Mobile Agent Rendezvous: A Survey

Evangelos Kranakis[1], Danny Krizanc[2], and Sergio Rajsbaum[3]

[1] School of Computer Science, Carleton University, Ottawa, ON, Canada.
[2] Department of Mathematics and Computer Science, Wesleyan University, Middletown, Connecticut 06459, USA.
[3] Instituto de Matemáticas, Universidad Nacional Autónoma de México (UNAM), Ciudad Universitaria, D. F. 04510, Mexico.

Abstract. Recent results on the problem of mobile agent rendezvous on distributed networks are surveyed with an emphasis on outlining the various approaches taken by researchers in the theoretical computer science community.

1 Introduction

Consider the following problem original proposed by Alpern [1] (as quoted in [3]):

> Two astronauts land on a spherical body that is much larger than the detection radius (within which they can see each other). The body does not have fixed orientation in space, nor does it have an axis of rotation, so that no common notion of position or direction is available to the astronauts for coordination. Given unit walking speeds for both astronauts, how should they move about so as to minimize the expected meeting time T (before they come within the detection radius)?

This is just one version of a problem that has been studied under many guises under most of which it is referred to as *rendezvous*. In all settings, a set of agents are placed in a domain and are required to all meet at the same place and time within the domain, i.e., rendezvous. The settings differ mainly in the types and properties of the agents and the types and properties of domains. Besides the astronaut example above, rendezvous has been studied in settings as various as ships at sea, mother and child at a mall and autonomous robots on a hilly terrain.

Recently the theoretical computer science community has taken up the challenge of the problem of rendezvous for autonomous software agents moving through a distributed network. Requiring such agents to meet in order to synchronize, share information, divide up duties, etc. would seem to be a natural fundamental operation useful as a subroutine in more complicated applications such as web-crawling, peer-to-peer lookup, meeting scheduling, etc. In this paper, we provide a short (perhaps biased) survey of recent work done on this version of the problem. The research done in other settings is extensive and many of the solutions can be applied here. But it is often the case that the models used and the concerns studied are sufficiently different as to require new approaches. Having said that, one ignores the earlier work at one's peril. For an excellent discussion of mainly continuous domains with randomized agents see the book by Alpern and Gal [3]. For work on robot rendezvous consider [26] and [28] as possible starting points.

2 The Model

The definition of rendezvous must begin with establishing the properties of the agents that will rendezvous and the domain in which rendezvous will occur. The model below captures the essence of what has been termed *the theory of mobile agent computing.*

2.1 Mobile agents

We are interested in modeling a set of software entities that act more or less autonomously from their originator and have the ability to move from node to node in a distributed network maintaining some sort of state with the nodes of the network providing some amount of (possibly longterm) storage and computational support. Either explicitly or implicitly such a mobile (software) agent has most often been modeled using a finite automaton consisting of a set of states and a transition function. The transition function takes as input the agent's current state as well as possibly the state of the node it resides in and outputs a new agent state, possible modifications to the current node's state and a possible move to another node. In some instances we consider probabilistic automaton which have available a source of randomness that is used as part of their input. Such agents are referred to as *randomized* agents.

An important property to consider is whether or not the agents are distinguishable, i.e., if they have distinct labels or identities. Agents without identities are referred to as *anonymous* agents. Anonymous agents are limited to running precisely the same program, i.e., they are identical finite automata. As the identity is assumed to be part of the starting state of the automaton, agents with identities have the potential to run different programs.

The knowledge the agent has about the network it is on and about the other agents can make a difference in the solvability and efficiency of rendezvous. For example, knowledge of the size of the network or its topology or the number of and identities of the other agents may be used as part of the program for rendezvous. If available to the agents, this information is assumed to part of its starting state. (One could imagine situations where the information is made available by the nodes of the network and not necessarily encoded in the agent.)

Other properties that may be considered in mobile agent computing include whether or not the agents have the ability to “clone” themselves, whether or not they have the ability to “merge” upon meeting (sometimes referred to as “sticky” agents) or whether or not they can send self-generated messages. At this point, most of the research on rendezvous ignores these properties and they will not be discussed below.

2.2 Distributed networks

The model of a distributed network is essentially inherited directly from the theory of distributed computing. We model the network by a graph whose vertices comprise the computing nodes and edges correspond to communication links.

The nodes of the network may or may not have distinct identities. In an *anonymous* network the nodes have no identities. In particular this means that an agent can not distinguish two nodes except perhaps by their degree. The outgoing edges of a node are usually thought of as distinguishable but an important distinction is made between a globally consistent edge-labelling versus a locally independent edge-labelling. A simple example is the case of a ring where clockwise and counterclockwise edges are marked consistently around the ring in one case, and the edges are arbitrarily - say by an adversary - marked 1 and 2 in the other case. If the labelling satisfies certain coding properties it is called a *sense of direction* [13]. Sense of direction has turned out to greatly effect the solvability and efficiency of solution of a number of problems in distributed computing and has been shown to be important in rendezvous as well.

Networks are also classified by how they deal with time. In a synchronous network there exists a global clock available to all nodes. This global clock is inherited by the agents. In particular it is usually assumed that in a single step an agent arrives at a node, performs some calculation, and exits the node and that all agents are performing these tasks "in sync". In an asynchronous network such a global clock is not available. The speed with which an agent computes or moves between nodes, while guaranteed to be finite, is not a priori determined.

Finally we have to consider the resources provided by the nodes to the agents. All nodes are assumed to provide enough space to store the agent temporarily and computing power for it to perform its tasks. (The case of malicious nodes refusing agents or even worse destroying agents - so-called *blackholes* - is also sometimes considered.) Beyond these basic services one considers nodes that might provide some form of long-term storage, i.e., state that is left behind when the agent leaves. In the rendezvous problem the idea of leaving an indistinguishable mark or *token* at a node (introduced in [5]) has been studied. More accommodating nodes might provide a *whiteboard* for agents to write messages to be left for themselves or for other agents.

3 The Rendezvous Problem

Given a particular agent model (e.g., deterministic, anonymous agents with knowledge they are on a ring of size n) and network model (e.g., anonymous, synchronous with tokens) a set of k agents distributed arbitrarily over the nodes of the network are said to *rendezvous* if after running their programs after some finite time they all occupy the same node of the network at the same time. It is generally assumed that two agents occupying the same node can recognize this fact (though in many instances this fact is not required for rendezvous to occur). As stated, rendezvous is assumed to occur at nodes. In some instances one considers the possibility of rendezvous on an edge, i.e., if both agents use the same edge (in opposite directions) at the same time. (For physical robots this makes sense. For software agents this perhaps is not so realistic but sometimes necessary to allow for the possibility of rendezvous at all - especially in instances where the network lacks a sense of direction.)

The first question one asks for an instance of rendezvous is whether or not it is solvable. There are many situations where it is not possible to rendezvous at all. This will depend upon both the properties of the agents (deterministic or randomized, anonymous or with identities, knowledge of the size of the network or not, etc.) and the network (synchronous or asynchronous, anonymous or with identities, tokens available or not, etc.). The solvability is also a function of the starting positions chosen for the agents. For example, if the agents start at the same node and can recognize this fact, rendezvous is possible in this instance. Given a situation where some starting positions are not solvable (i.e., rendezvous is not possible) but others are, we distinguish between algorithms that are guaranteed to finish for all starting positions, with successful rendezvous when possible but otherwise recognizing that rendezvous is impossible, versus algorithms that are only guaranteed to halt when rendezvous is possible. Algorithms of the former type are said to solve *rendezvous with detection.* (The distinction is perhaps analogous to Turing machines deciding versus accepting a language.)

For solvable instances of rendezvous one is interested in comparing the efficiency of different solutions. Much of the research focuses on the time required to rendezvous. In the synchronous setting the time is measured via the global clock. (In some situations, it makes a difference if the agents begin their rendezvous procedure at the same time or their is possible delay between start times.) In the asynchronous setting we adapt the standard time measures from the distributed computing model. Also of interest is the size of the program required by the agents to solve the problem. This is referred to as the memory requirement of the agents and is considered to be proportional to the base two logarithm of the number of states required by the finite state machine encoding the agent.

As is often the case, researchers are interested in examining the extremes in order to get an understanding of the limits a problem imposes. Over time it has become clear that for rendezvous symmetry (of the agents and the network) plays a central role in determining its solvability and the efficiency of its solutions. As such we divide our discussion below into the asymmetric and symmetric cases. For simplicity we restrict ourselves to the case of just two agents in most of the discussion below.

4 Asymmetric Rendezvous

Asymmetry in a rendezvous problem may arise from either the network or the agents.

4.1 Network asymmetry

A network is asymmetric if it has one or more uniquely distinguishable vertices. A simple example is the case of a network where all of the nodes have unique identities chosen from a subset of some totally ordered set such as the integers. In this case, the node labelled with the smallest identity (for example) is unique and may be used as a meeting point for a rendezvous algorithm. Uniqueness need not be conferred using node labels. For example, in a network where there is a unique node of degree one, it may be used as a focal point.

If a "map" of the graph with an agent's starting position marked on it is available to the agents then the problem of rendezvous is easily solved by just traversing the path to an agreed upon unique node. Algorithms that use an agreed upon meeting place are referred to by Alpern and Gal [3] as FOCAL strategies. In the case where the graph is not available in advance but the agents know that a focal point exists (e.g., they know the nodes are uniquely labelled and therefore there exists a unique minimum label node) this strategy reduces to the problem of graph traversal or graph exploration whereby all of the nodes (sometimes edges) of the graph are to be visited by an agent. This has been extensively studied in a number of contexts, e.g., [9, 25]. Much of the work in this area has looked at improving the efficiency (time or memory) for restricted classes of graphs, e.g., trees [11]. A closely related problem is that of robot exploration of an unknown environment with obstacles which can often be modeled using graphs. See for example [6].

4.2 Agent asymmetry

By agent asymmetry one generally means the agents have unique identities that allow them to act differently depending upon their values. In the simplest scenario of two agents, the agent with the smaller value could decide to wait at its starting position for the other agent to find it by exploring the graph as above. Alpern and Gal [3] refer to this as the Wait For Mommy (WFM) strategy and they show it to be optimal under certain conditions.

WFM depends upon the fact that the agents know in advance the identities associated with the other agents. In some situations this may be an unrealistic assumption. Yu and Yang [29] were the first to consider this problem. Under the assumption that the algorithm designer may assign the identities to the agents (as well as the existence of distinct whiteboards for each agent), they show that rendezvous may be achieved deterministically on a synchronous network in $O(nl)$ steps where n is the size of the network and l is the size of the identities assigned. The perhaps more interesting case where an adversary assigns the labels was first considered in [10]. Extensions to this work including showing rendezvous on an arbitrary graph is possible in time polynomial in n and l and that there exist graphs requiring $\Omega(n^2)$ time for rendezvous are described in [17, 18]. The case of an asynchronous network is considered in [8] where a (non-polynomial) upper bound is set for rendezvous in arbitrary graphs (assuming the agents have an upper bound on the size of the graph). Improvements (in some cases optimal) for the case of the ring network are discussed in each of the above papers.

5 Symmetric Rendezvous

In the case of symmetric rendezvous, both the (generally synchronous) network and the agents are assumed to be anonymous. Further one considers classes of networks that in the worst case contain highly symmetric networks that do not submit to a FOCAL strategy. As might be expected some mechanism is required to break symmetry in order for rendezvous to be possible. The use of randomization and of tokens to break symmetry have both been studied extensively.

5.1 Randomized rendezvous

Many authors have observed that rendezvous may be solved by anonymous agents on an anonymous network by having the agents perform a random walk. The expected time to rendezvous is then a (polynomial) function of the (size of the) network and is directly related to the cover time of the network. (See [24] for definitions relating to random walks.)

For example, it is straightforward to show that two agents performing a symmetric random walk on ring of size n will rendezvous within expected $O(n^2)$ time. This expected time can be improved by considering the following strategy (for a ring with sense of direction). Repeat the following until rendezvous is achieved: flip a (fair) coin and walk $n/2$ steps to the right if the result is heads, $n/2$ steps to the left if the result is tails. If the two agents choose different directions (which they do with probability $1/2$) then they will rendezvous (at least on an edge if not at a node). It is easy to see that expected time until rendezvous is $O(n)$. Alpern refers to this strategy as Coin Half Tour and studies it in detail in [2]. Note that the agents are required to count up to n and thus seem to require $O(\log n)$ bits of memory to perform this algorithm (whereas the straightforward random walk requires only a constant number of states to implement). This can be reduced to $O(\log \log n)$ bits and this can be shown to be tight [23] for achieving linear expected rendezvous time.

5.2 Rendezvous using tokens

The idea of using tokens or marks to break symmetry for rendezvous was first suggested in [5] and expanded upon for the case of the ring in [27]. The first observation to make is that rendezvous is impossible for deterministic agents with tokens (or whiteboards) on an even size ring when the agents start at distance $n/2$ as the agents will remain in symmetric positions indefinitely. However, this is the only starting position for the agents for which rendezvous is impossible. This leads one to consider algorithms for rendezvous with detection where rendezvous is achieved when possible and otherwise the agents detect they are in an impossible to rendezvous situation. In this case, a simple algorithm suffices (described here for the oriented case). Each agent marks their starting position with a token. They then travel once around the ring counting the distances between their tokens. If the two distances are the same, they halt declaring rendezvous impossible. If they are different they agree to meet (for example) in the middle of the shorter side.

Again, one observes that the algorithm as stated requires $O(\log n)$ bits of memory for each agent in order to keep track of the distances. Interestingly enough this can be reduced to $O(\log \log n)$ bits and this can be shown to be tight for unidirectional algorithms [20]. If we are allowed two movable tokens (i.e., the indistinguishable marks can be erased and written with up to two marks stored per node) then rendezvous with detection becomes possible with an agent of constant size [21].

Multi-agent rendezvous, i.e., more than two agents, on the ring is considered in [14, 16], the second reference establishing optimal memory bounds for the problem. Two agent rendezvous on the torus is studied in [21] where tradeoffs between memory and the number of (movable) tokens used are given. Finally [15] considers a model in which tokens may disappear or fail over time.

5.3 Whiteboards and blackholes

The use of whiteboards to achieve multiagent rendezvous on an arbitrary anonymous asynchronous network is studied in [4] where the problem is shown to be equivalent to that of leader election. In [12] rendezvous on an asynchronous ring with whiteboards in the presence of blackholes is considered.

6 Related Problems

As was mentioned above the rendezvous problem is intimately related under certain conditions to that of graph exploration [9, 25]. Besides this connection there are a number of “search” problems that are at least tangentially related to rendezvous in as much as techniques and/or models considered by researchers on these problems may be of some use to those working on rendezvous. Some examples are described below.

Imagine that the police have cornered a fugitive in a complicated warren of tunnels and rooms. They know that she is situated at one of the nodes and would like to capture her by searching the rooms. Their main concern is to decide what is the least number of police personnel necessary in order to be certain the fugitive does not evade capture. This number, referred to as the *search number* of a graph, was introduced by Megiddo et al. [22] and has since been studied by many authors.

Two friends check into an n room hotel. The hotel clerk is off duty and each doesn't know the room number of the other. They start calling rooms in order to find each other. What is the least number of calls they must make before at least one finds the other? This question was formalized and generalized as *mutual search* by Buhrman et al[7].

A friend has given you the name but not the address of a restaurant in downtown Manhattan. You start searching for the restaurant by asking passersby which direction the restaurant is in. Their answers are sometimes erroneous, even contradictory. What is the best strategy to use in order to find the restaurant quickly? This problem was formalized in [19].

7 Conclusions

Rendezvous is a natural and fundamental problem for mobile software agents traveling through a distributed network. This makes it a natural choice as a model problem to study in the development of the theory of mobile agent computing much like leader election and consensus form model problems for the traditional study of the theory of distributed algorithms. We have attempted to bring together the current state of our knowledge of the study of rendezvous in the theoretical computer science context. A lot of work remains to be done.

Acknowledgments

Research of the first author was supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants. Research of the third author was supported in part by PAPIIT-UNAM.

References

1. S. Alpern, Hide and Seek Games, Seminar, Institut fur Hohere Studien, Wien, July 1976.
2. S. Alpern, The Rendezvous Search Problem, *SIAM Journal of Control and Optimization*, 33, pp. 673-683, 1995.
3. S. Alpern and S. Gal, *The Theory of Search Games and Rendezvous*, Kluwer Academic Publishers, Norwell, Massachusetts, 2003.
4. L. Barriere, P. Flocchini, P. Fraigniaud, and N. Santoro, Election and Rendezvous of Anonymous Mobile Agents in Anonymous Networks with Sense of Direction, *Proceedings of the 9th Sirocco*, pp. 17-32, 2003.
5. V. Baston and S. Gal, Rendezvous Search When Marks Are Left at the Starting Points, *Naval Research Logistics*, 47, No. 6, pp. 722-731, 2001.
6. A. Blum, P. Raghavan and B. Schieber, Navigating in Unfamiliar Geometric Terrain, *SIAM Journal on Computing* 26 (1997), 110-137.
7. H. Buhrman, M. Franklin, J. Garay, J. Hoepman, J. Tromp and P. Vitanyi, Mutual Search, *Journal of the ACM* 46 (1999), 517-536.

8. G. De Marco, L Gargano, E. Kranakis, D. Krizanc, A. Pelc and U. Vacaro, Asynchronous deterministic rendezvous in graphs, *Proc. 30th MFCS*, 2005, 271-282.
9. X. Deng and C. H. Papadimitriou, Exploring an Unknown Graph, *Journal of Graph Theory* 32 (1999), 265-297.
10. A. Dessmark, P. Fraigniaud, and A. Pelc, Deterministic Rendezvous in Graphs, *Proc. 11th ESA*, 184-195, 2003.
11. K. Diks, P. Fraigniaud, E. Kranakis, A. Pelc, Tree Exploration with Little Memory, *Journal of Algorithms*, 51 (2004) 38-63.
12. S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro, Multiple Agents Rendezvous in a Ring in spite of a Black Hole, *Proc. Symposium on Principles of Distributed Systems (OPODIS '03)*, LNCS 3144, pp. 34-46, 2004.
13. P. Flocchini, B. Mans and N. Santoro, Sense of direction: definition, properties and classes, *Networks* 32 (1998), 29-53.
14. P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Multiple Mobile Agent Rendezvous in the Ring, *Proc. LATIN 2004*, LNCS 2976, pp. 599-608, 2004.
15. P. Flocchini, E. Kranakis, D. Krizanc, F. Luccio, N. Santoro and C. Sawchuk, Mobile Agent Rendezvous When Tokens Fail, *Proc. of Sirocco 2004*.
16. L. Gasieniec, E. Kranakis, D. Krizanc, X. Zhang, Optimal Memory Rendezvous of Anonymous Mobile Agents in a Uni-directional Ring, *Proc. of 32nd SOFSEM 2006*, to appear.
17. D. Kowalski and A. Pelc, Polynomial deterministic rendezvous in arbitrary graphs, *Proc. 15th ISAAC*, 2004.
18. D. Kowalski and A. Malinowski, How to meet in an anonymous network, *Proc. 13th Sirocco*, 2006.
19. E. Kranakis, and D. Krizanc, Searching with Uncertainty. *Proc. of 6th Sirocco*, 1999.
20. E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Mobile Agent Rendezvous Search Problem in the Ring, *Proc. International Conference on Distributed Computing Systems (ICDCS)*, pp. 592-599, 2003.
21. E. Kranakis, D. Krizanc, E. Markou, Mobile Agent Rendezvous in a Synchronous Torus. *Proc. of LATIN*, 653-664, 2006.
22. N. Megiddo, S. Hakimi, M. Garey, D. Johnson and C. Papadimitriou. The Complexity of Searching a Graph, *Journal of the ACM* 35 (1988), 18-44.
23. P. Morin, personal communication.
24. R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, 1995.
25. P. Panaite and A. Pelc, Exploring Unknown Undirected Graphs, *Journal of Algorithms* 33 (1999), 281-295.
26. N. Roy and G. Dudek, Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations, *Autonomous Robots* 11 (2001), 117-136.
27. C. Sawchuk, *Mobile Agent Rendezvous in the Ring*, PhD thesis, Carleton University, School of Computer Science, Ottawa, Canada, 2004.
28. I. Suzuki and M. Yamashita, Distributed anonymous mobile robots: Formation of geometric patterns, *SIAM Journal of Computing* 28 (1999), 1347-1363.
29. X. Yu and M. Yung, Agent Rendezvous: A Dynamic Symmetry-Breaking Problem, *Proceedings of ICALP*, LNCS 1099, 610-621, 1996.