A linear lower bound for incrementing a space-optimal integer representation in the bit-probe model

Michael Raskin, raskin@mccme.ru

LaBRI, Université de Bordeaux

July 13, 2017

A linear lower bound for incrementing a space-optimal integer representation in the bit-probe model

Michael Raskin, raskin@mccme.ru

LaBRI, Université de Bordeaux

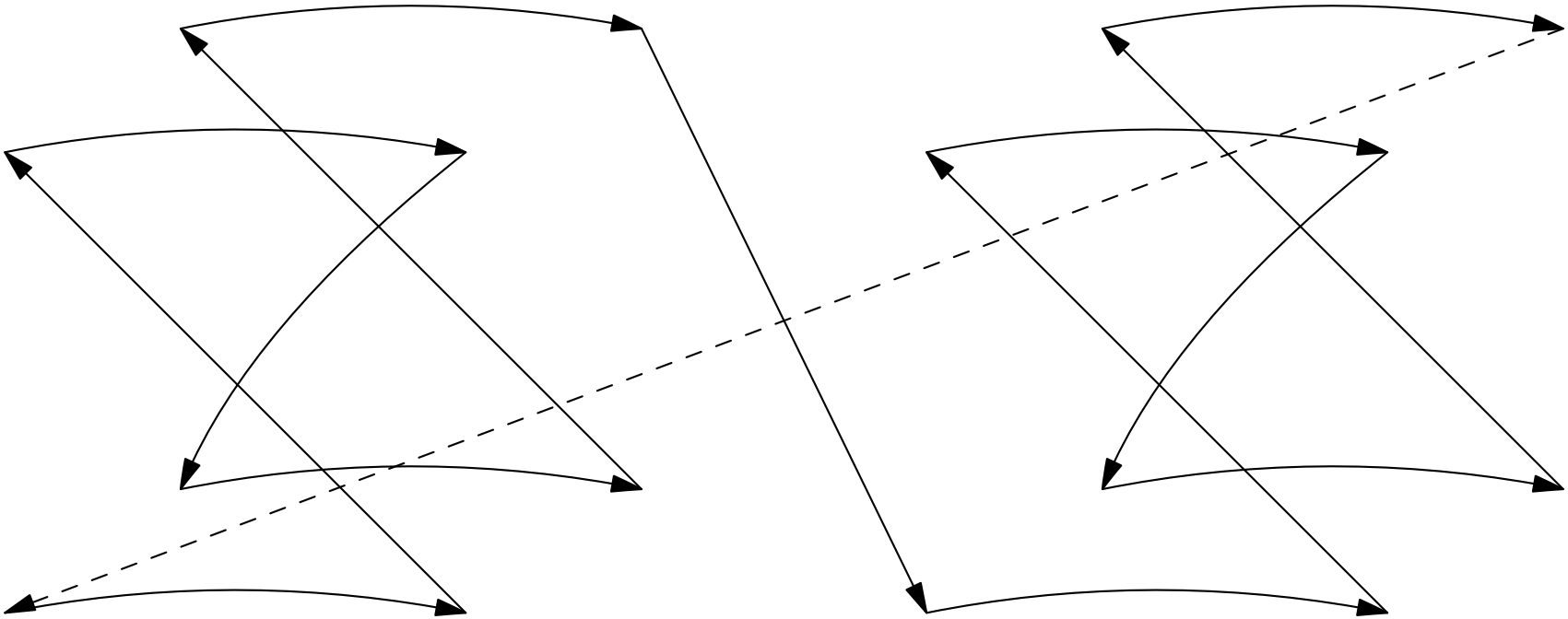
This work was mostly done in Aarhus University

July 13, 2017

How to count from 0 to $2^n - 1$ using n bits?

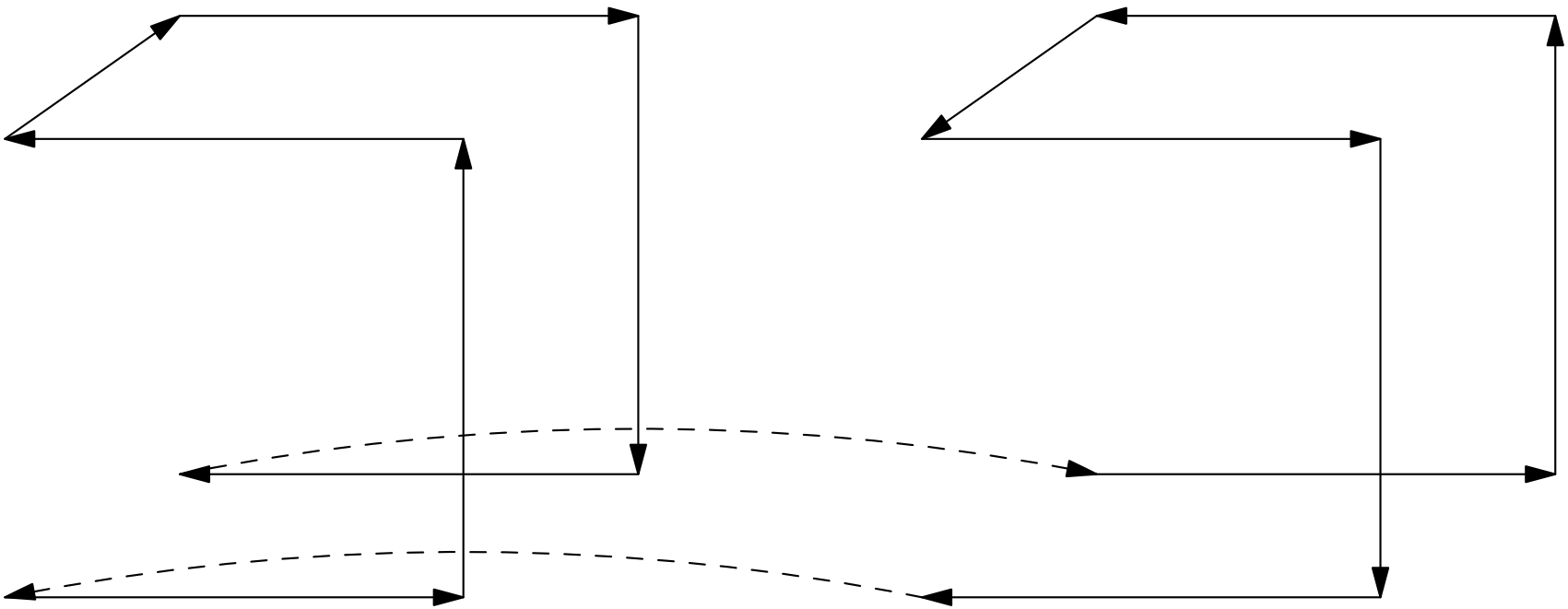
000, 001, 010, ...

Worst case (1111→0000): n reads, n writes



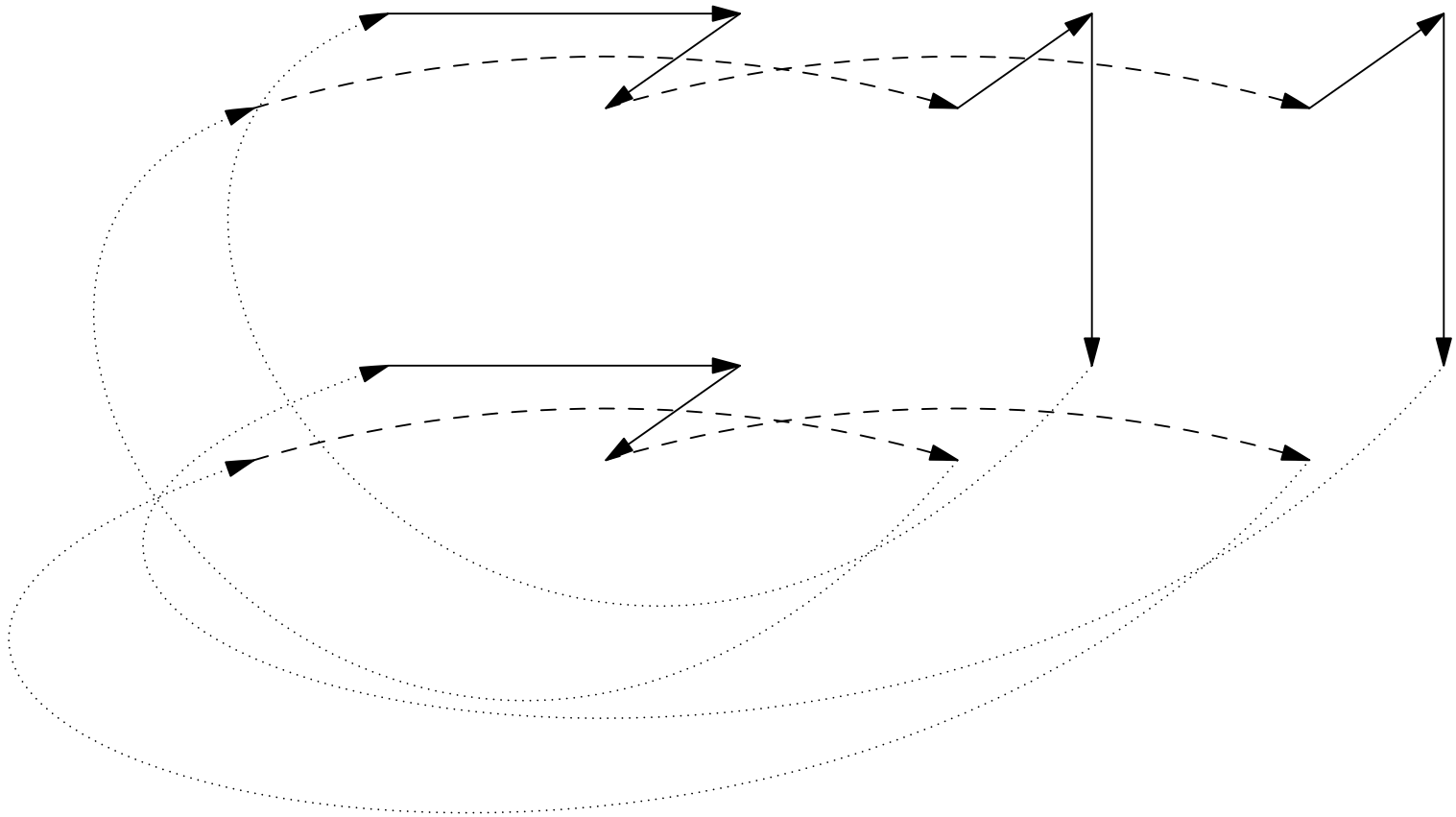
How to count from 0 to $2^n - 1$ using n bits?

[Gray1953] Frank Gray. Pulse code communication
Worst case: n reads, 1 write



How to count from 0 to $2^n - 1$ using n bits?

[BGPS2014]
Gerth Stølting Brodal, Mark Greve, Vineet Pandey, Srinivasa Rao Satti.
Integer Representations towards Efficient Counting in the Bit Probe Model
Worst case: $n - 1$ reads, 3 writes



How to count from 0 to $2^n - 1$ using n bits?

[Gray1953] n reads, 1 write
[BGPS2014] $n - 1$ reads, 3 writes

Can we do even better?

Redundant case

[RM2010] M. Ziaur Rahman, J. Ian Munro. Integer Representation and Counting in the Bit Probe Model.

One extra bit for representation, worst case: $\log_2 n + O(1)$ reads, $O(1)$ writes

(one possible idea: increment n-bit normal binary counter over n steps with Gray-coded pointer and carry bit)

Previous lower bound

[FMS1997]
Gudmund Skovbjerg Frandsen and Peter Bro Miltersen and Sven Skyum
Dynamic word problems

Cannot write unread bits
Every bit needs to be written at some point
Reading at most k bits $\rightarrow$ reading $2^k - 1$ distinct bits in total

Lower bound $\log_2 n$ *upper bound* $n - 1$

The main claim

Lower bound $\log_2 n$, upper bound $n - 1$

Redundant case: $\log_2 n + O(1)$

The main claim

Lower bound $\log_2 n$, upper bound $n - 1$

Redundant case: $\log_2 n + O(1)$

This talk: non-redundant counter needs to read at least $\lfloor \frac{n}{2} \rfloor$ bits

Hypercube shuffling

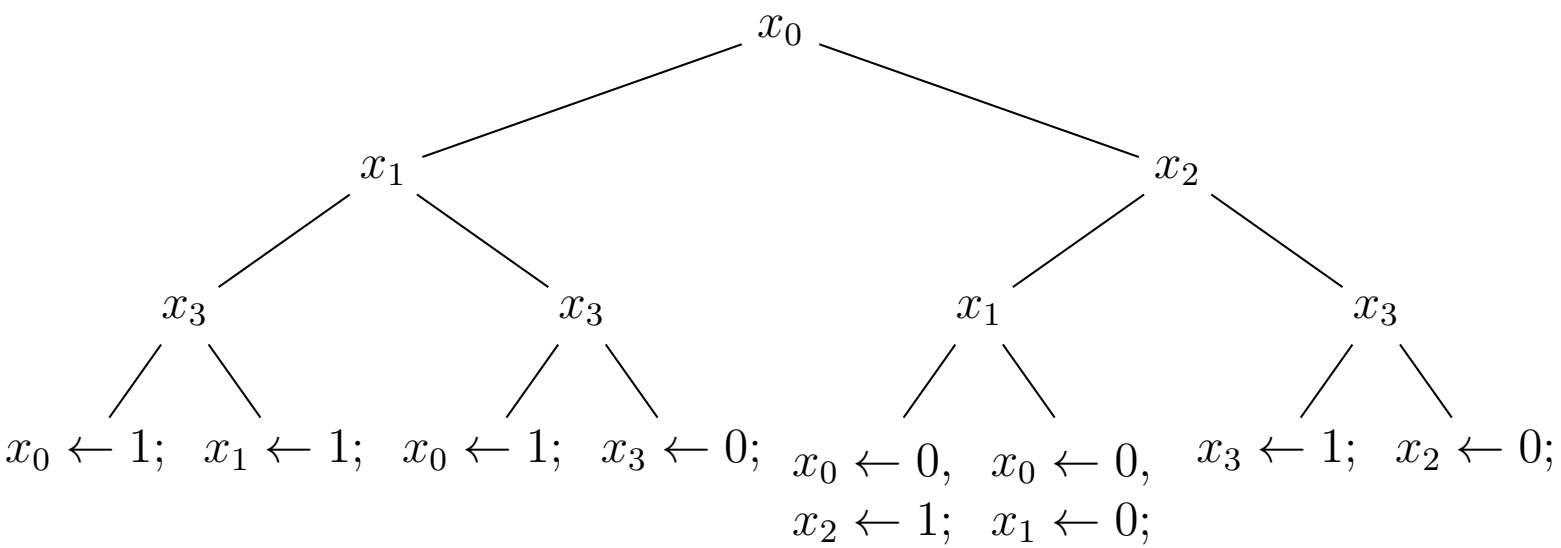
Code: a vertex of the n-dimensional hypercube

Fixing some bits: a block (a hypercube of lower dimension)

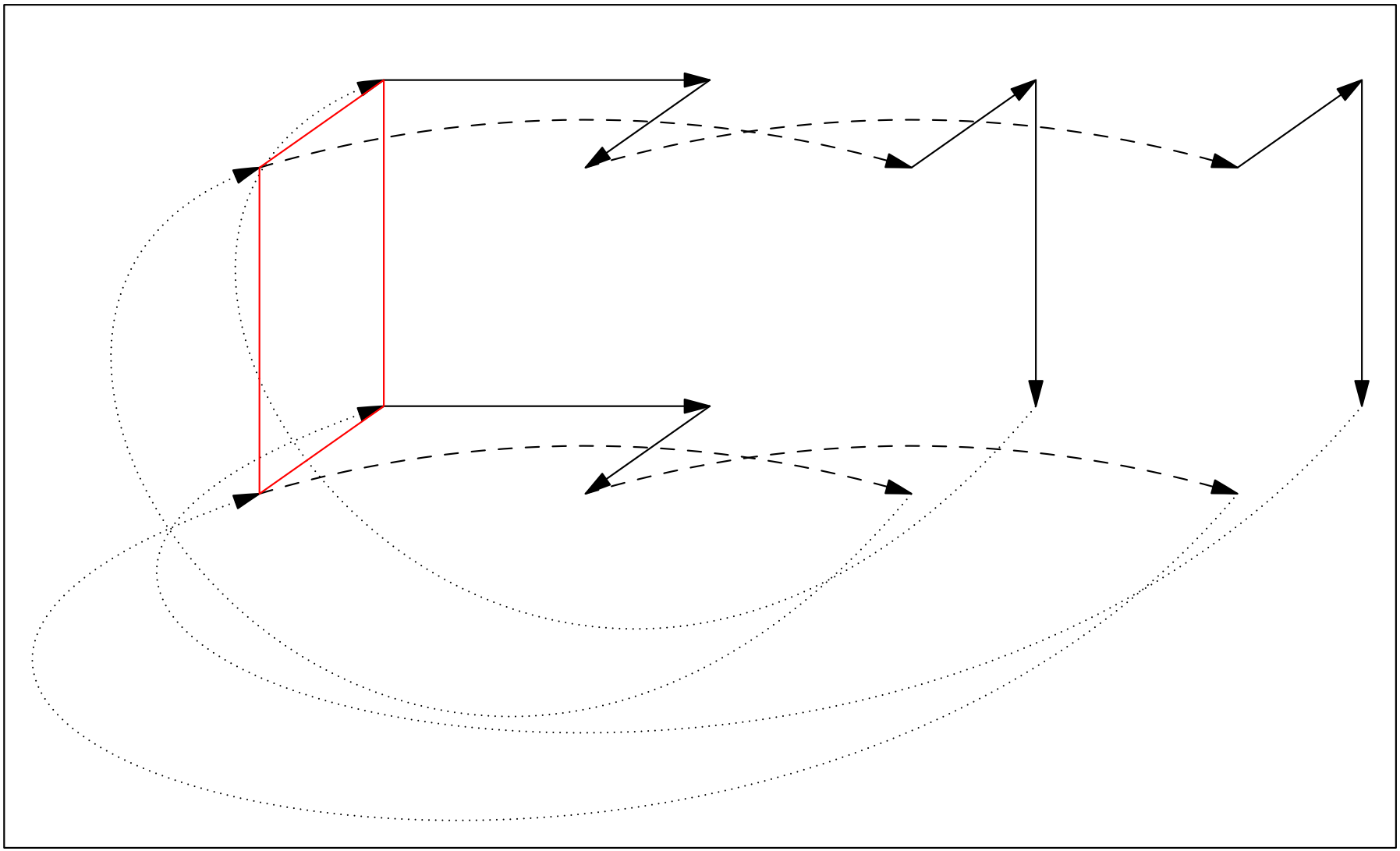
Changing fixed bits: parallel translation

The [BGPS2014] example

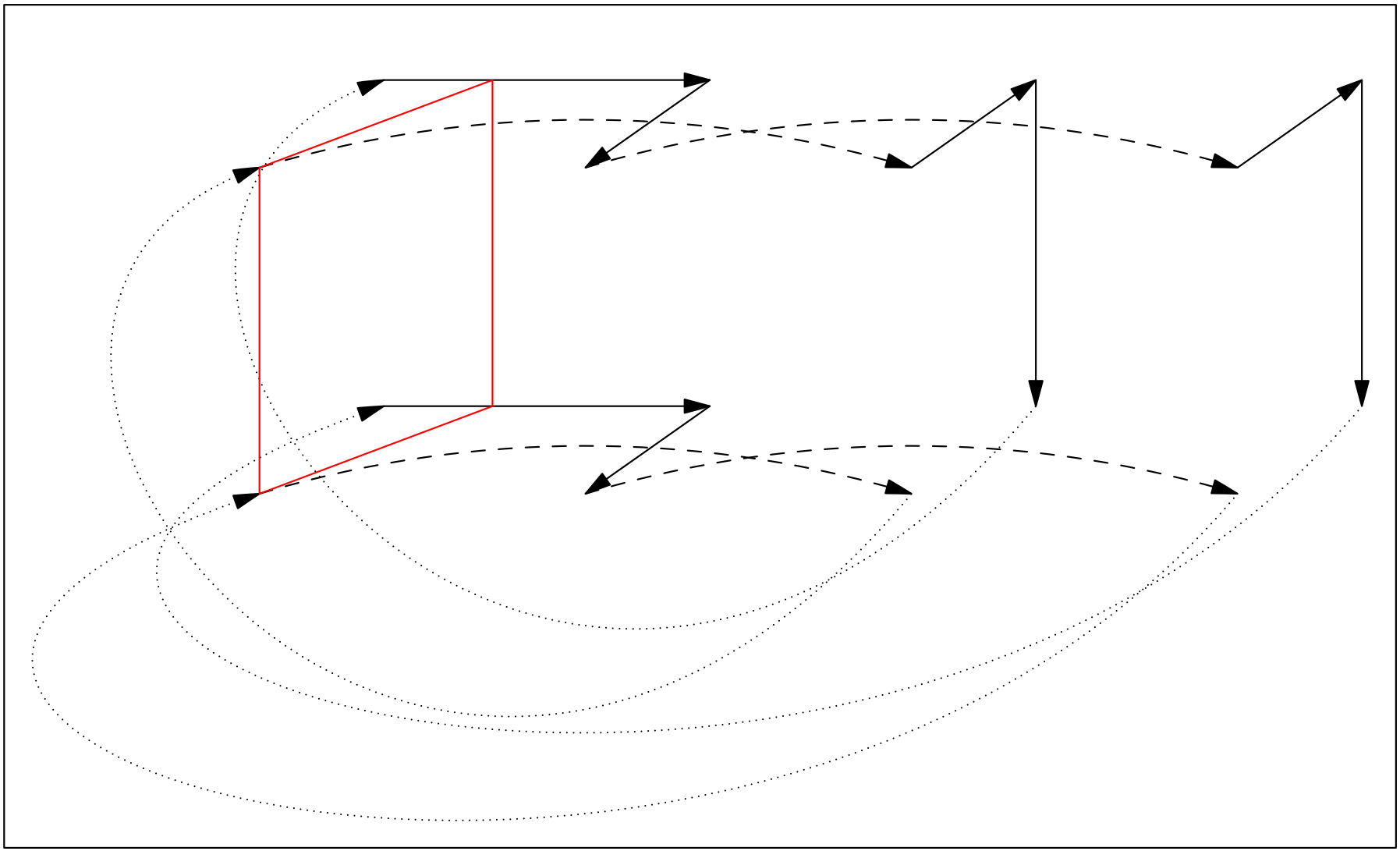
Initial representation: a decision tree



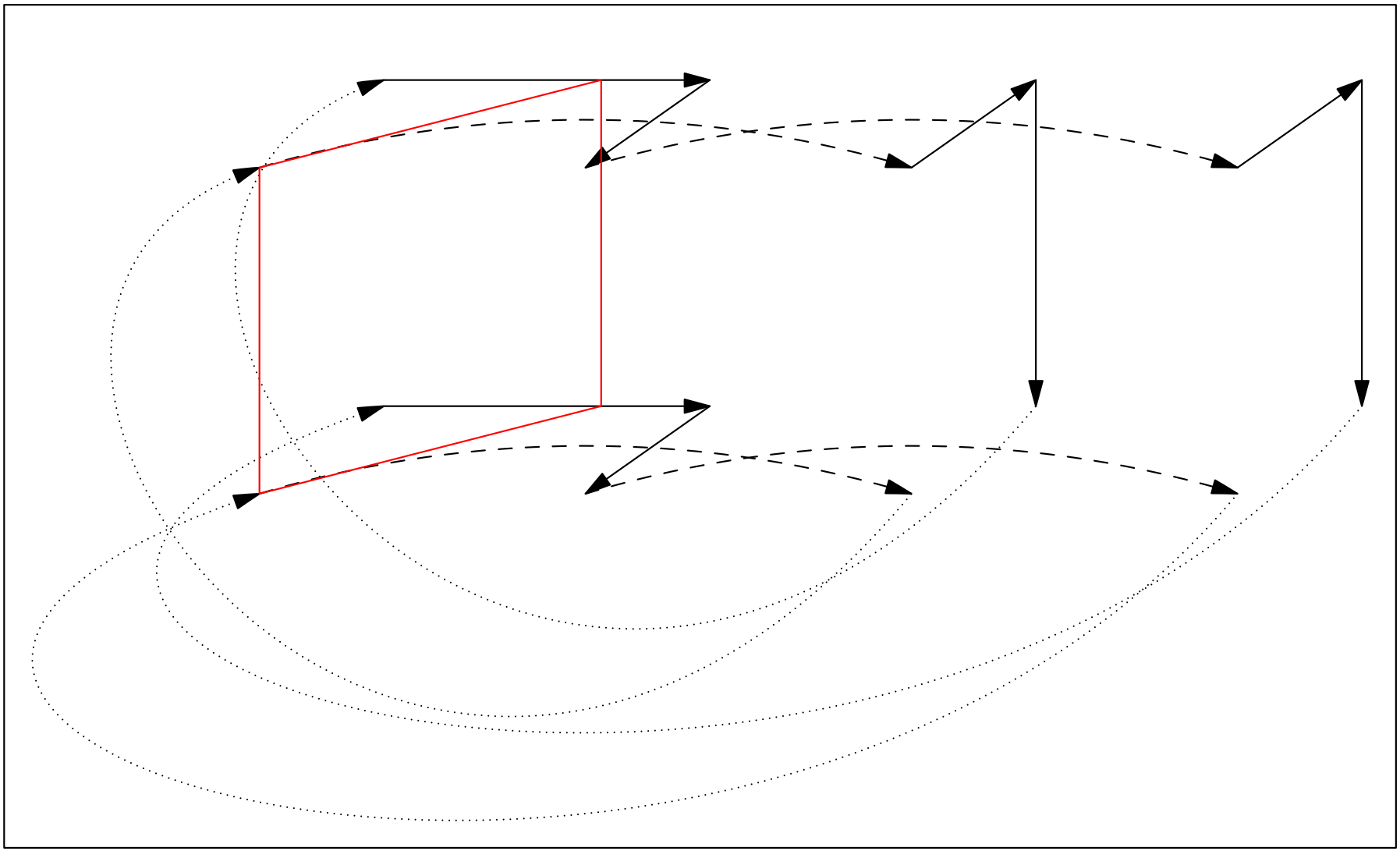
Hypercube shuffling and [BGPS2014]



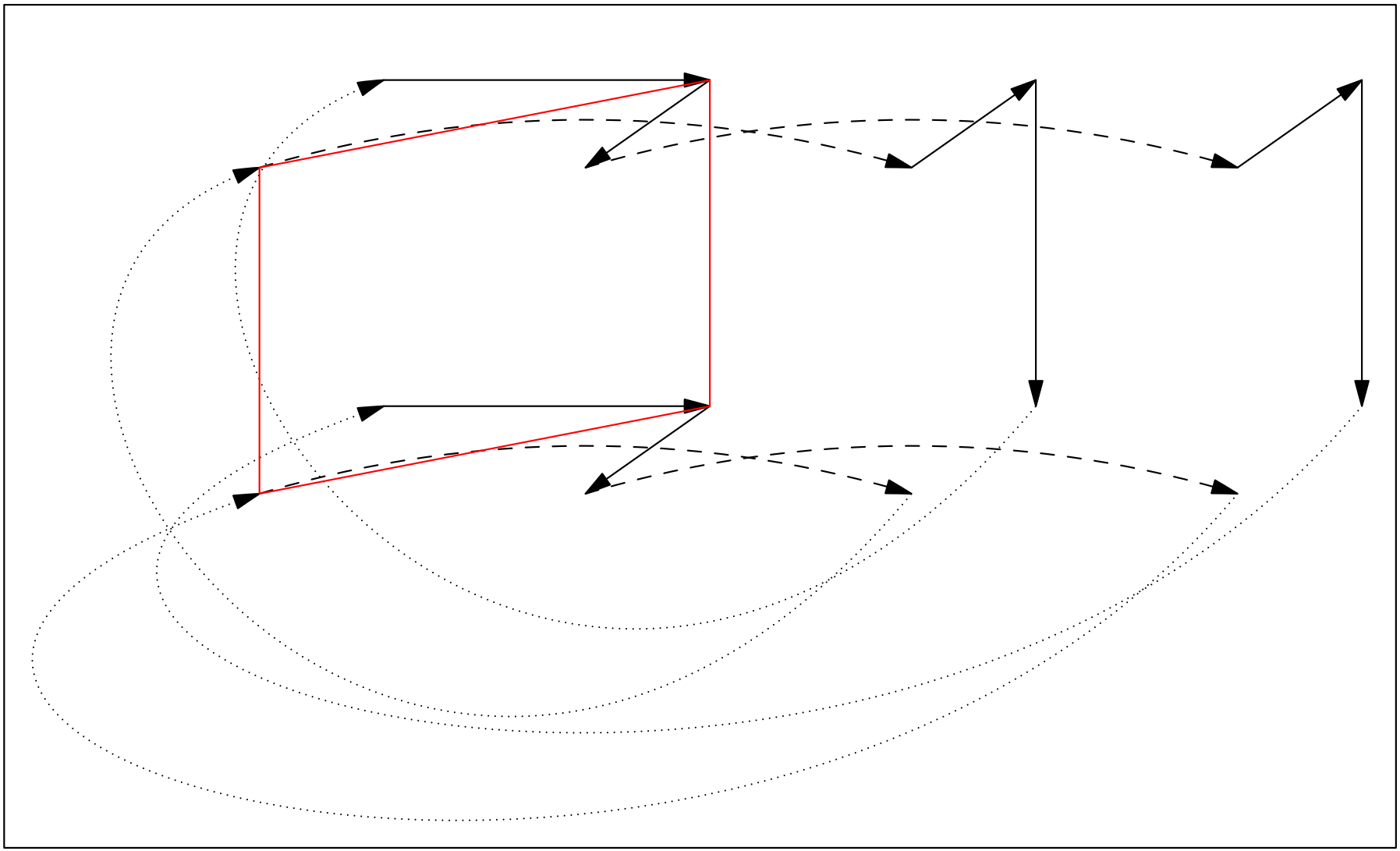
Hypercube shuffling and [BGPS2014]



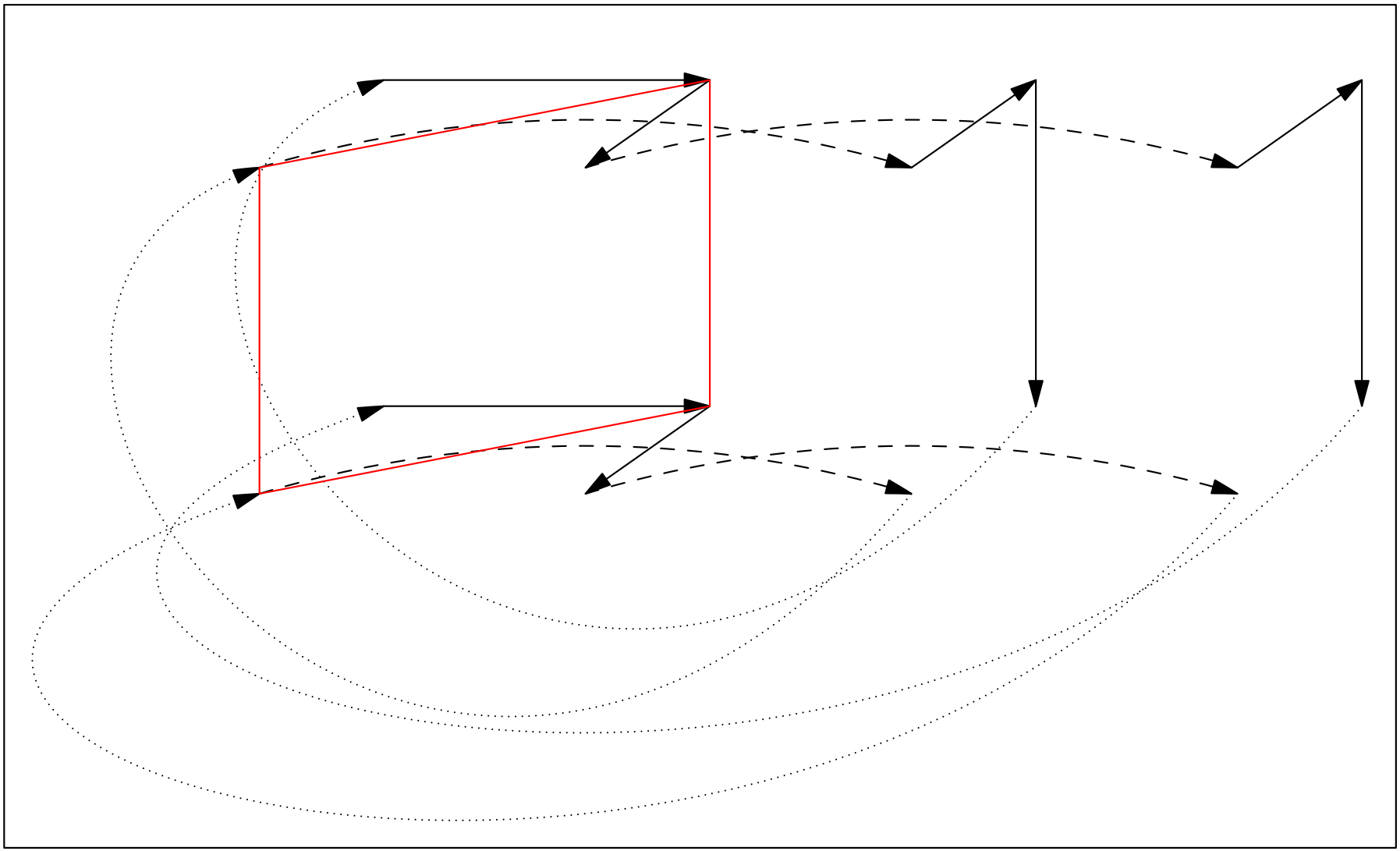
Hypercube shuffling and [BGPS2014]



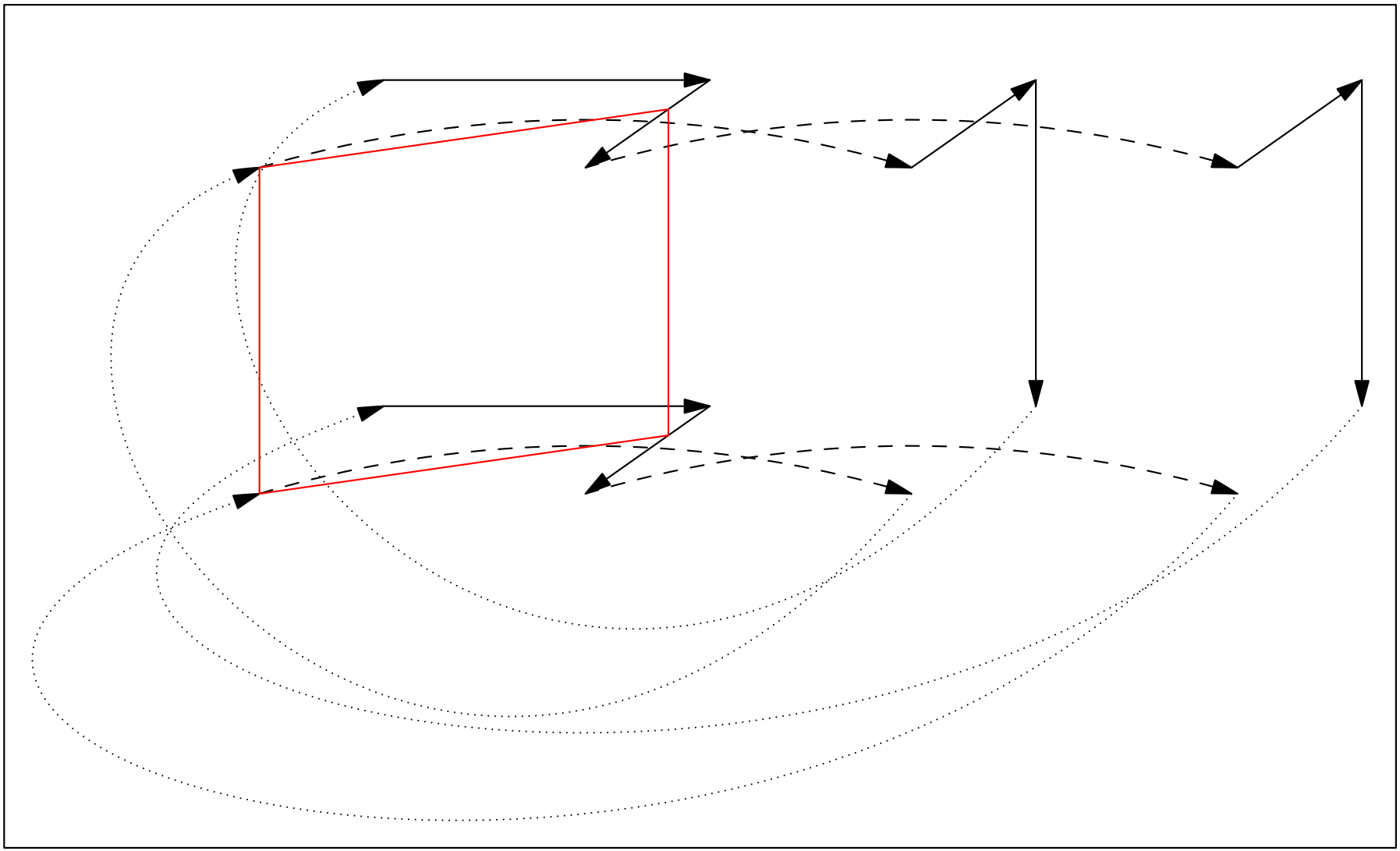
Hypercube shuffling and [BGPS2014]



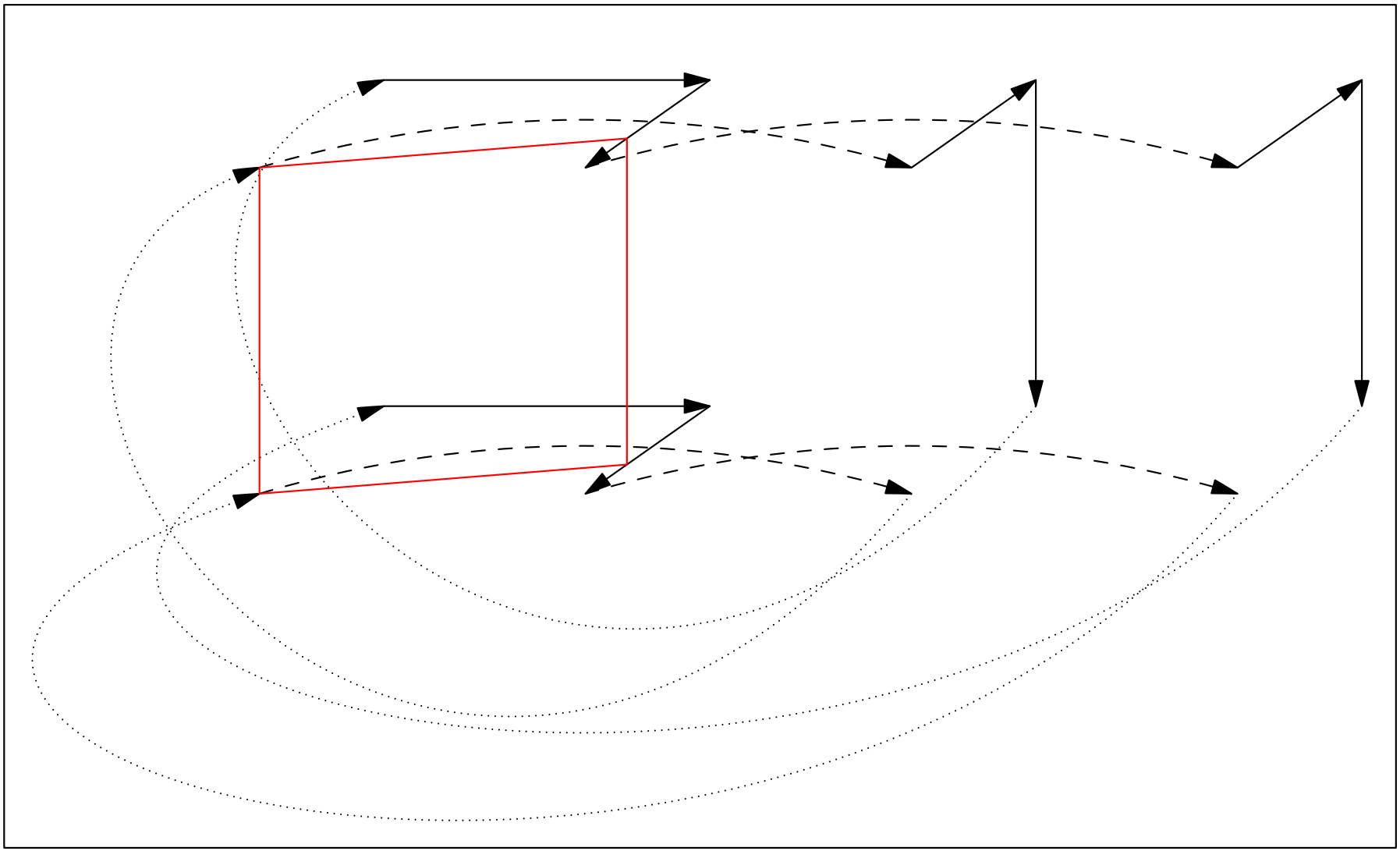
Hypercube shuffling and [BGPS2014]



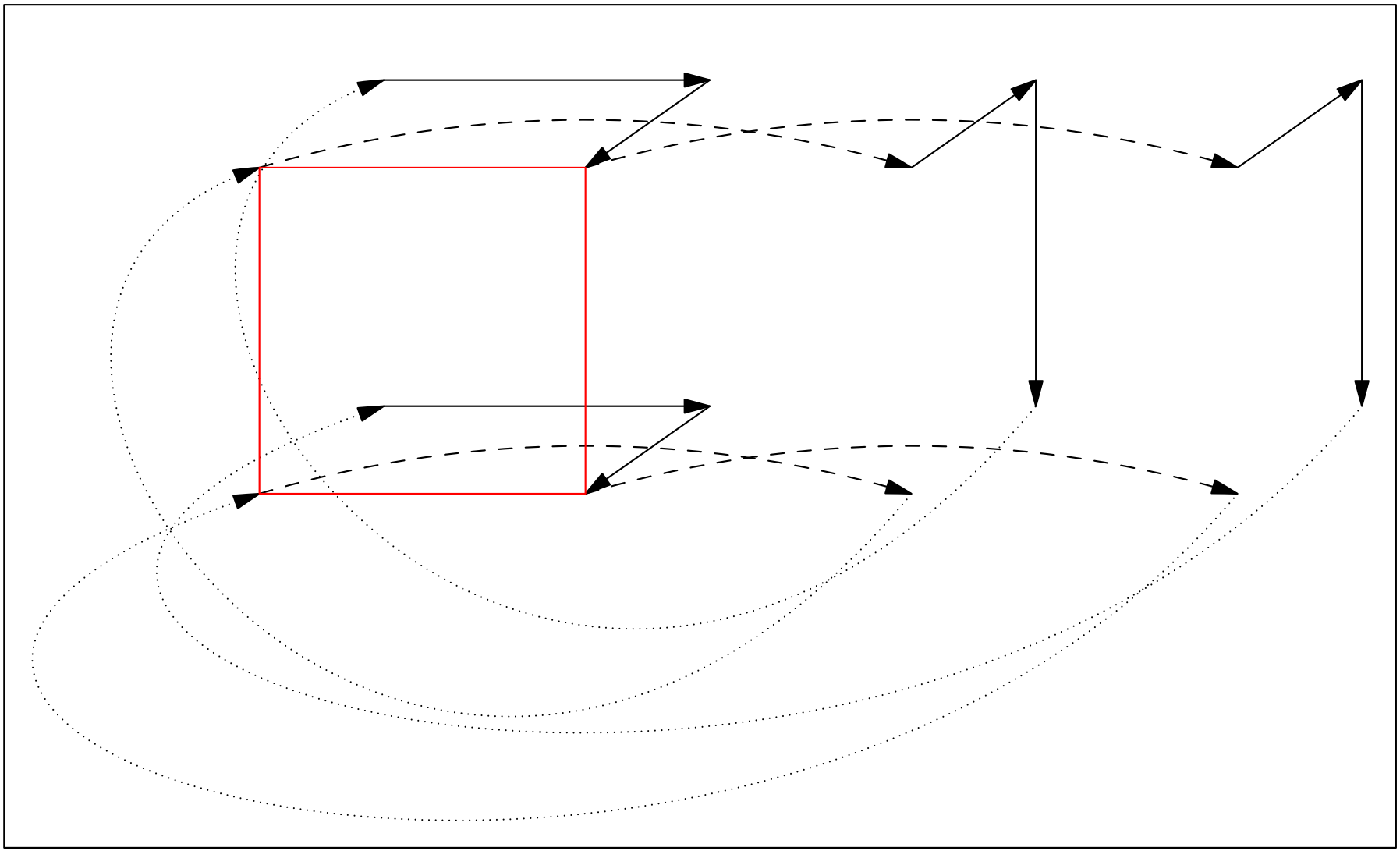
Hypercube shuffling and [BGPS2014]



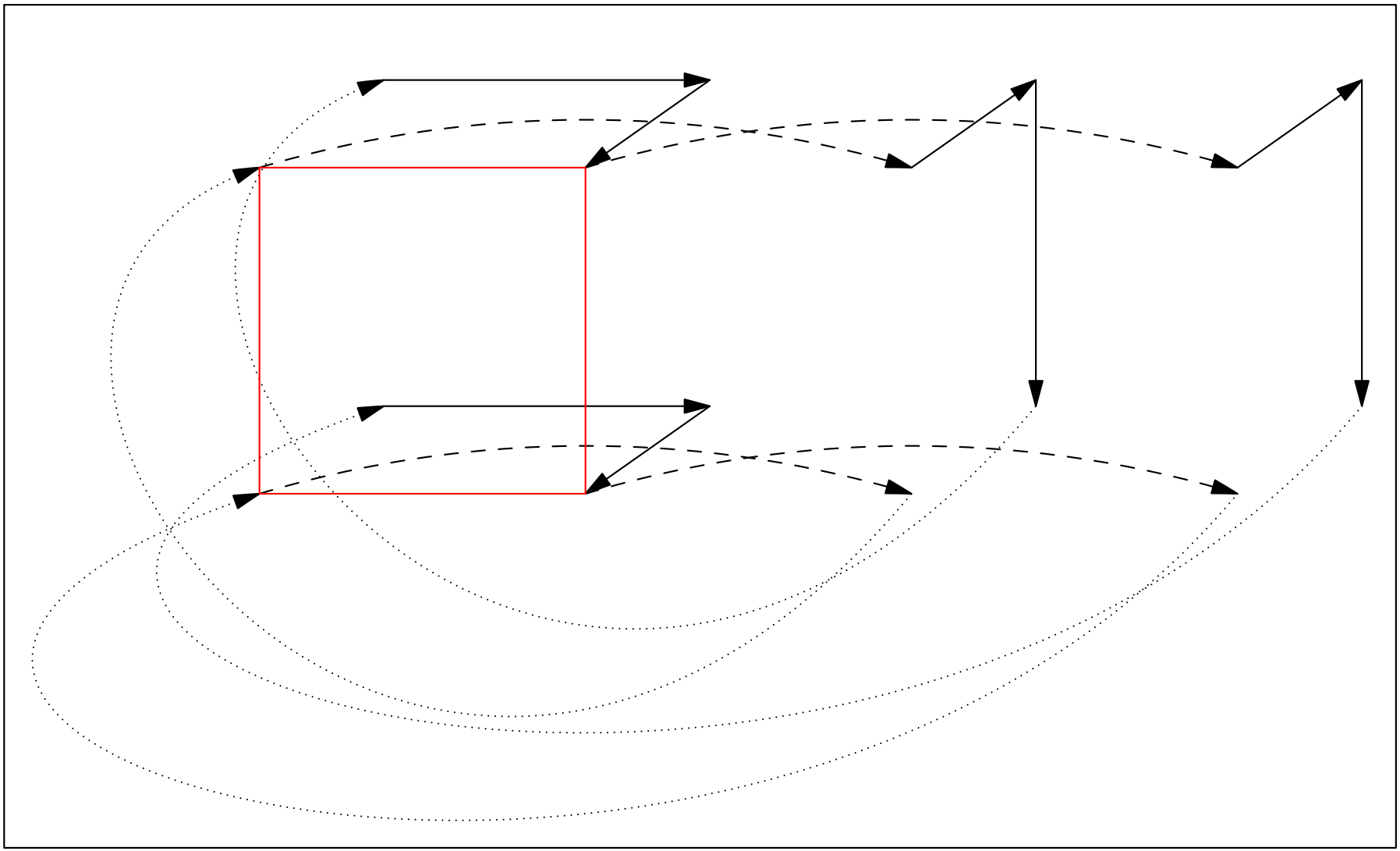
Hypercube shuffling and [BGPS2014]



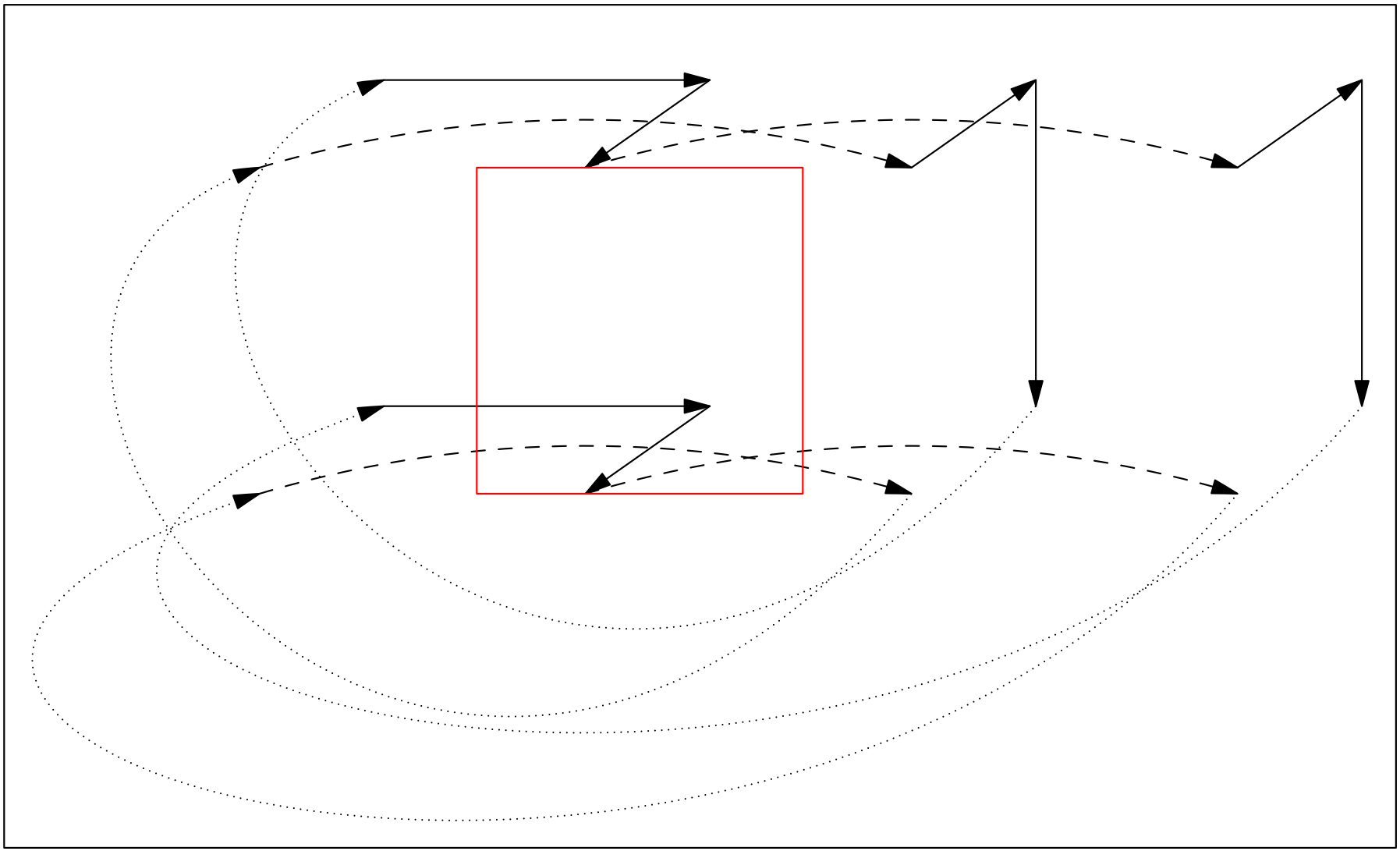
Hypercube shuffling and [BGPS2014]



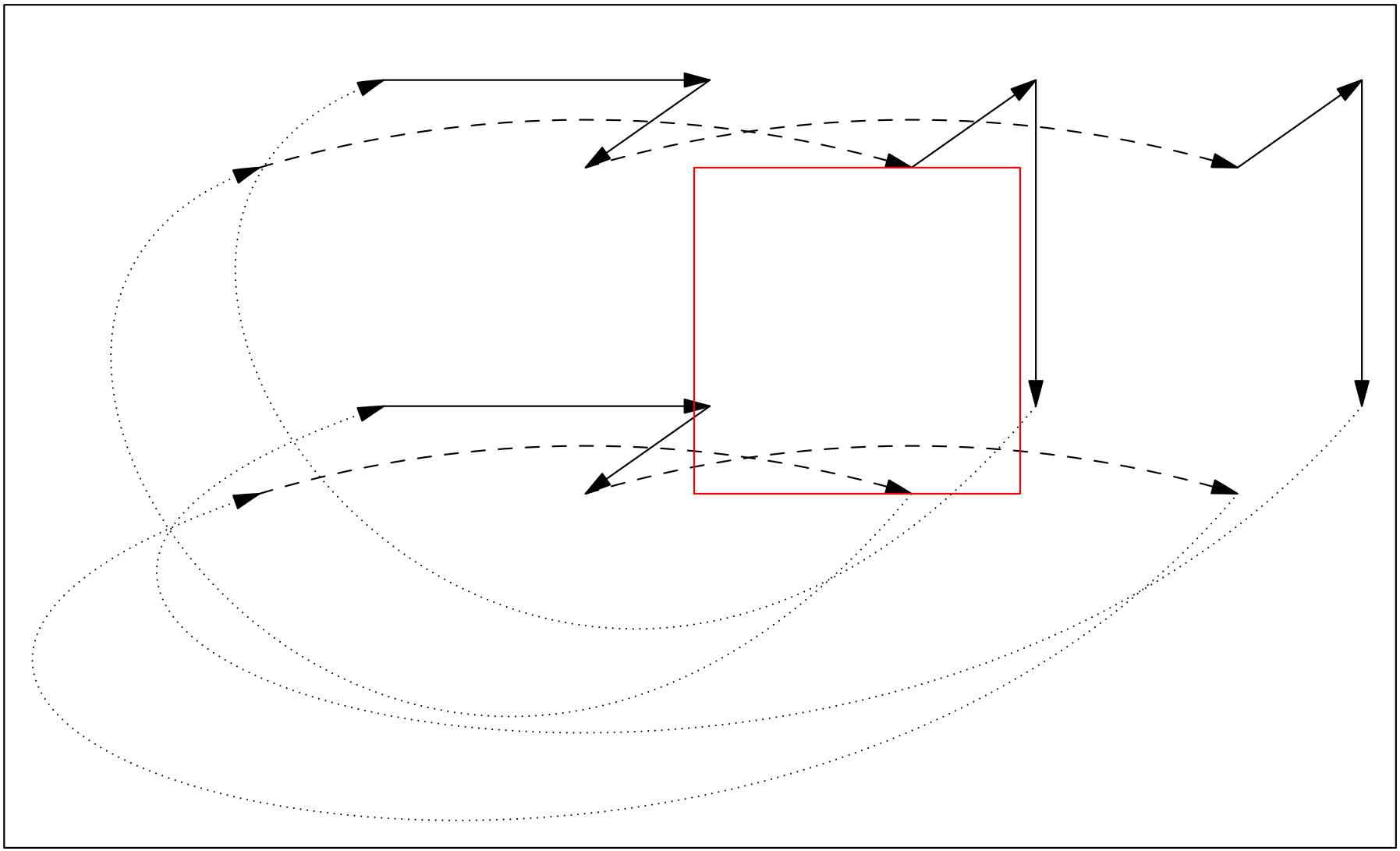
Hypercube shuffling and [BGPS2014]



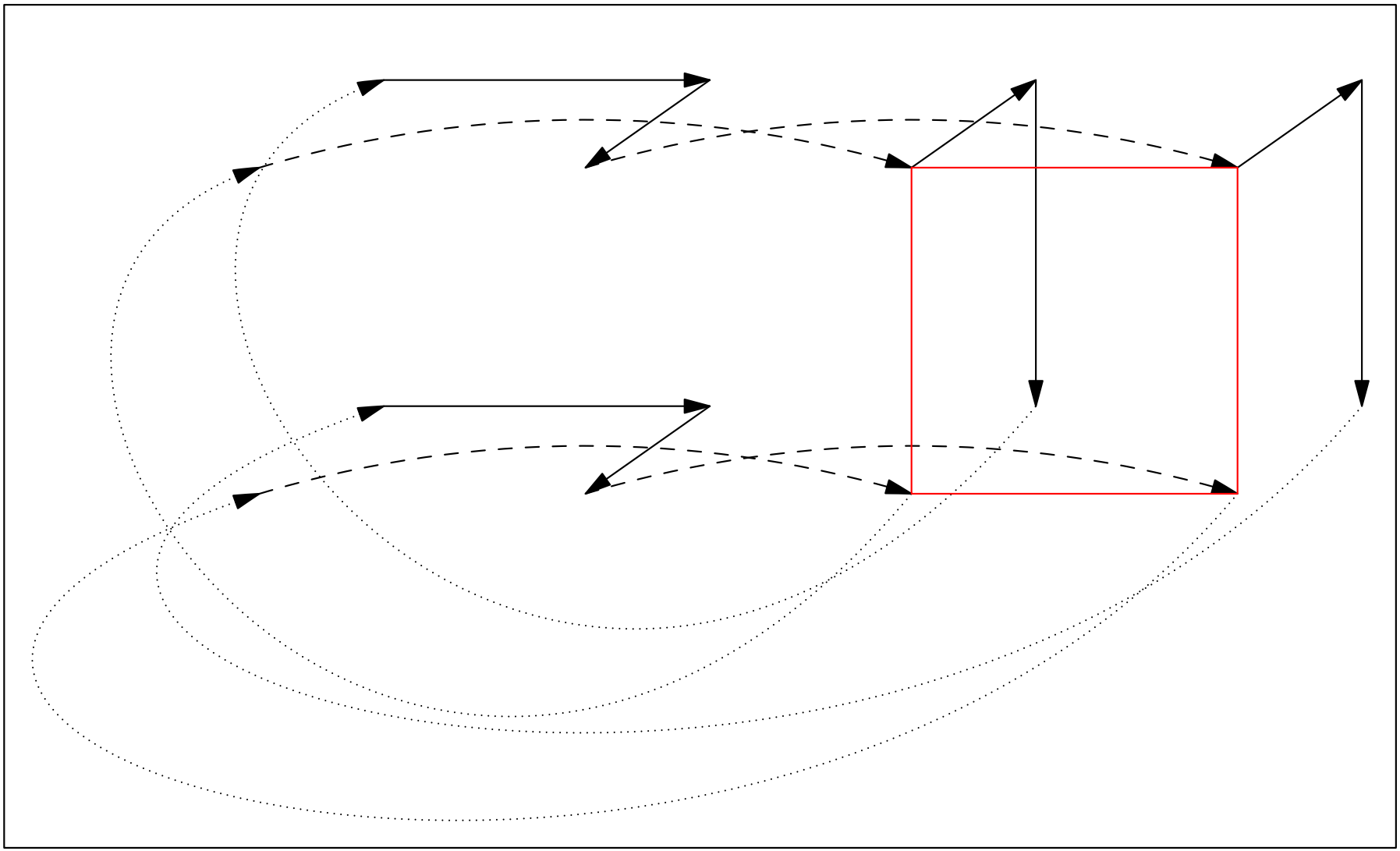
Hypercube shuffling and [BGPS2014]



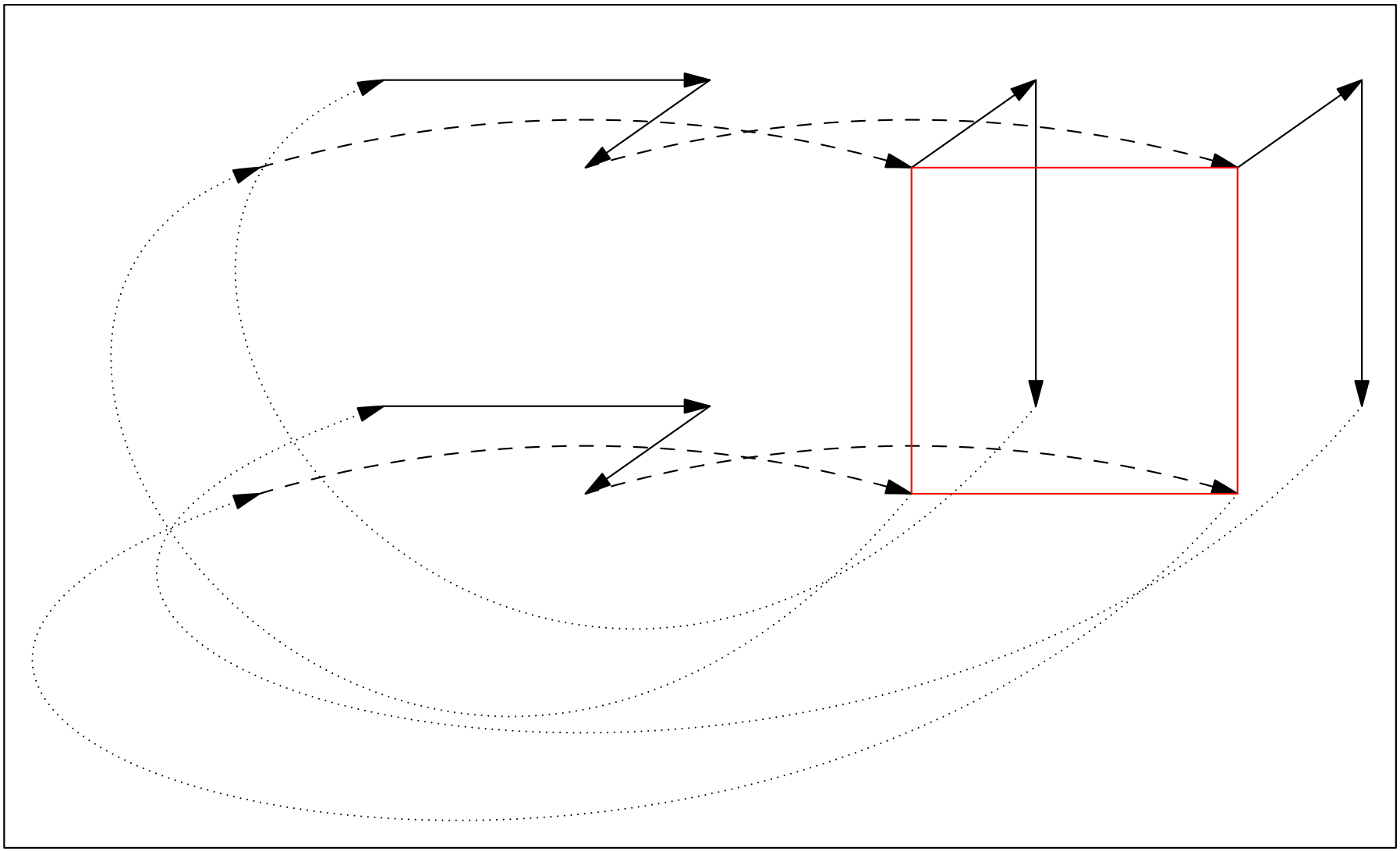
Hypercube shuffling and [BGPS2014]



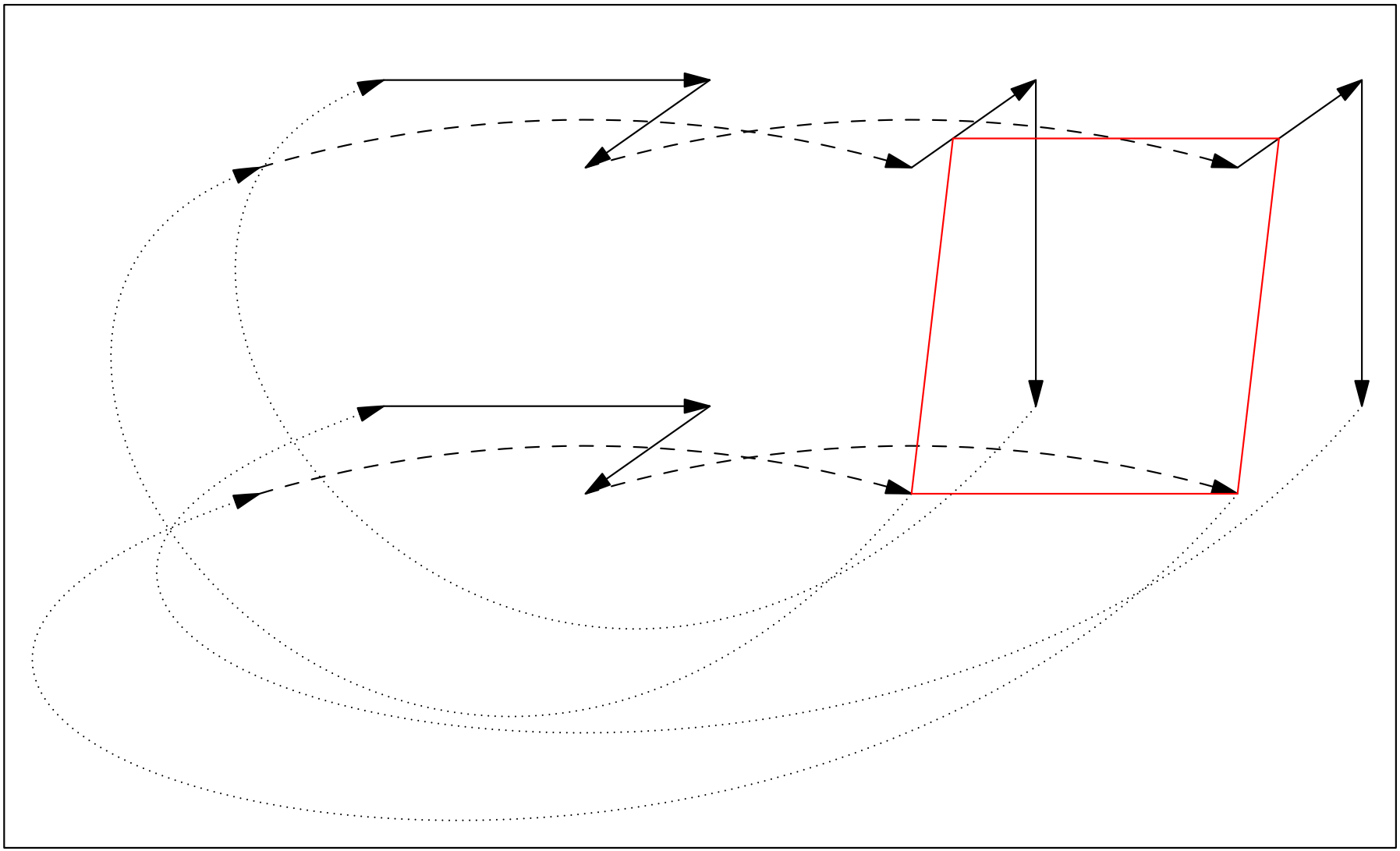
Hypercube shuffling and [BGPS2014]



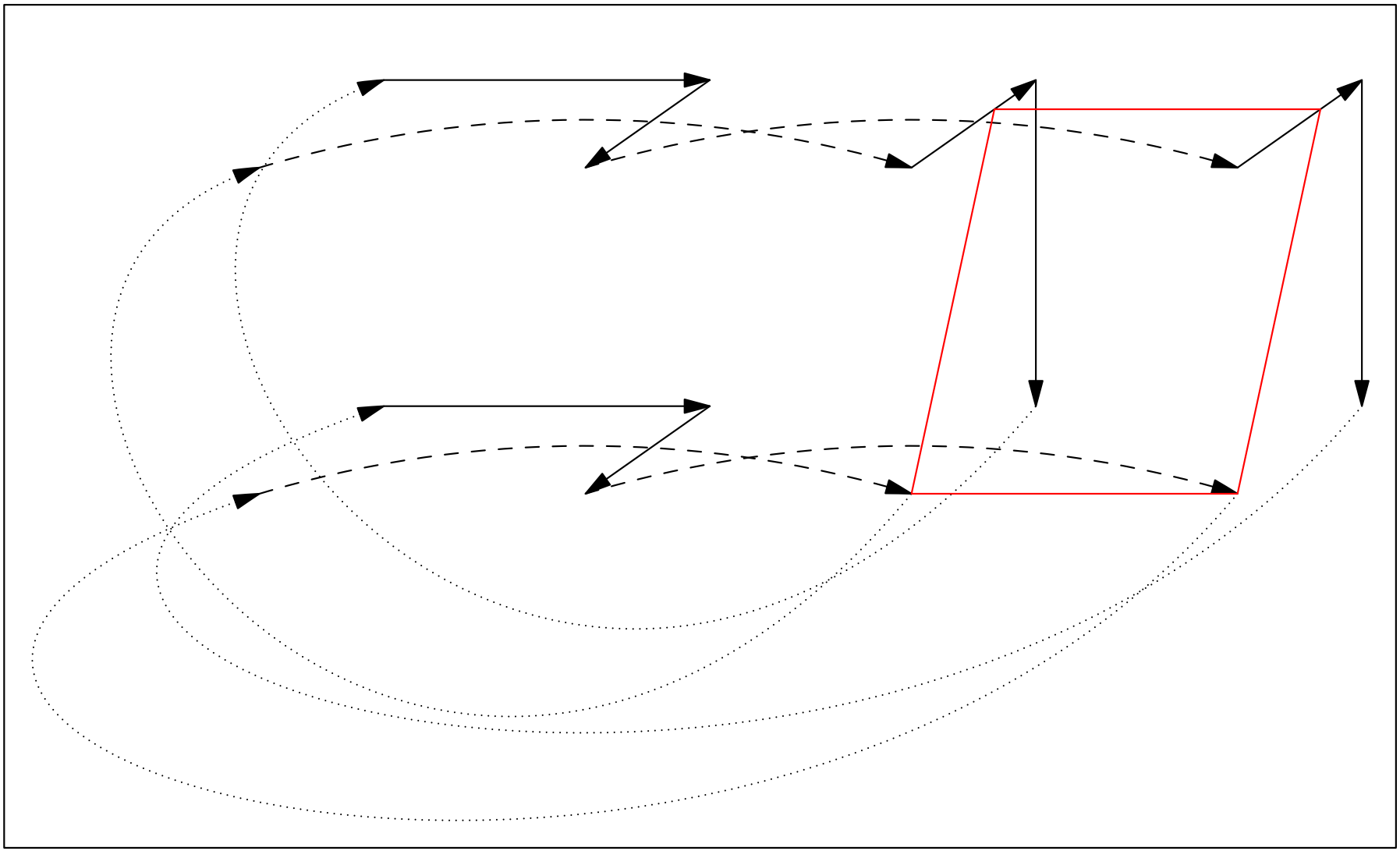
Hypercube shuffling and [BGPS2014]



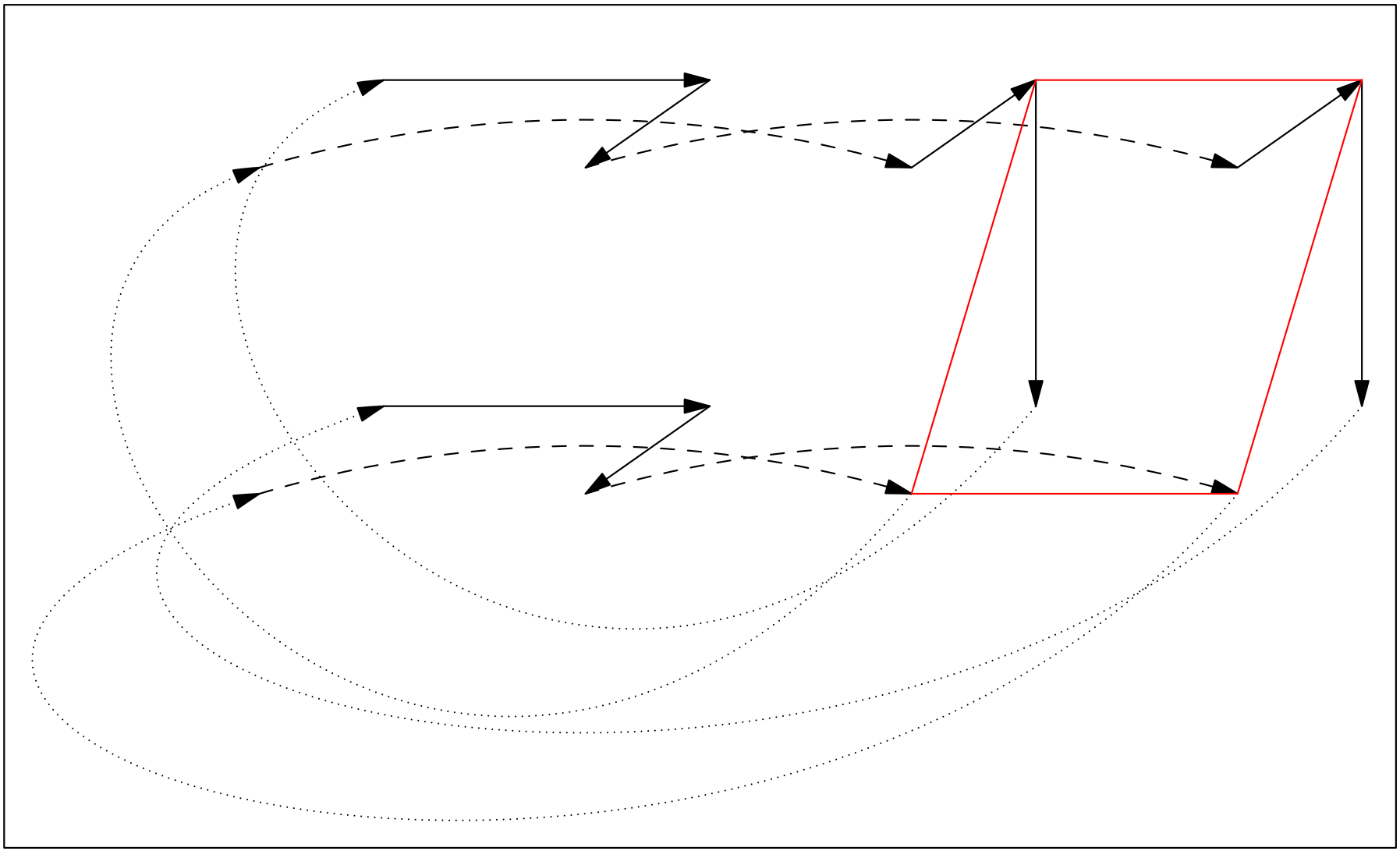
Hypercube shuffling and [BGPS2014]



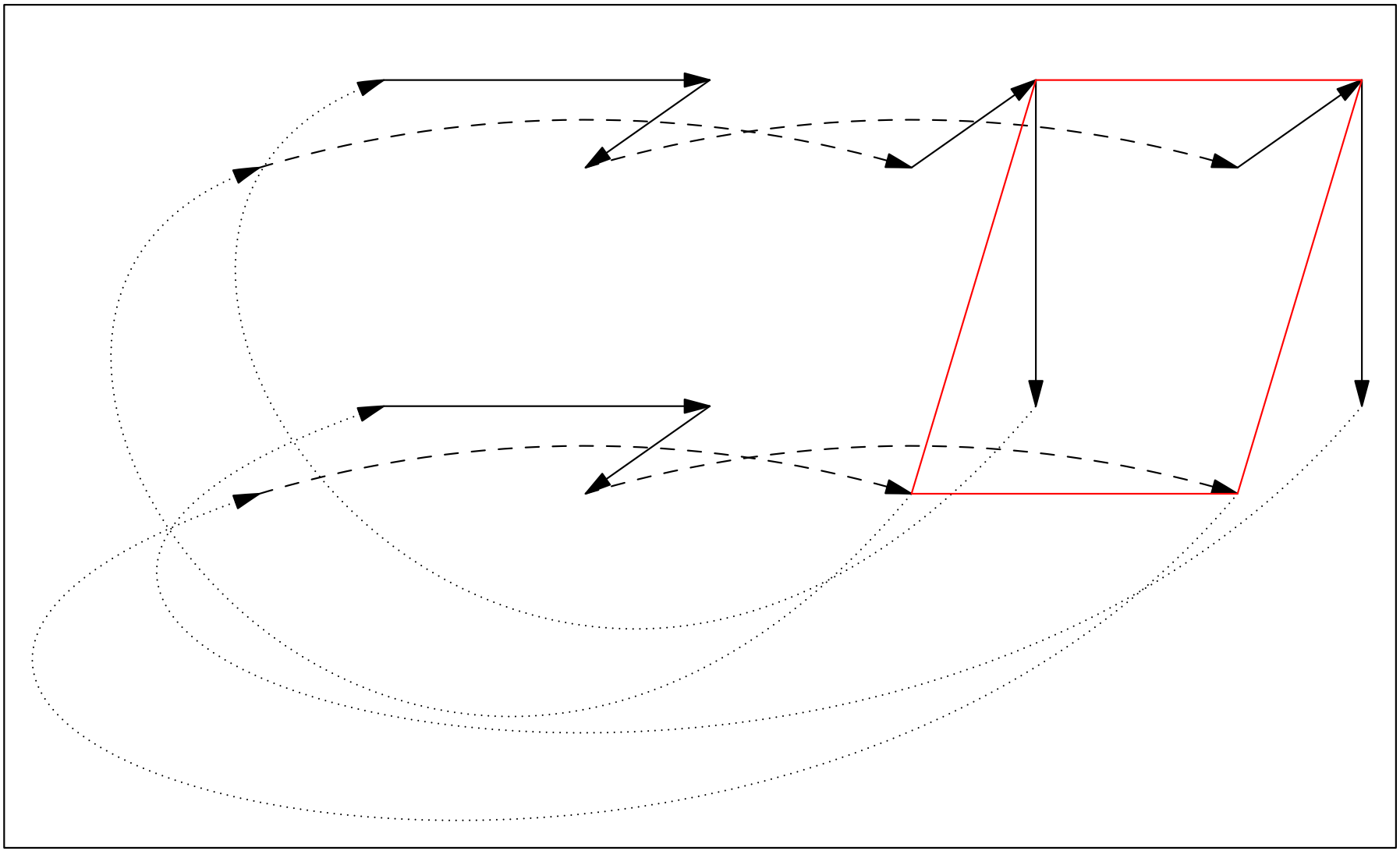
Hypercube shuffling and [BGPS2014]



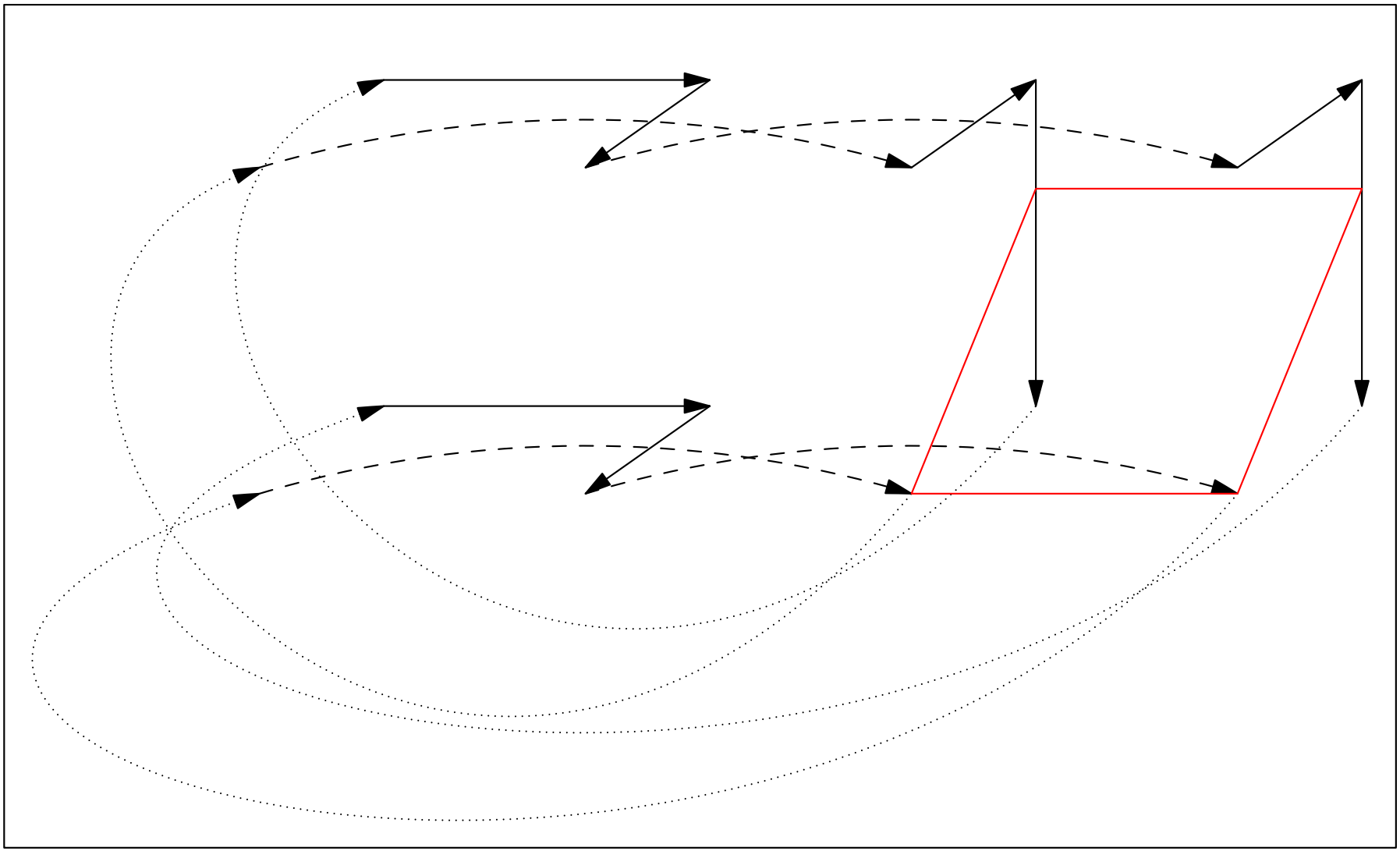
Hypercube shuffling and [BGPS2014]



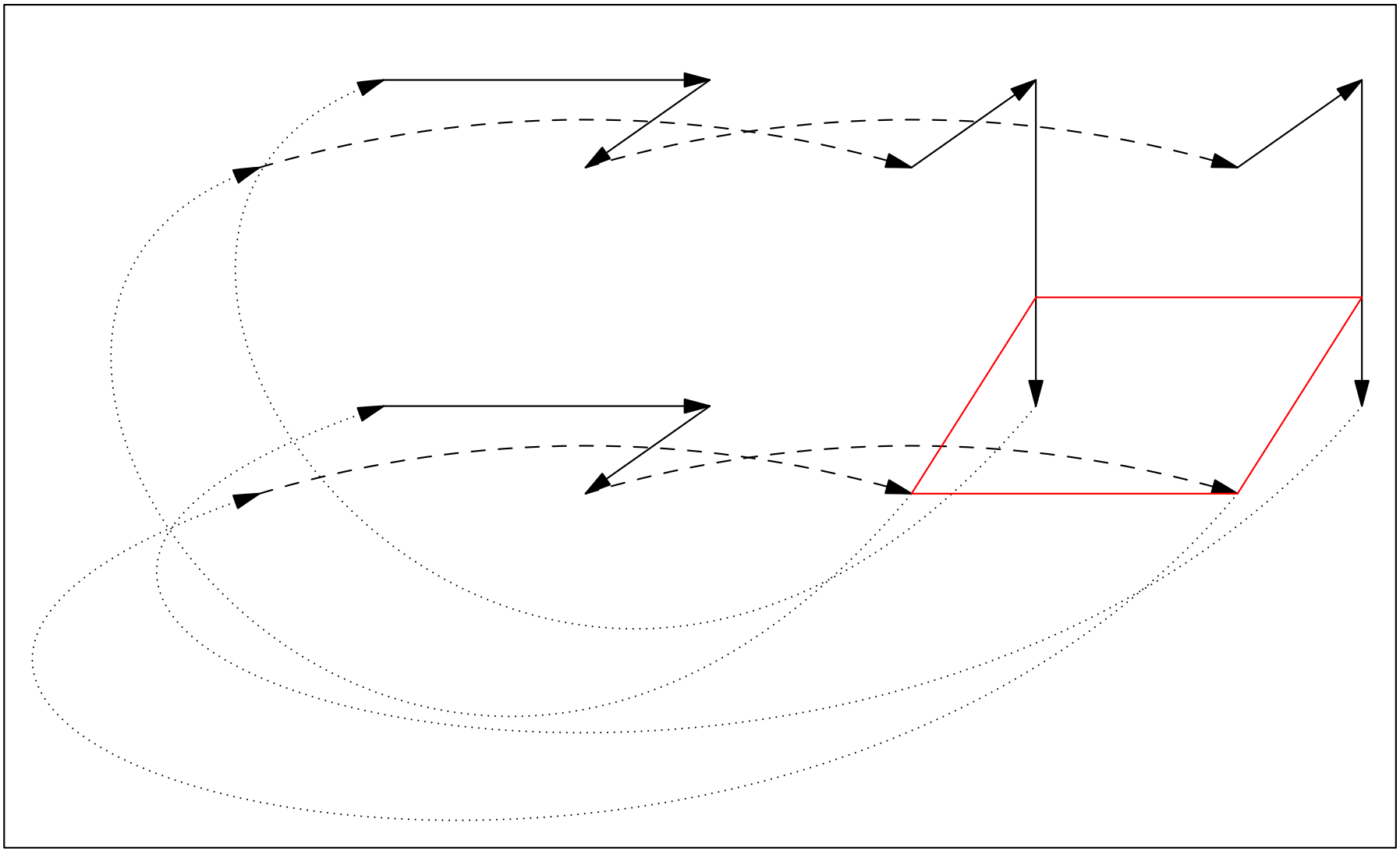
Hypercube shuffling and [BGPS2014]



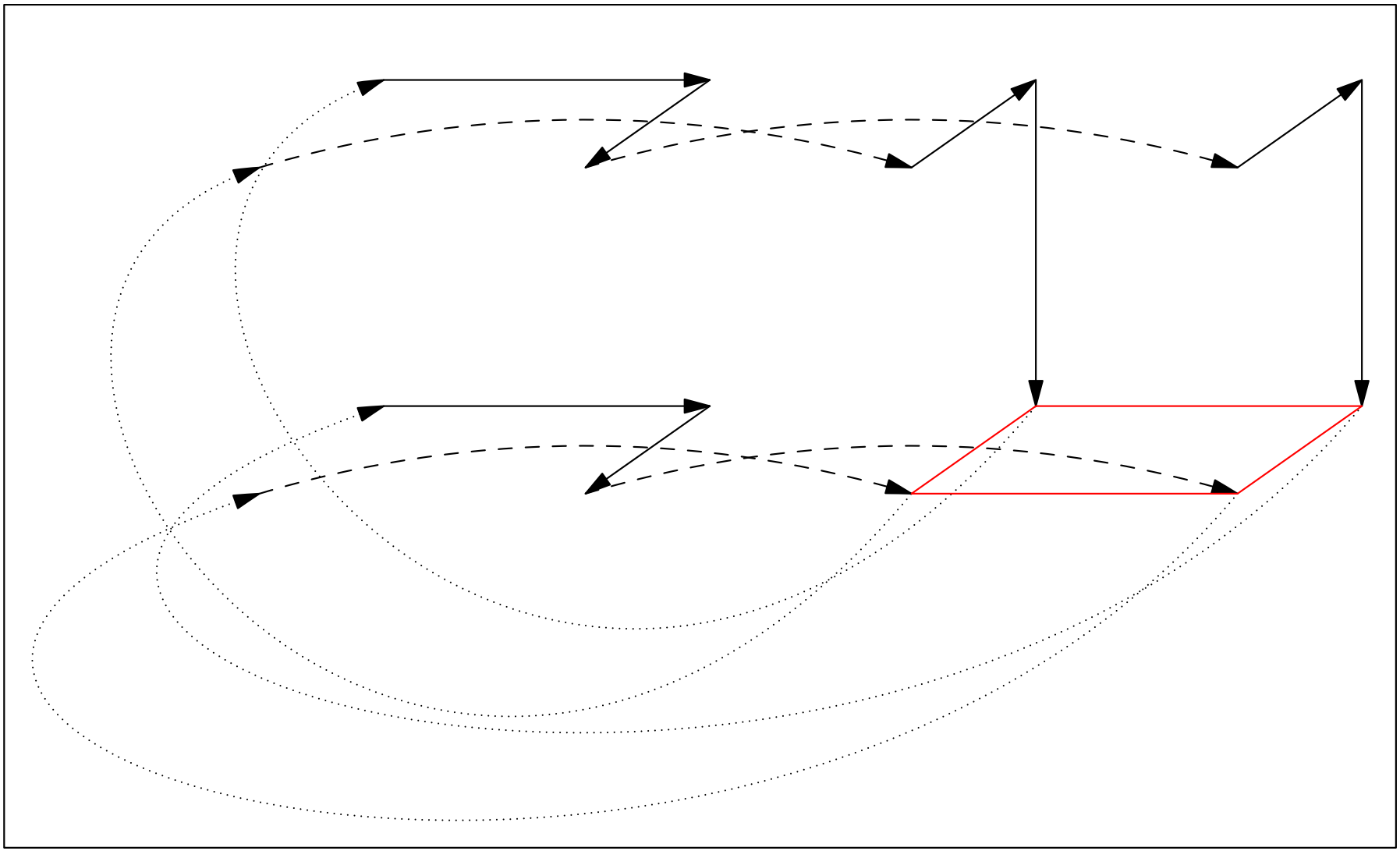
Hypercube shuffling and [BGPS2014]



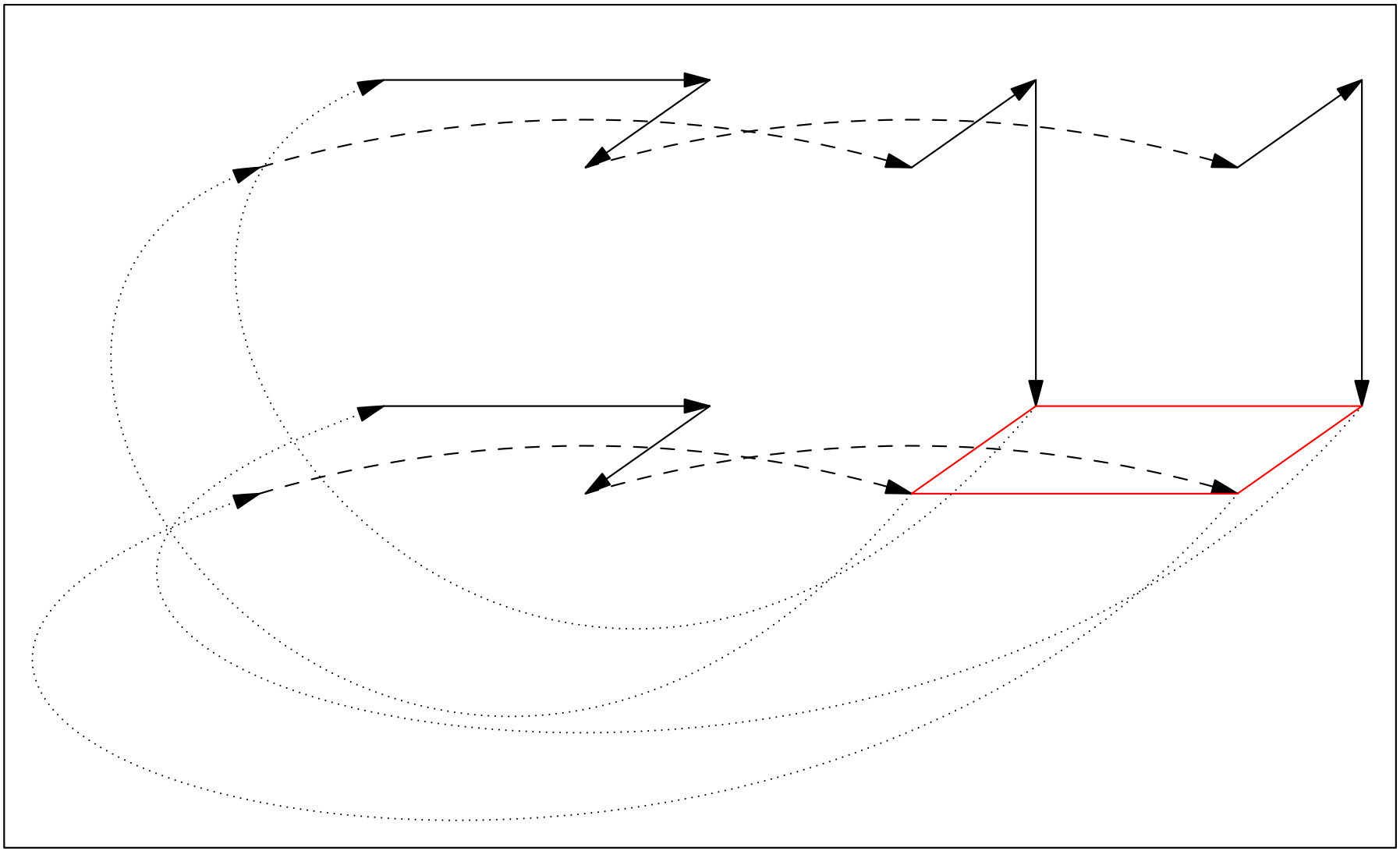
Hypercube shuffling and [BGPS2014]



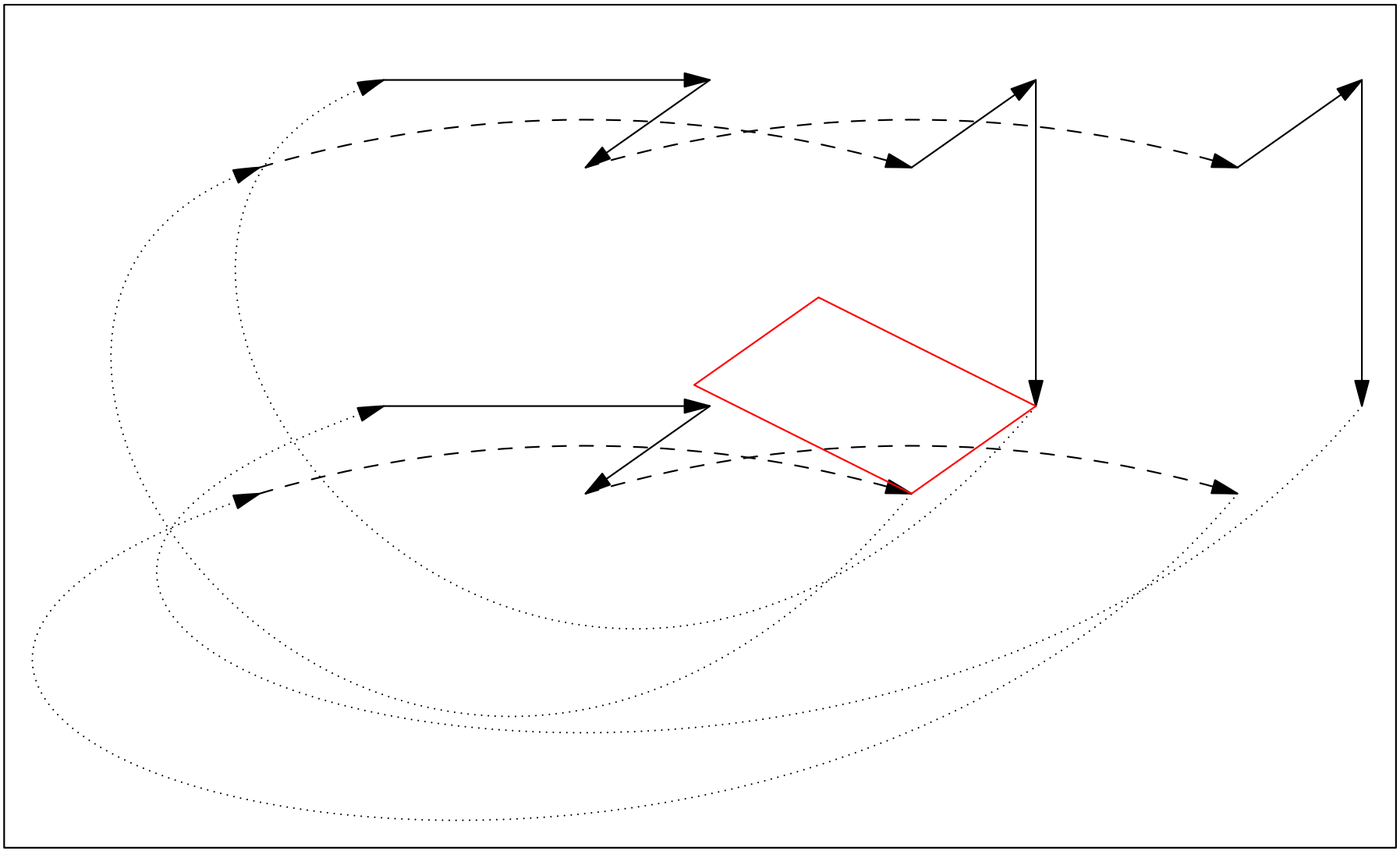
Hypercube shuffling and [BGPS2014]



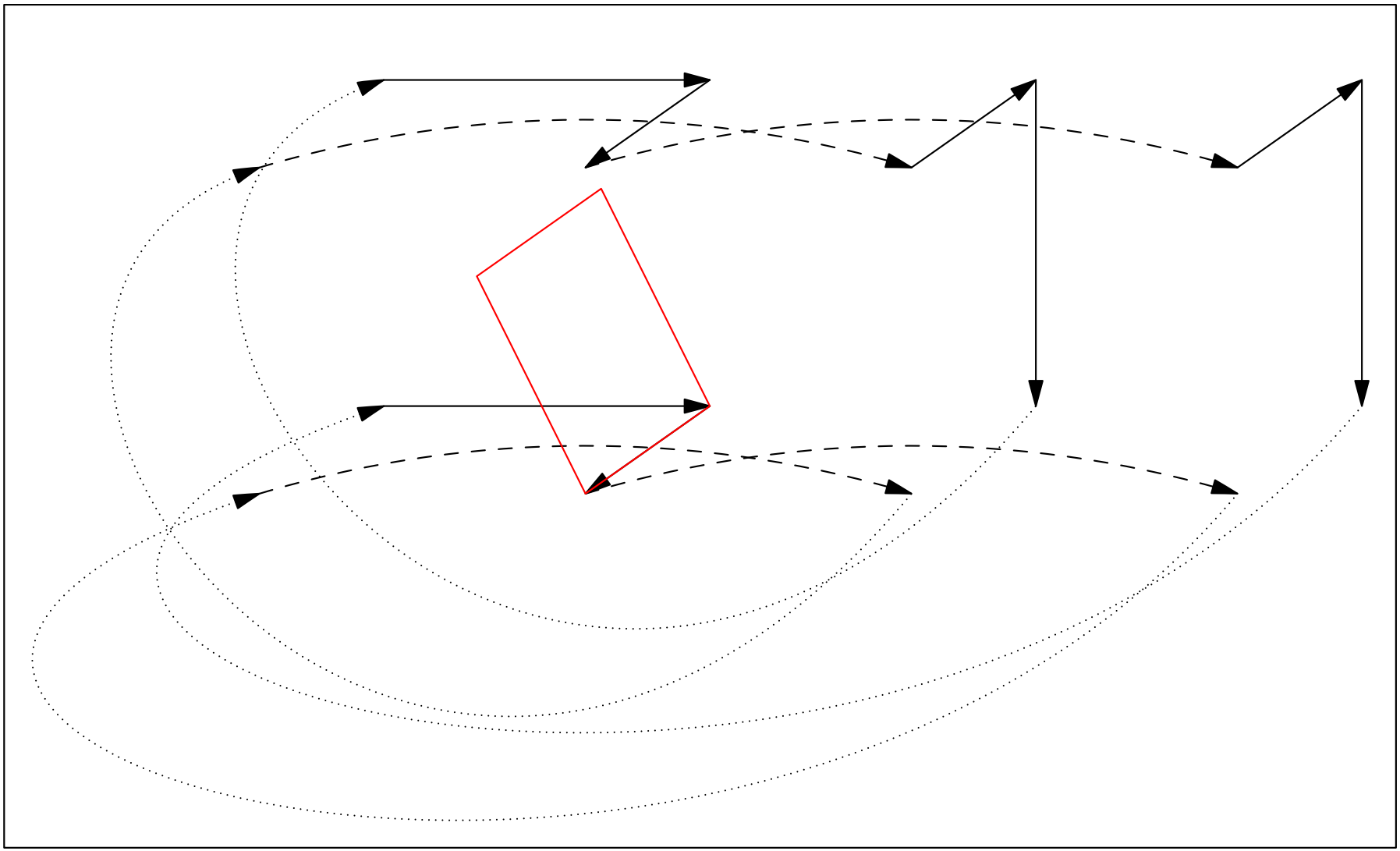
Hypercube shuffling and [BGPS2014]



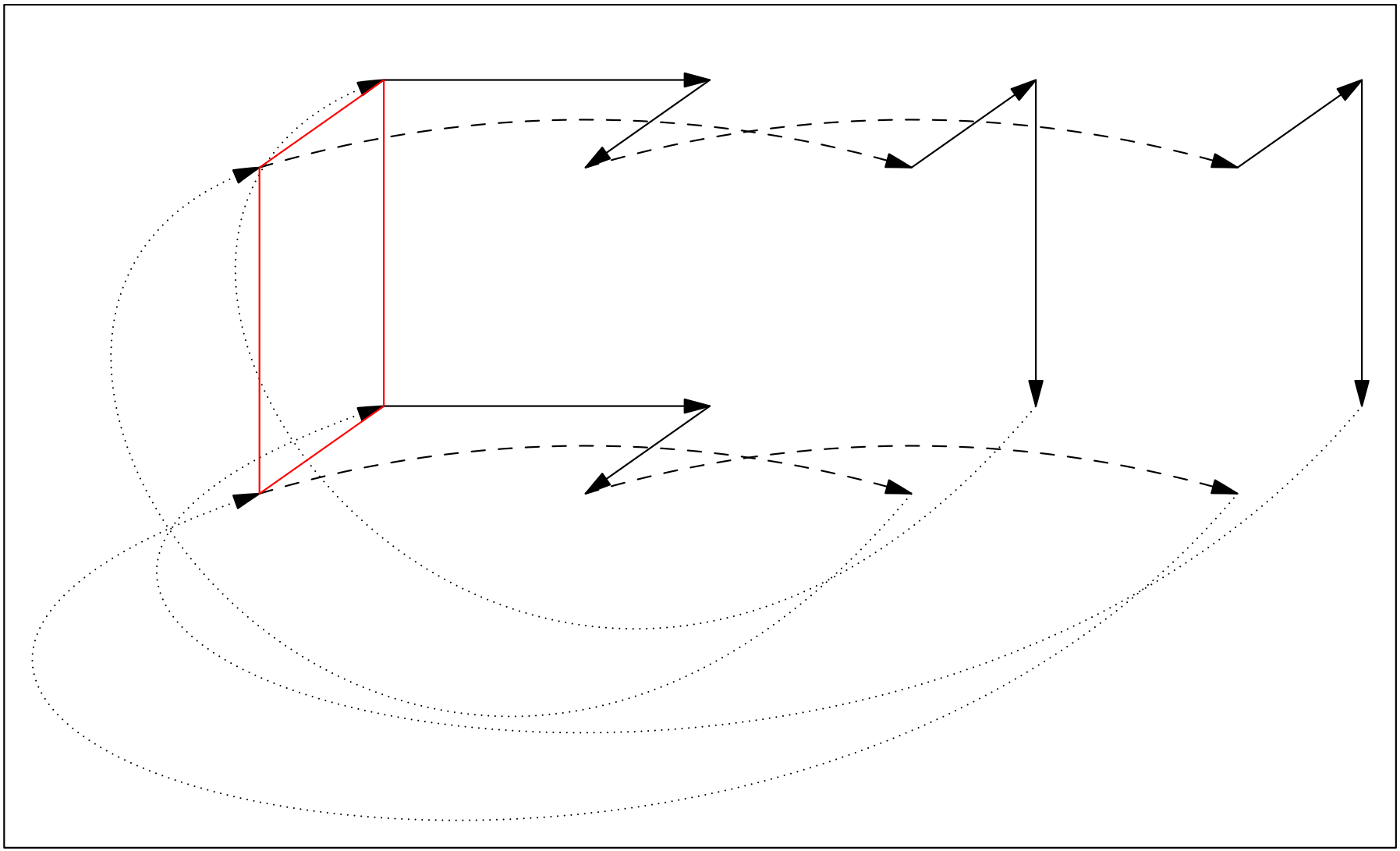
Hypercube shuffling and [BGPS2014]



Hypercube shuffling and [BGPS2014]



Hypercube shuffling and [BGPS2014]



Hypercube shuffling as a permutation

Increment is a function
$0 \to 1 \to \ldots \to 2^n - 1 \to 0$
Permutation *inc* on codes; a cycle of length 2^n

Lexicographical order of codes (first bit is least significant bit)
Inversion: $x < y$, $\sigma(x) > \sigma(y)$
Permutation sign: parity of total number of inversions
Permutation *inc* on codes; a cycle of length 2^n; and odd permutation

Enough to prove: shifting $\lceil \frac{n}{2} \rceil + 1$-dimensional blocks is even permutation

Hypercube enumeration

Leaves of decision tree — blocks in hypercube, a partition

Two partitions: before and after the translations

Enumerating hypercube vertices: lexicographically; block-by-block, lexicographically inside block

Blocks can be before or after shift

Hypercube enumeration

Before(k): code of k-th vertex in enumeration block-by-block before translations

After(k): code of k-th vertex in enumeration block-by-block after translations

$inc : \mathit{Before}(k) \mapsto \mathit{After}(k)$.

Needed: $inc = \mathit{After} \circ \mathit{Before}^{-1}$ is even unless $\lfloor \frac{n}{2} \rfloor$ bits read in the worst case.
Enough to show *Before* and *After* are even when we read fewer than $\lfloor \frac{n}{2} \rfloor$ bits

Blocks and the inversions

Inside the same block: monotonous enumeration, no inversions

Different blocks A, B: need parity of number of pairs
$(x, y) : x > y, x \in A, y \in B$

Total number is even $\rightarrow$ block order is irrelevant

Two blocks have $< \lfloor \frac{n}{2} \rfloor$ fixed coordinates each?
They share $\geqslant 2$ free coordinates

Inversions in similar pairs of codes

At least 2 common free positions.
p: most significant common free position

Compare codes in the pair; classify by most significant position where codes differ

Case 1. More significant than p, involution: flip both at p
Case 2. Less significant than p, equality at p, involution: flip both at p
Case 3. Exactly p, involution: flip at less significant free position

Putting things together

For each block there are no inversions inside the block.
For each two blocks, there is an even number of inversions between the two blocks.
Therefore, the total number of inversions is even, and the permutation (*Before* or *After*) is even.

Therefore $inc = After \circ Before^{-1}$ is an even permutation, particularly, not a cycle of length 2^n.

Therefore we need to read at least $\lfloor \frac{n}{2} \rfloor$ bits in the worst case.

Thanks for your attention.

Questions?