How to Meet Asynchronously (Almost) Everywhere

Jurek Czyzowicz[1], Arnaud Labourel[2], Andrzej Pelc[1]

[1]Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec, Canada

[2]LaBRI, University Bordeaux, Talence, France

Université de Bordeaux, le 4 Mars 2010

The rendezvous problem

The problem (graph scenario)

Two mobile agents must meet inside an unknown network.

Modelisation of the problem :

- Networks $\rightarrow$ graphs
- Mobiles agents $\rightarrow$ points moving from node to node along the edges of the graph
- Rendezvous $\rightarrow$ meeting of the two agents in a node or on an edge
- Cost $\rightarrow$ sum of the lengths of the trajectories of the agents until rendezvous

The rendezvous problem

The problem (geometric scenario)

Two mobile agents must meet in a geometric terrain.

Modelisation of the problem :

- Geometric terrain → A subset of plane with polygonal obstacles
- Mobiles agents → points moving in the terrain
- Agent's visibility range → A closed circle centered at agent's position
- Rendezvous → each agent belongs to the visibility range of the other agent
- Cost → sum of the lengths of the trajectories of the agents until rendezvous

Rendezvous in graphs

Graphs considered

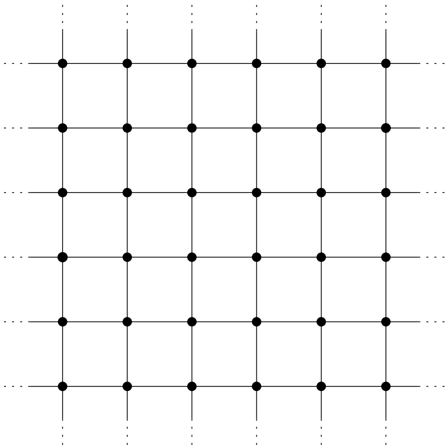
- Nodes are anonymous (do not have identifiers)
- Edges incident to a node are locally numbered (by port numbers)
- Graphs are connected, finite or infinite (with countable node set and edge set)

Movement of the agent

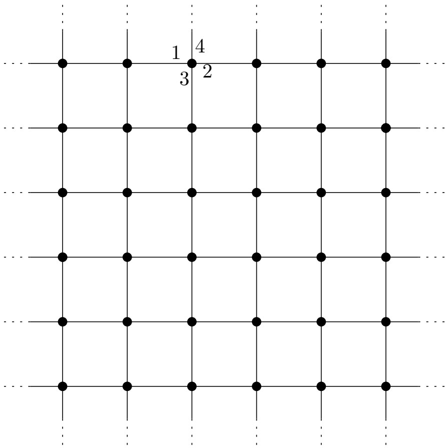
At each step, an agent :

- chooses a port number of the current node (outgoing port)
- moves along the corresponding edge
- accesses the target node of the traversed edge via the port number in the new node (incoming port)

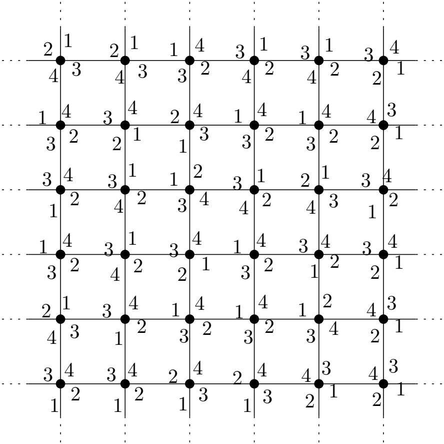
Example of a graph



Example of a graph



Example of a graph



Deterministic rendezvous

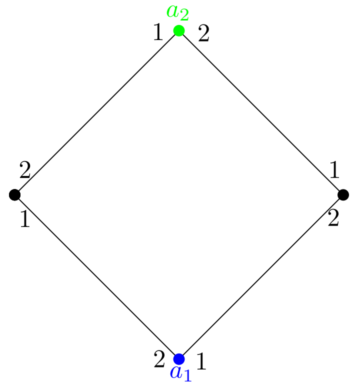
Deterministic Algorithm

The route (sequence of port numbers) followed by an agent only depends on :

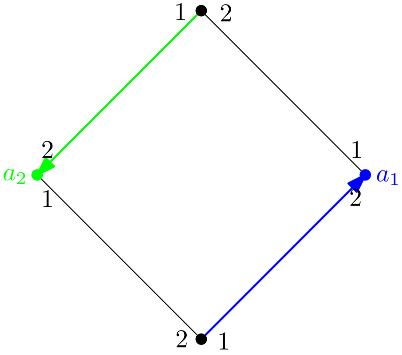
- the environment : the starting position of the agent and the graph (more precisely the part of the graph that the agent learned up to date)
- the identifier of the agent

Agent's identifier is required for deterministic model
$\Rightarrow$ Without identifiers, deterministic agents never meet in a ring because of symmetry.

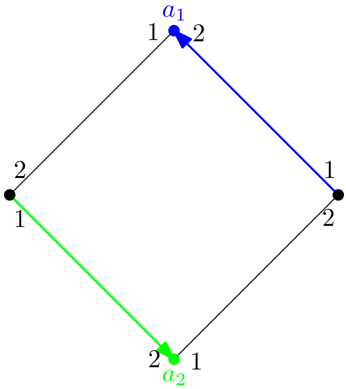
Deterministic rendezvous impossible without identifier



Deterministic rendezvous impossible without identifier



Deterministic rendezvous impossible without identifier



Asynchronous model

The agents

- Each agent chooses a sequence of steps forming its route (or the algorithm computing it)
- The agents try to choose their routes so they always meet

The omniscient adversary

- Tries to prevent the rendezvous
- Chooses the identities of the two agents and their starting positions
- Knows in advance the route chosen by the agent and determines the duration of each step of the route.

The route and the walk

The route

The route R chosen by the agent is a sequence of segments $(e_1, e_2, \dots)$. In stage i the agent traverses segment $e_i = [a_{i-1}, a_i]$, starting at a_{i-1} and ending in a_i. Stages are repeated indefinitely (until rendezvous).

The walk

Let $(t_1, t_2, \dots)$, where $t_1 = 0$, be an increasing sequence of reals, chosen by the adversary, that represent points in time. Let $f_i : [t_i, t_{i+1}] \to [a_i, a_{i+1}]$ be any continuous non-decreasing function, chosen by the adversary, such that $f_i(t_i) = a_i$ and $f_i(t_{i+1}) = a_{i+1}$. For any $t \in [t_i, t_{i+1}]$, we define $f(t) = f_i(t)$. The interpretation of the walk f is as follows: at time t the agent is at the point $f(t)$ of its route.

The asynchronous rendezvous problem

The feasibility

The asynchronous rendezvous problem has a solution if it is possible to choose a route for each agent $A_1, A_2, \ldots$ such that

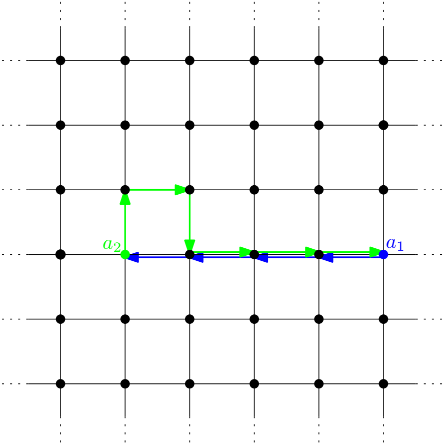
- for any choice of agents A_i, A_j
- for any starting positions of A_i, A_j
- for any walks of the agents A_i, A_j

the agents A_i, A_j will eventually meet

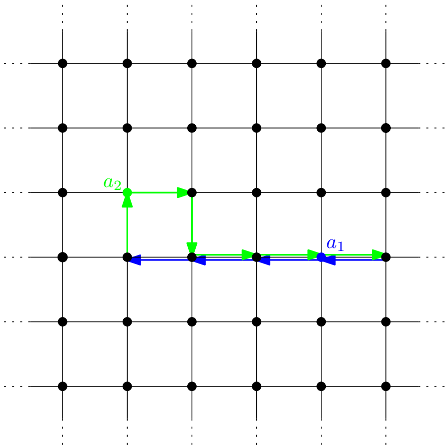
The cost

The cost of the asynchronous rendezvous is the maximum sum of lengths of routes of the two agents (taken over all possible actions by the adversary)

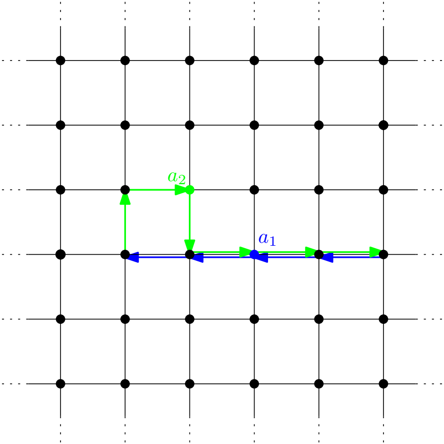
Example of routes that does not guarantee rendezvous



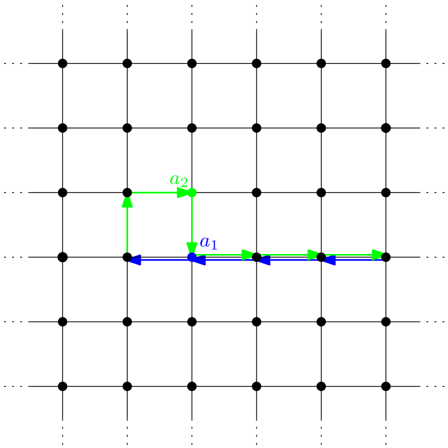
Example of routes that does not guarantee rendezvous



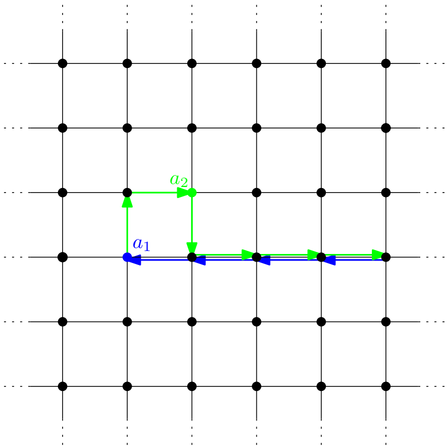
Example of routes that does not guarantee rendezvous



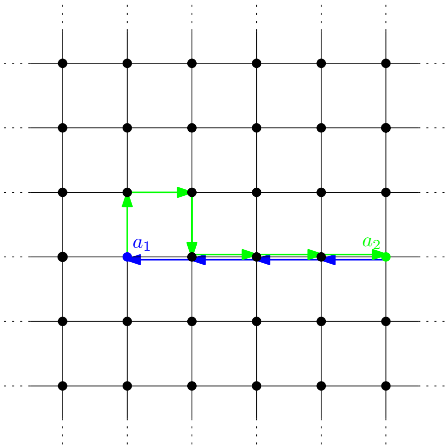
Example of routes that does not guarantee rendezvous



Example of routes that does not guarantee rendezvous



Example of routes that does not guarantee rendezvous



Total knowledge of the agents

Rendezvous in finite graph known to the agents [1]

Rendezvous algorithm at cost $\Theta(D|L_{\min}|)$ if each of the agent knows the graph, its starting position and the starting position of the other agent.

$|L_{\min}|$: size in bits of the smaller of the two identifiers of the agents
D: distance between the two starting positions of the agents

[1] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, U. Vaccaro, Asynchronous deterministic rendezvous in graphs, Theoretical Computer Science 355 (2006), 315-326.

Finite graphs partially known by the agents

Rendezvous in a graph partially known by the agents [1]

Rendezvous algorithm (cost exponential in the size of the graph) if the size of the graph is known by the agent.

[1] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, U. Vaccaro, Asynchronous deterministic rendezvous in graphs, Theoretical Computer Science 355 (2006), 315-326.

Graphs (finite or infinite) unknown to the agents

Rendezvous in graphs unknown to the agents

Deterministic asynchronous rendezvous algorithm in any countable anonymous graph (finite or infinite).

The rendezvous problem in the plane

The problem

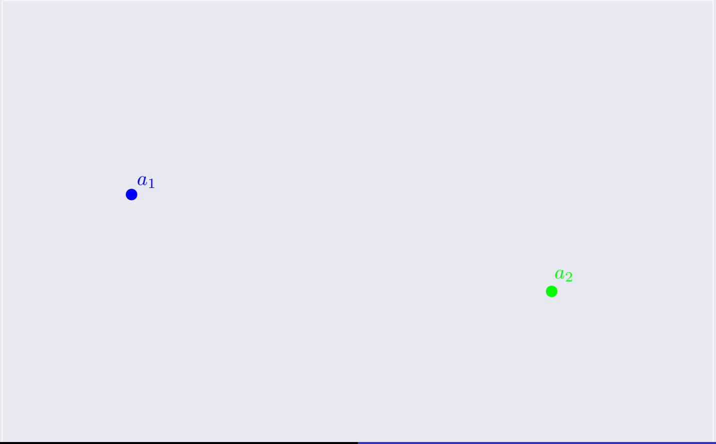
Two mobile agents must meet in the plane.

Modelisation of the problem :

- Mobiles agents $\rightarrow$ points moving inside the plane along a polygonal trajectory. The agents have coherent compasses showing North and a common unit of length.
- Agent's visibility range $\rightarrow$ A circle of radius $\epsilon > 0$ centered at agent's current position
- Rendezvous $\rightarrow$ each agent belongs to the visibility range of the other agent
- Cost $\rightarrow$ sum of the lengths of the trajectories of the agents until rendezvous

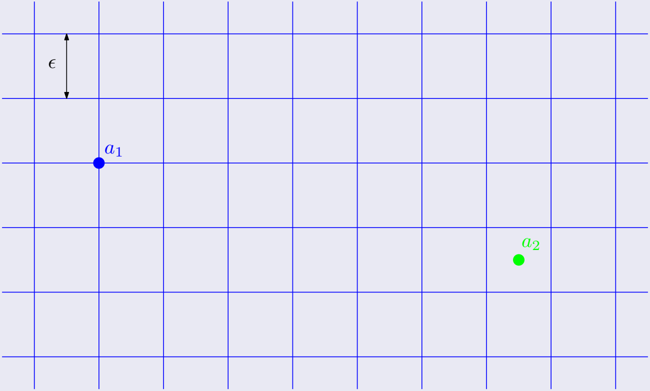
Proof of the rendezvous algorithm in the plane without obstacles

From plane to grid



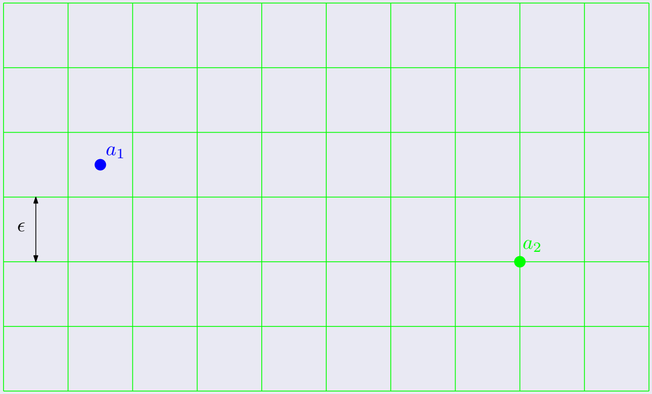
Proof of the rendezvous algorithm in the plane without obstacles

From plane to grid



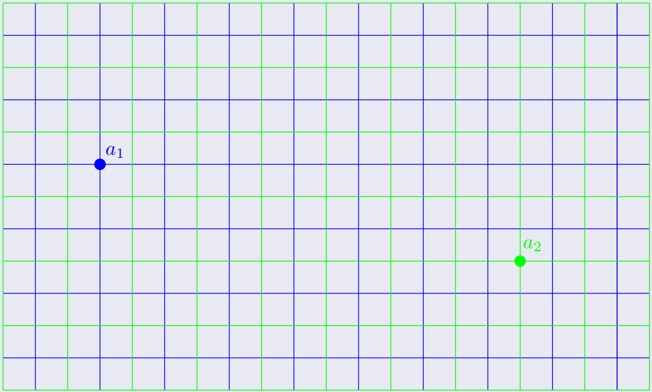
Proof of the rendezvous algorithm in the plane without obstacles

From plane to grid



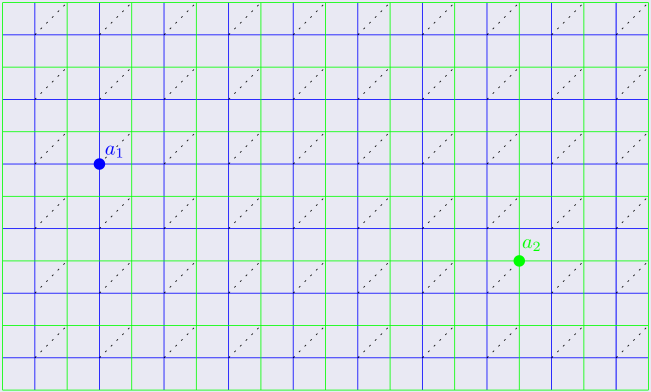
Proof of the rendezvous algorithm in the plane without obstacles

From plane to grid



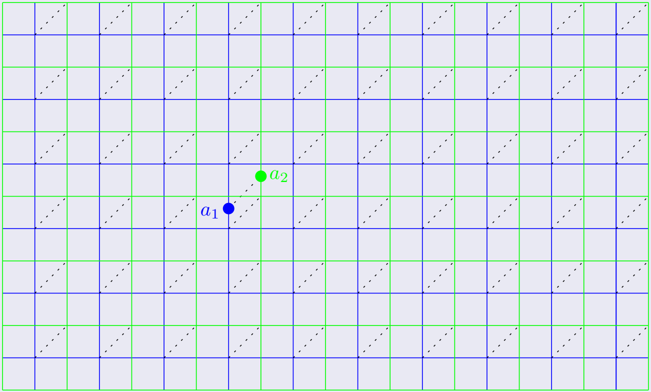
Proof of the rendezvous algorithm in the plane without obstacles

From plane to grid



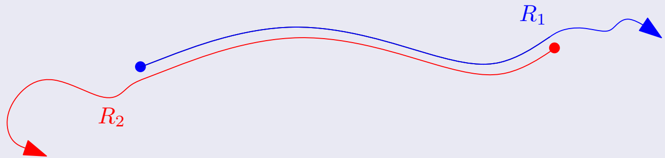
Proof of the rendezvous algorithm in the plane without obstacles

From plane to grid



General idea of the algorithm

The tunnel



Tunnel : sequence of edges $e_1, e_2, \ldots, e_k$ such that :

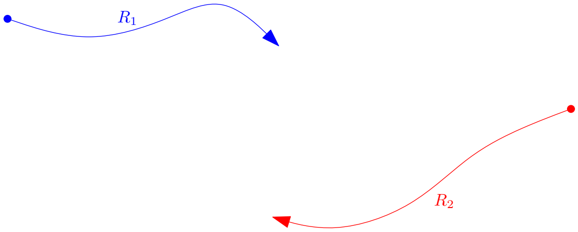
- route of agent 1 begins by $e_1, e_2, \ldots, e_k$
- route of agent 2 begins by $e_k, e_{k-1}, \ldots, e_1$

When the routes of the two agents form a tunnel the agents must meet

Forming tunnels

Fact

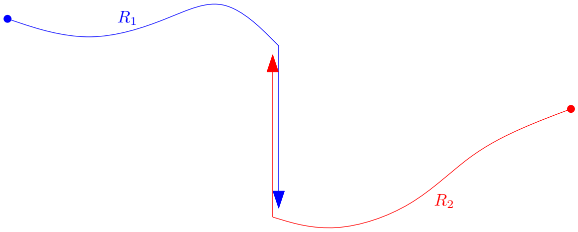
We can always extend two routes such that they form a tunnel.



Forming tunnels

Fact

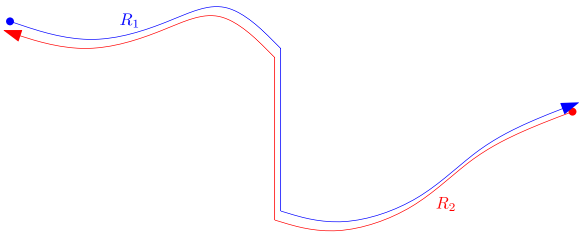
We can always extend two routes such that they form a tunnel.



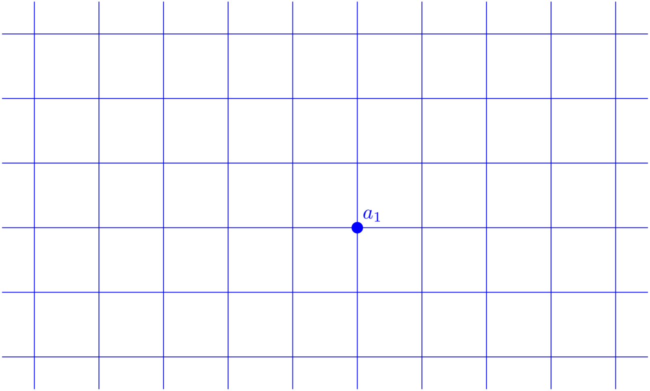
Forming tunnels

Fact

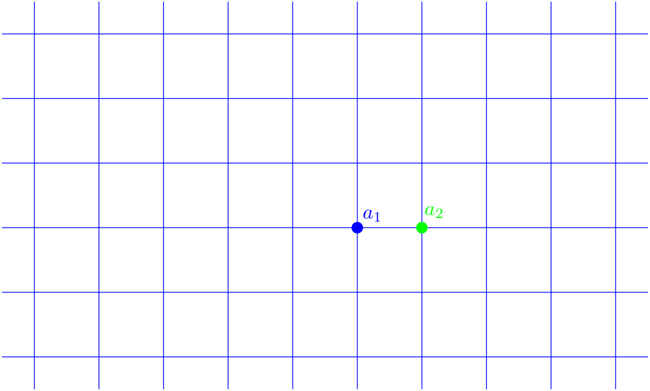
We can always extend two routes such that they form a tunnel.



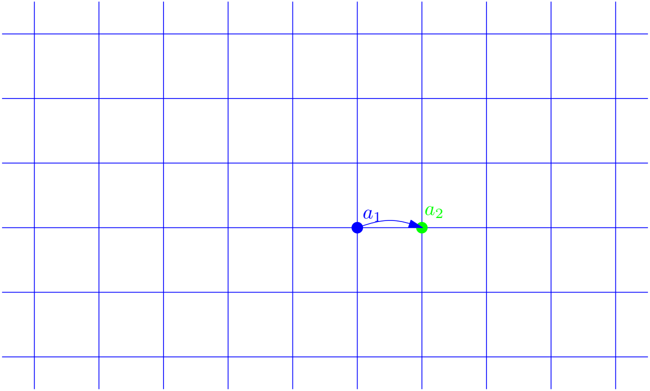
How to meet asynchronously two agents in the grid?



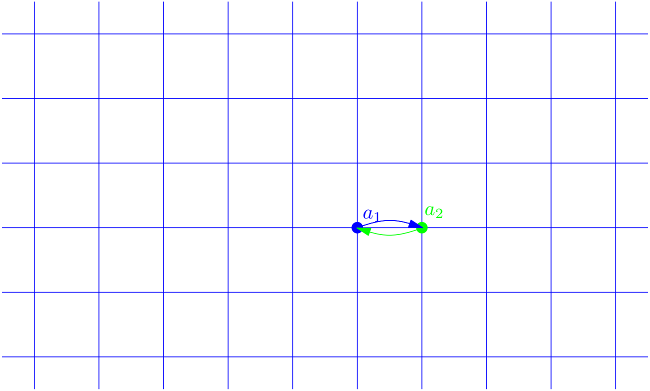
How to meet asynchronously two agents in the grid?



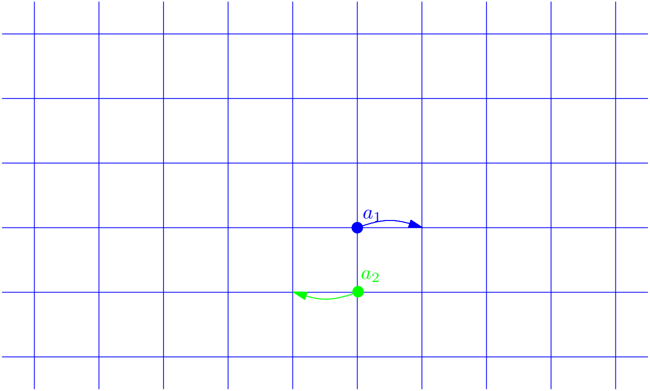
How to meet asynchronously two agents in the grid?



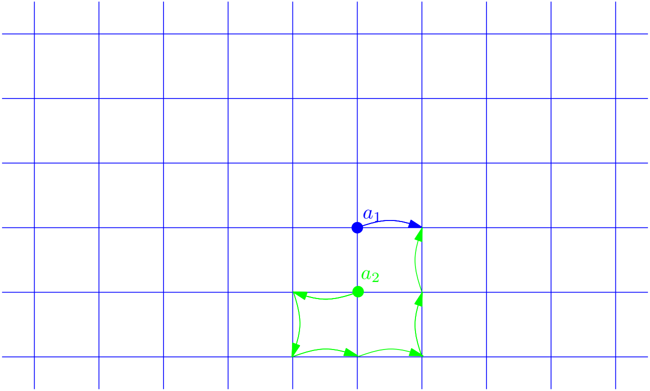
How to meet asynchronously two agents in the grid?



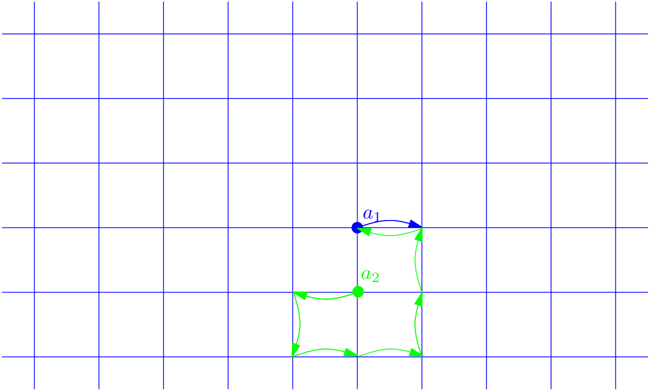
How to meet asynchronously two agents in the grid?



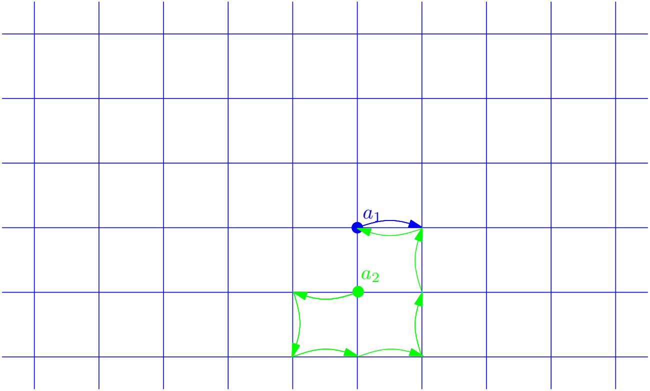
How to meet asynchronously two agents in the grid?



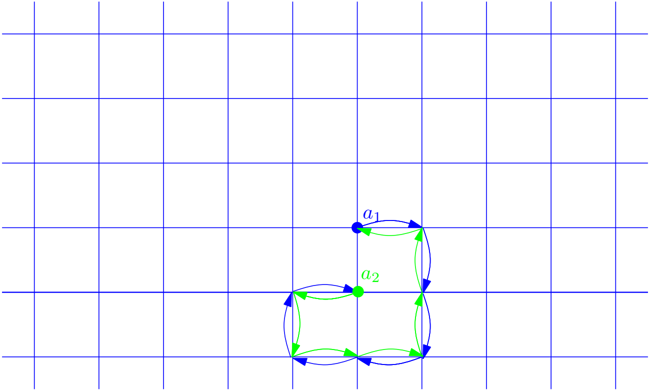
How to meet asynchronously two agents in the grid?



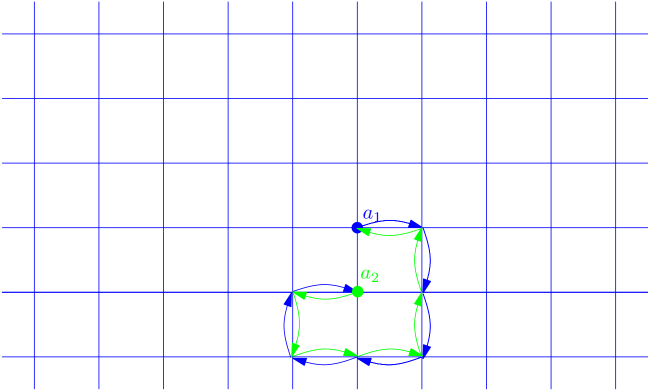
How to meet asynchronously two agents in the grid?



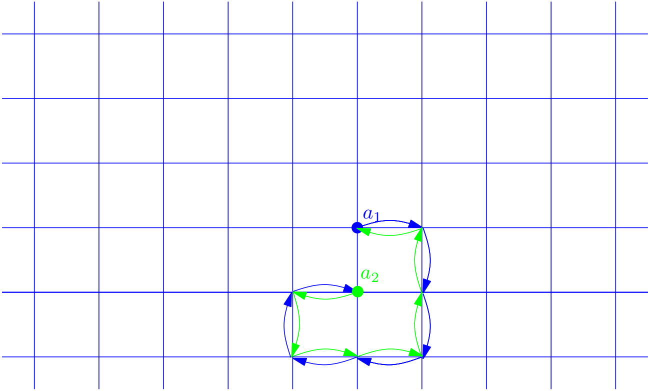
How to meet asynchronously two agents in the grid?



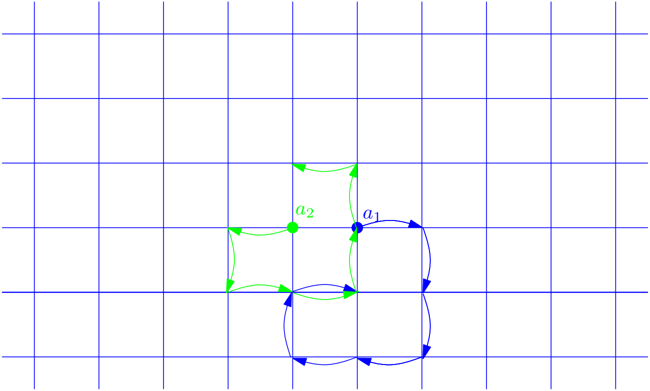
How to meet asynchronously two agents in the grid?



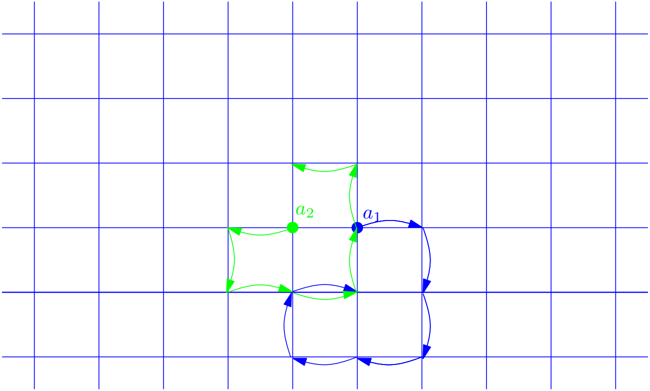
How to meet asynchronously two agents in the grid?



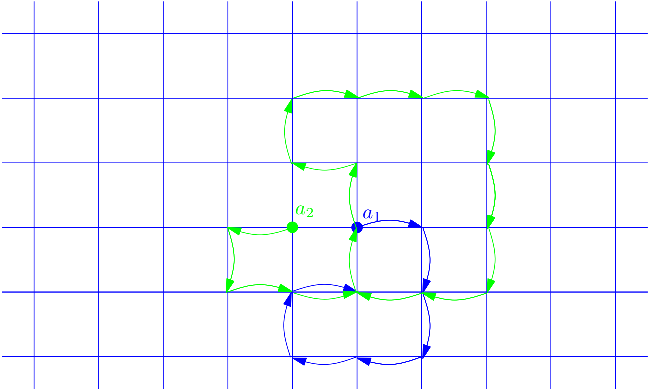
How to meet asynchronously two agents in the grid?



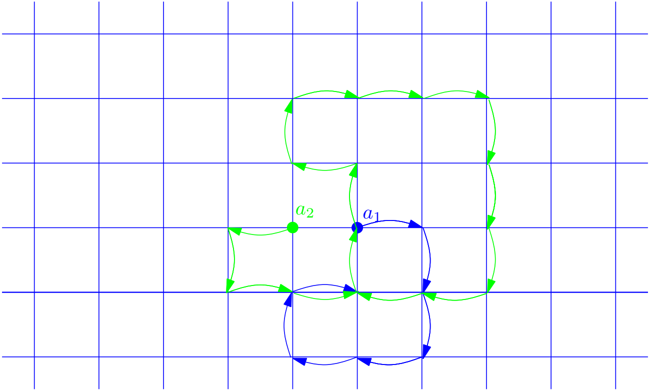
How to meet asynchronously two agents in the grid?



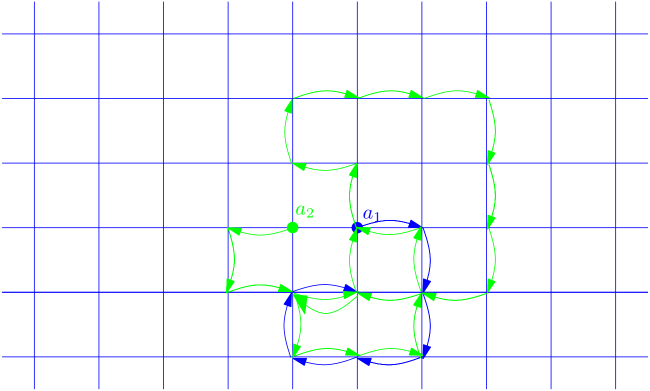
How to meet asynchronously two agents in the grid?



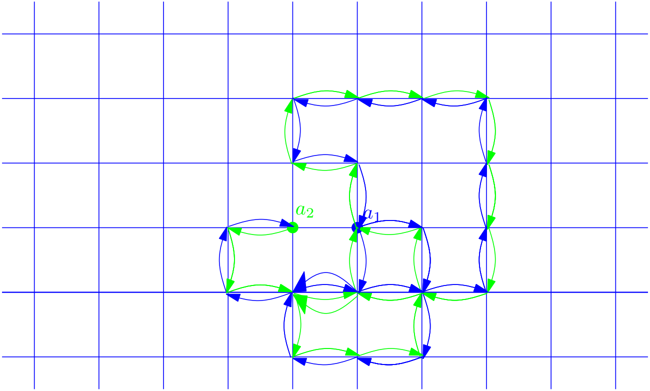
How to meet asynchronously two agents in the grid?



How to meet asynchronously two agents in the grid?



How to meet asynchronously two agents in the grid?



The rendezvous algorithm for a known graph

General idea of the algorithm

- We want to create a tunnel for every possible configuration of the algorithm (for each pair of agents and their possible initial positions in the graph)
- Each configuration is a triple consisting of two agents' identifiers and their relative position in the grid
- Enumerate all the configurations of the algorithm
- Iteratively, for any subsequent configuration in the enumeration, extend the routes of both involved agents to form a tunnel

The rendezvous algorithm for an unknown graph

General idea of the algorithm

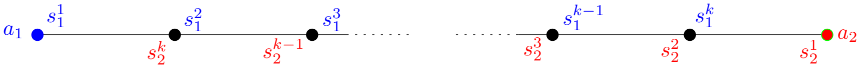
- Each configuration is a quadruple consisting of two agents' identifiers and two sequences of ports potentially traversed by the routes of both agents
- Enumerate all the configurations of the algorithm
- Iteratively, for any subsequent configuration in the enumeration, **if**
  - the quadruple contains your identifier, and
  - your route corresponds to a valid path in the graph, say from node v to w
  - the other route corresponds to the reverse path from w to v
- **then** extend the route to form a tunnel with the other route

Rendezvous algorithm

Initial configuration : quadruple (L_1, S_1, L_2, S_2)

- S_1, S_2 : two sequences of integers of same length.
- L_1, L_2 : identifiers of the two agents.

S_1 and S_2 correspond to sequence of ports number of a path linking the two starting positions of the agents.



$$S_1 = (s_1^1, s_1^2, s_1^3, \dots, s_1^{k-1}, s_1^k)$$
$$S_2 = (s_2^1, s_2^2, s_2^3, \dots, s_2^{k-1}, s_2^k)$$

Pseudo-code of the algorithm

Rendezvous algorithm

For each quadruple $\varphi_k = (i, S_1, j, S_2)$ **Do**

  If *identifier*(Agent) = i **Then**

    follow port s_1^1 then s_1^2 ... then s_1^k

    If $(\forall i \leq k, s_2^{k+1-i}$ is the outgoing port of $s_1^i)$**Then**

      execute route of agent j until quadruple φ_{k-1}

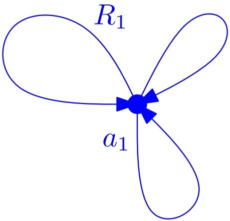
      extend the route of the agent i such that the routes of agents i et j form a tunnel

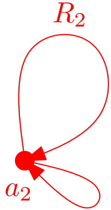
    return to the starting point

  If Identifier(Agent) = j **Then**

    Do the same with S_2 instead of S_1 and j instead of i.

Execution of the step of the main loop corresponding to the initial configuration of the agents

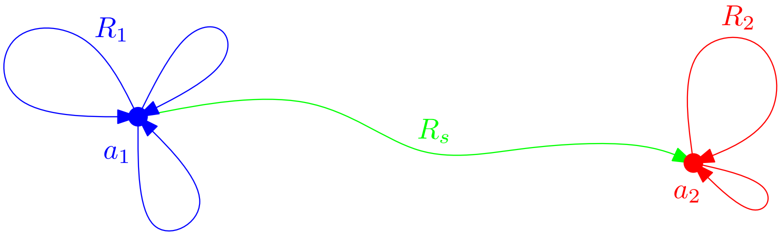




Route of agent a_1: R_1

Route of agent a_2: R_2

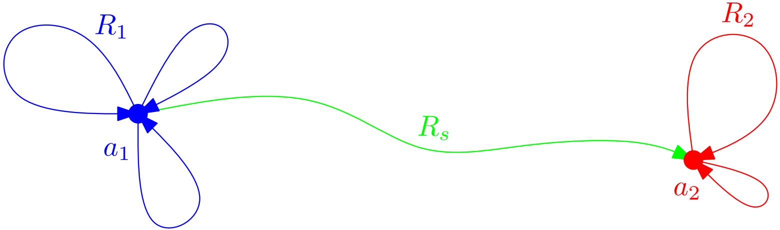
Execution of the step of the main loop corresponding to the initial configuration of the agents



Route of agent a_1: $R_1 + R_s$

Route of agent a_2: $R_2 + R_s^{-1}$

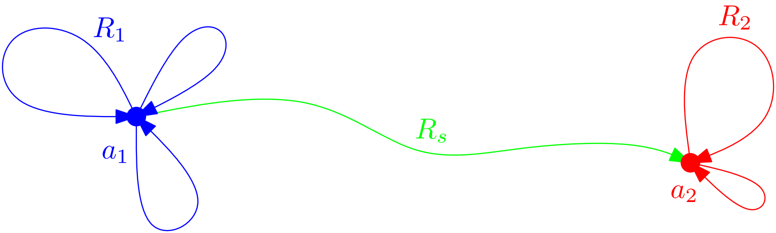
Execution of the step of the main loop corresponding to the initial configuration of the agents



Route of agent a_1: $R_1 + R_s + R_2$

Route of agent a_2: $R_2 + R_s^{-1} + R_1$

Execution of the step of the main loop corresponding to the initial configuration of the agents



Route of agent a_1: $R_1 + R_s + R_2 + R_s^{-1} + R_1^{-1} + R_s + R_2^{-1}$

Route of agent a_2: $R_2 + R_s^{-1} + R_1 + R_s + R_2^{-1} + R_s^{-1} + R_1^{-1}$

Cost of the rendezvous exponential or polynomial?

Open problem

Does there exist an asynchronous deterministic algorithm in finite graph such that the cost of rendezvous is polynomial in the size of the graph?

Thanks for your attention !