Subexponential lower bounds for randomized pivoting rules for the simplex algorithm

Oliver Friedmann[1] Thomas Dueholm Hansen[2] Uri Zwick[3]

[1] Department of Computer Science,
University of Munich, Germany.

[2] Department of Management Science and Engineering,
Stanford University, USA.

[3] School of Computer Science,
Tel Aviv University, Israel.

The Fields Institute, November 29, 2013

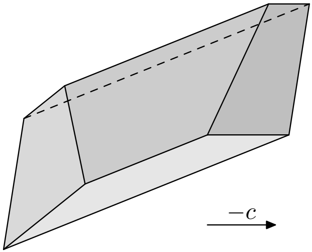
Outline

- Linear programming and the simplex algorithm.
- Related work and results.
- The simplex algorithm for shortest paths.
- Framework: Lower bounds for the simplex algorithm utilizing shortest paths.
- On the lower bound for RandomEdge.
- (On the lower bound for RandomFacet.)
- Open problems.

Linear programming

- Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$.
- A **linear program** (LP) in **standard form** is an optimization problem of the form:

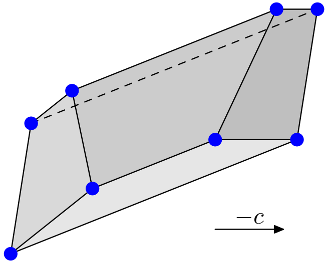
$$\begin{array}{llcl} \min & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$



- The set of **feasible solutions** is a **convex polytope**.

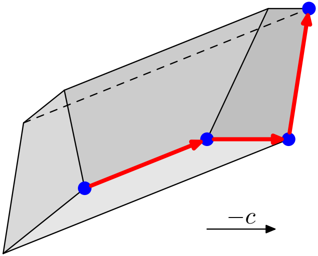
Basic feasible solutions

$$\begin{array}{lrcl} \min & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$



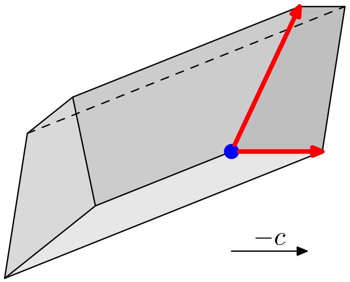
- A **basis** is a subset $B \subseteq \{1, \ldots, n\}$ of m columns of A such that the corresponding matrix $A_B \in \mathbb{R}^{m \times m}$ is invertible.
- Every **basis** defines a **basic feasible solution** $x^B = A_B^{-1} b$ by setting **non-basic** variables, x_i for $i \notin B$, to zero.
- **Vertices** (or corners) are **basic feasible solutions**.

$$\begin{array}{llcl} \min & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$



- **Pivoting**: Exchange a basic and a non-basic variable in a **basis** to move from one **basic feasible solution** to another.
- A **basic feasible solution** is **optimal** if there are no **improving pivots** w.r.t. its **basis**.
- The **simplex algorithm**: Repeatedly perform **improving pivots**.

Pivoting rules



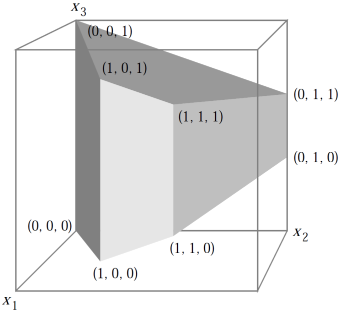
- Several **improving pivots** may be available for a given **basis**. The edge is chosen by a **pivoting rule**.
- I.e., a pivoting rule decides which basic and non-basic variables to exchange.

Deterministic pivoting rules

- LargestCoefficient, Dantzig (1947)
  - The non-basic variable with **most negative reduced cost** enters the basis.
- Bland's rule, Bland (1977)
  - Pick the available variable with the **smallest index**, both for entering and leaving the basis.
  - This pivoting rule is guaranteed not to cycle.
- Others:
  - LargestIncrease
  - SteepestEdge
  - ShadowVertex
  - LeastEntered
  - ...

Exponential lower bounds

- Klee and Minty (1972): The LargestCoefficient pivoting rule may require exponentially many steps; the Klee-Minty cube.[1]
- Essentially all known natural deterministic pivoting rules are now known to be exponential:
  - LargestIncrease: Jeroslow (1973).
  - SteepestEdge: Goldfarb and Sit (1979).
  - Bland's rule: Avis and Chvátal (1978).
  - ShadowVertex: Murty (1980), Goldfarb (1983).
  - See Amenta and Ziegler (1996) for a unified view.



[1] Picture from Gärtner, Henk and Ziegler (1998)

Randomized pivoting rules

- RandomEdge
    - Perform **uniformly random** **improving pivots**.

Randomized pivoting rules

- RandomEdge
  - Perform **uniformly random** **improving pivots**.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a **uniformly random facet** that contains the current vertex, and **recursively find an optimal solution** within that facet. If possible, make an **improving pivot** leaving the facet and repeat.

Randomized pivoting rules

- RandomEdge
  - Perform **uniformly random** **improving pivots**.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a **uniformly random facet** that contains the current vertex, and **recursively find an optimal solution** within that facet. If possible, make an **improving pivot** leaving the facet and repeat.
  - Expected **subexponential** time: $2^{O(\sqrt{m \log n})}$ expected steps.

Randomized pivoting rules

- RandomEdge
  - Perform **uniformly random** **improving pivots**.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a **uniformly random facet** that contains the current vertex, and **recursively find an optimal solution** within that facet. If possible, make an **improving pivot** leaving the facet and repeat.
  - Expected **subexponential** time: $2^{O(\sqrt{m \log n})}$ expected steps.
- Randomized Bland's rule
  - Randomly **permute** the variables and use Bland's rule.

Randomized pivoting rules

- RandomEdge
  - Perform **uniformly random** **improving pivots**.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a **uniformly random facet** that contains the current vertex, and **recursively find an optimal solution** within that facet. If possible, make an **improving pivot** leaving the facet and repeat.
  - Expected **subexponential** time: $2^{O(\sqrt{m \log n})}$ expected steps.
- Randomized Bland's rule
  - Randomly **permute** the variables and use Bland's rule.
- No subexponential upper bounds are known for RandomEdge and Randomized Bland's rule.

Randomized pivoting rules

- RandomEdge
  - Perform **uniformly random** **improving pivots**.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a **uniformly random facet** that contains the current vertex, and **recursively find an optimal solution** within that facet. If possible, make an **improving pivot** leaving the facet and repeat.
  - Expected **subexponential** time: $2^{O(\sqrt{m \log n})}$ expected steps.
- Randomized Bland's rule
  - Randomly **permute** the variables and use Bland's rule.
- No subexponential upper bounds are known for RandomEdge and Randomized Bland's rule.
- Prior to our work no superpolynomial lower bounds were known for randomized pivoting rules.

Results

We prove lower bounds for the expected number of pivoting steps:

RandomEdge:	$2^{\Omega(m^{1/4})}$
RandomFacet:	$2^{\tilde{\Omega}(m^{1/3})}$
Randomized Bland's rule:	$2^{\tilde{\Omega}(m^{1/2})}$

where m is the number of equality constraints, and the number of variables is $n = \tilde{O}(m)$.

Results

We prove lower bounds for the expected number of pivoting steps:

RandomEdge:	$2^{\Omega(m^{1/4})}$
RandomFacet:	$2^{\tilde{\Omega}(m^{1/3})}$
Randomized Bland's rule:	$2^{\tilde{\Omega}(m^{1/2})}$

where m is the number of equality constraints, and the number of variables is $n = \tilde{O}(m)$.

- Note: In our SODA 2011 paper we studied a modified RandomFacet pivoting rule and **incorrectly** claimed that the expected running time was the same. We have repaired the analysis, but with a worse bound.

Results

We prove lower bounds for the expected number of pivoting steps:

RandomEdge:	$2^{\Omega(m^{1/4})}$
RandomFacet:	$2^{\tilde{\Omega}(m^{1/3})}$
Randomized Bland's rule:	$2^{\tilde{\Omega}(m^{1/2})}$

where m is the number of equality constraints, and the number of variables is $n = \tilde{O}(m)$.

- Note: In our SODA 2011 paper we studied a modified RandomFacet pivoting rule and **incorrectly** claimed that the expected running time was the same. We have repaired the analysis, but with a worse bound.
- Initially, we used **Markov decision processes** for the constructions. We now use **shortest paths** for RandomFacet and Randomized Bland's rule.

Techniques

- Previous lower bounds were proved by studying linear programs directly.
- The new lower bounds are based on linear programs for shortest paths and Markov decision processes (MDPs), for which the behavior of the simplex algorithm can be more easily understood.
  - MDPs can be viewed as **stochastic shortest paths**: edges can result in stochastic transitions.
- We prove lower bounds for corresponding PolicyIteration algorithms for MDPs, which immediately translate to lower bounds for the simplex algorithm.

Related work

- Friedmann (2009) and Fearnley (2010) gave a similar lower bound construction for Howard's PolicyIteration algorithm for solving MDPs (and *parity games*).

Related work

- Friedmann (2009) and Fearnley (2010) gave a similar lower bound construction for Howard's PolicyIteration algorithm for solving MDPs (and *parity games*).
- Friedmann (2011) used the same technique to prove a lower bound of subexponential form, $2^{\Omega(\sqrt{m})}$, for Zadeh's LeastEntered pivoting rule (1980).

Related work

- Friedmann (2009) and Fearnley (2010) gave a similar lower bound construction for Howard's PolicyIteration algorithm for solving MDPs (and *parity games*).
- Friedmann (2011) used the same technique to prove a lower bound of subexponential form, $2^{\Omega(\sqrt{m})}$, for Zadeh's LeastEntered pivoting rule (1980).

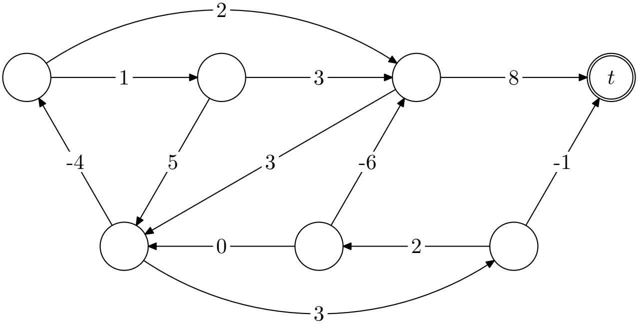
Superpolynomial lower bounds for RandomEdge and RandomFacet were previously only known in an abstract setting (Acyclic Unique Sink Orientations):

- Matoušek (1994): $2^{\Omega(\sqrt{m})}$ lower bound for RandomFacet.
- Matoušek and Szabó (2006): $2^{\Omega(m^{1/3})}$ lower bound for RandomEdge.

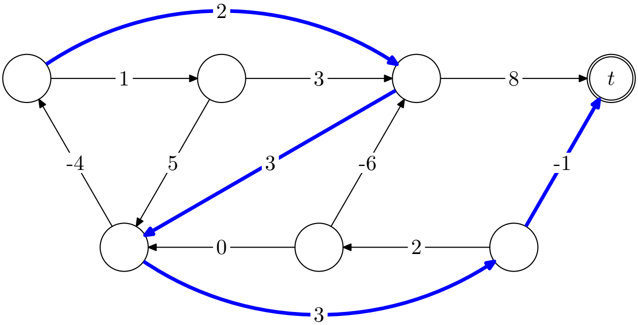
Outline

- Linear programming and the simplex algorithm.
- Related work and results.
- ⇒ The simplex algorithm for shortest paths.
- Framework: Lower bounds for the simplex algorithm utilizing shortest paths (and Markov decision processes).
- On the lower bound for RandomEdge.
- (On the lower bound for RandomFacet.)
- Summary of open problems.

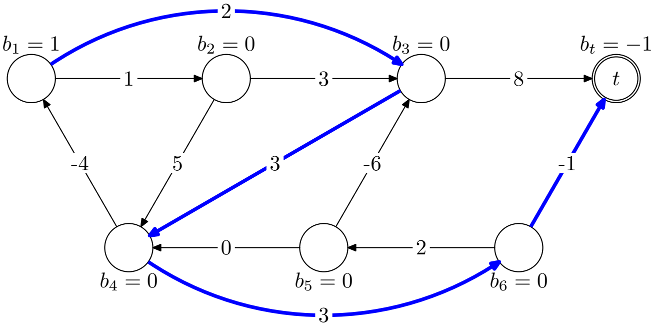
Single target shortest paths



Single target shortest paths

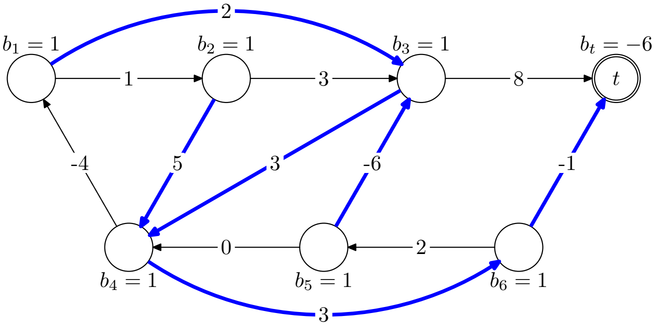


Single target shortest paths



$$
\begin{aligned}
\text{minimize} \quad & \sum_{(u,v)\in E} c_{(u,v)} x_{(u,v)} \\
s.t. \quad \forall v \in V: \quad & \sum_{w:(v,w)\in E} x_{(v,w)} - \sum_{u:(u,v)\in E} x_{(u,v)} = b_v \\
\forall (u,v) \in E: \quad & x_{(u,v)} \geq 0
\end{aligned}
$$

Single target shortest paths



$$\text{minimize} \quad \sum_{(u,v)\in E} c_{(u,v)} x_{(u,v)}$$

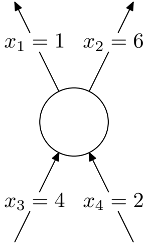
$$s.t.\ \ \forall v \in V: \quad \sum_{w:(v,w)\in E} x_{(v,w)} - \sum_{u:(u,v)\in E} x_{(u,v)} = b_v$$

$$\forall (u,v) \in E: \quad x_{(u,v)} \geq 0$$

Basic feasible solutions

$$
\begin{array}{lrcl}
\text{minimize} & \displaystyle\sum_{(u,v)\in E} c_{(u,v)} x_{(u,v)} & & \\
s.t.\ \ \forall v \neq t: & \displaystyle\sum_{w:(v,w)\in E} x_{(v,w)} - \sum_{u:(u,v)\in E} x_{(u,v)} & = & 1 \\
\forall (u,v) \in E: & x_{(u,v)} & \geq & 0
\end{array}
$$

Flow conservation:



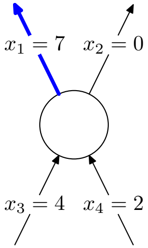
$x_1 + x_2 = 1 + x_3 + x_4$

- The flow through every vertex is at least 1.

Basic feasible solutions

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(u,v)\in E} c_{(u,v)} x_{(u,v)} & \\
s.t. \;\; \forall v \neq t : & \displaystyle\sum_{w:(v,w)\in E} x_{(v,w)} - \sum_{u:(u,v)\in E} x_{(u,v)} & = \;\; 1 \\
\quad \forall (u,v) \in E : & x_{(u,v)} & \geq \;\; 0
\end{array}
$$

Flow conservation:

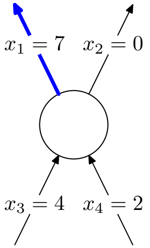


$$x_1 + x_2 = 1 + x_3 + x_4$$

- The flow through every vertex is at least 1.
- For a **basic feasible solution**, at most one edge leaving every vertex has non-zero flow.

$$
\begin{array}{lrcl}
\text{minimize} & \displaystyle\sum_{(u,v)\in E} c_{(u,v)} x_{(u,v)} & & \\
s.t.\ \ \forall v \neq t : & \displaystyle\sum_{w:(v,w)\in E} x_{(v,w)} - \sum_{u:(u,v)\in E} x_{(u,v)} & = & 1 \\
\forall (u,v) \in E : & x_{(u,v)} & \geq & 0
\end{array}
$$

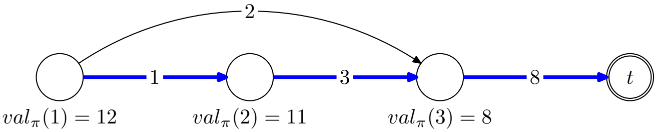
Flow conservation:



$$x_1 + x_2 = 1 + x_3 + x_4$$

- The flow through every vertex is at least 1.
- For a **basic feasible solution**, at most one edge leaving every vertex has non-zero flow.
- There is a one-to-one correspondence between **basic feasible solutions** and shortest paths trees (or **policies**).

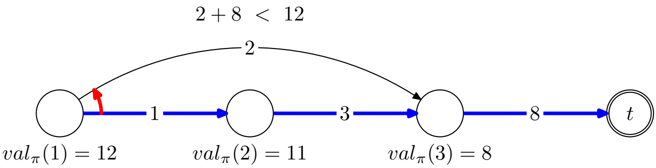
Improving pivots



- For every **policy** π (shortest paths tree), let $val_\pi(v)$ be the length of the path from v to t in π:

$$\forall (u,v) \in \pi : \quad val_\pi(u) \; = \; c_{(u,v)} + val_\pi(v)$$

Improving pivots



- For every **policy** π (shortest paths tree), let $val_\pi(v)$ be the length of the path from v to t in π:

$$\forall (u,v) \in \pi : \quad val_\pi(u) \; = \; c_{(u,v)} + val_\pi(v)$$

- An edge (u, v) is an **improving pivot** (or **improving switch**) w.r.t. π if it improves the value of u:

$$c_{(u,v)} + val_\pi(v) \; < \; val_\pi(u)$$

- Multiple **improving switches** can be performed in parallel, which gives a more general class of algorithms:

Function PolicyIteration(π)

while $\exists$ **improving switch** *w.r.t.* π **do**

    Update π by performing **improving switches**

return π

- Multiple **improving switches** can be performed in parallel, which gives a more general class of algorithms:

Function PolicyIteration(π)

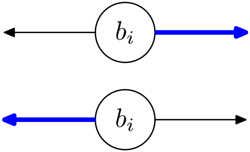
while $\exists$ **improving switch** *w.r.t.* π **do**
  Update π by performing **improving switches**

return π

- The simplex algorithm is the special case where only one **improving switch** is performed in every iteration.

Lower bound constructions: The first idea

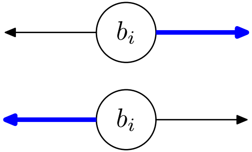
- To prove a lower bound for a given pivoting rule, we construct a family of graphs (or MDPs) G_n such that the corresponding PolicyIteration algorithm simulates an *n*-**bit binary counter**.
- We define a way to interpret a **policy** π as a configuration of the binary counter:



$\Rightarrow \quad bit_i(\pi) = 0$

$\Rightarrow \quad bit_i(\pi) = 1$

- To prove a lower bound for a given pivoting rule, we construct a family of graphs (or MDPs) G_n such that the corresponding PolicyIteration algorithm simulates an ***n*-bit binary counter**.
- We define a way to interpret a **policy** π as a configuration of the binary counter:

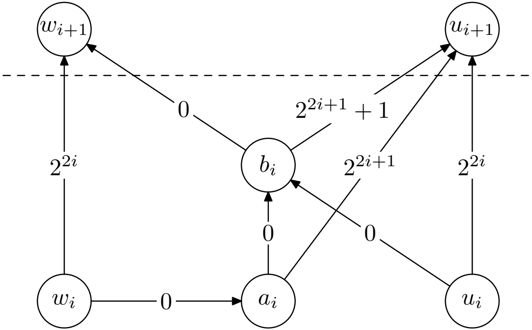


$$\Rightarrow \quad bit_i(\pi) = 0$$

$$\Rightarrow \quad bit_i(\pi) = 1$$

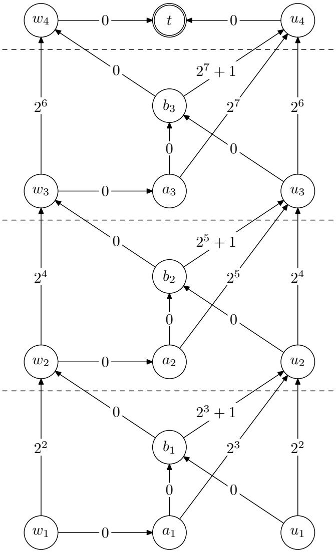
- We then show that (with high probability) a run of the PolicyIteration algorithm generates all 2^n counting configurations.

A simplified construction

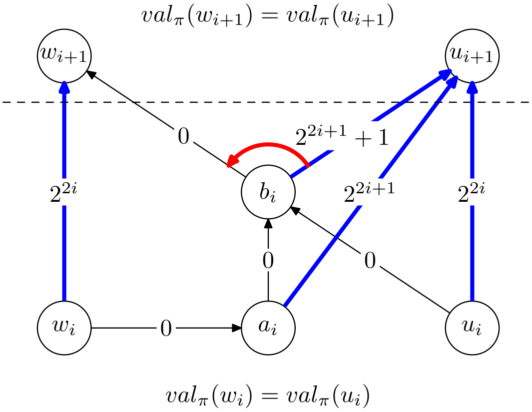


- The graph is acyclic, and every bit i is associated with a level consisting of four vertices: b_i, a_i, w_i, u_i.
- $w_{n+1} = u_{n+1} = t$.

A simplified construction, $n = 3$

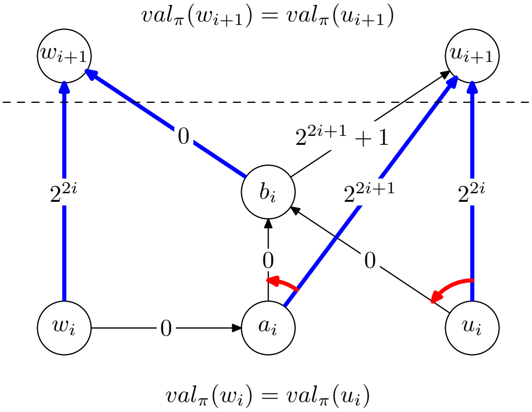


Case: $val_\pi(w_{i+1}) = val_\pi(u_{i+1})$



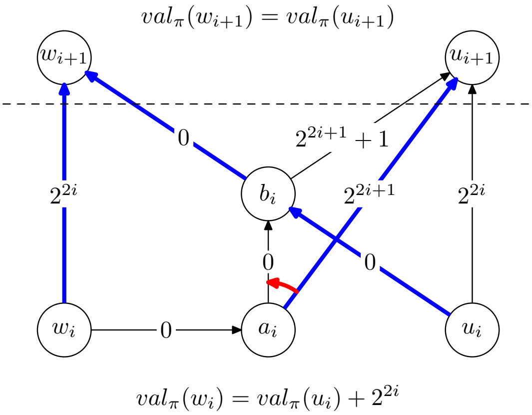
- $bit_i(\pi) = 0$, stable.

Case: $val_\pi(w_{i+1}) = val_\pi(u_{i+1})$



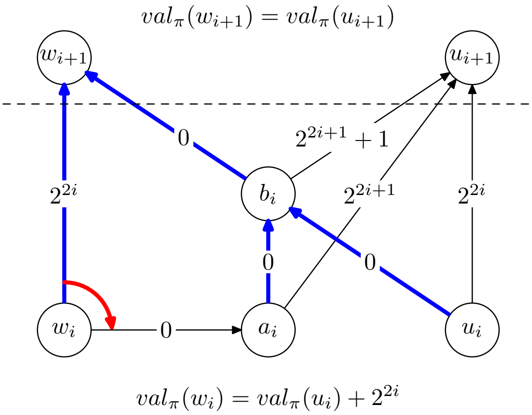
- $bit_i(\pi) = 1$, transitioning.

Case: $val_\pi(w_{i+1}) = val_\pi(u_{i+1})$



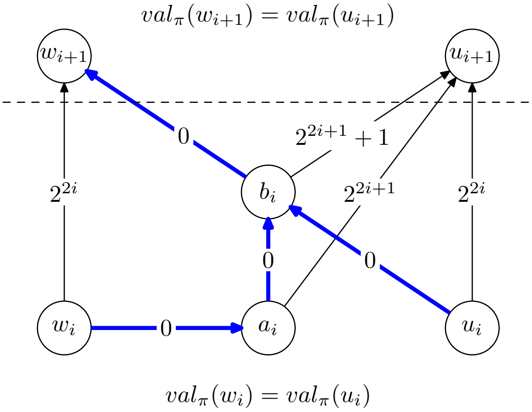
- $bit_i(\pi) = 1$, transitioning, lower bits are unstable: reset.

Case: $val_\pi(w_{i+1}) = val_\pi(u_{i+1})$

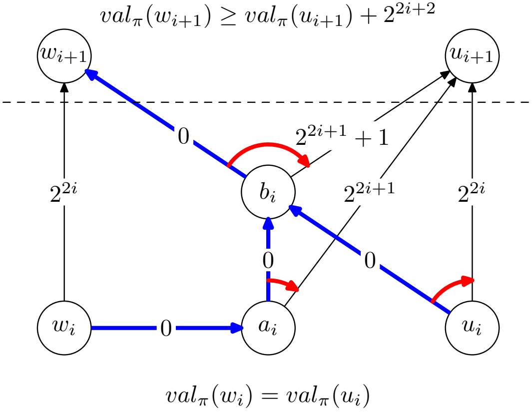


- $bit_i(\pi) = 1$, transitioning, lower bits are unstable: reset.

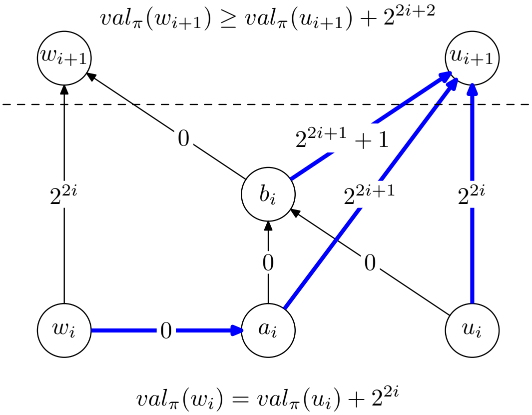
Case: $val_\pi(w_{i+1}) = val_\pi(u_{i+1})$



- $bit_i(\pi) = 1$, stable.



- $bit_i(\pi) = 1$, unstable, resetting.



- $bit_i(\pi) = 0$, stable, lower bits are unstable: reset.
- w_i is updated when bit $i+1$ stabilizes.

Lower bound for Bland's rule

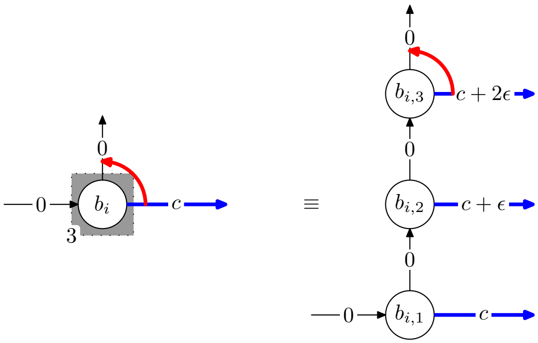
- Bland's rule for shortest paths: Perform the first **improving switch** according to a permutation of the edges.

- Bland's rule for shortest paths: Perform the first **improving switch** according to a permutation of the edges.
- It is easy to define a permutation of the edges, σ, such that we get the described behavior, giving an exponential lower bound:
  - (b_i, w_{i+1}) edges are placed last, and $\sigma(b_i, w_{i+1}) < \sigma(b_j, w_{j+1})$ for $i < j$.
  - (a_i, b_i) edges are placed next, and $\sigma(a_i, b_i) < \sigma(a_j, b_j)$ for $i < j$.
  - The remaining edges are placed first in arbitrary order.

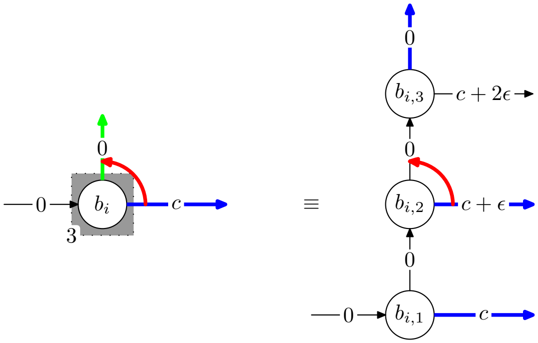
- BLAND'S RULE for shortest paths: Perform the first **improving switch** according to a permutation of the edges.
- It is easy to define a permutation of the edges, σ, such that we get the described behavior, giving an exponential lower bound:
  - (b_i, w_{i+1}) edges are placed last, and $\sigma(b_i, w_{i+1}) < \sigma(b_j, w_{j+1})$ for $i < j$.
  - (a_i, b_i) edges are placed next, and $\sigma(a_i, b_i) < \sigma(a_j, b_j)$ for $i < j$.
  - The remaining edges are placed first in arbitrary order.
- To implement a lower bound for RANDOMEDGE we need a gadget to delay improving switches like (b_i, w_{i+1}) and (a_i, b_i).

Outline

- Linear programming and the simplex algorithm.
- Related work and results.
- The simplex algorithm for shortest paths.
- Framework: Lower bounds for the simplex algorithm utilizing shortest paths (and Markov decision processes).
- ⇒ On the lower bound for RandomEdge.
- (On the lower bound for RandomFacet.)
- Summary of open problems.

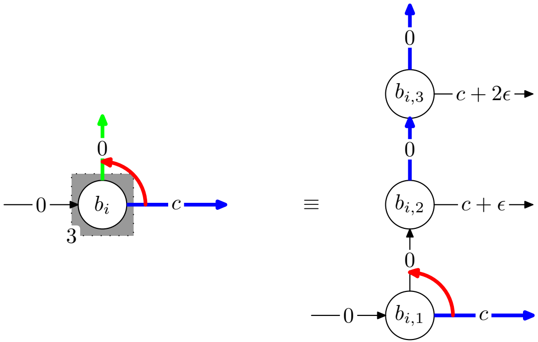


- By replacing a vertex by a chain of vertices, a specific sequence of improving switches has to be performed to get the same effect as performing one improving switch originally.
- At any time there is only one edge for which it is improving to move into the chain.



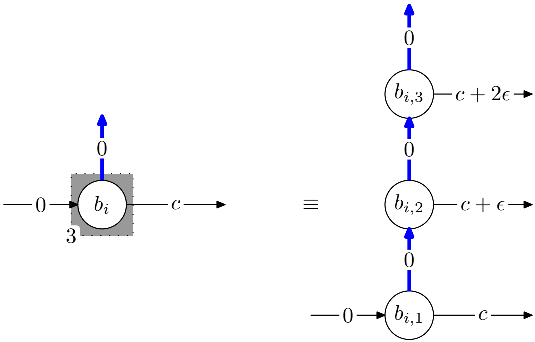
- By replacing a vertex by a chain of vertices, a specific sequence of improving switches has to be performed to get the same effect as performing one improving switch originally.
- At any time there is only one edge for which it is improving to move into the chain.

Delaying events



- By replacing a vertex by a chain of vertices, a specific sequence of improving switches has to be performed to get the same effect as performing one improving switch originally.
- At any time there is only one edge for which it is improving to move into the chain.

Delaying events



- By replacing a vertex by a chain of vertices, a specific sequence of improving switches has to be performed to get the same effect as performing one improving switch originally.
- At any time there is only one edge for which it is improving to move into the chain.

Competing chains

- Suppose a short chain of length ℓ_i is competing with a longer chain of length ℓ_{i+1}.
- There is exactly one improving switch in both chains, and RandomEdge performs either one of them with equal probability.

Competing chains

- Suppose a short chain of length ℓ_i is competing with a longer chain of length ℓ_{i+1}.
- There is exactly one improving switch in both chains, and $\textsc{RandomEdge}$ performs either one of them with equal probability.
- Let X be the number of heads observed in $\ell_i + \ell_{i+1}$ coin tosses, then by a **Chernoff bound**:

$$Pr[X \leq \ell_i] \leq e^{\frac{(\ell_{i+1}-\ell_i)^2}{2(\ell_{i+1}+\ell_i)}}$$

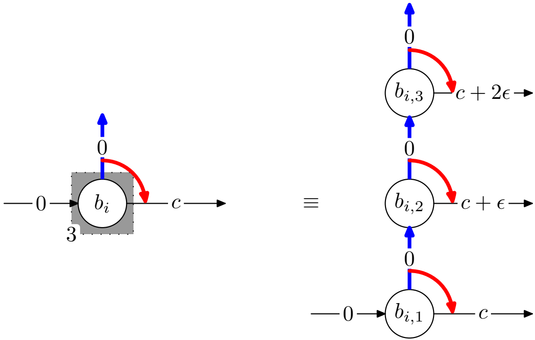
- Setting $\ell_k = \Theta(k^2 n)$, the probability of failure, $X < \ell_i$, is at most e^{-n}.

- Suppose a short chain of length ℓ_i is competing with a longer chain of length ℓ_{i+1}.
- There is exactly one improving switch in both chains, and RandomEdge performs either one of them with equal probability.
- Let X be the number of heads observed in $\ell_i + \ell_{i+1}$ coin tosses, then by a **Chernoff bound**:

$$Pr[X \leq \ell_i] \leq e^{\frac{(\ell_{i+1}-\ell_i)^2}{2(\ell_{i+1}+\ell_i)}}$$

- Setting $\ell_k = \Theta(k^2 n)$, the probability of failure, $X < \ell_i$, is at most e^{-n}.
- With n such chains this results in $N = O(n^4)$ vertices, giving a lower bound of $2^{\Omega(N^{1/4})}$ expected pivoting steps for RandomEdge.

Fast resetting



- Moving in the other directions happens much faster since all edges are improving switches simultaneously.

Problem

- We need to reset the progress made in chains at higher bits in order for the analysis to work.

Problem

- We need to reset the progress made in chains at higher bits in order for the analysis to work.
- The graph should not be acyclic: higher bits must have access to lower bits.

Problem

- We need to reset the progress made in chains at higher bits in order for the analysis to work.
- The graph should not be acyclic: higher bits must have access to lower bits.
- No vertex in a chain should get the benefit of setting a bit before this event occurs.
  - RandomEdge solves shortest paths in $O(NM)$ expected iterations, where N is the number of vertices and M is the number of edges.

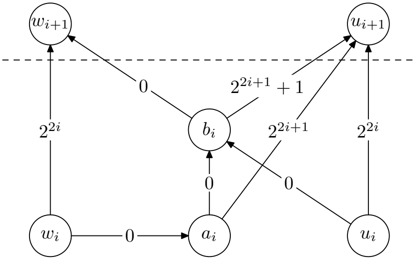
Problem

- We need to reset the progress made in chains at higher bits in order for the analysis to work.
- The graph should not be acyclic: higher bits must have access to lower bits.
- No vertex in a chain should get the benefit of setting a bit before this event occurs.
  - RANDOMEDGE solves shortest paths in $O(NM)$ expected iterations, where N is the number of vertices and M is the number of edges.
  - Use the power of MDPs: Introduce stochastic transitions.

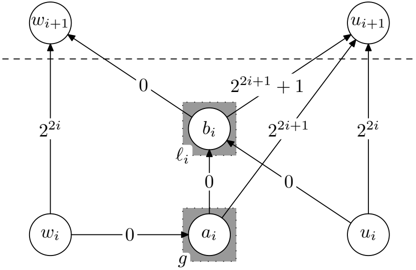
Problem

- We need to reset the progress made in chains at higher bits in order for the analysis to work.
- The graph should not be acyclic: higher bits must have access to lower bits.
- No vertex in a chain should get the benefit of setting a bit before this event occurs.
  - RandomEdge solves shortest paths in $O(NM)$ expected iterations, where N is the number of vertices and M is the number of edges.
  - Use the power of MDPs: Introduce stochastic transitions.
- To reset b_i-chains we need additional c_i-chains, resulting in alternating behavior.

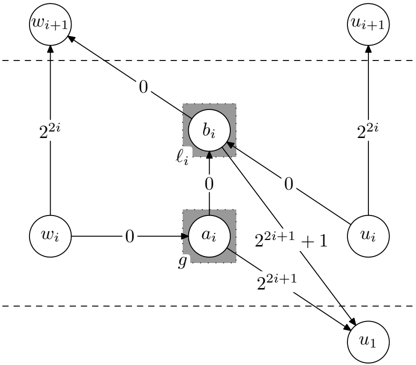
Construction for RandomEdge



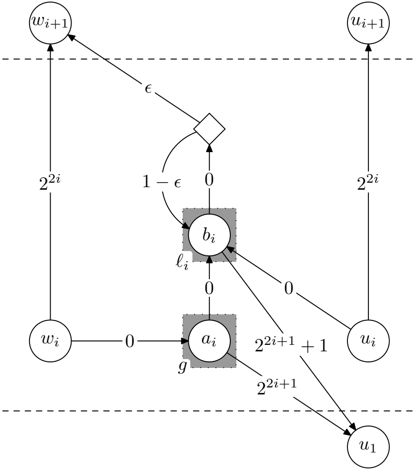
Construction for RandomEdge

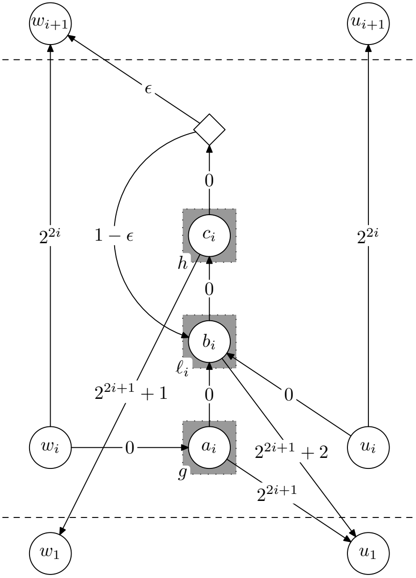


Construction for RANDOMEDGE



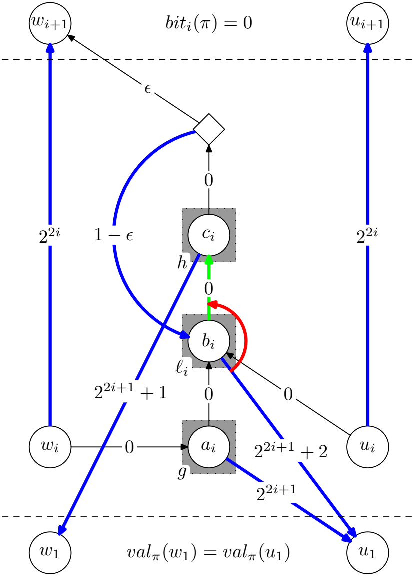
Construction for RandomEdge



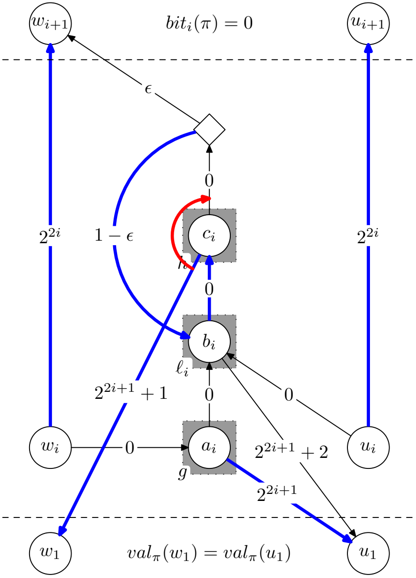


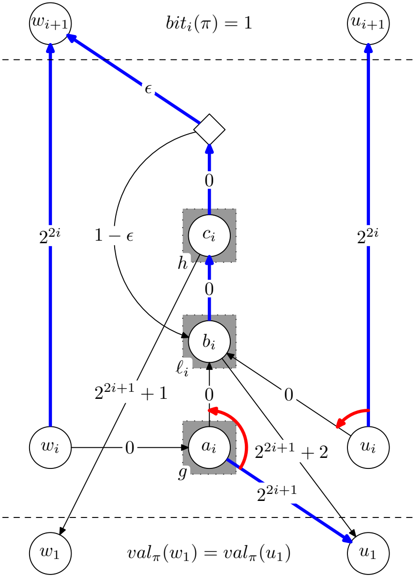
w_{i+1}
u_{i+1}
ϵ
0
2^{2i}
$1-\epsilon$
c_i
2^{2i}
h
0
b_i
ℓ_i
$2^{2i+1}+1$
0
0
w_i
0
a_i
$2^{2i+1}+2$
u_i
g
2^{2i+1}
w_1
u_1

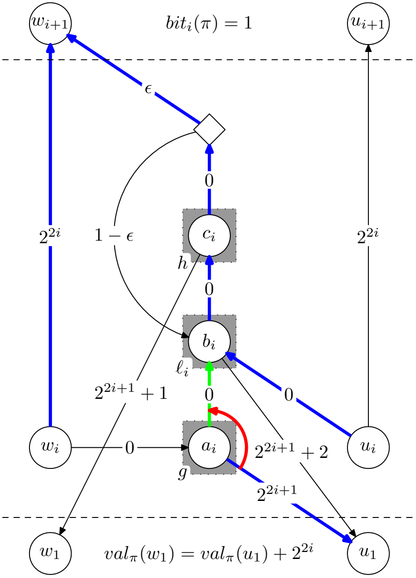
Setting a bit



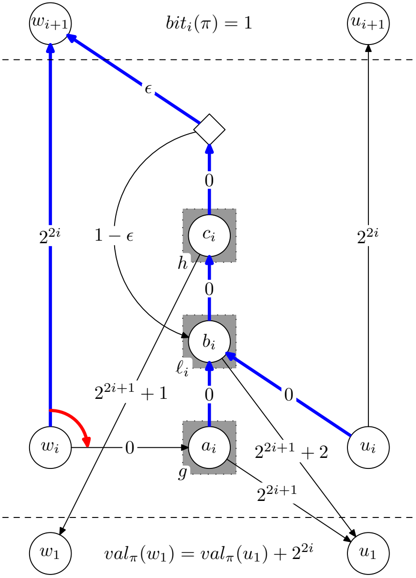
Setting a bit



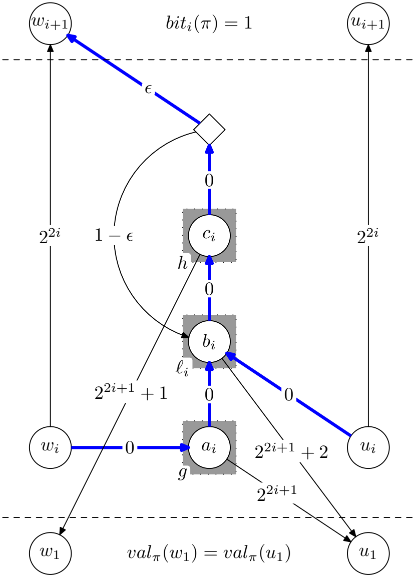

w_{i+1}
$bit_i(\pi) = 1$
u_{i+1}
ϵ
0
2^{2i}
$1 - \epsilon$
c_i
2^{2i}
h
0
b_i
ℓ_i
$2^{2i+1} + 1$
0
0
w_i
0
a_i
$2^{2i+1} + 2$
u_i
g
2^{2i+1}
w_1
$val_\pi(w_1) = val_\pi(u_1)$
u_1


w_{i+1}
$bit_i(\pi) = 1$
u_{i+1}
ϵ
0
2^{2i}
$1-\epsilon$
c_i
2^{2i}
h
0
b_i
ℓ_i
$2^{2i+1}+1$
0
0
w_i
0
a_i
$2^{2i+1}+2$
u_i
g
2^{2i+1}
w_1
$val_\pi(w_1) = val_\pi(u_1) + 2^{2i}$
u_1

Setting a bit



Setting a bit



RandomEdge lower bound

Theorem (Friedmann, Hansen, and Zwick (2011))

The worst-case expected number of pivoting steps performed by RandomEdge *on linear programs with* m *equalities and* $n = 2m$ *non-negative variables is* $2^{\Omega(m^{1/4})}$.

Outline

- Linear programming and the simplex algorithm.
- Related work and results.
- The simplex algorithm for shortest paths.
- Framework: Lower bounds for the simplex algorithm utilizing shortest paths (and Markov decision processes).
- On the lower bound for RandomEdge.
- ⇒ (On the lower bound for RandomFacet.)
- Summary of open problems.

The RandomFacet pivoting rule

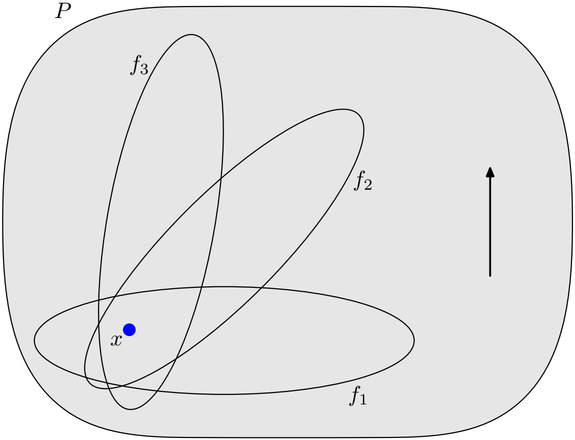
- RandomFacet, Kalai (1992):
  1. Pick a uniformly random facet f that contains the current basic feasible solution x.
  2. Recursively find the optimal solution x' within the picked facet f.
  3. If possible, make an improving pivot from x', leaving the facet f, and repeat from (1). Otherwise return x'.

The RandomFacet pivoting rule

- RandomFacet, Kalai (1992):
  1. Pick a uniformly random facet f that contains the current basic feasible solution x.
  2. Recursively find the optimal solution x' within the picked facet f.
  3. If possible, make an improving pivot from x', leaving the facet f, and repeat from (1). Otherwise return x'.

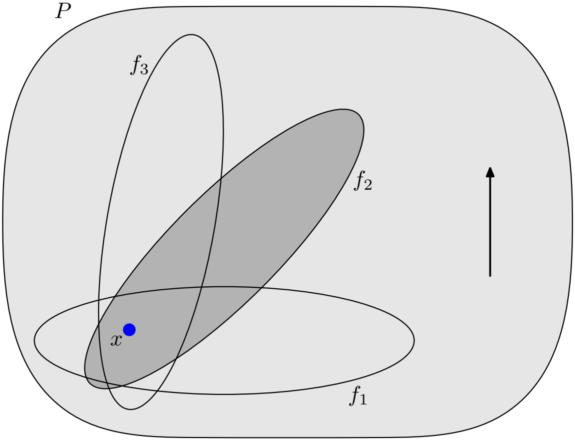
- A **dual** variant of the RandomFacet pivoting rule was discovered independently by Matoušek, Sharir, and Welzl (1992).

The RandomFacet pivoting rule



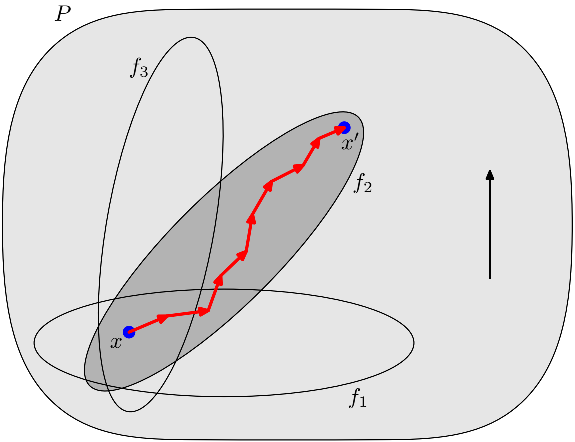
- Pick a uniformly random facet f_i that contains the current basic feasible solution x.

The RandomFacet pivoting rule

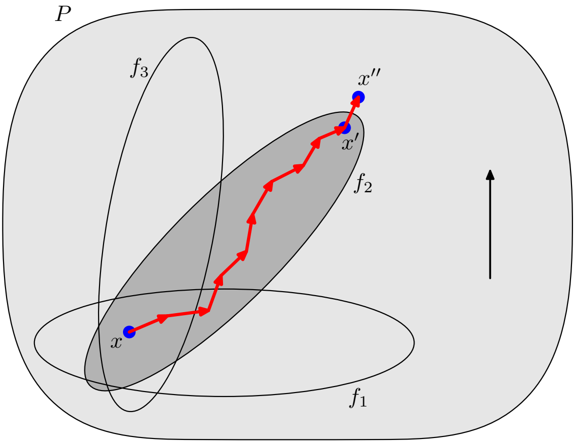


- Pick a uniformly random facet f_i that contains the current basic feasible solution x.

The RandomFacet pivoting rule

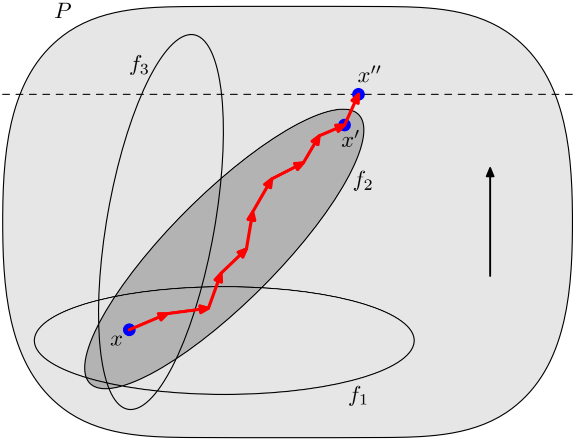


- Recursively find the optimal solution x' within the picked facet f_i.



- If possible, make an improving pivot from x', leaving the facet f_i, and repeat from the beginning. Otherwise return x'.

The RandomFacet pivoting rule



- Note that if the facets $f_1, \ldots, f_d$ containing x are ordered according to their optimal value, then from x'' we never visit $f_1, \ldots, f_i$ again.

- The number of pivoting steps for a linear program with dimension d and n inequalities is at most:

$$f(d,n) \leq f(d-1,n-1) + 1 + \frac{1}{d}\sum_{i=1}^{d} f(d,n-i)$$

with $f(d,n) = 0$ for $n \leq d$.

The RandomFacet pivoting rule

- The number of pivoting steps for a linear program with dimension d and n inequalities is at most:

$$f(d,n) \leq f(d-1,n-1) + 1 + \frac{1}{d}\sum_{i=1}^{d} f(d,n-i)$$

  with $f(d,n) = 0$ for $n \leq d$.
- Solving the corresponding recurrence gives:

$$f(d,n) \ \leq \ 2^{O(\sqrt{(n-d)\log n})}$$

Interpretation for shortest paths

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{(u,v)\in E} c_{(u,v)} x_{(u,v)} & \\
s.t.\ \ \forall v \neq t: & \displaystyle\sum_{w:(v,w)\in E} x_{(v,w)} - \sum_{u:(u,v)\in E} x_{(u,v)} & = \ 1 \\
\quad \forall (u,v) \in E: & \hfill x_{(u,v)} & \geq \ 0
\end{array}
$$

- Staying within a facet means that the corresponding inequality is tight, meaning that a variable is fixed to zero. This corresponds to removing the edge.
- The RandomFacet pivoting rule removes random unused edges and solves the corresponding problem recursively.

Interpretation for shortest paths

$$
\begin{array}{llcl}
\text{minimize} & \displaystyle\sum_{(u,v)\in E} c_{(u,v)} x_{(u,v)} & & \\
s.t.\ \ \forall v \neq t: & \displaystyle\sum_{w:(v,w)\in E} x_{(v,w)} - \sum_{u:(u,v)\in E} x_{(u,v)} & = & 1 \\
\quad \forall (u,v) \in E: & x_{(u,v)} & \geq & 0
\end{array}
$$

- Staying within a facet means that the corresponding inequality is tight, meaning that a variable is fixed to zero. This corresponds to removing the edge.
- The RandomFacet pivoting rule removes random unused edges and solves the corresponding problem recursively.
- Note that delaying a switch, as for Bland's rule, can also be viewed as removing the edge.

Different challenges

- When constructing lower bounds for RandomFacet, the challenge is to make sure that certain edges are not removed before certain other edges.

Different challenges

- When constructing lower bounds for RandomFacet, the challenge is to make sure that certain edges are not removed before certain other edges.
- Suppose an edge e must not be removed before another edge e'.

Different challenges

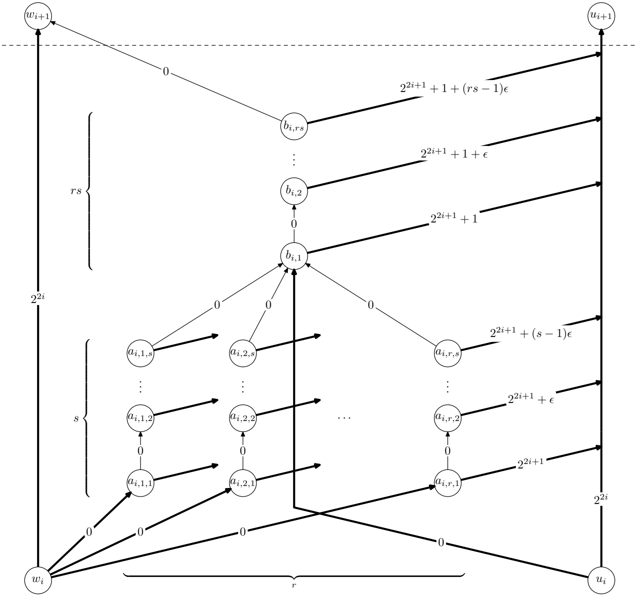
- When constructing lower bounds for RANDOMFACET, the challenge is to make sure that certain edges are not removed before certain other edges.
- Suppose an edge e must not be removed before another edge e'.
- To achieve this with high probability we make use of redundancy: Let e and e' be copied k times, in such a way that we only require that at least one copy of e' is removed before all copies of e are removed.

Different challenges

- When constructing lower bounds for RandomFacet, the challenge is to make sure that certain edges are not removed before certain other edges.
- Suppose an edge e must not be removed before another edge e'.
- To achieve this with high probability we make use of redundancy: Let e and e' be copied k times, in such a way that we only require that at least one copy of e' is removed before all copies of e are removed.
- The probability of failure, i.e. removing all k copies of e before one copy of e', is then:

$$\prod_{i=1}^{k} \frac{i}{i+k} = \frac{(k!)^2}{(2k)!} \leq \frac{1}{2^k}$$

Lower bound construction



Analysis: simulate a "randomized bitcounter"

Start with n bits with value 0:	00000
Pick a random bit i and fix it:	$00\underline{0}00$
Count recursively with the remaining $n-1$ bits:	$11\underline{0}11$
Increment the i'th bit:	$11\underline{1}11$
Reset the $i-1$ lower bits:	$11\underline{1}00$
Count recursively with the $i-1$ lower bits:	11100

- Expected number of increments:

$$\begin{aligned} f(0) &= 0 \\ f(n) &= f(n-1) + 1 + \frac{1}{n}\sum_{i=0}^{n-1} f(i) \qquad \text{for} \qquad n > 0 \end{aligned}$$

- Solving the recurrence gives: $\quad f(n) = 2^{\Theta(\sqrt{n})}$

Open problems

- Subexponential upper bounds for RandomEdge and Randomized Bland's rule?
- Close the gap between the $2^{\tilde{\Omega}(m^{1/3})}$ and $2^{O(\sqrt{m \log n})}$ bounds for RandomFacet for linear programs.
- The polynomial Hirsch conjecture: A polynomial upper bound for the diameter of polytopes?
- Strongly polynomial time algorithm for linear programming? A variant of the simplex algorithm?
  - This question remains open already for Markov decision processes.

The end

Thank you for listening!

On the diameter of polytopes

- The **diameter** of a polytope P is the maximum distance between any two vertices in the edge graph of P.
- The diameter gives a lower bound for any pivoting rule for the simplex algorithm.
- Hirsch conjecture (1957): The diameter of any n-facet convex polytope in d-dimensional Euclidean space is at most $n - d$.
- Kalai and Kleitman (1992): $O(n^{\log n})$ upper bound on the diameter.
- Counter-example by Santos (2010): Existence of polytopes with diameter $(1 + \epsilon)(n - d)$.
    - It remains open whether the diameter is polynomial, or even linear, in n and d.
- Our results are unrelated to the diameter: The constructed polytopes have low diameter.