Deterministic Rendezvous in Networks

Andrzej Pelc

Université du Québec en Outaouais, Canada

Outline

Outline

Problem

Two or more mobile entities, starting at distinct initial positions, have to meet. This task, called rendezvous, has numerous applications in domains ranging from human interaction and animal behavior to programming of autonomous mobile robots and software agents.

Problem

Two or more mobile entities, starting at distinct initial positions, have to meet. This task, called rendezvous, has numerous applications in domains ranging from human interaction and animal behavior to programming of autonomous mobile robots and software agents.

Examples (human)

- Two astronauts land in two distant places on a spherical body, without any orientation, and have to minimize the expected time of getting within their detection radius.
- Two parachutists that have to meet after separately parachuting from a plane.
- Rescuers want to find a lost hiker in the mountains.

Examples (animals)

- Gathering of migratory birds or undersea animals.
- Penguin parents that need to find their offspring coming back with food.
- Searching for a mate for species living in small densities over a large territory

Examples (networks and robotics)

- Software agents, i.e., mobile pieces of software that travel in a communication network in order to perform maintenance of its components or to collect data distributed in nodes of the network. Such software agents need to meet periodically, in order to exchange collected data and plan further moves.
- Autonomous mobile robots that start in different locations of a planar terrain or a labyrinth and have to meet, e.g., in order to exchange information obtained while exploring the terrain and coordinate further actions.

Main alternative scenarios

- The environment in which the agents navigate: it can be either a terrain in the plane, or a network modeled as an undirected graph.
- In the second case, the agents may be capable or incapable of marking nodes.
- The way in which the agents move: it can be either deterministic or randomized.

In this talk we survey results concerning deterministic rendezvous in networks with unmarked nodes.

Main alternative scenarios

- The environment in which the agents navigate: it can be either a terrain in the plane, or a network modeled as an undirected graph.
- In the second case, the agents may be capable or incapable of marking nodes.
- The way in which the agents move: it can be either deterministic or randomized.

In this talk we survey results concerning deterministic rendezvous in networks with unmarked nodes.

Common assumptions

- Modeling the network as a simple undirected connected graph.
- Anonymity of the underlying network:
  the absence of distinct names of nodes that can be perceived by the navigating agents.
- A node of degree d has ports $0, 1, \ldots, d-1$ corresponding to the incident edges. Ports at each node can be perceived by an agent visiting this node, but there is no coherence assumed between port labelings at different nodes.

Outline

Breaking symmetry

Three ways to break symmetry in the deterministic scenario:

- Distinguishing the agents: each of them has a label and the labels are different. Each agent knows its label, but we do not need to assume that it knows the label of the other agent.
- Exploiting either non-symmetries of the network itself, or the differences of the initial positions of the agents, even in a symmetric network.
- Marking nodes (unmovable tokens, movable tokens, whiteboards): NOT in this talk.

Amount of memory

The memory of the agents is unbounded and agents are modeled as Turing machines,
or
the memory is bounded, in which case the model of input/output automata (finite state machines) is usually used.

Synchronous vs. asynchronous scenario

In the synchronous scenario agents move from node to node in synchronous rounds and the meeting has to occur at a node.

Asynchrony is captured in two different ways.

- In one model, an adversary decides when each agent moves, but the move itself is instantaneous, thus it is possible to require meeting at a node, as previously.
- In the second model, the agent chooses an edge but the adversary determines the actual walk of the agent on this edge and can, e.g., speed up or slow down the agent. Under this scenario it may be impossible to meet at a node, and thus the requirement is relaxed to that of meeting at a node or inside an edge.

Synchronous vs. asynchronous scenario

In the synchronous scenario agents move from node to node in synchronous rounds and the meeting has to occur at a node. Asynchrony is captured in two different ways.

- In one model, an adversary decides when each agent moves, but the move itself is instantaneous, thus it is possible to require meeting at a node, as previously.
- In the second model, the agent chooses an edge but the adversary determines the actual walk of the agent on this edge and can, e.g., speed up or slow down the agent. Under this scenario it may be impossible to meet at a node, and thus the requirement is relaxed to that of meeting at a node or inside an edge.

What is being sought, apart from meeting

- Feasibility: for what classes of networks and what initial positions is rendezvous possible under a particular scenario.
- Minimizing rendezvous cost: the (maximum or average) number of steps made by an agent until rendezvous.
- Minimizing memory with which agents have to be equipped in order to solve the rendezvous problem in a given class of networks.

Outline

Synchronous rendezvous

Agents move in synchronous steps. In every step, an agent may either remain at the same node or move to an adjacent node. Rendezvous means that all agents are at the same node in the same step. Agents that cross each other when moving along the same edge, do not notice this fact.

Two scenarios:

- simultaneous startup, when both agents start executing the algorithm at the same time,
- arbitrary delay, when starting times are arbitrarily decided by an adversary.

Synchronous rendezvous

Agents move in synchronous steps. In every step, an agent may either remain at the same node or move to an adjacent node. Rendezvous means that all agents are at the same node in the same step. Agents that cross each other when moving along the same edge, do not notice this fact.
Two scenarios:

- simultaneous startup, when both agents start executing the algorithm at the same time,
- arbitrary delay, when starting times are arbitrarily decided by an adversary.

Cost of synchronous rendezvous

[Dessmark, Fraigniaud, Kowalski, Pelc (Algorithmica 2006)]

Agents have different labels, which are positive integer numbers, and each agent knows its own label (which is a parameter of the common algorithm that they use), but is unaware of the label of the other agent. In general, agents do not know the topology of the graph in which they have to meet.

The smaller of the two labels is denoted by l. The delay (the difference between startup times of the agents) is denoted by τ. n denotes the number of nodes in the graph, and D – the distance between initial positions of agents.

Cost of synchronous rendezvous

[Dessmark, Fraigniaud, Kowalski, Pelc (Algorithmica 2006)]

Agents have different labels, which are positive integer numbers, and each agent knows its own label (which is a parameter of the common algorithm that they use), but is unaware of the label of the other agent. In general, agents do not know the topology of the graph in which they have to meet.

The smaller of the two labels is denoted by l. The delay (the difference between startup times of the agents) is denoted by τ. n denotes the number of nodes in the graph, and D – the distance between initial positions of agents.

Cost of synchronous rendezvous (cont.)

Theorem

Rendezvous can be completed at cost $O(n + \log l)$ on any n-node tree, even with arbitrary delay. This is optimal.

Theorem

With simultaneous startup, the optimal cost of rendezvous on any ring is $\Theta(D \log l)$.

Cost of synchronous rendezvous (cont.)

Theorem

Rendezvous can be completed at cost $O(n + \log l)$ on any n-node tree, even with arbitrary delay. This is optimal.

Theorem

With simultaneous startup, the optimal cost of rendezvous on any ring is $\Theta(D \log l)$.

Cost of synchronous rendezvous (cont.)

Theorem

With arbitrary delay, $\Omega(n + D \log l)$ is a lower bound on the cost required for rendezvous in a n-node ring. Under this scenario, two rendezvous algorithms for the ring : an algorithm with cost $O(n \log l)$, for known n, and an algorithm with cost $O(l\tau + ln^2)$, if n is unknown.

For arbitrary connected graphs, the main contribution of [DFKP] is a deterministic rendezvous algorithm with cost polynomial in n, τ and $\log l$.

Theorem

There exists an algorithm that solves the rendezvous problem for any n-node graph G, for any labels $L_1 > L_2 = l$ of agents and for any delay τ between startup times, in cost $\mathcal{O}(n^5\sqrt{\tau \log l} \log n + n^{10} \log^2 n \log l)$.

Cost of synchronous rendezvous (cont.)

Problem:
Does there exist a deterministic rendezvous algorithm for arbitrary connected graphs with cost polynomial in n and l (or even in n and $\log l$) but independent of τ?
A positive answer due to [Kowalski, Malinowski (TCS 2008)]

Theorem

There exists a rendezvous algorithm, working in arbitrary connected graphs for an arbitrary delay τ, whose complexity is $O(\log^3 l + n^{15} \log^{12} n)$, i.e., is independent of τ and polynomial in n and $\log l$.

Cost of synchronous rendezvous (cont.)

Both above algorithms use a non-constructive ingredient, i.e, a combinatorial object whose existence is proved using the probabilistic method. Each of the agents can deterministically find such an object by exhaustive search, which keeps the algorithm deterministic, but may significantly increase the time of local computations. In the described model the time of these computations does not contribute to cost which is measured by the number of steps, regardless of the time taken to compute each step.

Cost of synchronous rendezvous (cont.)

Does there exists a rendezvous algorithm for which both the cost and the time of local computations are polynomial in n and $\log l$? Such an algorithm would have to eliminate any non-constructive ingredients.

[Ta-Shma, Zwick (SODA 2007)]: Yes.
(Using Universal Exploration Sequences)

Cost of synchronous rendezvous (cont.)

Does there exists a rendezvous algorithm for which both the cost and the time of local computations are polynomial in n and $\log l$? Such an algorithm would have to eliminate any non-constructive ingredients.

[Ta-Shma, Zwick (SODA 2007)]: Yes.
(Using Universal Exploration Sequences)

View of a node

Definition

Let G be a graph and v a node of G. The view from v is the infinite rooted tree $V(v)$ with labeled ports, defined recursively as follows. $V(v)$ has the root x_0 corresponding to v. For every node v_i, $i = 1, \ldots, k$, adjacent to v, there is a neighbor x_i in $V(v)$ such that the port number at v corresponding to edge $\{v, v_i\}$ is the same as the port number at x_0 corresponding to edge $\{x_0, x_i\}$, and the port number at v_i corresponding to edge $\{v, v_i\}$ is the same as the port number at x_i corresponding to edge $\{x_0, x_i\}$. Node x_i, for $i = 1, \ldots, k$, is now the root of the view from v_i.

Symmetric nodes

A pair (u, v) of distinct nodes is called *symmetric*, if these nodes have the same view of the graph. Initial positions forming a symmetric pair of nodes are crucial when considering the feasibility of rendezvous in arbitrary graphs by identical (anonymous) agents. Indeed, rendezvous is feasible, if and only if the initial positions of the agents are not a symmetric pair.

Optimizing memory size for synchronous rendezvous by identical agents

Agents are identical copies A and A' of the same abstract state machine, starting at two distinct nodes v_A and $v_{A'}$. We will refer to such identical machines as a pair of agents.

A pair of agents is said to solve the rendezvous problem with delay τ in a class C of graphs, if, for any graph in the class C and for any initial positions that are not symmetric, both agents are eventually in the same node of the graph in the same round, provided that they start with delay τ.

Optimizing memory size for synchronous rendezvous by identical agents

Agents are identical copies A and A' of the same abstract state machine, starting at two distinct nodes v_A and $v_{A'}$. We will refer to such identical machines as a pair of agents.

A pair of agents is said to solve the rendezvous problem with delay τ in a class C of graphs, if, for any graph in the class C and for any initial positions that are not symmetric, both agents are eventually in the same node of the graph in the same round, provided that they start with delay τ.

Optimizing memory size (cont.)

The optimization criterion is the size of the memory of the agents, measured by the number of states of the corresponding automaton, or equivalently by the number of bits on which these states are encoded. An automaton with K states requires $\Theta(\log K)$ bits of memory.

Optimizing memory size: rv in trees

[Fraigniaud, Pelc (DISC 2008)]

Theorem

The minimum size of memory of the agents that can solve the rendezvous problem in the class of trees with at most n nodes is $\Theta(\log n)$.

A rendezvous algorithm for arbitrary delay τ is provided, that uses only a logarithmic number of memory bits, and it is observed that $\Omega(\log n)$ is also a lower bound on the number of bits of memory that enable rendezvous in all trees of size linear in n.

Optimizing memory size: rv in trees (cont.)

Due to the above lower bound, a universal pair of finite agents achieving rendezvous in the class of all trees cannot exist. However, the lower bound uses a counterexample of a tree with maximum degree linear in the size of the tree.

Does there exist a pair of finite agents solving the rendezvous problem in all trees of bounded degree? The answer is no.

Optimizing memory size: rv in trees (cont.)

Due to the above lower bound, a universal pair of finite agents achieving rendezvous in the class of all trees cannot exist. However, the lower bound uses a counterexample of a tree with maximum degree linear in the size of the tree.

Does there exist a pair of finite agents solving the rendezvous problem in all trees of bounded degree? The answer is no.

Optimizing memory size: rv in trees (cont.)

Due to the above lower bound, a universal pair of finite agents achieving rendezvous in the class of all trees cannot exist. However, the lower bound uses a counterexample of a tree with maximum degree linear in the size of the tree.

Does there exist a pair of finite agents solving the rendezvous problem in all trees of bounded degree? The answer is no.

Optimizing memory size: rv in trees (cont.)

Theorem

For any pair of identical finite agents, there is a line on which these agents cannot solve the rendezvous problem, even with simultaneous start.

As a function of the size of the trees, this impossibility result indicates a lower bound $\Omega(\log \log n)$ bits on the memory size for rendezvous in bounded degree trees of at most n nodes.

Optimizing memory size: rv in trees (cont.)

[Fraigniaud, Pelc (SPAA 2010)]

Theorem

If the delay between startup times of agents is arbitrary, then the lower bound on memory required for rendezvous is $\Omega(\log n)$ bits, even for the line of length n.

This lower bound matches the upper bound from [FP 2008], which shows that the minimum size of memory of the agents that can solve the rendezvous problem in the class of bounded degree trees with at most n nodes is $\Theta(\log n)$.

Optimizing memory size: rv in trees (cont.)

By contrast, for simultaneous start, it is shown in [FP 2010] that the amount of memory needed for rendezvous depends on two parameters of the tree: the number n of nodes and the number ℓ of leaves.

Theorem

There exist two identical agents with $O(\log \ell + \log\log n)$ bits of memory that solve the rendezvous problem with simultaneous start in all trees with n nodes and ℓ leaves.

For the class of trees with $O(\log n)$ leaves, this shows an exponential gap in minimum memory size needed for rendezvous between the scenario with arbitrary delay and with delay zero.

Optimizing memory size: rv in trees (cont.)

By contrast, for simultaneous start, it is shown in [FP 2010] that the amount of memory needed for rendezvous depends on two parameters of the tree: the number n of nodes and the number ℓ of leaves.

Theorem

There exist two identical agents with $O(\log \ell + \log \log n)$ bits of memory that solve the rendezvous problem with simultaneous start in all trees with n nodes and ℓ leaves.

For the class of trees with $O(\log n)$ leaves, this shows an exponential gap in minimum memory size needed for rendezvous between the scenario with arbitrary delay and with delay zero.

Optimizing memory size: rv in trees (cont.)

The following matching lower bound is also shown in [FP 2010]:

Theorem

For infinitely many integers ℓ, there is a class of arbitrarily large trees with maximum degree 3 and with ℓ leaves, for which rendezvous requires $\Omega(\log \ell)$ bits of memory.

This lower bound, together with the result from [FP 2008] showing that $\Omega(\log \log n)$ bits of memory are required for rendezvous in the line of length n, implies that the upper bound $O(\log \ell + \log \log n)$ cannot be improved even for trees with maximum degree 3.

Optimizing memory size: rv in arbitrary graphs

[Czyzowicz, Kosowski, Pelc (PODC 2010)]

Theorem

The minimum size of the memory of agents that guarantees deterministic rendezvous when it is feasible, is $\Theta(\log n)$, where n is the size of the graph, regardless of the delay between the startup times of the agents.

The authors construct identical agents equipped with $\Theta(\log n)$ memory bits that solve the rendezvous problem in all graphs with at most n nodes, when starting with any delay τ,

and they prove a matching lower bound $\Omega(\log n)$ on the number of memory bits needed to accomplish rendezvous, even for simultaneous start. In fact, this lower bound is achieved already on the class of rings.

Optimizing memory size: rv in arbitrary graphs

[Czyzowicz, Kosowski, Pelc (PODC 2010)]

Theorem

The minimum size of the memory of agents that guarantees deterministic rendezvous when it is feasible, is $\Theta(\log n)$, where n is the size of the graph, regardless of the delay between the startup times of the agents.

The authors construct identical agents equipped with $\Theta(\log n)$ memory bits that solve the rendezvous problem in all graphs with at most n nodes, when starting with any delay τ, and they prove a matching lower bound $\Omega(\log n)$ on the number of memory bits needed to accomplish rendezvous, even for simultaneous start. In fact, this lower bound is achieved already on the class of rings.

Asynchronous rendezvous

Two models of asynchrony.

The first model is adapted from the CORDA model, originally used for rendezvous in the plane.

This model assumes that the agents are very weak in terms of memory (they cannot remember any past events), but they have significant sensory power (they can take snapshots of the entire network, including other agents' positions in it).

The Look-Compute-Move model

- Agents are anonymous and execute the same algorithm.
- They start at different nodes and operate in Look-Compute-Move cycles.
- In one cycle, a robot takes a snapshot of the current configuration (Look), then, based on the perceived configuration, makes a decision to stay idle or to move to one of its adjacent nodes (Compute), and in the latter case makes an instantaneous move to this neighbor (Move). Agents do not remember previous cycles.

The Look-Compute-Move model (cont.)

Cycles are performed asynchronously for each agent. This means that the time between Look, Compute, and Move operations is finite but unbounded, and is decided by the adversary for each agent in each cycle. The only constraint is that moves are instantaneous, and hence any agent performing a Look operation sees all other agents at nodes of the ring and not on edges, while performing a move.

Rendezvous in the ring

Agents initially situated at different nodes of the ring, have to gather at the same node (not determined in advance) and remain in it.
Main difficulty of rendezvous (gathering): agents have to break symmetry by agreeing on a common meeting node, in spite of the asynchrony in executing cycles.

Multiplicity detection

An important capability studied in the literature on rendezvous is the multiplicity detection. This is the ability of the agents to perceive, during the Look operation, if there is one or more agents at a given node (or at a given point in the case of the plane). In the case of the ring, it is proved that without this capability, rendezvous of more than one agent is always impossible. Thus multiplicity detection is assumed.

During a Look operation, an agent can only tell if at some node there are no agents, there is one agent, or there are more than one agents: an agent does not see a difference between a node occupied by a or b agents, for distinct $a, b > 1$.

Impossibility results

[Klasing, Markou, Pelc: TCS 2008]

Proposition

1. Gathering any 2 robots is impossible on any ring.
2. If multiplicity detection is not available then gathering any $k > 1$ robots is impossible on any ring.

Impossibility results (cont.)

Theorem

Gathering is impossible for any periodic configuration.

A configuration is edge-edge symmetric if it has exactly one axis of symmetry and this axis does not contain any node.

Theorem

Gathering is impossible for any edge-edge symmetric configuration.

Impossibility results (cont.)

Theorem

Gathering is impossible for any periodic configuration.

A configuration is edge-edge symmetric if it has exactly one axis of symmetry and this axis does not contain any node.

Theorem

Gathering is impossible for any edge-edge symmetric configuration.

Impossibility results (cont.)

Theorem

Gathering is impossible for any periodic configuration.

A configuration is edge-edge symmetric if it has exactly one axis of symmetry and this axis does not contain any node.

Theorem

Gathering is impossible for any edge-edge symmetric configuration.

Gathering an odd number of agents

Theorem

Gathering is possible for any non-periodic configuration of an odd number of agents.

This solves the gathering problem on the ring for any odd number of agents.

Gathering an odd number of agents

Theorem

Gathering is possible for any non-periodic configuration of an odd number of agents.

This solves the gathering problem on the ring for any odd number of agents.

Gathering in the remaining case

[Klasing, Kosowski, Navarra (OPODIS 2008)]

Theorem

Gathering is possible for any non-periodic configuration without edge-edge symmetry of more than 18 agents.

Open problem: Characterize gatherable configurations of n agents, where $4 \leq n \leq 18$ and n is even.

The second model of asynchrony

- Rendezvous can occur either at a node or inside an edge.
- Agents have distinct labels and each of them knows its own label, but not that of the other agent.
- Nodes are anonymous and ports at each node are labeled in a possibly incoherent way.

The second model of asynchrony (cont.)

Since meetings inside an edge are allowed, unwanted crossings of edges have to be avoided. Thus, we consider an embedding of the underlying graph in the three-dimensional Euclidean space, with nodes of the graph being points of the space and edges being pairwise disjoint line segments joining them. For any graph such an embedding exists. Agents are modeled as points moving inside this embedding.

The second model of asynchrony (cont.)

Asynchrony is modeled as follows:

When the agent, situated at a node v at time t_0 has to traverse an edge modeled as a segment $[v, w]$, the adversary performs the following choice.

It selects a time point $t_1 > t_0$ and any continuous function $f : [t_0, t_1] \longrightarrow [v, w]$, with $f(t_0) = v$ and $f(t_1) = w$. This function models the actual movement of the agent inside the line segment $[v, w]$ in the time period $[t_0, t_1]$.

The second model of asynchrony (cont.)

Hence this movement can be at arbitrary speed, the agent may be forced by the adversary to go back and forth, as long as it does not leave the segment and the movement is continuous. We say that at time $t \in [t_0, t_1]$ the agent is in point $f(t) \in [v, w]$. Moreover, the adversary chooses the starting time of the agent. Hence an agent's trajectory is represented by the concatenation of the functions chosen by the adversary for consecutive edges that the agent traverses.

The second model of asynchrony (cont.)

For a given algorithm, given starting nodes of agents and a given sequence of adversarial decisions in an embedding of a graph G, a rendezvous occurs, if both agents are at the same point of the embedding at the same time.

Rendezvous is feasible in a given graph, if there exists an algorithm for agents such that for any embedding of the graph, any (adversarial) choice of two distinct labels of agents, any starting nodes and any sequences of adversarial decisions, the rendezvous does occur.

The cost of rendezvous is defined as the worst-case number of edge traversals by both agents (the last partial traversal counted as a complete one for both agents), where the worst case is taken over all decisions of the adversary.

Minimizing the cost (infinite line)

[De Marco, Gargano, Kranakis, Krizanc, Pelc, Vaccaro (TCS 2006)]

Theorem

For agents initially situated at a distance D in an infinite line, there is a rendezvous algorithm with cost $O(D|L_{min}|^2)$ when D is known, and $O((D+|L_{max}|)^3)$ if D is unknown, where $|L_{min}|$ and $|L_{max}|$ are the lengths of the shorter and longer label of the agents, respectively.

This has been improved in [Stachowiak (SOFSEM 2009)] to $O(D\log^2 D + D\log D|L_{max}| + D|L_{min}|^2 + |L_{max}||L_{min}|\log|L_{min}|)$ for unknown D.

Minimizing the cost (ring)

[DGKKPV 2006]

Theorem

For agents initially situated at a distance D in a n-node ring, there is a rendezvous algorithm with cost $O(n|L_{min}|)$ (and this is optimal), if the size n of the ring is known, and with cost $O(n|L_{max}|)$, if it is unknown.

In both these algorithms the knowledge of the initial distance D between agents is not assumed, and for D of the order of n their complexity is better than that of infinite line algorithms. On the other hand, for small D and small labels of agents, the opposite is true.

Feasibility

[DGKKPV 2006]

Theorem

Asynchronous rendezvous is feasible for arbitrary graphs, if an upper bound on the size of the graph is known.

As an open problem, the authors state the question if asynchronous deterministic rendezvous is feasible in arbitrary graphs of unknown size. The solution from [DGKKPV 2006] heavily uses the knowledge of the upper bound on the size.

Feasibility (cont.)

This problem has been solved in [Czyzowicz, Labourel, Pelc (SODA 2010)] in the following strong way:

Theorem

There exists an algorithm that accomplishes asynchronous rendezvous in any connected countable (finite or infinite) graph, for arbitrary starting nodes.

Hence not only is rendezvous always possible, without the knowledge of any upper bound on the size of a finite (connected) graph, but it is also possible for all infinite (countable and connected) graphs.

Feasibility (cont.)

Does there exist a deterministic asynchronous rendezvous algorithm, working for all connected finite unknown graphs, with cost polynomial in the labels of the agents and in the size of the graph?

Rendezvous with GPS on the grid

- Each agent knows its initial position on the grid, but not the position of the other agent.
- Ports at each node of the grid are coherently labeled N,E,S,W.

Bampas, Czyzowicz, Gasieniec, Ilcinkas, Labourel [DISC 2010]:

Theorem

Rendezvous can be achieved at cost $O(D^2 polylog(D))$, where D is the initial distance between the agents.

Rendezvous with GPS on the grid

- Each agent knows its initial position on the grid, but not the position of the other agent.
- Ports at each node of the grid are coherently labeled N,E,S,W.

Bampas, Czyzowicz, Gasieniec, Ilcinkas, Labourel [DISC 2010]:

Theorem

Rendezvous can be achieved at cost $O(D^2 polylog(D))$, where D is the initial distance between the agents.

Outline

Important assumptions

Assumptions that turn out to have a significant impact concern labeling vs. anonymity of the agents, their synchronous vs. asynchronous behavior, the amount of memory with which they are equipped, and the capacity (vs. lack of it) of marking nodes.

Research directions

Problem 1

How the capacity of communication between agents influences the feasibility and performance of rendezvous?

Marking nodes by tokens or using whiteboards is only one possible way of communicating. It is intrinsically local, i.e., it assumes that the agent receiving information must visit a node at which information has been left. A much more powerful way of communicating would be wireless, in which agents currently at remote nodes of the network can nevertheless exchange messages.

Research directions (cont.)

Problem 2

What are the trade-offs between memory of the agents and their sensory capabilities?

In the model adapted from CORDA, an extreme scenario is considered: agents cannot remember any information from the previous cycles, but they can take a snapshot of the entire network during the Look action. It is more realistic to assume that agents have slightly larger memory of past events, but their sensory capabilities are more limited, e.g, they can only perceive the part of the configuration at a given radius from their current position. Studying feasibility of rendezvous under such more balanced scenarios would involve characterizing initial configurations for which rendezvous is possible.

Research directions (cont.)

Problem 3

Rendezvous under the paradigm of advice.

It has been mentioned that providing nodes with distinct labels greatly simplifies the task of rendezvous, as agents can meet at a predetermined node. However, this requires adding a large amount of information, this information has to be distributed among nodes and be visible to the agents.

Research directions (cont.)

The problem stated in the spirit of the advice paradigm is:

What are the trade-offs between the amount of information (advice) given to the anonymous agents and the efficiency of rendezvous?