Random-Edge is slower than Random-Facet on abstract cubes

Thomas Dueholm Hansen[1] Uri Zwick[2]

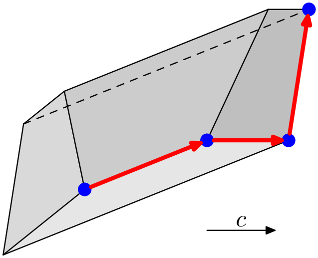
[1] Department of Computer Science,
Aarhus University, Denmark.

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

August 13, 2016

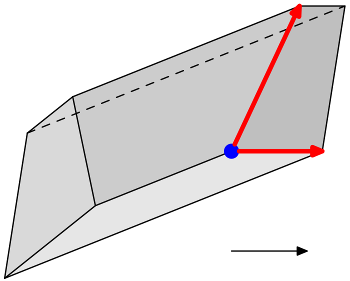
The simplex algorithm, Dantzig [1947]

$$\begin{array}{lrcl} \max & c^T x & & \\ s.t. & Ax & \leq & b \end{array}$$



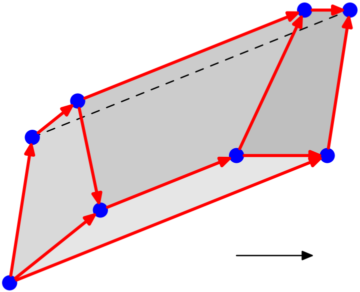
- **Linear programming**: Maximize a **linear objective function** subject to **linear constraints**.
- **The simplex algorithm**: Move from **vertex** to **vertex** along **edges** while improving the objective.
  - This operation is called a **pivot**.

Pivoting rules



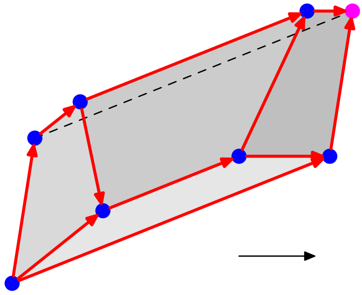
- A **pivoting rule** chooses which **improving pivot** to make.
- RANDOM-EDGE: Repeatedly use a uniformly random **improving pivot**.

Orientations



- The objective function defines an **orientation** of the **edges**.
- Many pivoting rules only rely on this orientation.

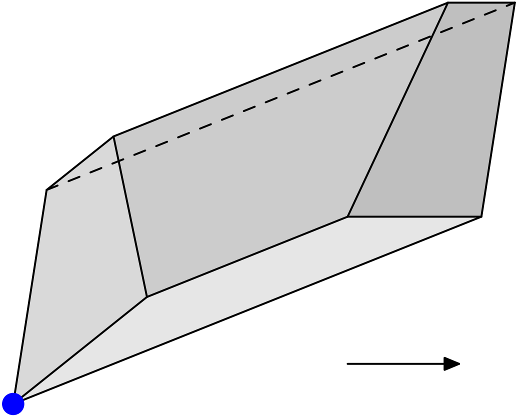
Orientations



- The objective function defines an **orientation** of the **edges**.
- Many pivoting rules only rely on this orientation.
- RANDOM-EDGE: Perform a **random walk** until reaching the **sink** where all **edges** are incoming.

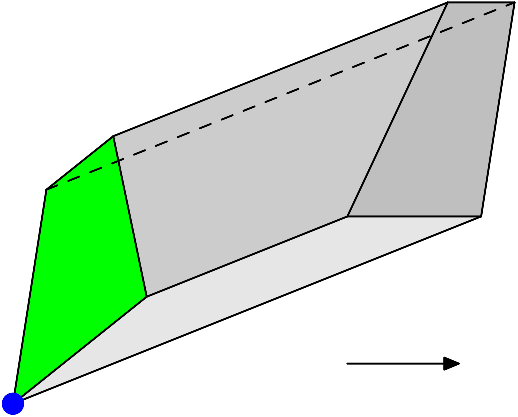
The RANDOM-FACET pivoting rule

- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].

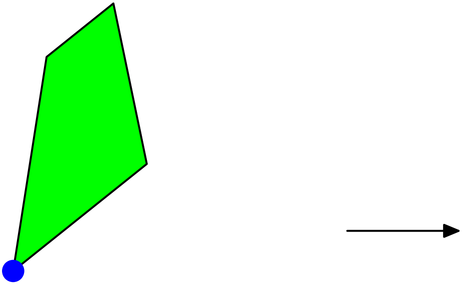


The RANDOM-FACET pivoting rule

- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].

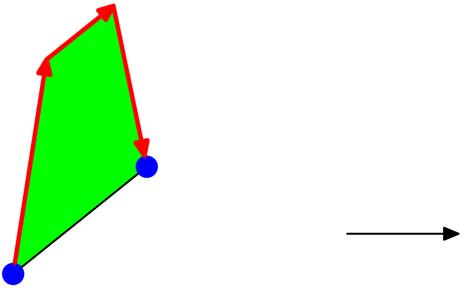


- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].



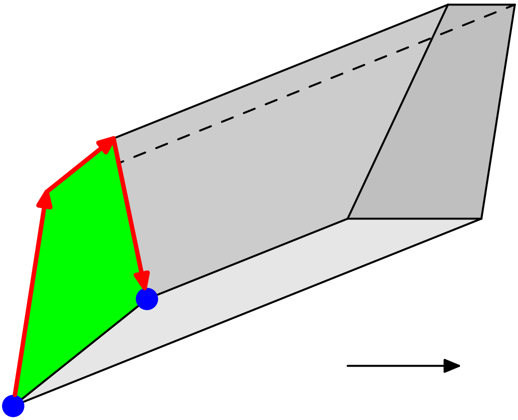
The RANDOM-FACET pivoting rule

- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].

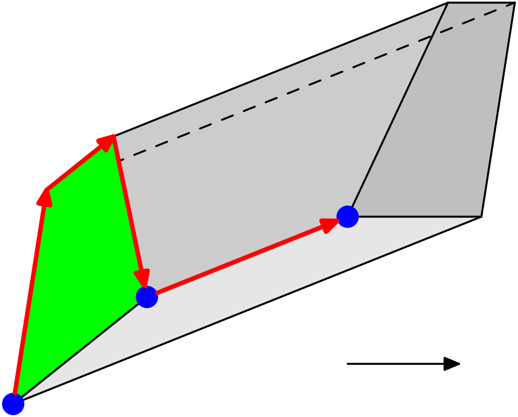


The RANDOM-FACET pivoting rule

- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].

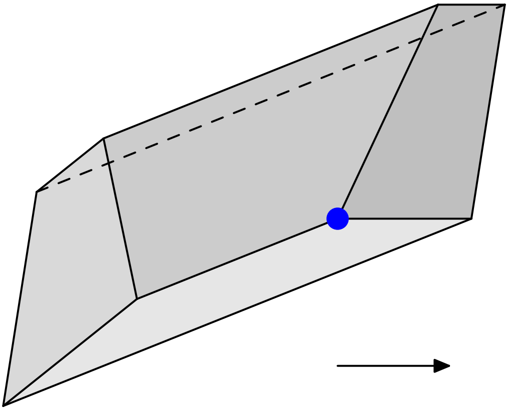


- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].



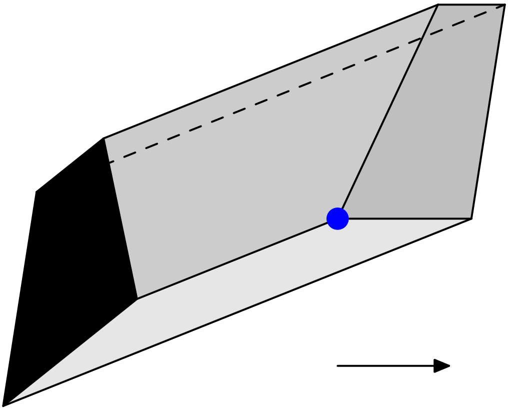
The RANDOM-FACET pivoting rule

- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].

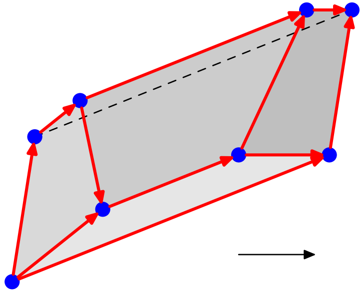


The RANDOM-FACET pivoting rule

- RANDOM-FACET: Introduced independently by Kalai [1992] and by Matoušek, Sharir and Welzl [1992].

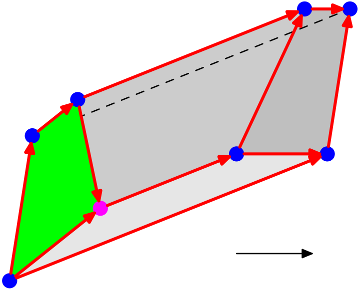


Properties of the orientation



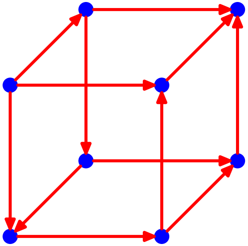
1. The graph is **acyclic**.

Properties of the orientation



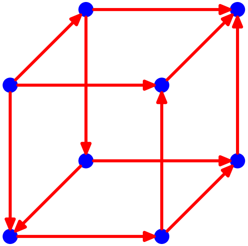
1. The graph is **acyclic**.
2. The **unique sink property**: In every **face** there is a **unique** **sink** (optimal **vertex** within the face).

Abstract cubes

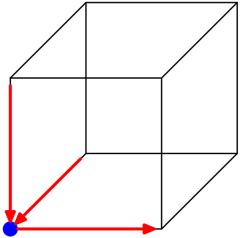


- **Acyclic unique sink orientations** (AUSOs) (or **abstract objective functions**) are orientations that are
  1. **acyclic** and
  2. have the **unique sink property**.

Abstract cubes

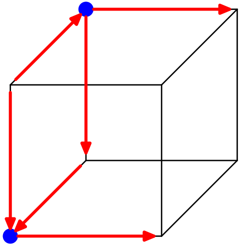


- **Acyclic unique sink orientations** (AUSOs) (or **abstract objective functions**) are orientations that are
  1. **acyclic** and
  2. have the **unique sink property**.
- AUSOs can be defined for arbitrary polytopes. We focus on the case where the underlying polytope is a **hypercube**.



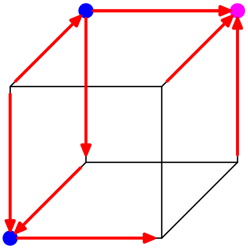
- An algorithm asks an **oracle** for the orientation of the **edges** adjacent to a **vertex**.

Acyclic unique sink orientations



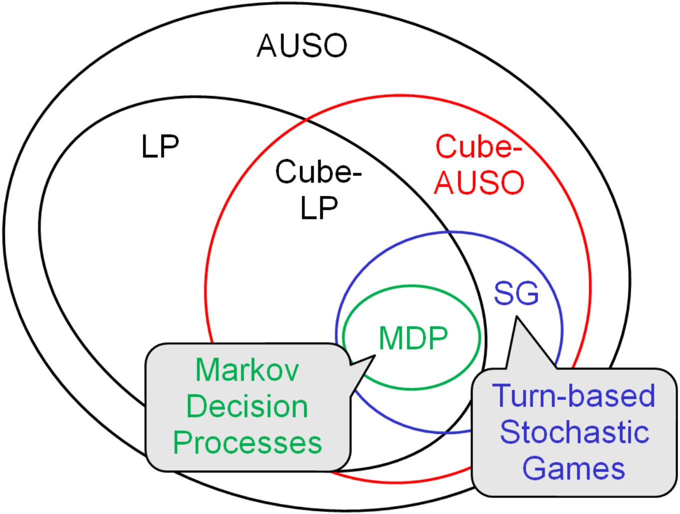
- An algorithm asks an **oracle** for the orientation of the **edges** adjacent to a **vertex**.

Acyclic unique sink orientations



- An algorithm asks an **oracle** for the orientation of the **edges** adjacent to a **vertex**.
- **Goal:** Find the global **sink** with as few oracle calls as possible.

AUSOs and some applications



Results about Random-Facet

Bounds for the expected number of steps performed by Random-Facet on n-dimensional AUSOs with m facets.

- Kalai [1992] and Matoušek, Sharir and Welzl [1992]: $2^{O(\sqrt{(m-n)\log n})}$
- Gärtner [2002]: $2^{O(\sqrt{n})}$ for abstract cubes

Results about Random-Facet

Bounds for the expected number of steps performed by Random-Facet on n-dimensional AUSOs with m facets.

- Kalai [1992] and Matoušek, Sharir and Welzl [1992]: $2^{O(\sqrt{(m-n)\log n})}$
- Gärtner [2002]: $2^{O(\sqrt{n})}$ for abstract cubes
  - This is the best known bound for solving AUSOs on cubes.

Results about Random-Facet

Bounds for the expected number of steps performed by Random-Facet on n-dimensional AUSOs with m facets.

- Kalai [1992] and Matoušek, Sharir and Welzl [1992]: $2^{O(\sqrt{(m-n)\log n})}$
- Gärtner [2002]: $2^{O(\sqrt{n})}$ for abstract cubes
  - This is the best known bound for solving AUSOs on cubes.
- Matoušek [1994]: $2^{\Omega(\sqrt{n})}$ for abstract cubes ($m = 2n$)
- Friedmann, Hansen, and Zwick [2011]: $2^{\tilde{\Omega}(\sqrt[3]{m})}$ for linear programs

Results about RANDOM-EDGE

Bounds for the expected number of steps performed by RANDOM-EDGE on n-dimensional AUSOs of cubes.

- Matoušek and Szabó [2004]: $2^{\Omega(\sqrt[3]{n})}$
- Friedmann, Hansen, and Zwick [2011]: $2^{\Omega(\sqrt[4]{n})}$ for linear programs (that are cubes)

Results about RANDOM-EDGE

Bounds for the expected number of steps performed by RANDOM-EDGE on n-dimensional AUSOs of cubes.

- Matoušek and Szabó [2004]: $2^{\Omega(\sqrt[3]{n})}$
- Friedmann, Hansen, and Zwick [2011]: $2^{\Omega(\sqrt[4]{n})}$ for linear programs (that are cubes)
- Gärtner and Kaibel [2007]: $O(2^n/n^{\log n})$
- Hansen, Paterson, and Zwick [2014]: $O(1.80^n)$

Results about Random-Edge

Bounds for the expected number of steps performed by Random-Edge on n-dimensional AUSOs of cubes.

- Matoušek and Szabó [2004]: $2^{\Omega(\sqrt[3]{n})}$
- Friedmann, Hansen, and Zwick [2011]: $2^{\Omega(\sqrt[4]{n})}$ for linear programs (that are cubes)
- Gärtner and Kaibel [2007]: $O(2^n/n^{\log n})$
- Hansen, Paterson, and Zwick [2014]: $O(1.80^n)$
- **We show**: $2^{\Omega(\sqrt{n \log n})}$

Results about Random-Edge

Bounds for the expected number of steps performed by Random-Edge on n-dimensional AUSOs of cubes.

- Matoušek and Szabó [2004]: $2^{\Omega(\sqrt[3]{n})}$
- Friedmann, Hansen, and Zwick [2011]: $2^{\Omega(\sqrt[4]{n})}$ for linear programs (that are cubes)
- Gärtner and Kaibel [2007]: $O(2^n/n^{\log n})$
- Hansen, Paterson, and Zwick [2014]: $O(1.80^n)$
- **We show**: $2^{\Omega(\sqrt{n \log n})}$
  - Thus Random-Edge is slower than Gärtner's $2^{O(\sqrt{n})}$ upper bound for Random-Facet.

Results about Random-Edge

Bounds for the expected number of steps performed by Random-Edge on n-dimensional AUSOs of cubes.

- Matoušek and Szabó [2004]: $2^{\Omega(\sqrt[3]{n})}$
- Friedmann, Hansen, and Zwick [2011]: $2^{\Omega(\sqrt[4]{n})}$ for linear programs (that are cubes)
- Gärtner and Kaibel [2007]: $O(2^n/n^{\log n})$
- Hansen, Paterson, and Zwick [2014]: $O(1.80^n)$
- **We show**: $2^{\Omega(\sqrt{n \log n})}$
  - Thus Random-Edge is slower than Gärtner's $2^{O(\sqrt{n})}$ upper bound for Random-Facet.
  - Improving the bound further requires significantly new ideas.

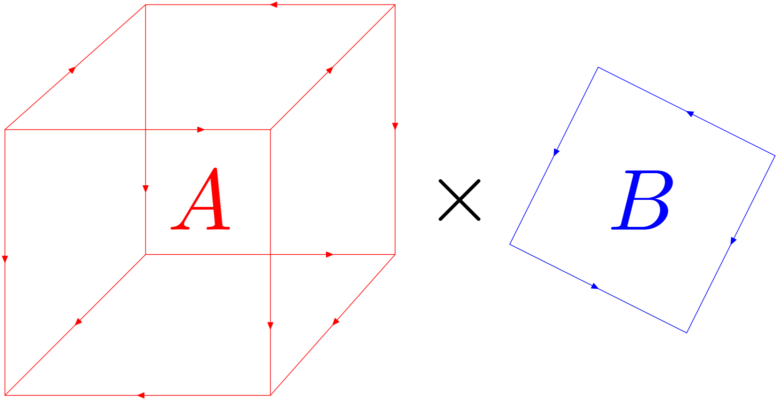
Results about RANDOM-EDGE

Bounds for the expected number of steps performed by RANDOM-EDGE on n-dimensional AUSOs of cubes.

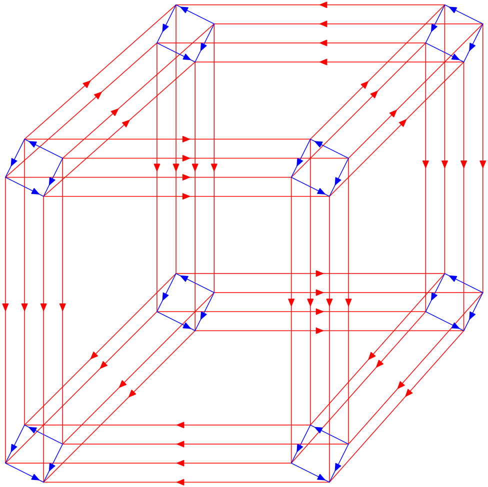
- Matoušek and Szabó [2004]: $2^{\Omega(\sqrt[3]{n})}$
- Friedmann, Hansen, and Zwick [2011]: $2^{\Omega(\sqrt[4]{n})}$ for linear programs (that are cubes)
- Gärtner and Kaibel [2007]: $O(2^n/n^{\log n})$
- Hansen, Paterson, and Zwick [2014]: $O(1.80^n)$
- **We show**: $2^{\Omega(\sqrt{n \log n})}$
  - Thus RANDOM-EDGE is slower than Gärtner's $2^{O(\sqrt{n})}$ upper bound for RANDOM-FACET.
  - Improving the bound further requires significantly new ideas.

Open problem: Is RANDOM-EDGE subexponential?

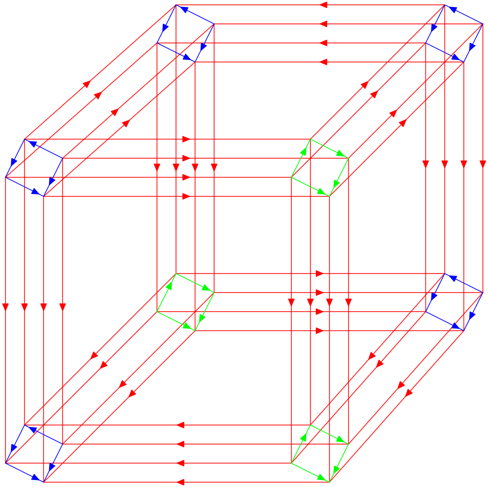
Product of AUSOs [Schurr and Szabó, 2004]



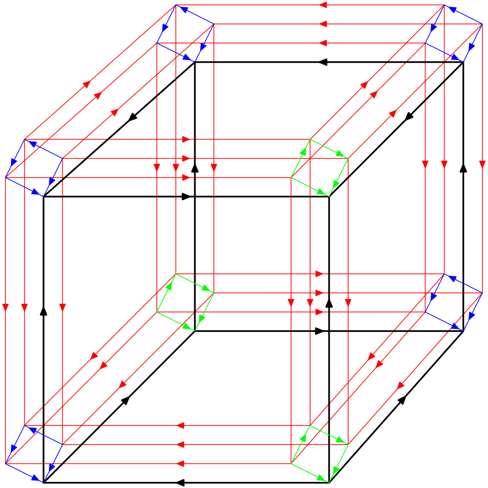
Product of AUSOs [Schurr and Szabó, 2004]



Product of AUSOs [Schurr and Szabó, 2004]



Hypersink replacement [Schurr and Szabó, 2004]

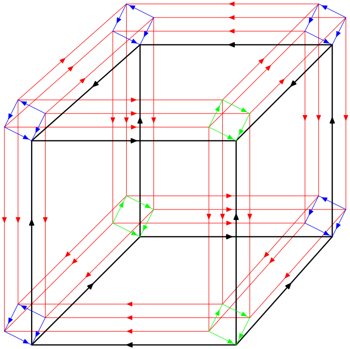


Matoušek and Szabó [2004]

- Let A be an AUSO for which Random-Edge performs T steps with high probability.
- **Goal**: Construct a slightly larger AUSO C for which Random-Edge performs $2T$ steps with high probability.

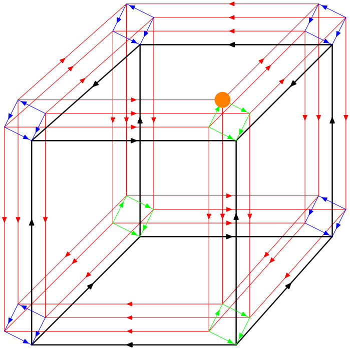
Matoušek and Szabó [2004]

- Let A be an AUSO for which Random-Edge performs T steps with high probability.
- **Goal**: Construct a slightly larger AUSO C for which Random-Edge performs $2T$ steps with high probability.
- **Construction**: Randomized product $C = A \times_R B$.



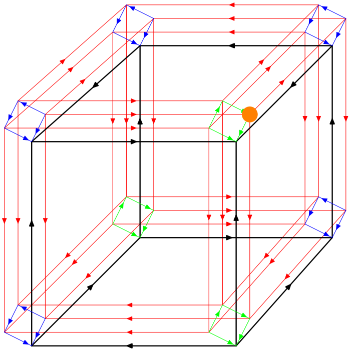
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



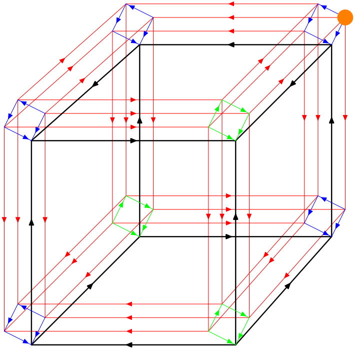
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



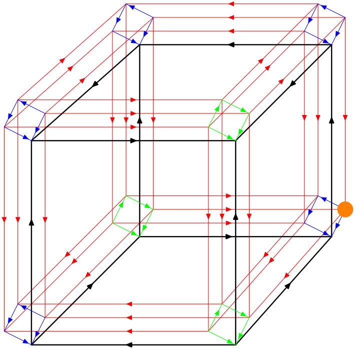
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



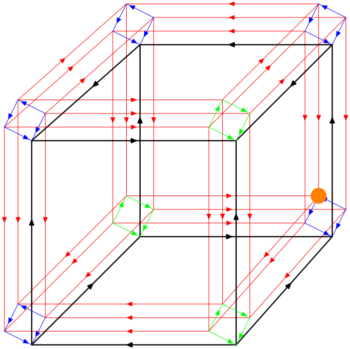
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



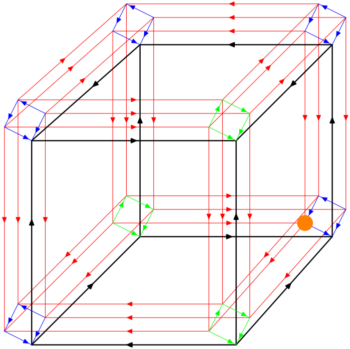
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



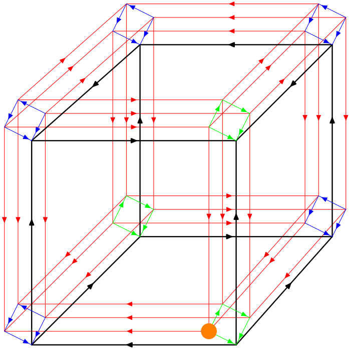
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



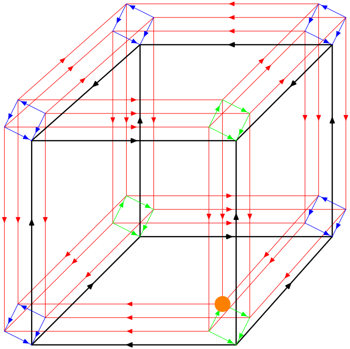
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



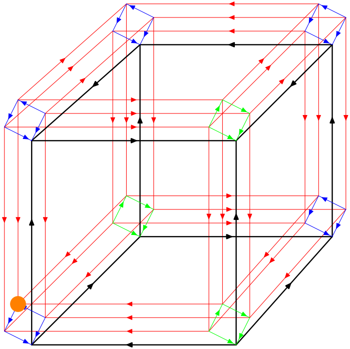
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



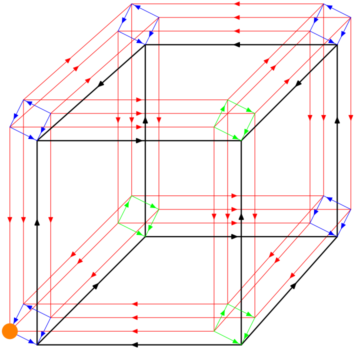
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



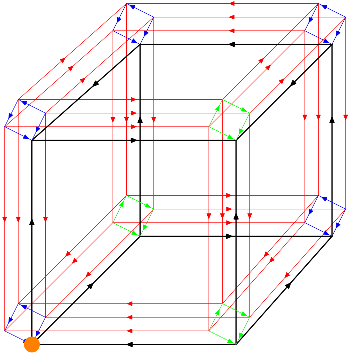
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



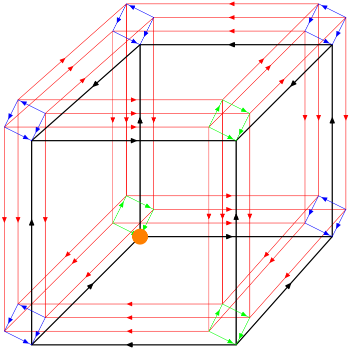
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



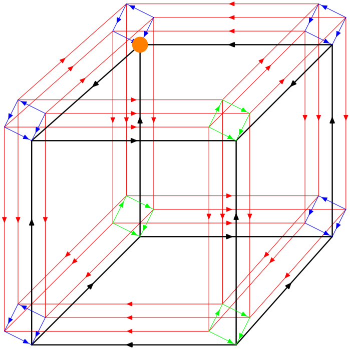
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



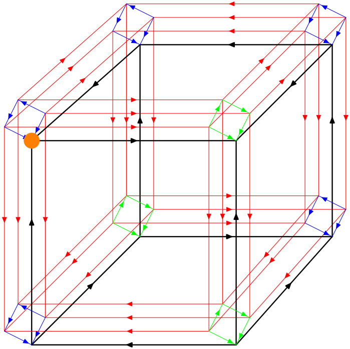
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.



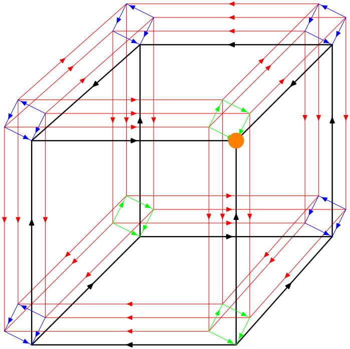
Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.

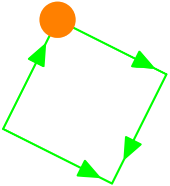


Random walk on product AUSO

- Every step in A brings us to a previously unvisited copy of B.
- Every copy of B has its coordinates randomly permuted.
- The hypersink is a randomly translated copy of A: This corresponds to starting from a uniformly random vertex of A.

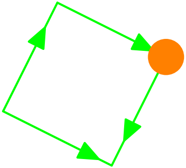


Random walk with reshuffles



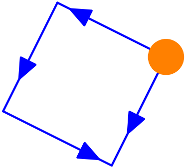
- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
    - A larger k delays progress in B.

Random walk with reshuffles



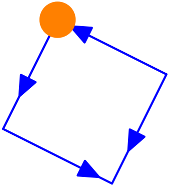
- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles



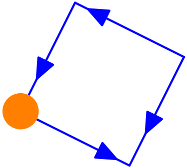
- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles



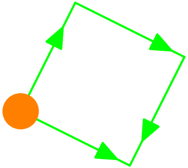
- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles



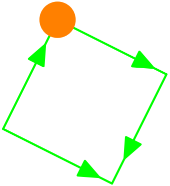
- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles



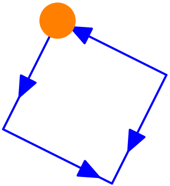
- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles

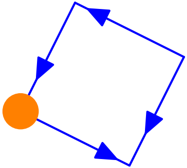


- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles

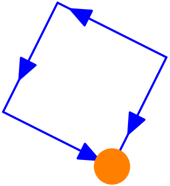


- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.



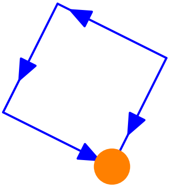
- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles

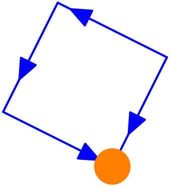


- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.

Random walk with reshuffles



- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.
- Matoušek and Szabó [2004] use many copies of the **Klee-Minty cube** to get a large k.



- **Main challenge**: Ensure that B does not reach its sink before A.
- $\textsc{Random-Reshuffle}_k$: Random walk on B where at least k edges are always available in A.
  - A larger k delays progress in B.
- Matoušek and Szabó [2004] use many copies of the **Klee-Minty cube** to get a large k.
- We simplify and improve their analysis by using only two copies (and therefore $k = 2$) of a **path AUSO**.

Path AUSOs

- Every vertex of a hypercube can be identified by a binary vector.
- **Path AUSO**: The i-th edge is outgoing iff the i-th coordinate is 0 and all previous coordinates are 1, or the i-th coordinate is 1 and some previous coordinate is 0.

In/**Out** : 11100110

- Every vertex of a hypercube can be identified by a binary vector.
- **Path AUSO**: The i-th edge is outgoing iff the i-th coordinate is 0 and all previous coordinates are 1, or the i-th coordinate is 1 and some previous coordinate is 0.

In/**Out** : 11100100

- Every vertex of a hypercube can be identified by a binary vector.
- **Path AUSO**: The i-th edge is outgoing iff the i-th coordinate is 0 and all previous coordinates are 1, or the i-th coordinate is 1 and some previous coordinate is 0.

In/**Out** : 11100000

- Every vertex of a hypercube can be identified by a binary vector.
- **Path AUSO**: The i-th edge is outgoing iff the i-th coordinate is 0 and all previous coordinates are 1, or the i-th coordinate is 1 and some previous coordinate is 0.

In/**Out** : 11110000

Path AUSOs

- Every vertex of a hypercube can be identified by a binary vector.
- **Path AUSO**: The i-th edge is outgoing iff the i-th coordinate is 0 and all previous coordinates are 1, or the i-th coordinate is 1 and some previous coordinate is 0.

In/**Out** : 11111000

Path AUSOs

- Every vertex of a hypercube can be identified by a binary vector.
- **Path AUSO**: The *i*-th edge is outgoing iff the *i*-th coordinate is 0 and all previous coordinates are 1, or the *i*-th coordinate is 1 and some previous coordinate is 0.

In/**Out** : 11111100

Path AUSOs

- Every vertex of a hypercube can be identified by a binary vector.
- **Path AUSO**: The i-th edge is outgoing iff the i-th coordinate is 0 and all previous coordinates are 1, or the i-th coordinate is 1 and some previous coordinate is 0.

$$\textbf{In}/\textbf{Out}: \quad 11011101$$

- A reshuffle permutes the coordinates; the number of 0's and 1's remain unchanged.

In/**Out** : 11100110 $k = 5,\ j = 2$

- Suppose a vertex has k 1's, j of which are non-leading.
- Let $r \geq 2/(j+3)$ be the reshuffle probability for Random-Reshuffle$_2$ on B.

In/**Out** : 11100110 $\quad k = 5,\ j = 2$

- Suppose a vertex has k 1's, j of which are non-leading.
- Let $r \geq 2/(j+3)$ be the reshuffle probability for Random-Reshuffle$_2$ on B.
- In the next step in B, the number of 1's increases with probability:

$$p \;=\; (1-r)\cdot\frac{1}{j+1} + r\cdot\sum_{j'=0}^{k}\frac{\binom{n-(k-j'+1)}{j'}}{\binom{n}{k}}\frac{1}{j'+1}$$

- **Lemma**: For $8 \leq k \leq n-9$, the number of 1's increases with probability at most $5/12$.

$\textsc{Random-Reshuffle}_2$ on a path AUSO

In/**Out** :   11100110 $k = 5, \; j = 2$

- Suppose a vertex has k 1's, j of which are non-leading.
- Let $r \geq 2/(j+3)$ be the reshuffle probability for $\textsc{Random-Reshuffle}_2$ on B.
- In the next step in B, the number of 1's increases with probability:

$$p \;=\; (1-r) \cdot \frac{1}{j+1} + r \cdot \sum_{j'=0}^{k} \frac{\binom{n-(k-j'+1)}{j'}}{\binom{n}{k}} \frac{1}{j'+1}$$

- **Lemma**: For $8 \leq k \leq n-9$, the number of 1's increases with probability at most $5/12$.
- The process can be analyzed as a **biased random walk** on $\{0, 1, \ldots, n-17\}$.

Choosing the size of B

By analyzing the biased random walk on $\{0, 1, \ldots, n\}$ we get:

Lemma

Let P_m be the m-dimensional path AUSO. There are constants $\alpha, \beta > 0$ such that the probability that $\textsc{Random-Reshuffle}_2$ *on P_m, starting from a random vertex, performs less than $2^{\alpha m}$ steps before reaching the sink is at most $2^{-\beta m}$.*

- We let $A_0 = P_m$ and $A_i = A_{i-1} \times_R^2 P_m$ for $i > 1$.
- We show that the probability that $\textsc{Random-Edge}$ performs less than 2^{ℓ} steps when started at a random vertex of A_ℓ, where $\ell < \alpha m$, is at most $4 \cdot 2^{\ell - \beta m}$.

Choosing the size of B

By analyzing the biased random walk on $\{0, 1, \ldots, n\}$ we get:

Lemma

Let P_m be the m-dimensional path AUSO. There are constants $\alpha, \beta > 0$ such that the probability that $\textsc{Random-Reshuffle}_2$ *on P_m, starting from a random vertex, performs less than $2^{\alpha m}$ steps before reaching the sink is at most $2^{-\beta m}$.*

- We let $A_0 = P_m$ and $A_i = A_{i-1} \times_R^2 P_m$ for $i > 1$.
- We show that the probability that $\textsc{Random-Edge}$ performs less than 2^ℓ steps when started at a random vertex of A_ℓ, where $\ell < \alpha m$, is at most $4 \cdot 2^{\ell - \beta m}$.
- Choosing $\ell = \Theta(m)$ gives a $2^{\Omega(\sqrt{n})}$ lower bound, where $n = \Theta(\ell m)$.

Improving the bound further

- We show that RANDOM-RESHUFFLE$_2$ on the m-dimensional path AUSO in **two steps** almost always increases the number of 1's with probability at most $O(1/\sqrt{m})$.
- The improved analysis gives a $2^{\Omega(\sqrt{n \log n})}$ lower bound.

Improving the bound further

- We show that $\textsc{Random-Reshuffle}_2$ on the m-dimensional path AUSO in **two steps** almost always increases the number of 1's with probability at most $O(1/\sqrt{m})$.
- The improved analysis gives a $2^{\Omega(\sqrt{n \log n})}$ lower bound.
- **Can we do better?**

Improving the bound further

- We show that $\textsc{Random-Reshuffle}_2$ on the m-dimensional path AUSO in **two steps** almost always increases the number of 1's with probability at most $O(1/\sqrt{m})$.
- The improved analysis gives a $2^{\Omega(\sqrt{n \log n})}$ lower bound.
- **Can we do better? No**:
  - Any m-dimensional AUSO has a path of length at most m to its sink from every vertex.
  - This is true for any choice of B in $A_i = A_{i-1} \times_R B$.
  - Random-Edge on A_ℓ follows this path in B with probability at least $1/n^m$, where m is the dimension of B and $n = m\ell$ is the dimension of A_ℓ.

$$\#steps \;\leq\; \min\{2^\ell, n^{n/\ell}\} poly(n)$$

  - It is impossible to get a better ℓ relative to m, regardless of the choice of B.

Concluding remarks

- We gave a $2^{\Omega(\sqrt{n \log n})}$ lower bound for Random-Edge on abstract cubes (AUSOs), showing that Random-Edge is slower than Random-Facet for this problem.
- **Open problem**: Is Random-Edge subexponential? Can the lower bound be further improved?
- **Open problem**: Improve the $2^{\Omega(\sqrt[4]{n})}$ lower bound for linear programming by Friedmann, Hansen, and Zwick [2011].
- **Open problem**: Is there an algorithm for AUSOs that is faster than $2^{O(\sqrt{n})}$?
    - Schurr and Szabó [2004]: Any deterministic algorithm requires $\Omega(n^2/\log n)$ queries.

Concluding remarks

- We gave a $2^{\Omega(\sqrt{n \log n})}$ lower bound for Random-Edge on abstract cubes (AUSOs), showing that Random-Edge is slower than Random-Facet for this problem.
- **Open problem**: Is Random-Edge subexponential? Can the lower bound be further improved?
- **Open problem**: Improve the $2^{\Omega(\sqrt[4]{n})}$ lower bound for linear programming by Friedmann, Hansen, and Zwick [2011].
- **Open problem**: Is there an algorithm for AUSOs that is faster than $2^{O(\sqrt{n})}$?
  - Schurr and Szabó [2004]: Any deterministic algorithm requires $\Omega(n^2/\log n)$ queries.

Thank you for listening!