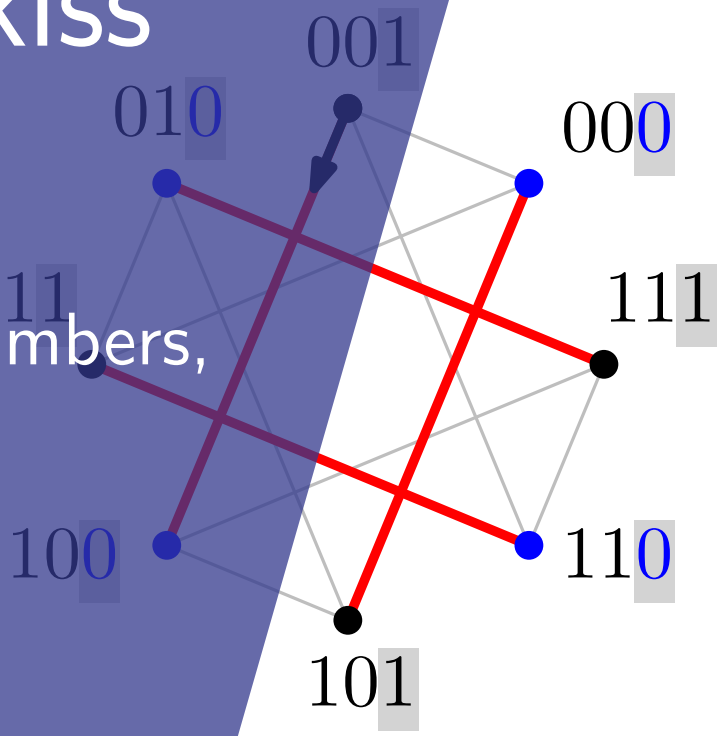
The angular blowing-a-kiss problem

Kevin Buchin, Irina Kostitsyna, Roel Lambers, and Martijn Struijs

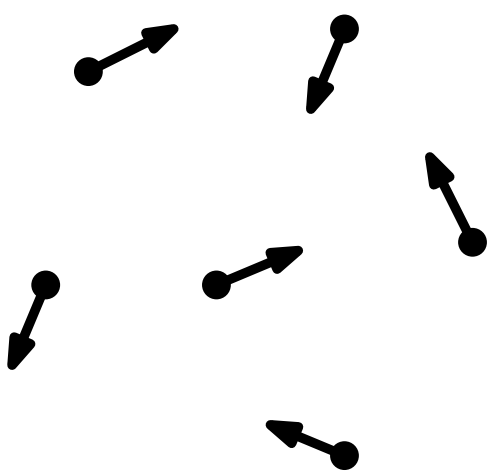


TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

The angular blowing-a-kiss problem

TU/e

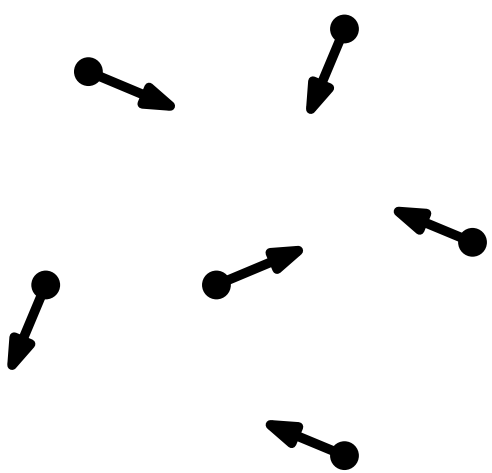
- n oriented agents in the plane



The angular blowing-a-kiss problem

TU/e

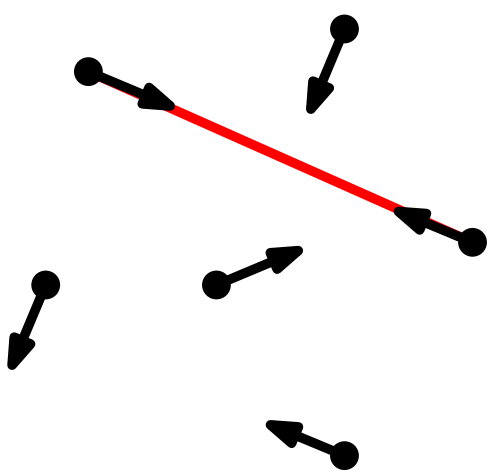
- n oriented agents in the plane
- stationary, but can rotate



The angular blowing-a-kiss problem

TU/e

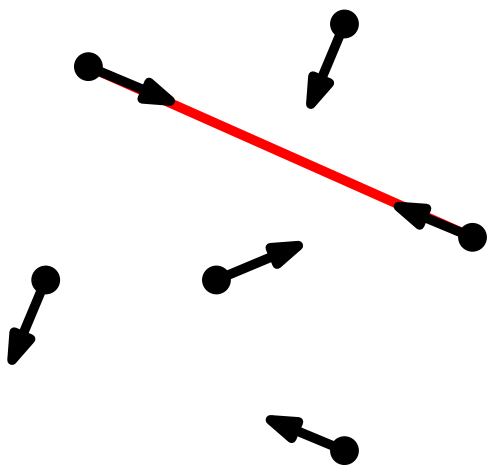
- n oriented agents in the plane
- stationary, but can rotate
- a pair of agents oriented towards eachother can *scan*



The angular blowing-a-kiss problem

TU/e

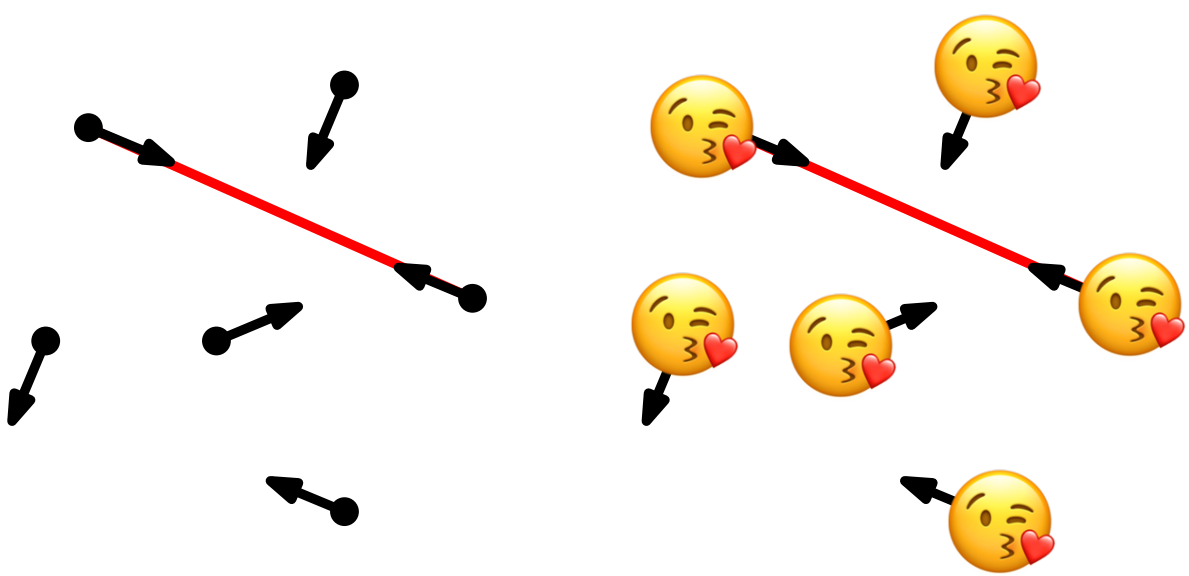
- n oriented agents in the plane
- stationary, but can rotate
- a pair of agents oriented towards eachother can *scan*
- how long does it take to scan all pairs?



The angular blowing-a-kiss problem

TU/e

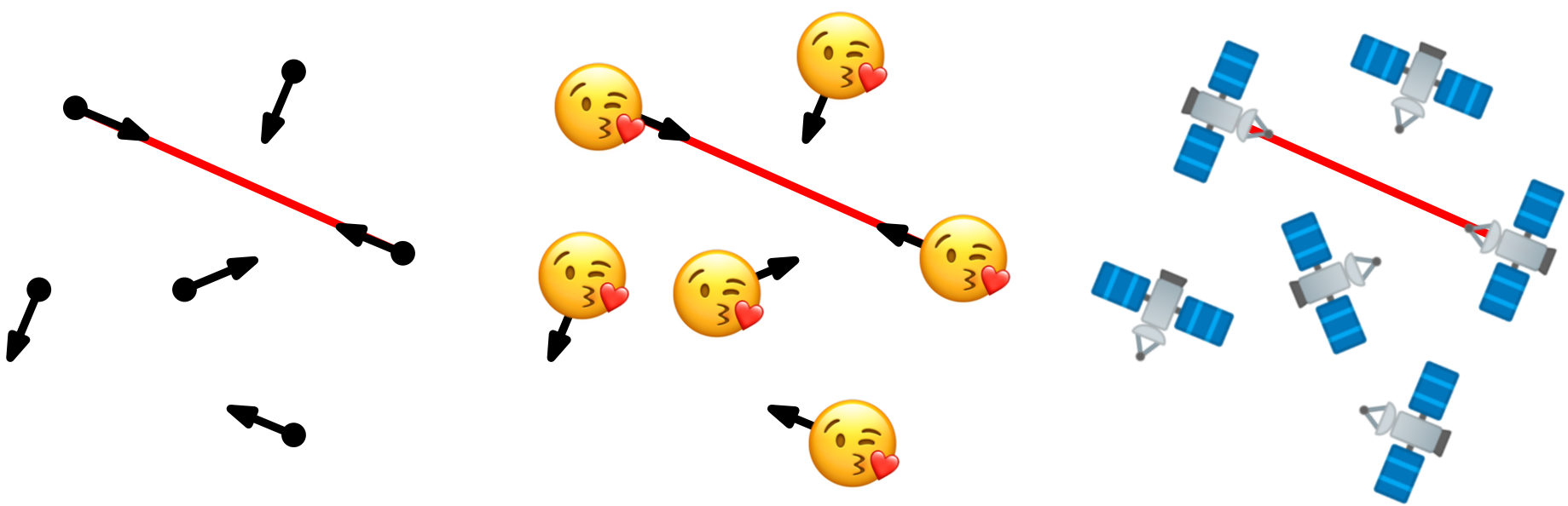
- n oriented agents in the plane
- stationary, but can rotate
- a pair of agents oriented towards eachother can *scan*
- how long does it take to scan all pairs?



The angular blowing-a-kiss problem

TU/e

- n oriented agents in the plane
- stationary, but can rotate
- a pair of agents oriented towards eachother can *scan*
- how long does it take to scan all pairs?

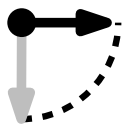


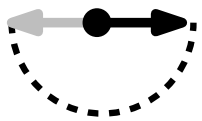
Related work

- kissing problem [Bender et al. 2014]
- angular freeze-tag [Fekete, Krupke 2018]
- scan cover for general graphs [Fekete, Kleist, Krupke 2020]

Problem description

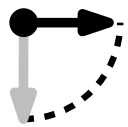
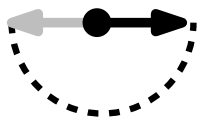
- rotations take time proportional to their angle

 $\pi/2$ time

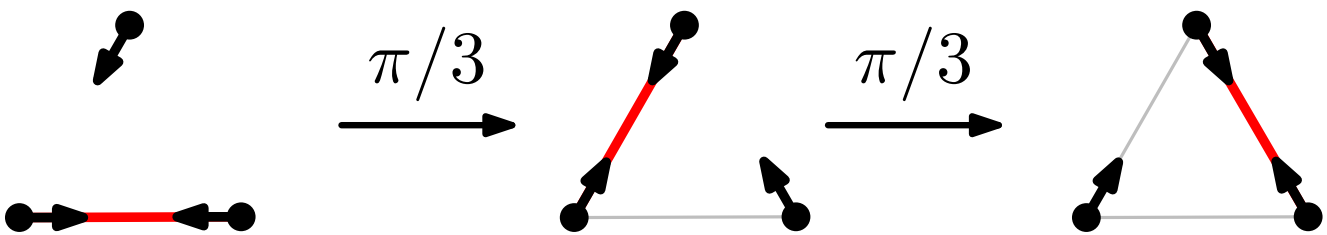
 π time

Problem description

- rotations take time proportional to their angle

 $\pi/2$ time  π time

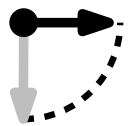
- goal: find a schedule that minimizes makespan

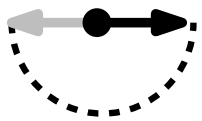


makespan: $2\pi/3$

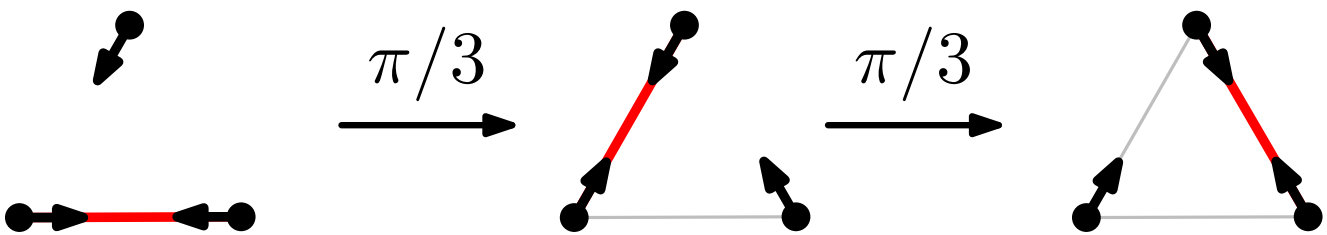
Problem description

- rotations take time proportional to their angle

 $\pi/2$ time

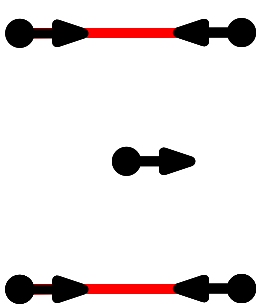
 π time

- goal: find a schedule that minimizes makespan

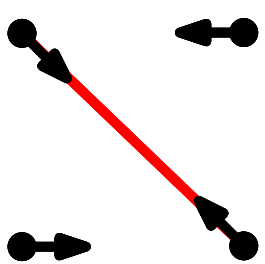


makespan: $2\pi/3$

- *synchronous* schedule: $\lfloor n/2 \rfloor$ simultaneous scans per round



synchronous

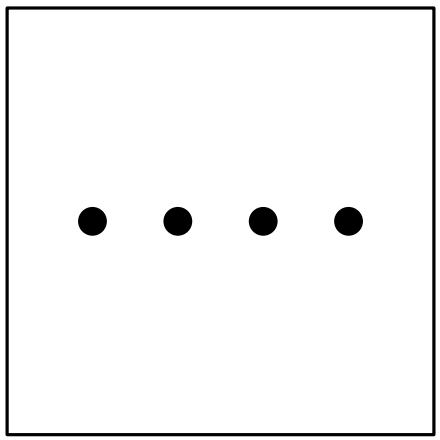


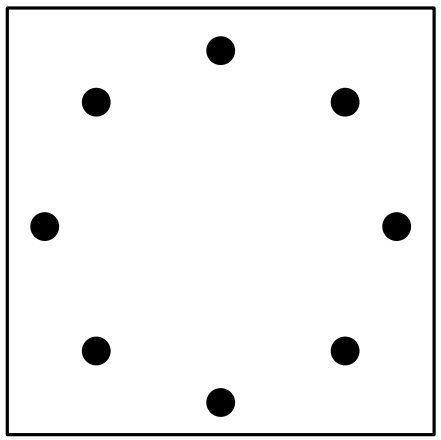
asynchronous

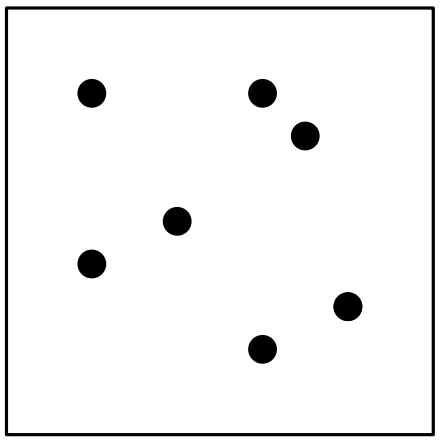
Results

	line/1D	uniform circle	general 2D
async.	$\pi(\lceil \log n \rceil - 1)$	$\sim \pi \log n$	$\pi(\frac{3}{2}\lceil \log n \rceil - \frac{1}{2})$
sync.	$\pi(\lceil \log n \rceil - 1)$	$2\pi \log n$ [a]	–
LB	$\pi(\lceil \log n \rceil - 1)$	$\sim \pi \log n$	–

[a] when n is a power of 2



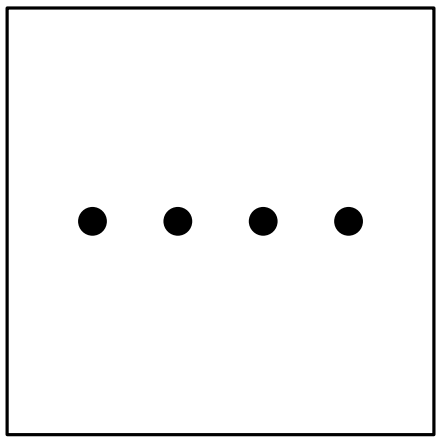


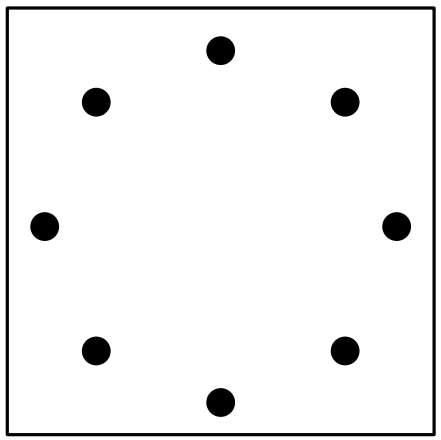


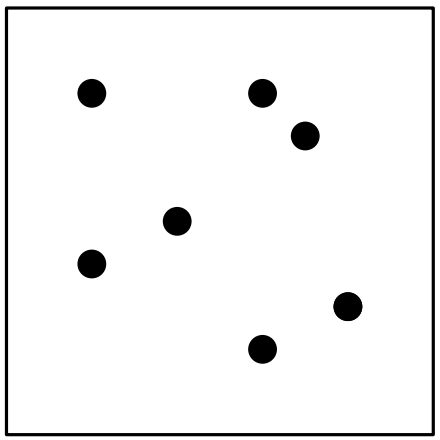
Results

	line/1D	uniform circle	general 2D
async.	$\pi(\lceil \log n \rceil - 1)$	$\sim \pi \log n$	$\pi(\frac{3}{2}\lceil \log n \rceil - \frac{1}{2})$
sync.	$\pi(\lceil \log n \rceil - 1)$	$2\pi \log n$ [a]	–
LB	$\pi(\lceil \log n \rceil - 1)$	$\sim \pi \log n$	–

[a] when n is a power of 2

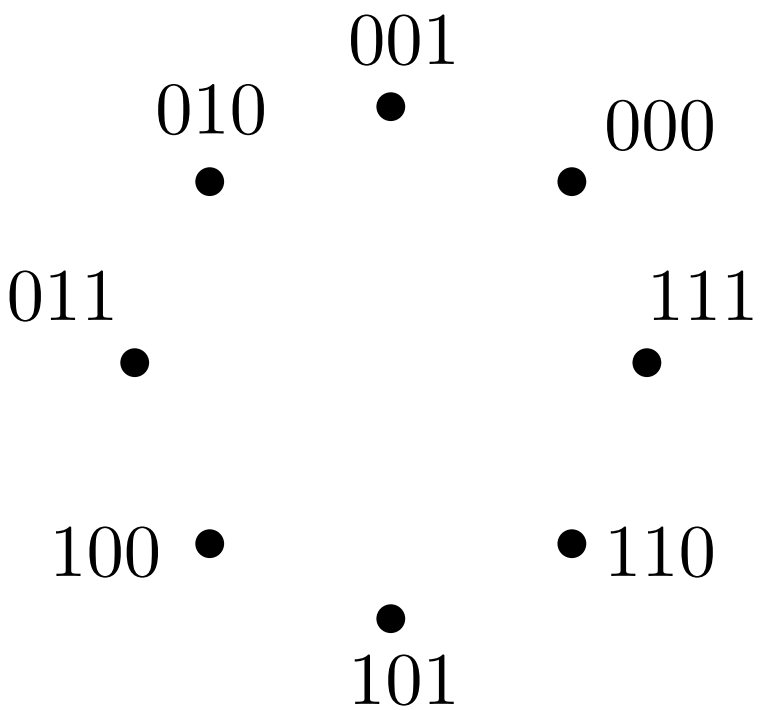






Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit



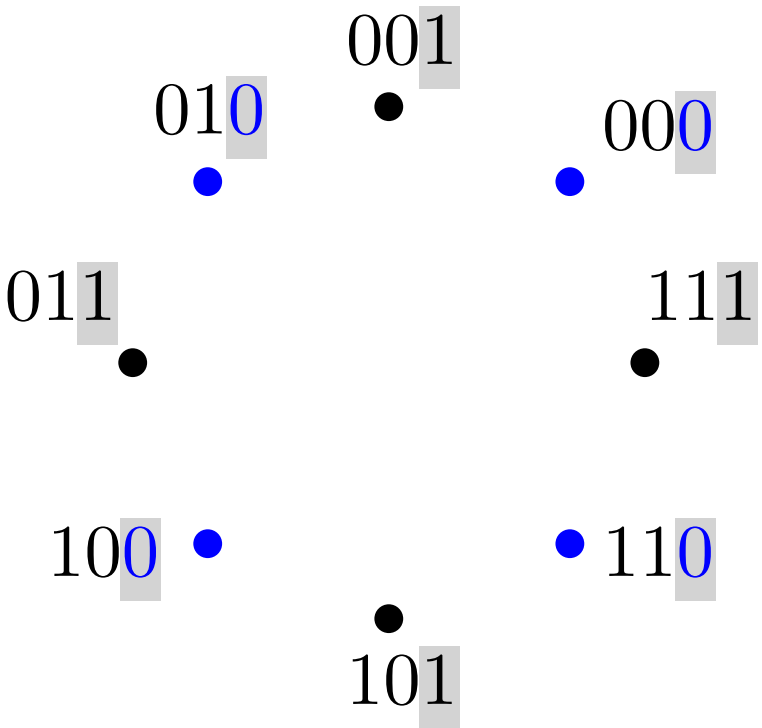
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 1:



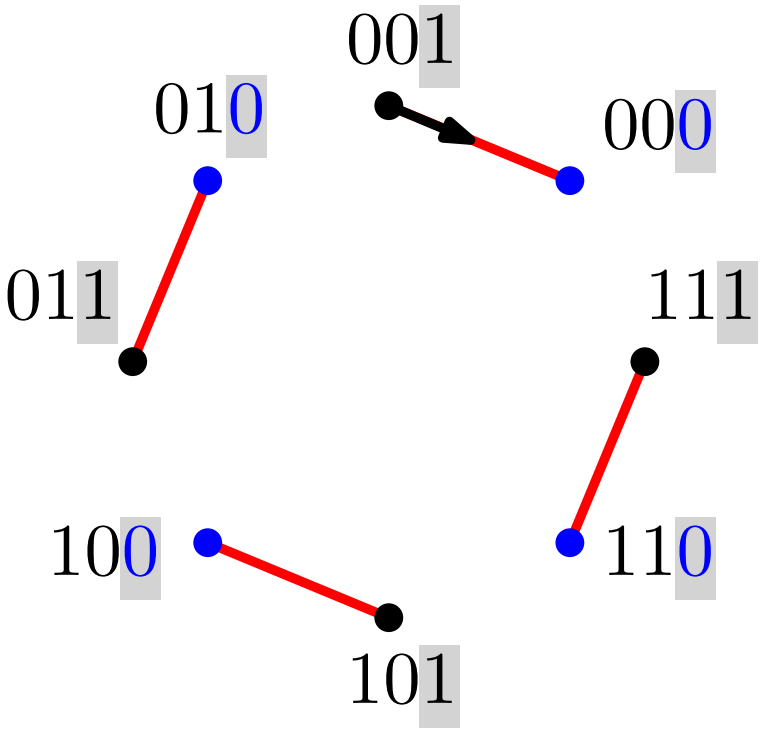
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 1:



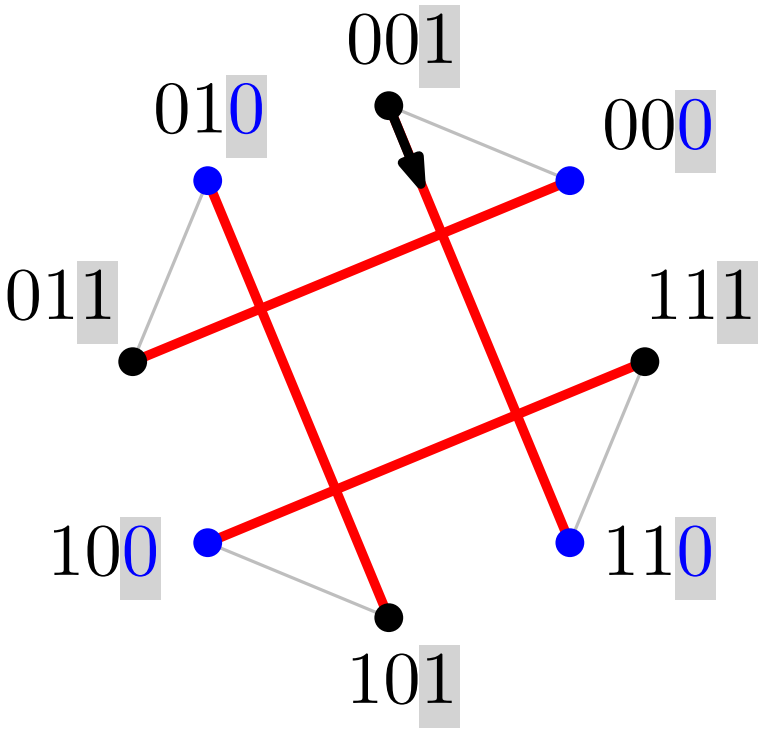
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 1:



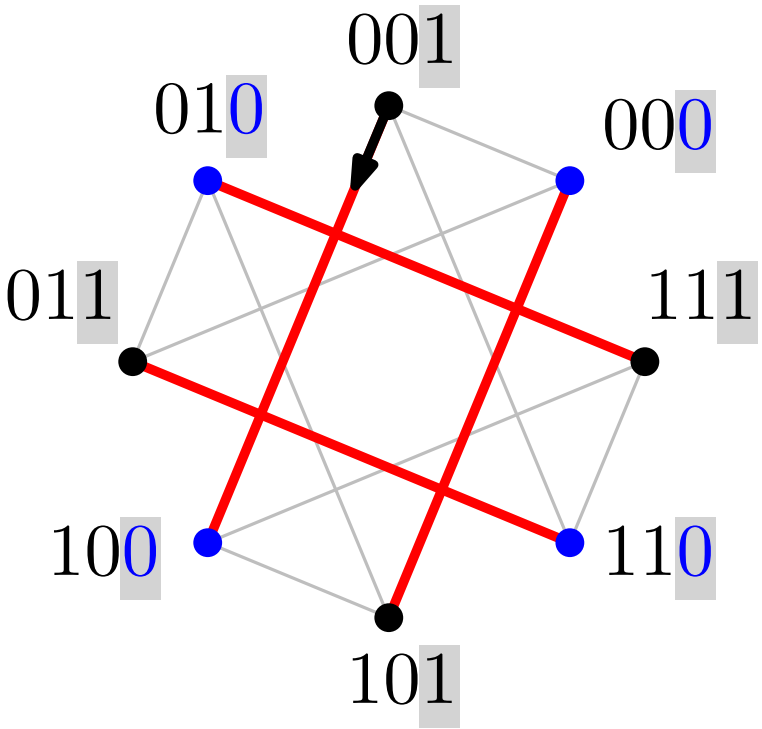
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 1:



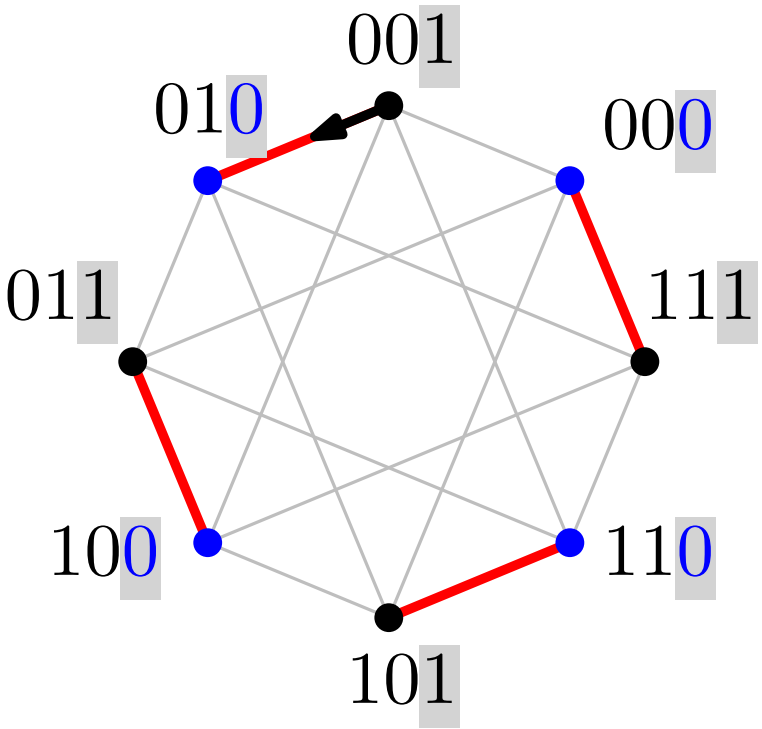
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 1:



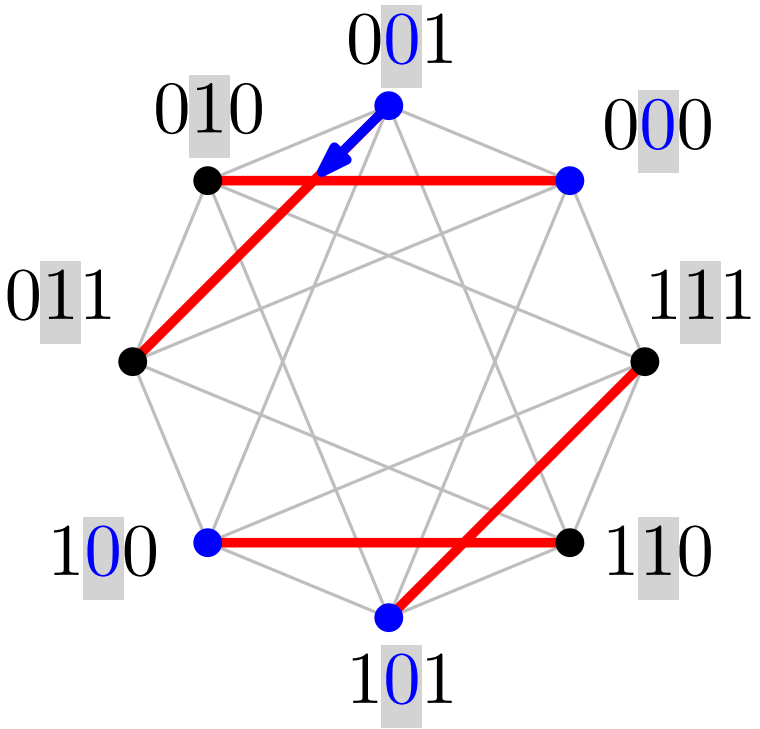
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 2:



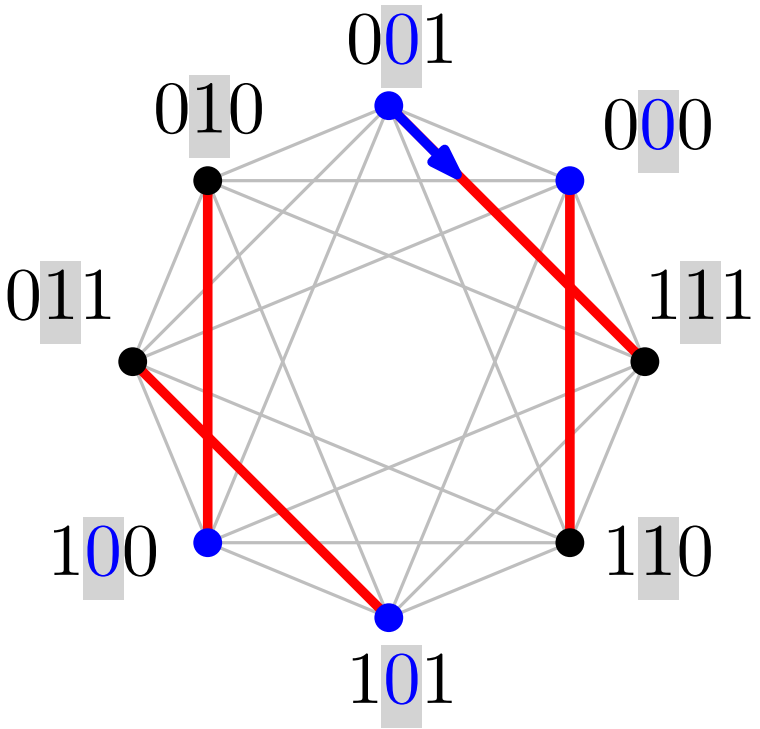
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 2:



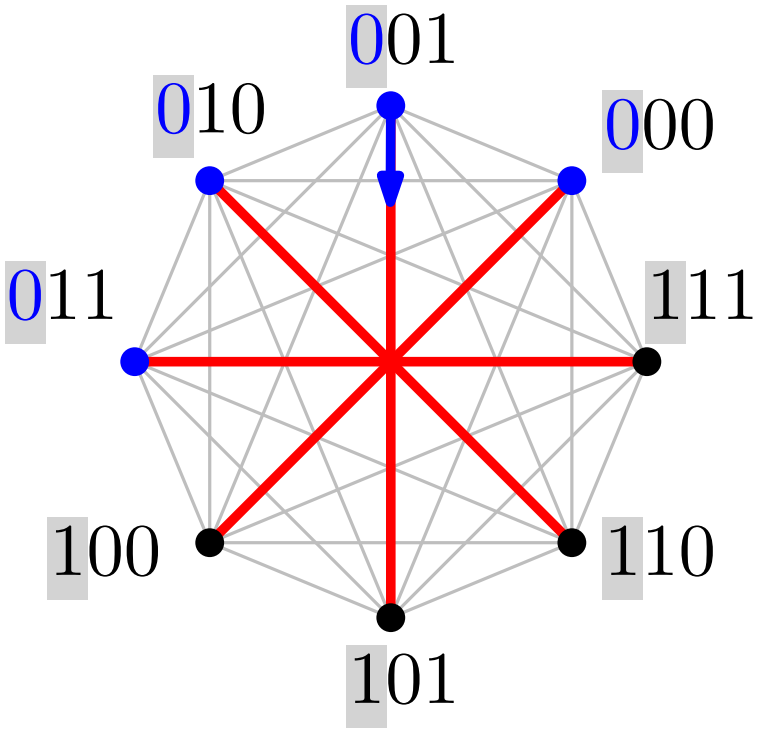
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 3:



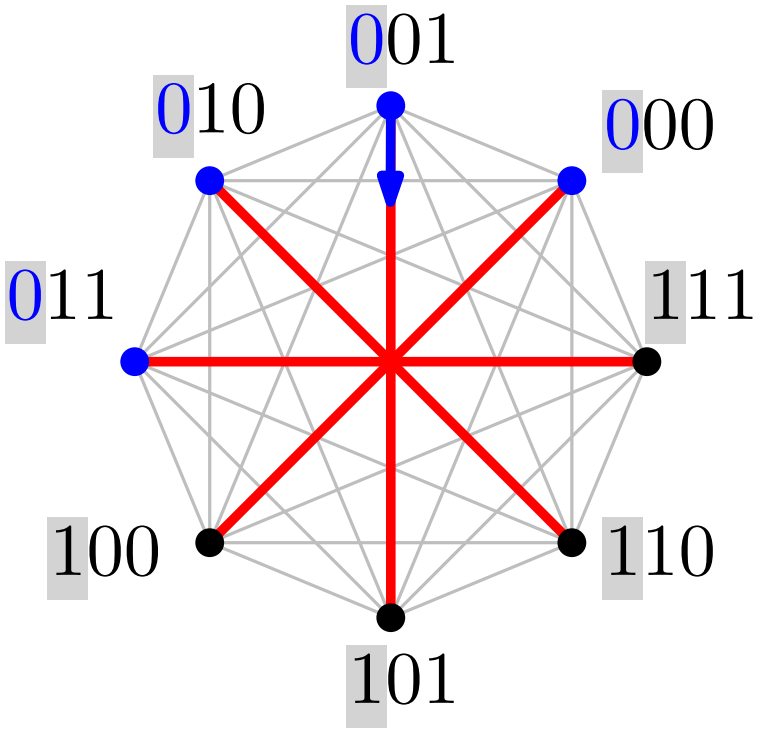
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

phase 3:



- phase ℓ:
  scan pairs at distance $2^{\ell-1}(2i+1)$
  for $i = 0, \ldots, 2^{k-\ell} - 1$

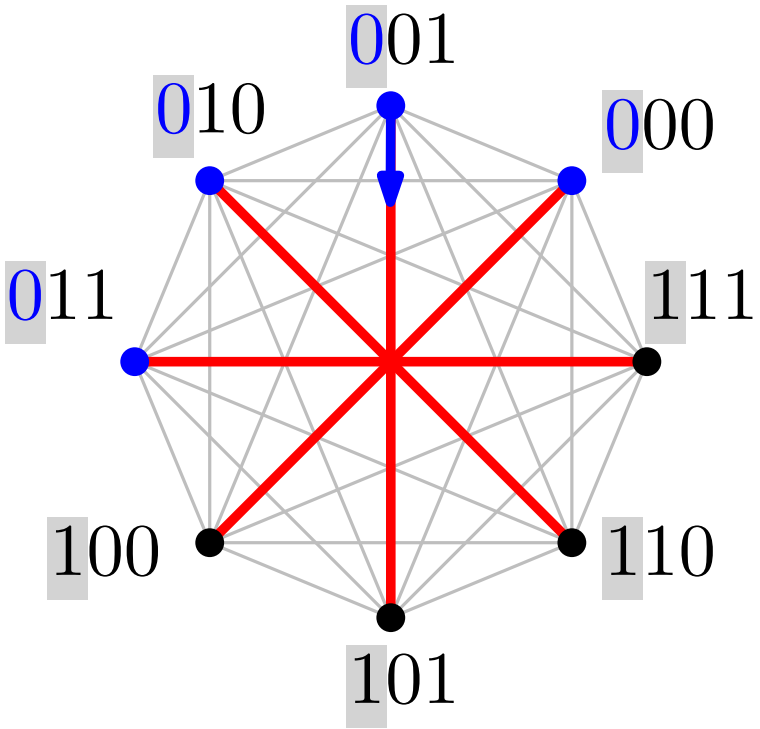
1-bit agents rotate clockwise

0-bit agents rotate counter-clockwise

Synchronous on a circle with $n = 2^k$

for phase $\ell = 1, \ldots, k$: scan pairs that differ in ℓ-th bit

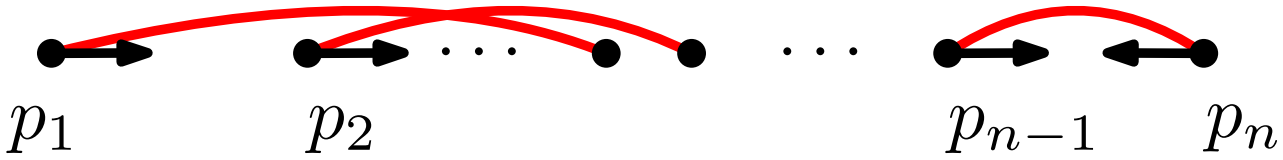
phase 3:



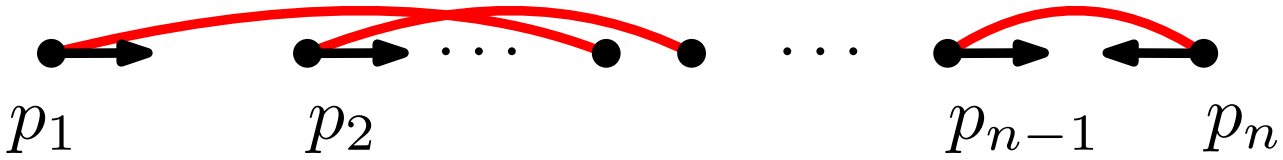
- phase ℓ:
  scan pairs at distance $2^{\ell-1}(2i+1)$
  for $i = 0, \ldots, 2^{k-\ell} - 1$
- costs:
  π per phase to scan pairs
  π between phases
  $2\pi \log n$ in total

1-bit agents rotate clockwise
0-bit agents rotate counter-clockwise

Synchronous on a line



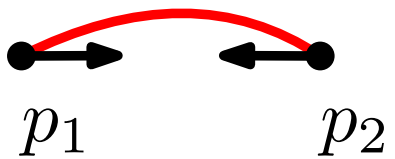
Synchronous on a line



Algorithm:

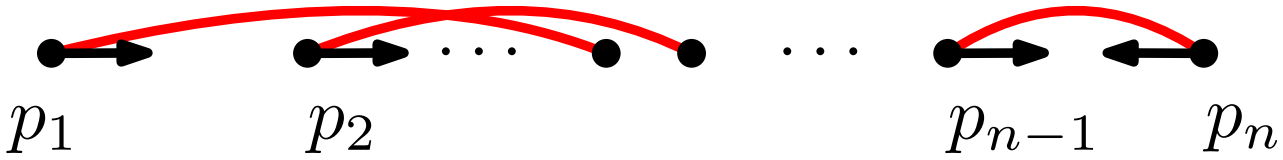
iteratively construct schedule S_n with rules:

base case: S_2, with 0 time



1. $S_k \to S_{k-1}$ if k even, with the same time
2. $S_k \to S_{2k}$, adding π time

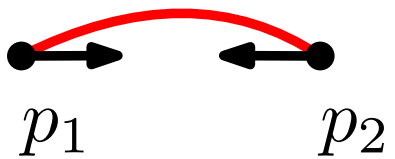
Synchronous on a line



Algorithm:

iteratively construct schedule S_n with rules:

base case: S_2, with 0 time



1. $S_k \to S_{k-1}$ if k even, with the same time
2. $S_k \to S_{2k}$, adding π time

Time:

$S_2 \rightsquigarrow S_n$ with $\lceil \log(n/2) \rceil$ applications of rule 2,
so $\pi(\lceil \log n \rceil - 1)$ time in total

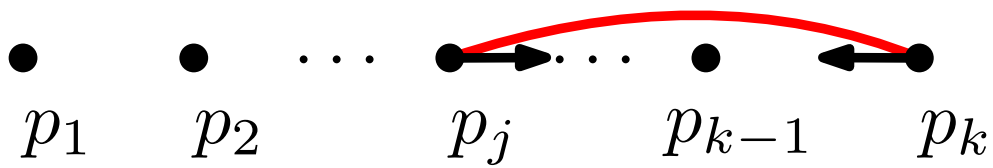
Synchronous on a line

1. $S_k \to S_{k-1}$ if k even, in the same time

Synchronous on a line

1. $S_k \to S_{k-1}$ if k even, in the same time

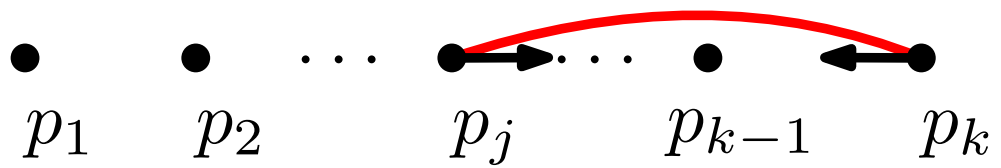
round i of S_k:



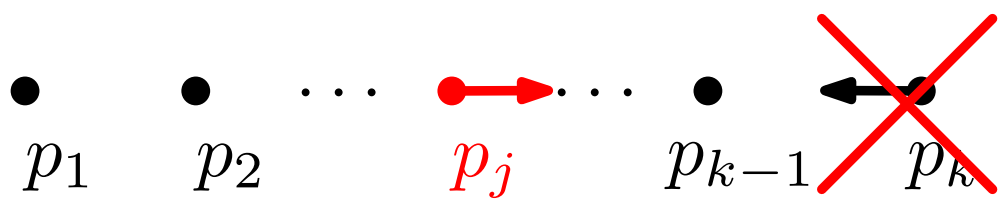
Synchronous on a line

1. $S_k \to S_{k-1}$ if k even, in the same time

round i of S_k:

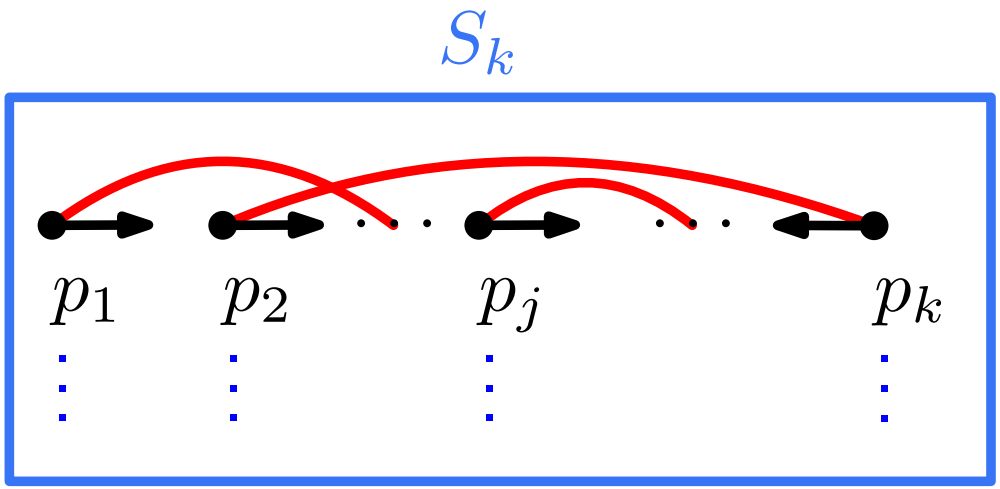


round i of S_{k-1}:
bye: p_j

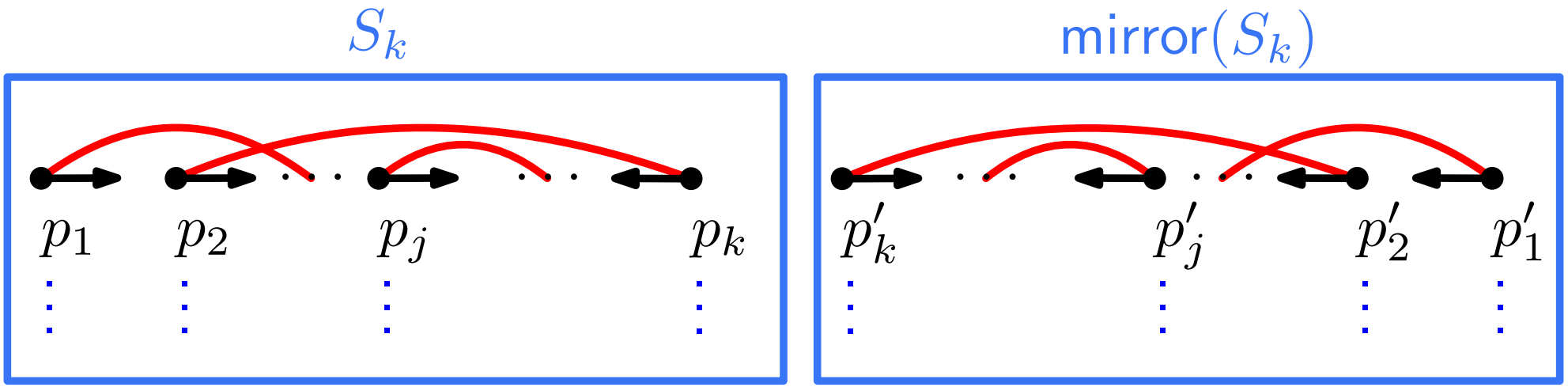


2. $S_k \to S_{2k}$, adding π time

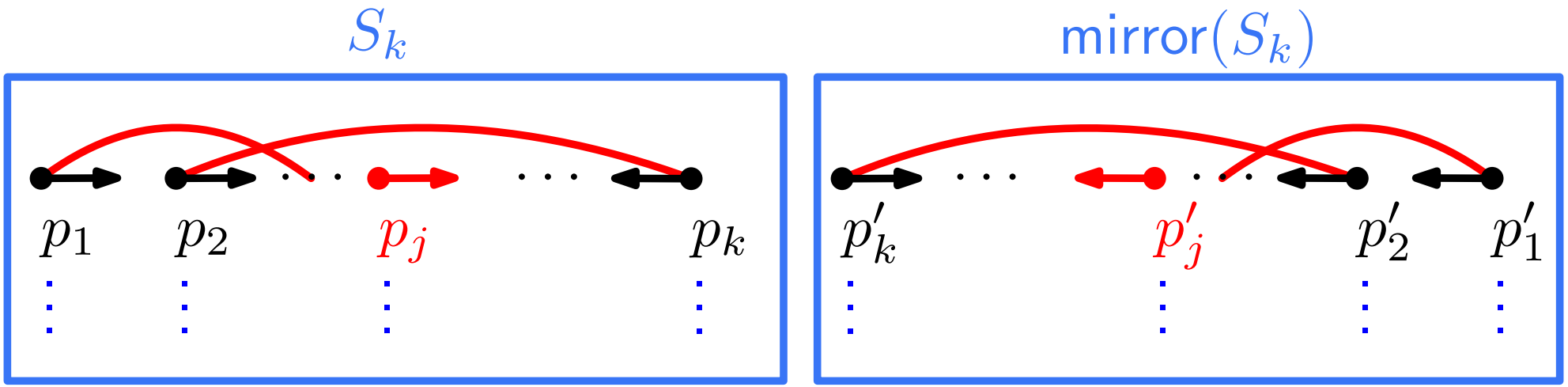
2. $S_k \to S_{2k}$, adding π time



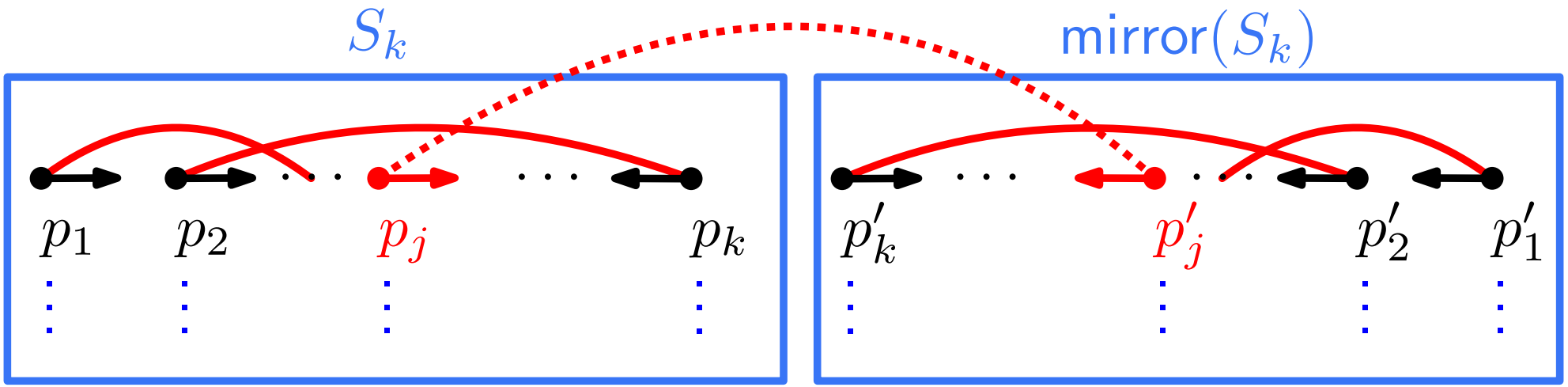
2. $S_k \to S_{2k}$, adding π time



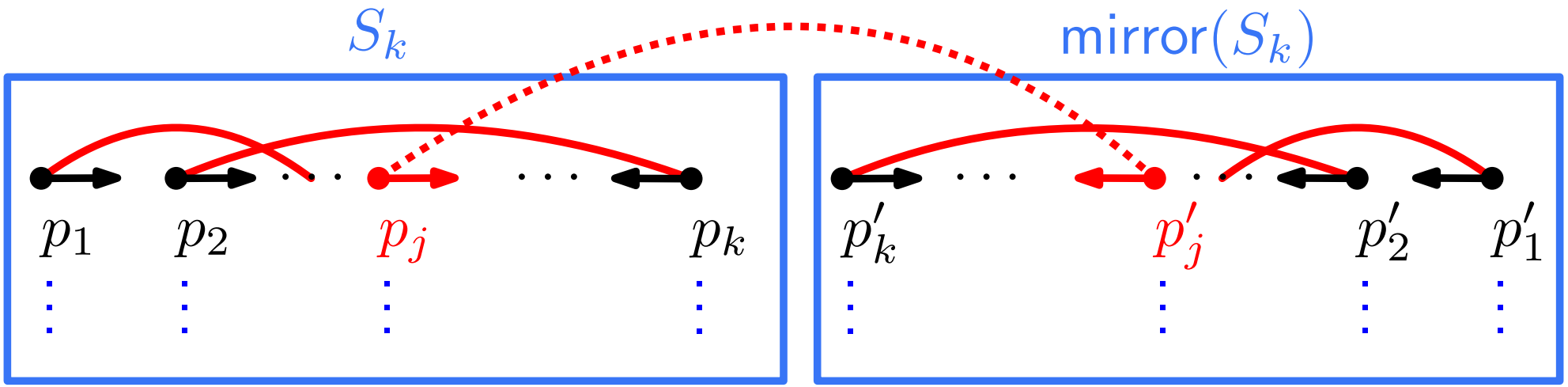
2. $S_k \to S_{2k}$, adding π time



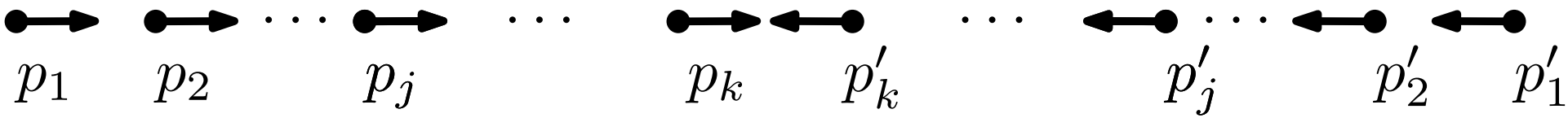
2. $S_k \to S_{2k}$, adding π time



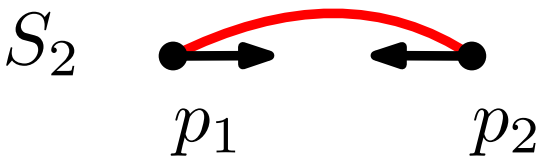
2. $S_k \to S_{2k}$, adding π time



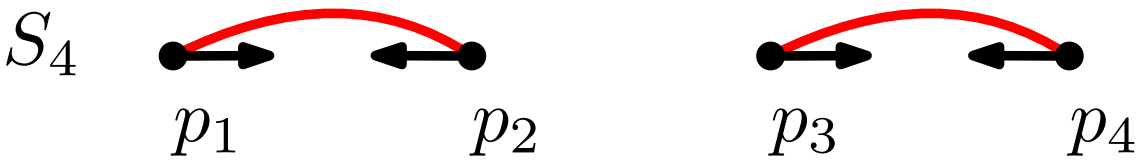
final rounds: scan bipartite graph between left and right



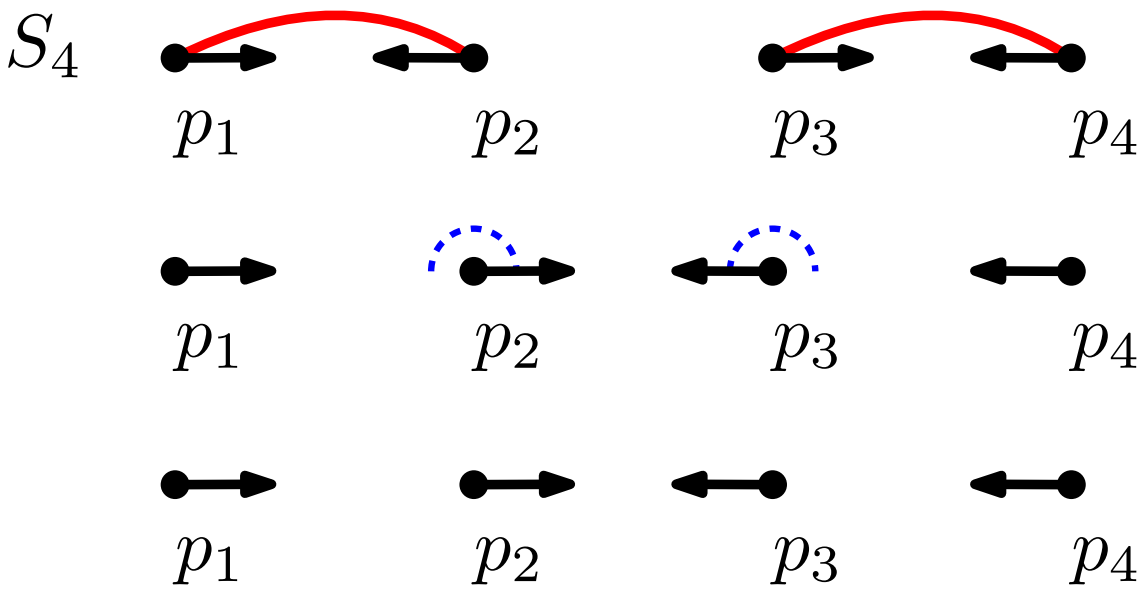
Example: $S_2 \to S_4 \to S_3 \to S_6$



Example: $S_2 \to S_4 \to S_3 \to S_6$

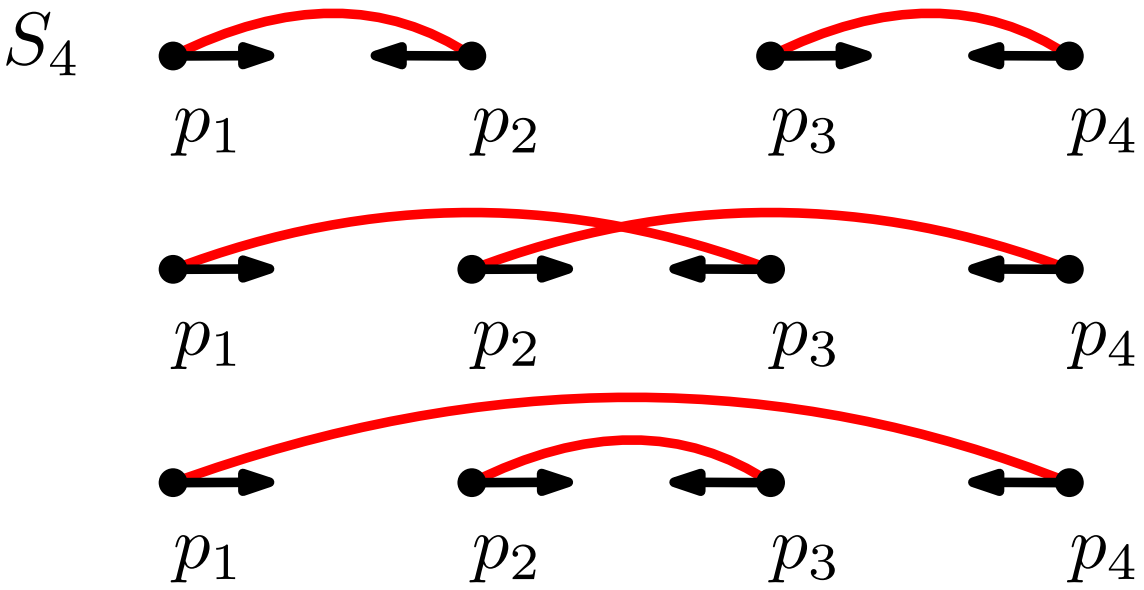


Example: $S_2 \to S_4 \to S_3 \to S_6$



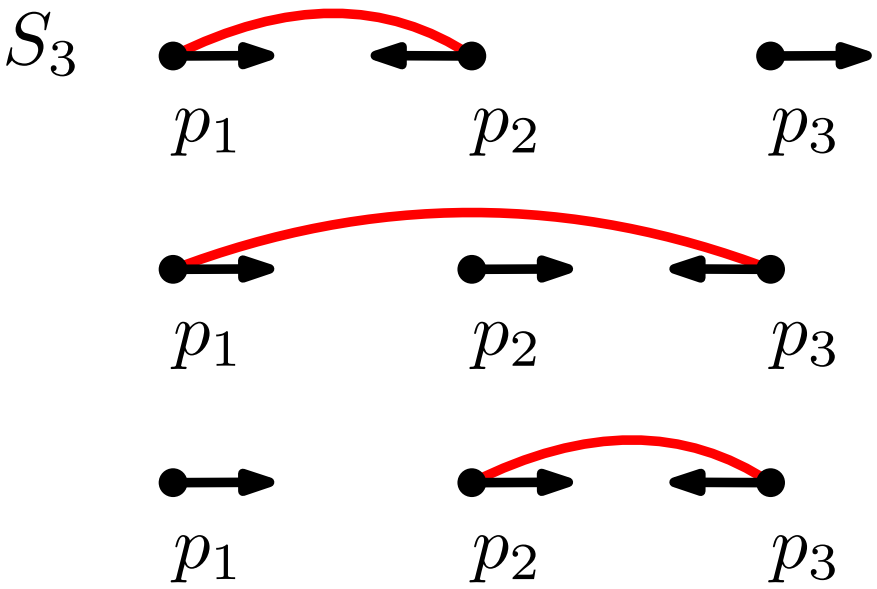
Example: $S_2 \to S_4 \to S_3 \to S_6$

TU/e

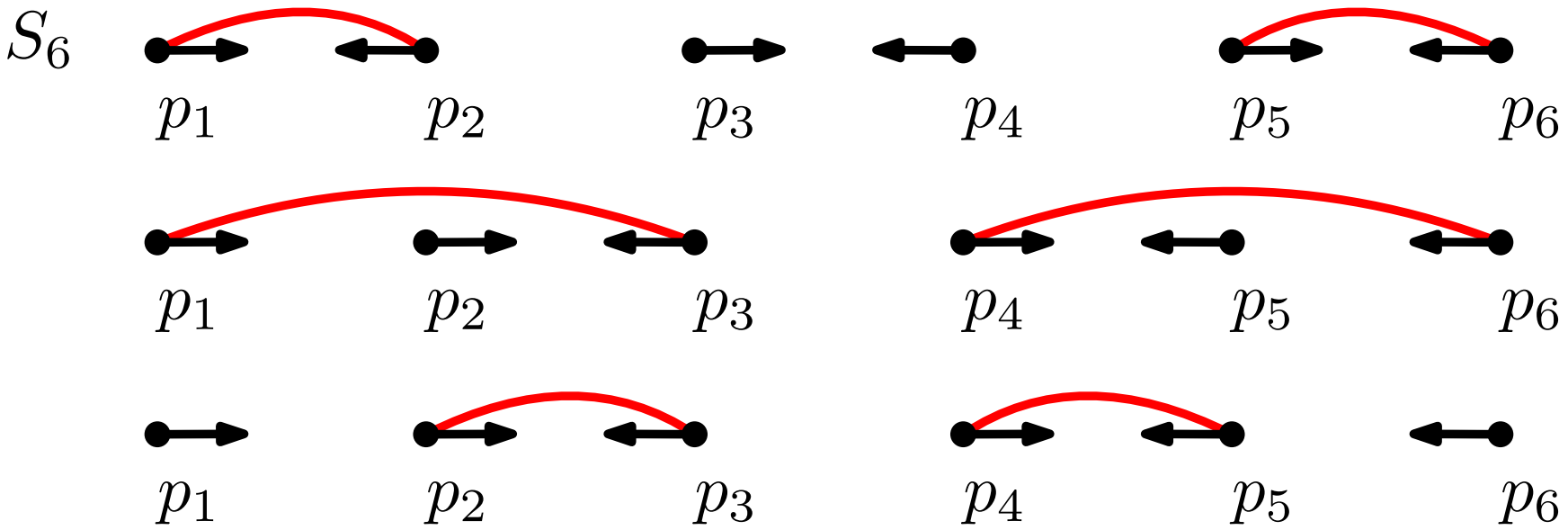


Example: $S_2 \to S_4 \to S_3 \to S_6$

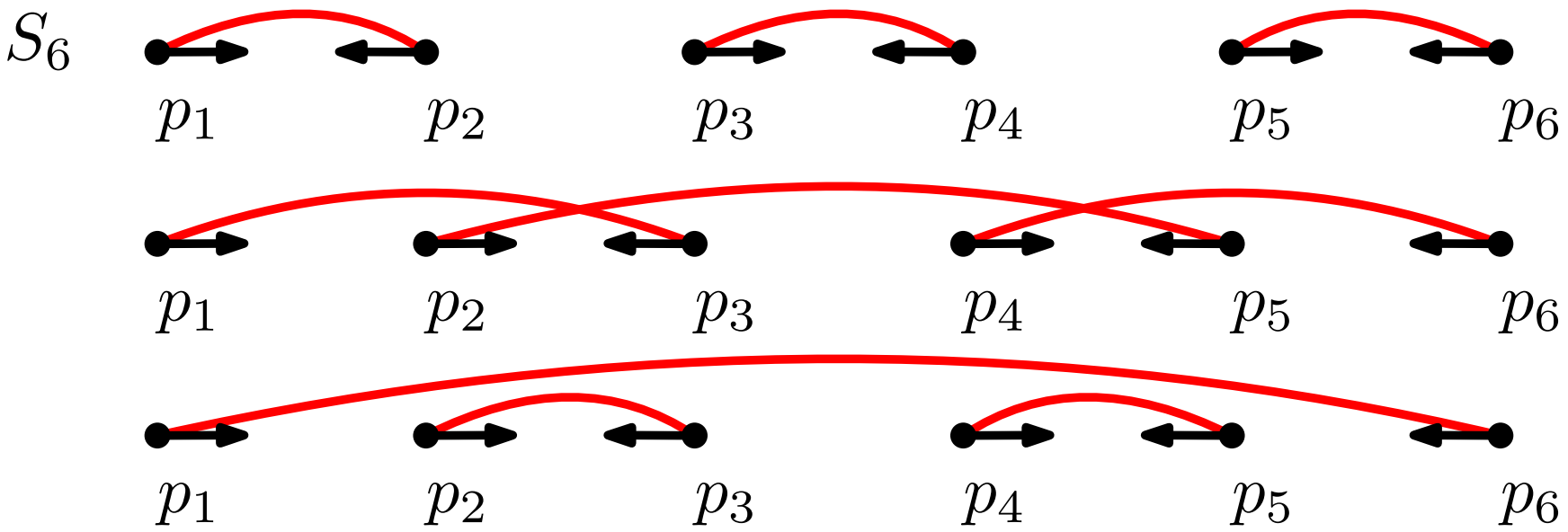
TU/e



Example: $S_2 \to S_4 \to S_3 \to S_6$

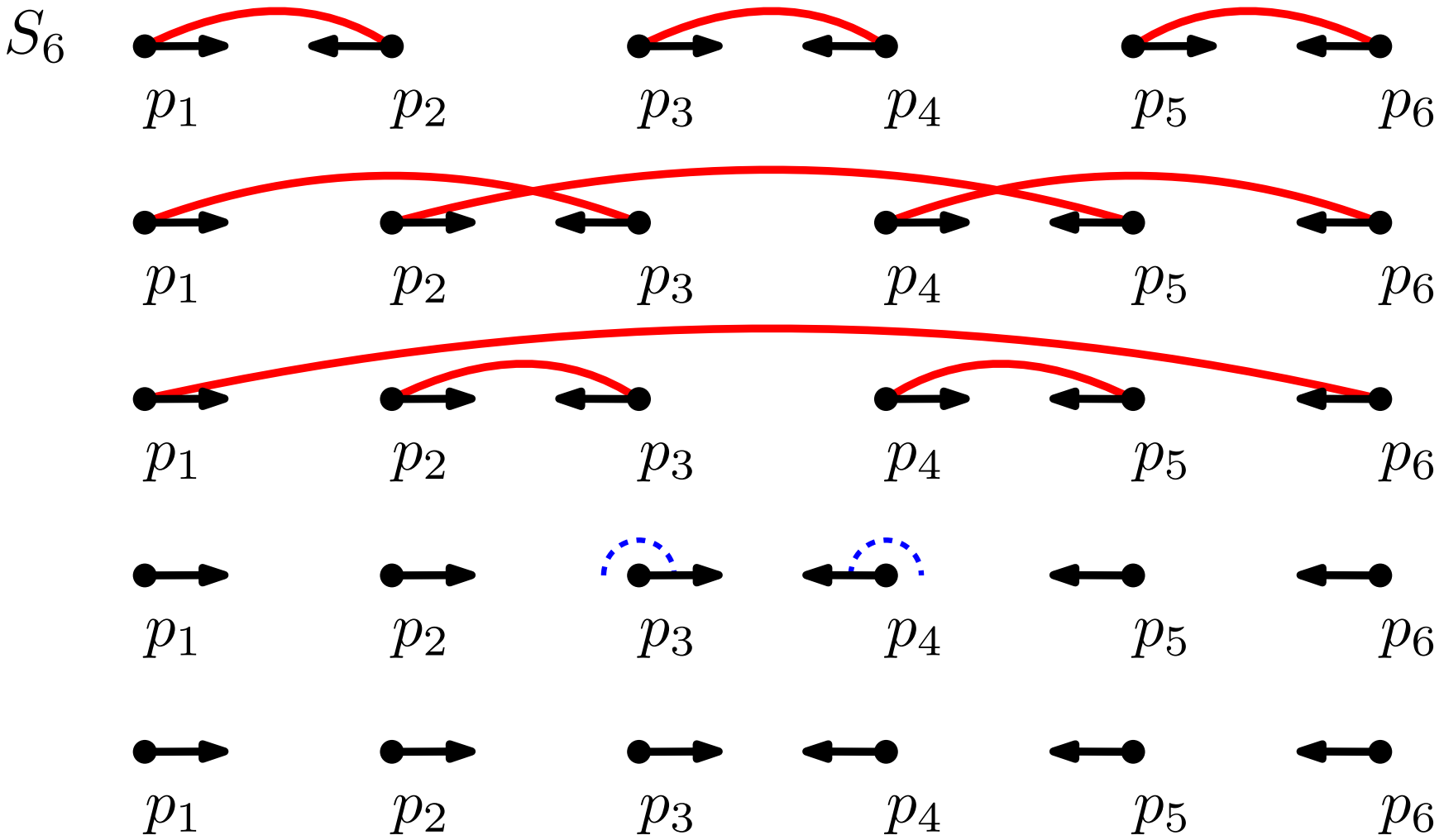


Example: $S_2 \to S_4 \to S_3 \to S_6$

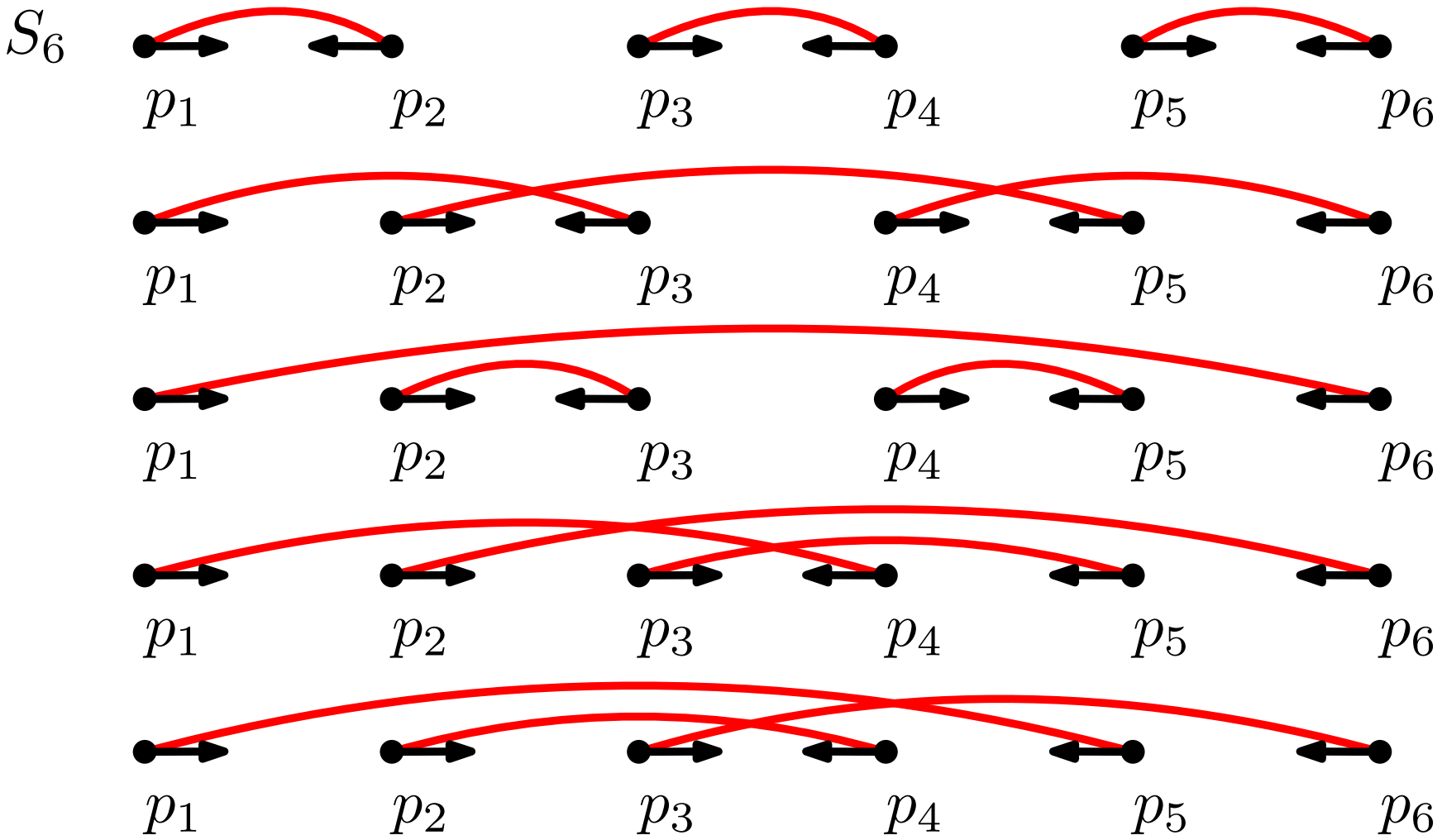


Example: $S_2 \to S_4 \to S_3 \to S_6$

TU/e



Example: $S_2 \to S_4 \to S_3 \to S_6$



Conclusion

TU/e

Results:

	line/1D	uniform circle	general 2D
async.	$\pi(\lceil \log n \rceil - 1)$	$\sim \pi \log n$	$\pi(\frac{3}{2}\lceil \log n \rceil - \frac{1}{2})$
sync.	$\pi(\lceil \log n \rceil - 1)$	$2\pi \log n$ [a]	–
LB	$\pi(\lceil \log n \rceil - 1)$	$\sim \pi \log n$	–

[a] when n is a power of 2

Work in progress:

- synchronous solutions for more general cases
- complexity of finding optimal solutions in 2D