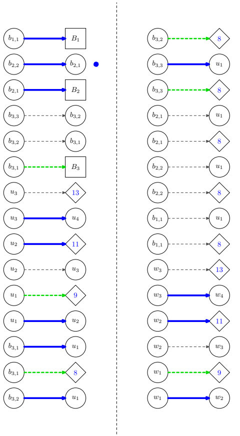
Round Robin Lower Bound Construction

0

1

1





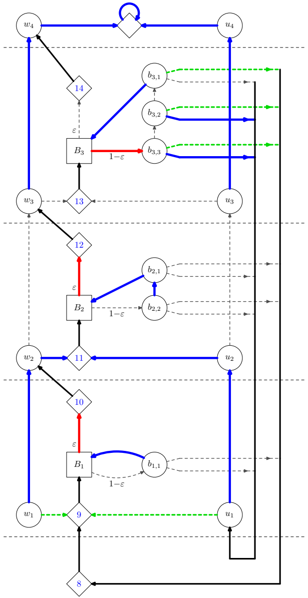
Set single edge of third bit.

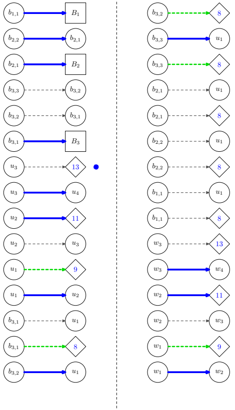
Round Robin Lower Bound Construction

0

1

1





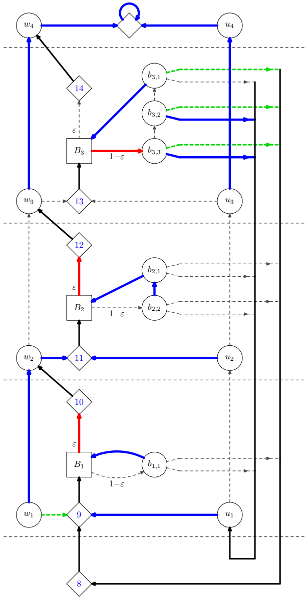
Update selector of first bit.

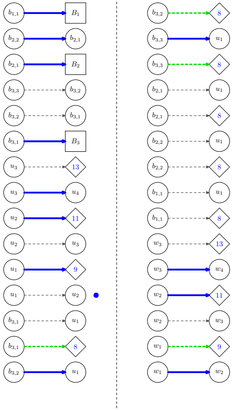
Round Robin Lower Bound Construction

0

1

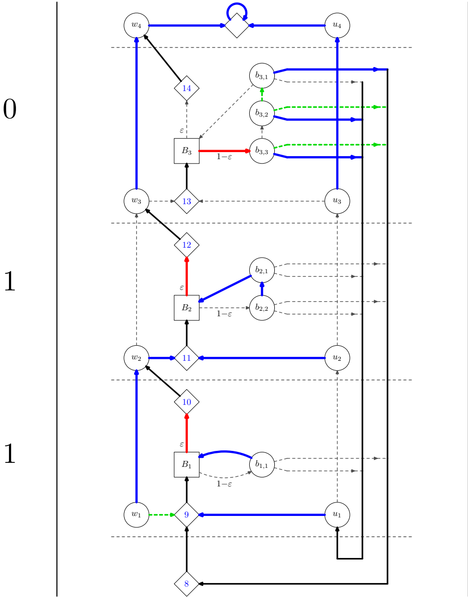
1

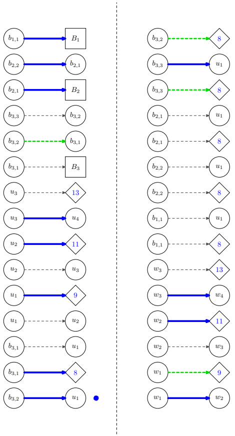




Reset single edge of third bit.

Round Robin Lower Bound Construction





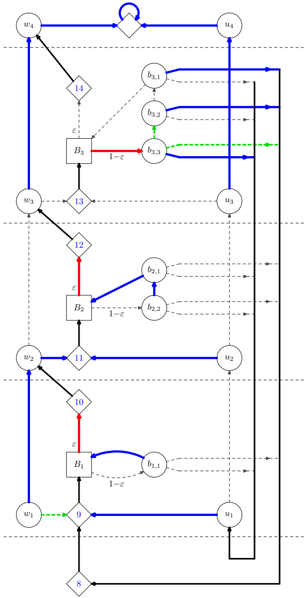
Reset second edge of third bit.

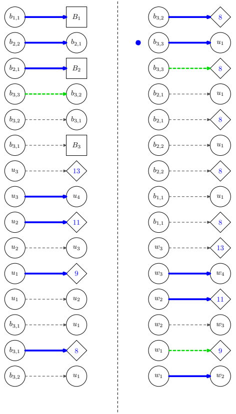
Round Robin Lower Bound Construction

0

1

1





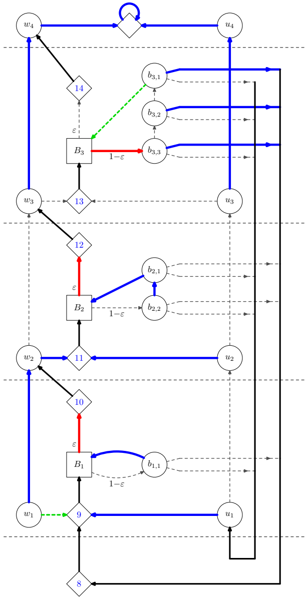
Reset last edge of third bit.

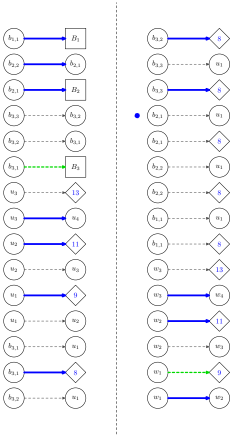
Round Robin Lower Bound Construction

0

1

1





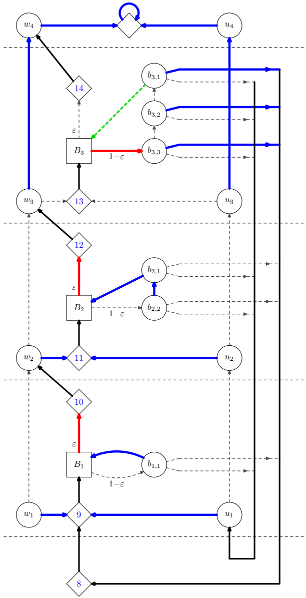
Update uplink of first bit.

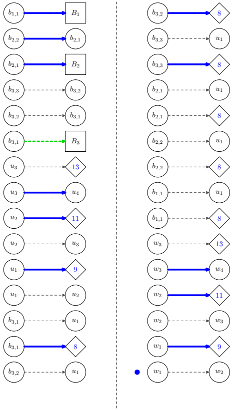
Round Robin Lower Bound Construction

0

1

1





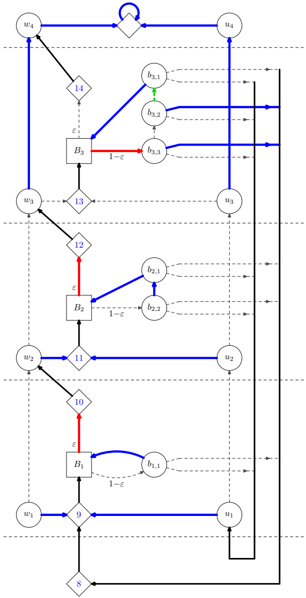
Set single edge of third bit.

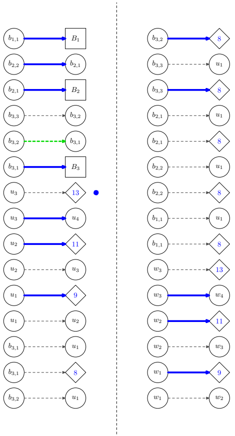
Round Robin Lower Bound Construction

0

1

1





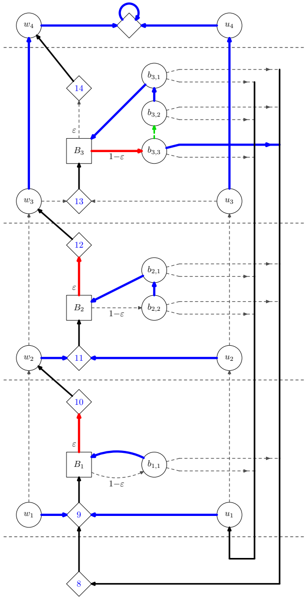
Set second edge of third bit.

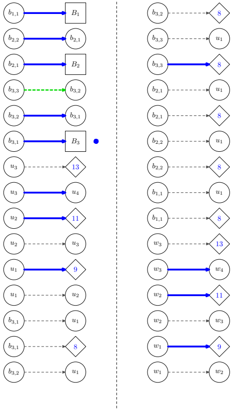
Round Robin Lower Bound Construction

0

1

1





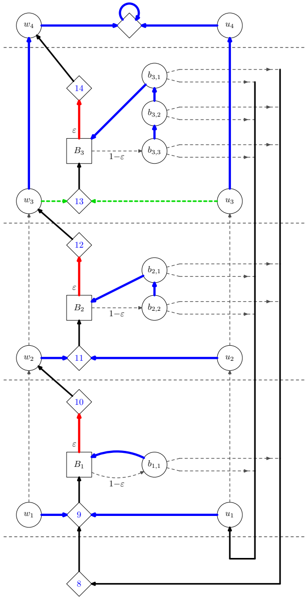
Set last edge of third bit, i.e. set third bit.

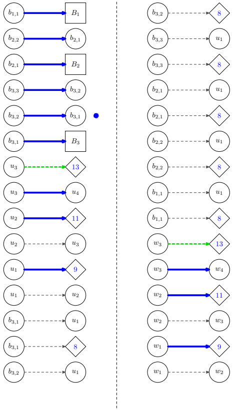
Round Robin Lower Bound Construction

1

R

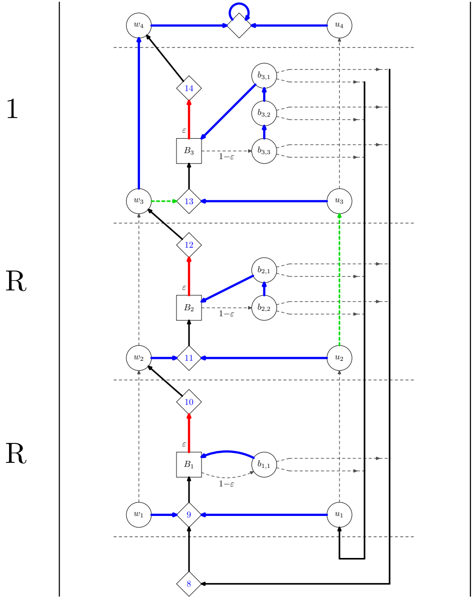
R

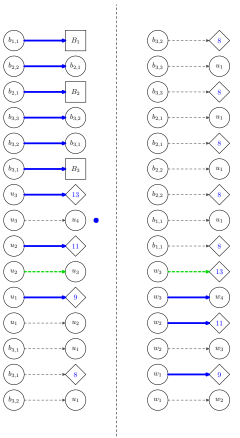




Update selector of third bit.

Round Robin Lower Bound Construction





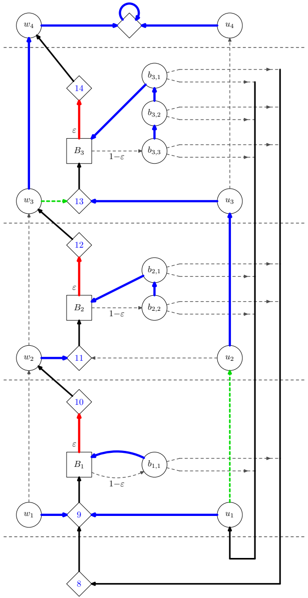
Update selector of second bit.

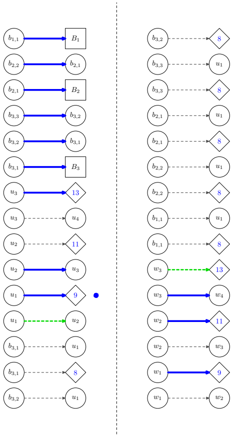
Round Robin Lower Bound Construction

1

R

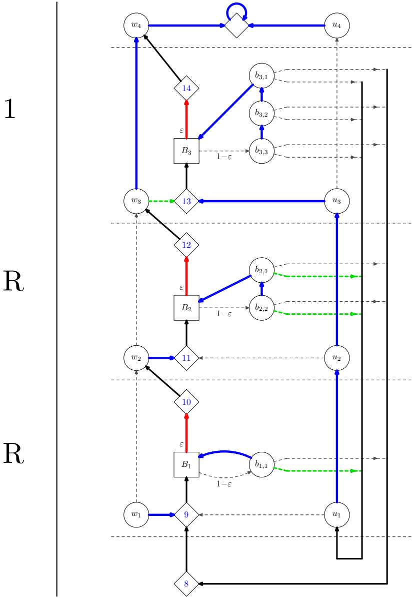
R

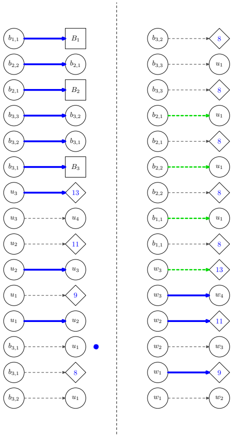




Update selector of first bit.

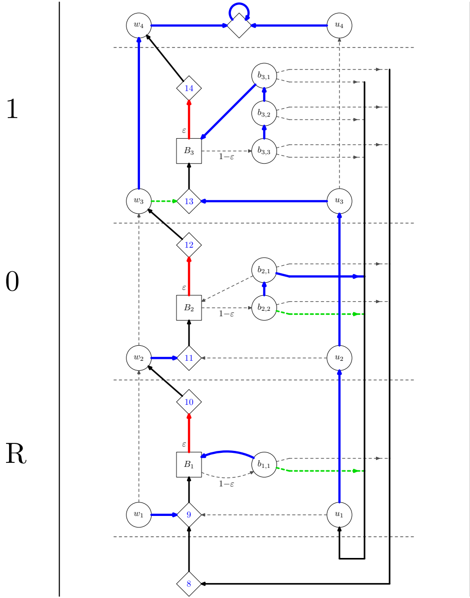
Round Robin Lower Bound Construction

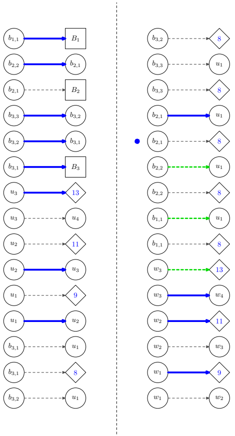




Reset first edge of second bit,
i.e. reset second bit.

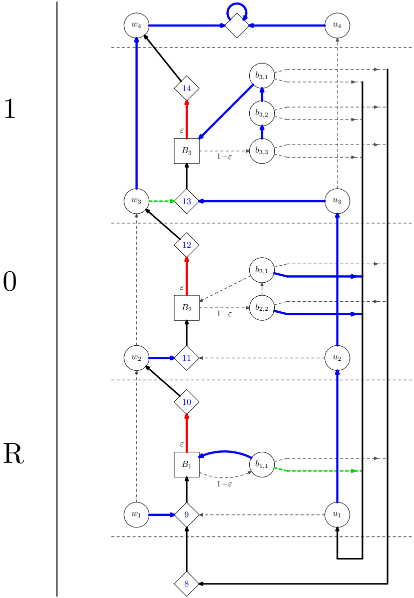
Round Robin Lower Bound Construction

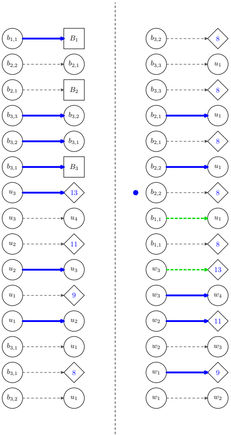




Reset other edge of second bit.

Round Robin Lower Bound Construction





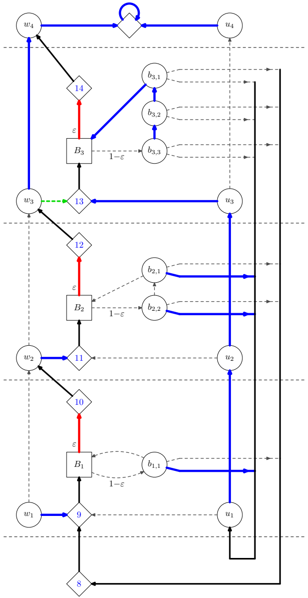
Reset first bit.

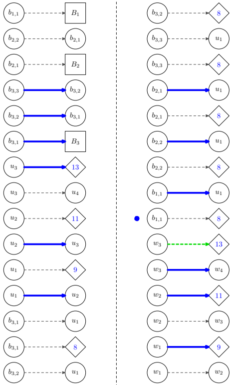
Round Robin Lower Bound Construction

1

0

0





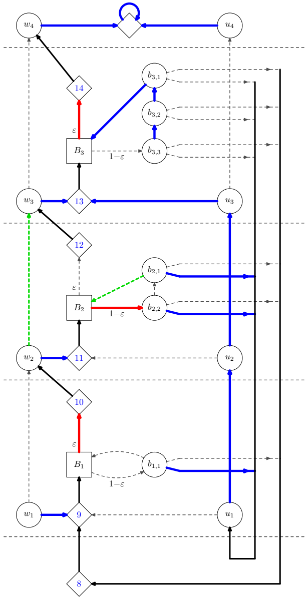
Update uplink of third bit.

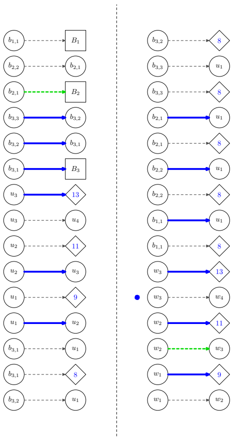
Round Robin Lower Bound Construction

1

0

0





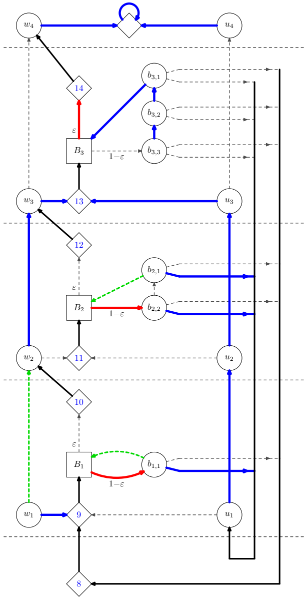
Update uplink of second bit.

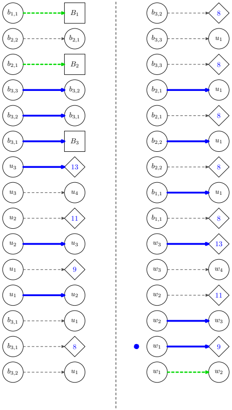
Round Robin Lower Bound Construction

1

0

0





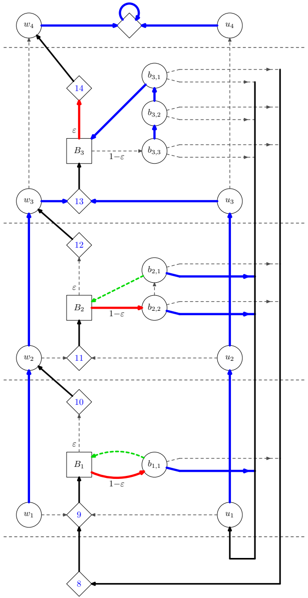
Update uplink of first bit.

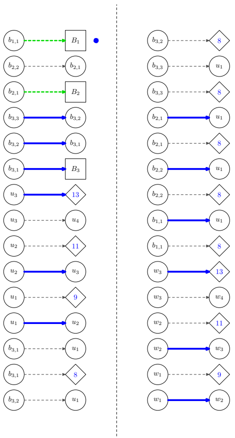
Round Robin Lower Bound Construction

1

0

0





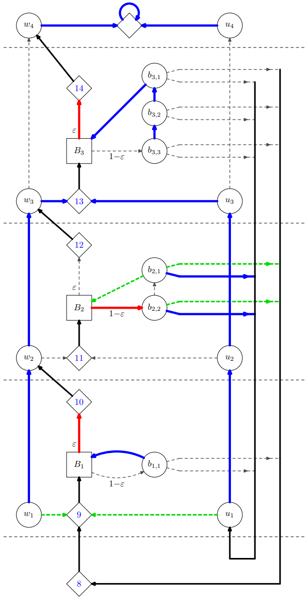
Set first bit.

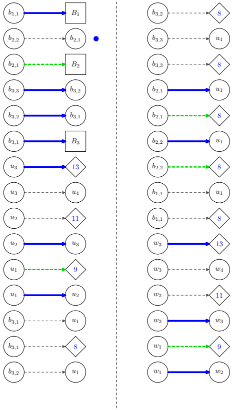
Round Robin Lower Bound Construction

1

0

1





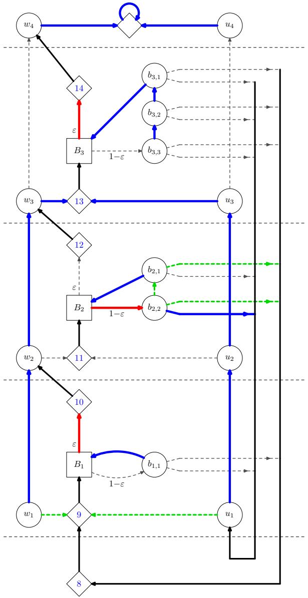
Set single edge of second bit.

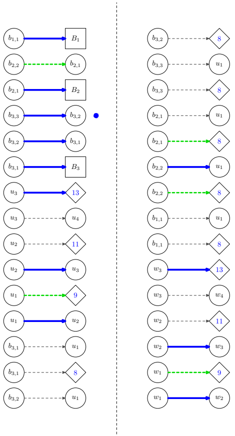
Round Robin Lower Bound Construction

1

0

1





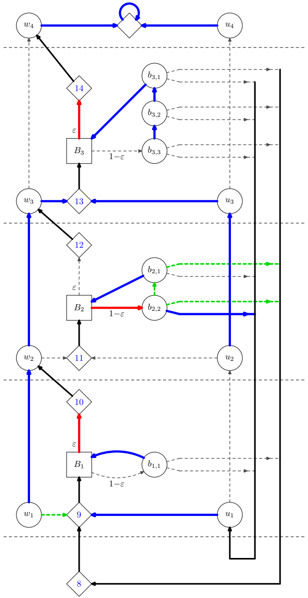
Other edge of second bit now improving, but smaller. Update selector of first bit.

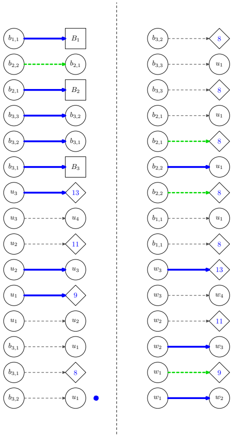
Round Robin Lower Bound Construction

1

0

1





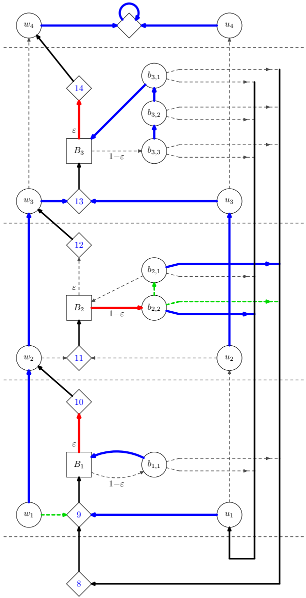
Reset single edge of second bit.

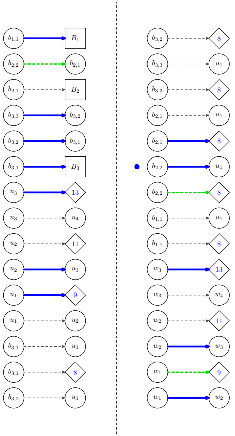
Round Robin Lower Bound Construction

1

0

1





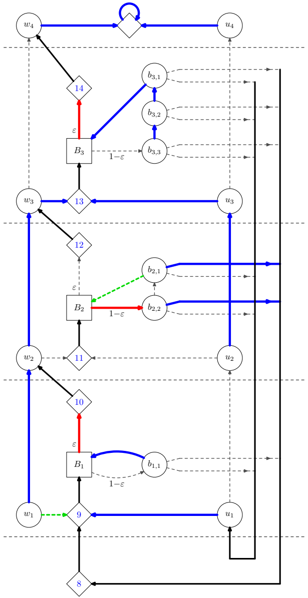
Reset last edge of second bit.

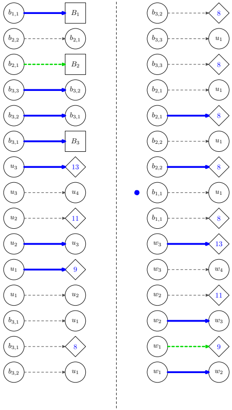
Round Robin Lower Bound Construction

1

0

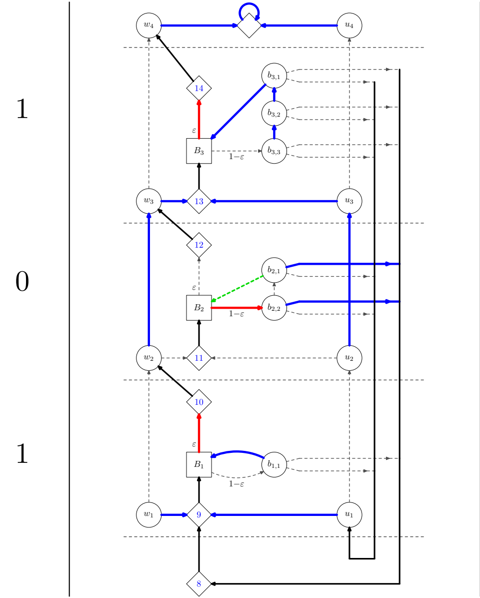
1

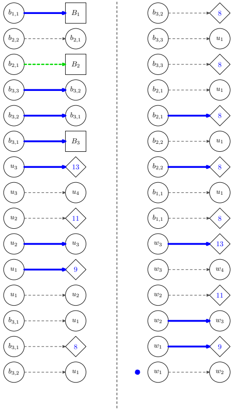




Update uplink of first bit.

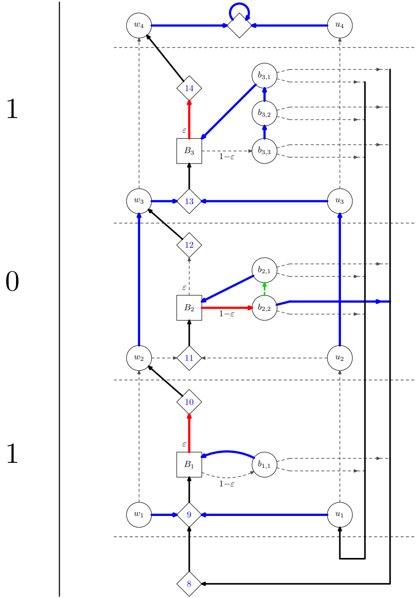
Round Robin Lower Bound Construction

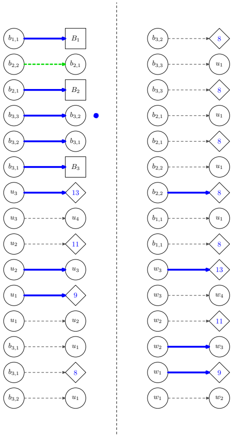




Set single edge of second bit.

Round Robin Lower Bound Construction





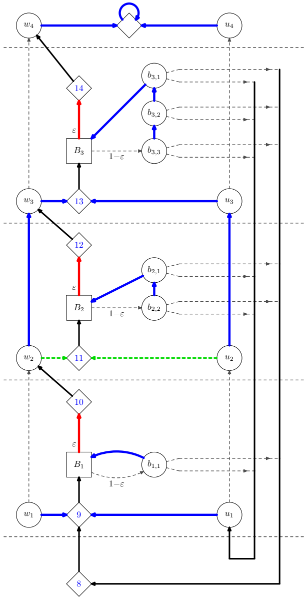
Set other edge of second bit, i.e. set second bit.

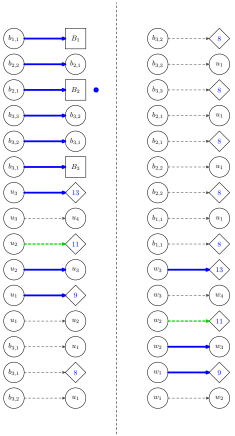
Round Robin Lower Bound Construction

1

1

R





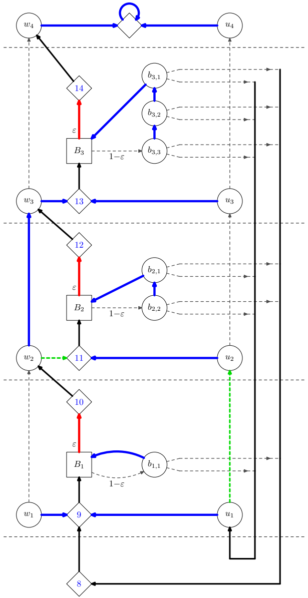
Update selector of second bit.

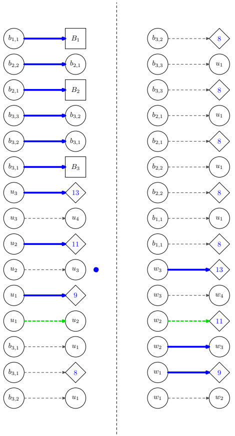
Round Robin Lower Bound Construction

1

1

R





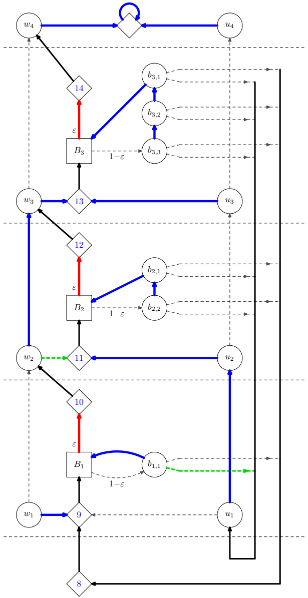
Update selector of first bit.

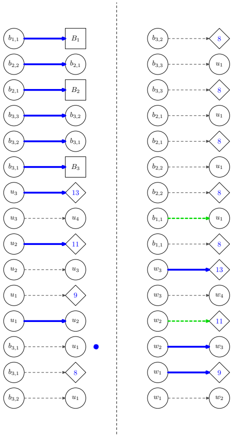
Round Robin Lower Bound Construction

1

1

R





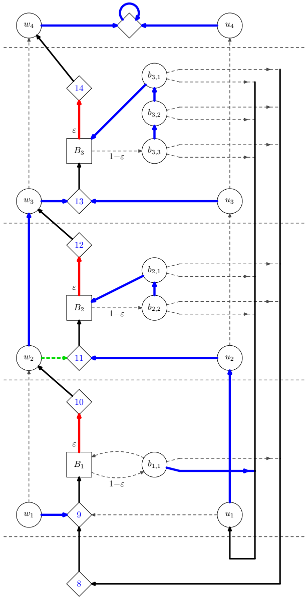
Reset first bit.

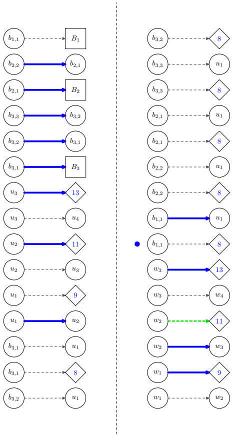
Round Robin Lower Bound Construction

1

1

0





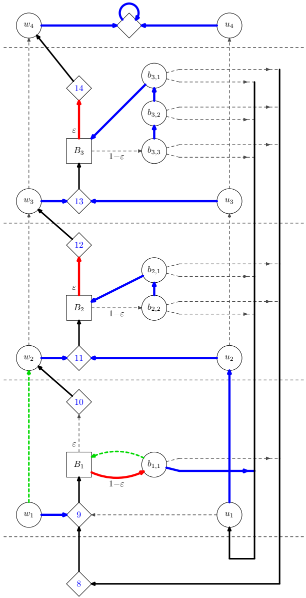
Update uplink of second bit.

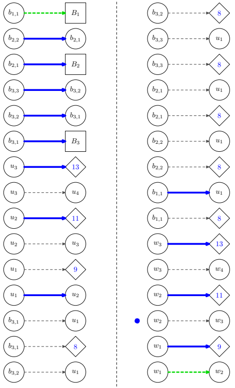
Round Robin Lower Bound Construction

1

1

0





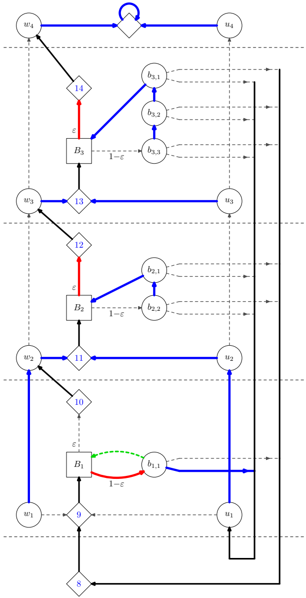
Update uplink of first bit.

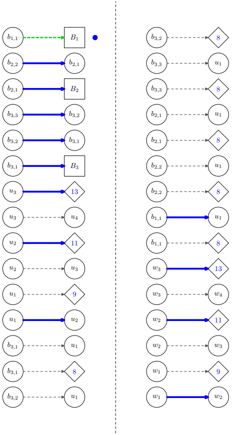
Round Robin Lower Bound Construction

1

1

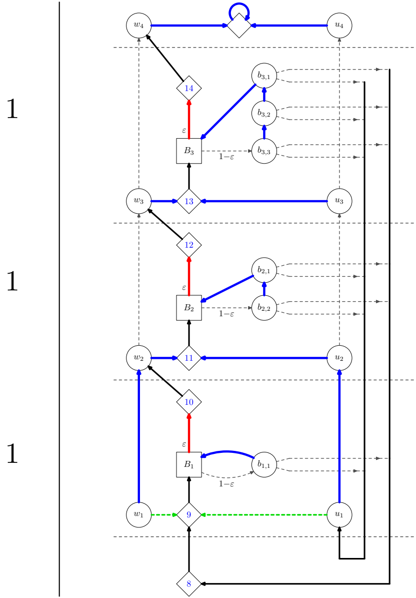
0

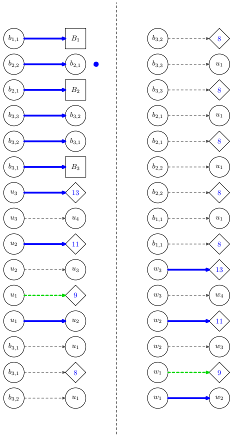




Set first bit.

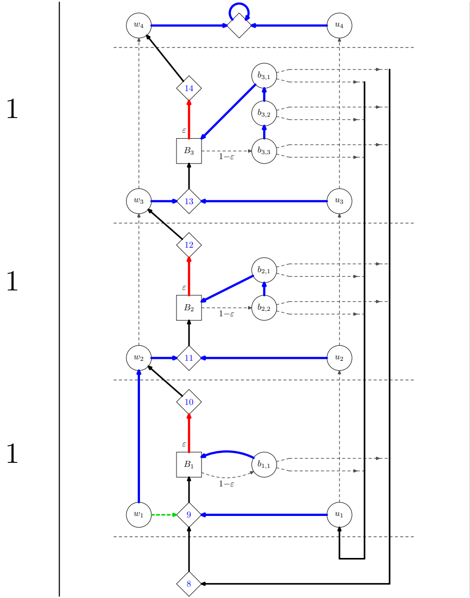
Round Robin Lower Bound Construction

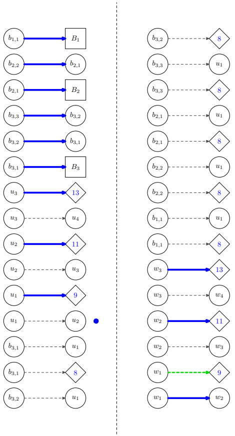




Update selector of first bit.

Round Robin Lower Bound Construction





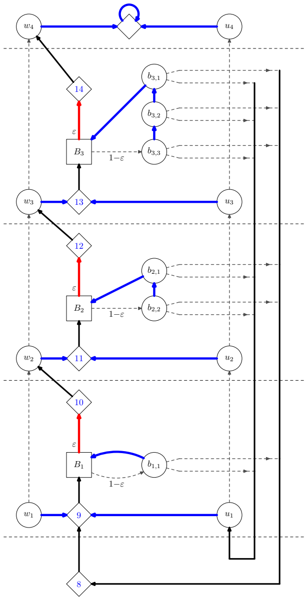
Update uplink of first bit.

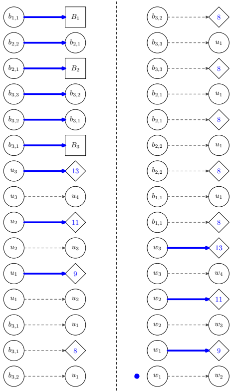
Round Robin Lower Bound Construction

1

1

1





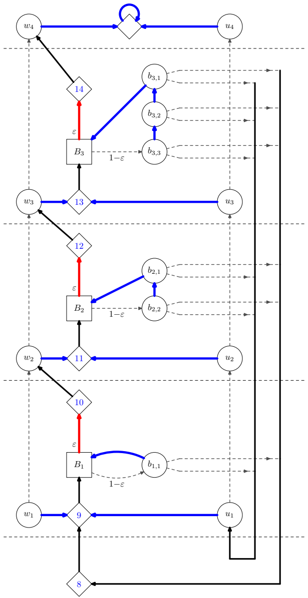
Finished.

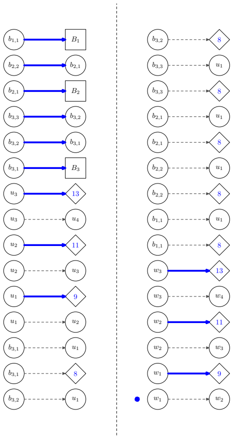
Round Robin Lower Bound Construction

1

1

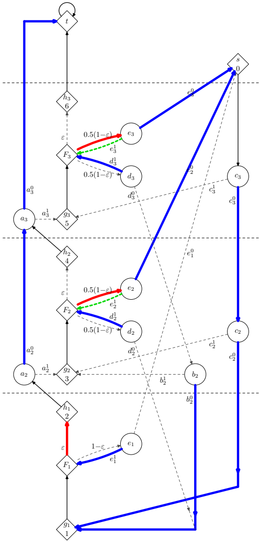
1





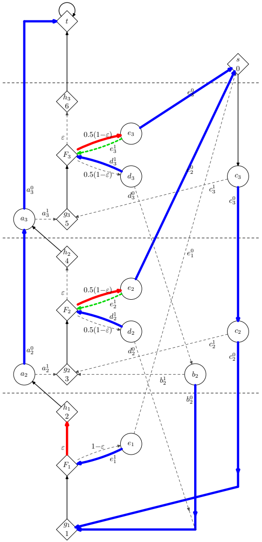
Subexponential lower bound
Some out degrees are 3

New construction



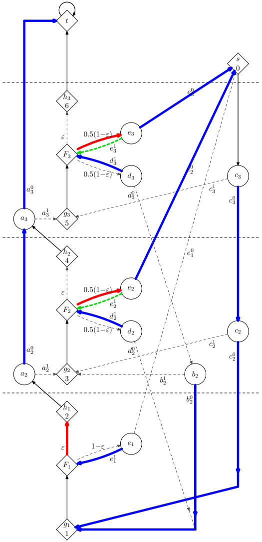
- Lower bound is exponential in MDP size

New construction



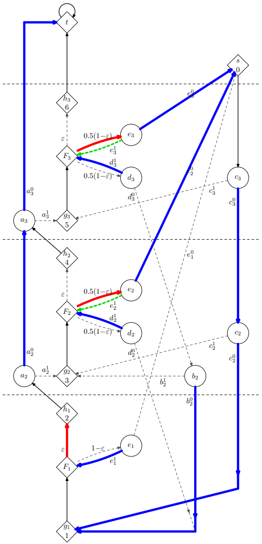
- Lower bound is exponential in MDP size
- Binary node-out-degree

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

Corresponding LP

(LHS variables show binary node-out-degree)

$$
\begin{array}{lll}
\max & \sum_{i=1}^{n} \left(\left(a_i^1 + b_i^1 + c_i^1\right) \cdot \left(\Omega(g_i) + \varepsilon \cdot \Omega(h_i)\right) + \left(d_i^1 + e_i^1\right) \cdot \varepsilon \cdot \Omega(h_i) + e_i^0 \cdot \Omega(s) \right) \\
\text{s.t.} & (a_i) & a_i^1 + a_i^0 = 1 + a_{i-1}^0 + \varepsilon \cdot (a_{i-1} + b_{i-1} + c_{i-1} + d_{i-1} + e_{i-1}) \\
 & (b_i) & b_i^1 + b_i^0 = 1 + b_{i+1}^0 + d_{i+1}^0 \\
 & (c_i) & c_i^1 + c_i^0 = 1 + \begin{cases} c_{i+1}^0 & \text{if } i < n \\ \sum_{j=1}^{n} e_j^0 & \text{if } i = n \end{cases} \\
 & (d_i) & d_i^1 + d_i^0 = 1 + (a_i^1 + b_i^1 + c_i^1 + d_i^1 + e_i^1) \cdot \begin{cases} \frac{1-\varepsilon}{2} & \text{if } i > 1 \\ 1 - \varepsilon & \text{if } i = 1 \end{cases} \\
 & (e_i) & e_i^1 + e_i^0 = 1 + (a_i^1 + b_i^1 + c_i^1 + d_i^1 + e_i^1) \cdot \begin{cases} \frac{1-\varepsilon}{2} & \text{if } i > 1 \\ 1 - \varepsilon & \text{if } i = 1 \end{cases}
\end{array}
$$

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!