Technical Report CS-2012-04
Guaranteeing Rendezvous of Mobile Agents Despite Noise

Luther A. Tychonievich and **James P. Cohoon**
Department of Computer Science
University of Virginia
Charlottesville, VA 22903

Abstract

Rendezvous is process of having mutually-oblivious agents locate one another without communication in an unknown environment. Most approaches to rendezvous are either based on random walks or utilize environmental features like landmarks. We investigate the question, "can agents experiencing uncertainty in timing and uncertainty in their own locations be guaranteed to rendezvous in finite time?" We present theoretic upper bounds on the level of uncertainty agents in featureless environments can handle while guaranteeing finite-time rendezvous; we also present algorithms that realize a lower bound on uncertainty. Together, these bounds help define the impact of uncertainty on rendezvous.

1 Introduction

The objective of rendezvous algorithms is to have mobile computational agents (robots) locate one another in an unknown or featureless environment. Rendezvous is a foundational problem in cooperative artificial intelligence: before agents can coordinate their actions, individual agents must come close enough to initiate contact.

Since its introduction in 1960 (Schelling, 1960), the rendezvous problem has received considerable attention by theoreticians and artificial intelligence researchers. Algorithms have been developed that rendezvous probabilistically, that congregate at common environmental landmarks, and that guarantee finite-time rendezvous in noise-free environments. We continue this investigation by looking at finite-time rendezvous algorithms for agents experiencing noise in both timing and position information without dependence on particular environmental features.

We present a collection of procedures that cause each agent in an arbitrarily-large set to find each other agents in bounded time. The procedures vary in complexity, runtime, and in the assumed capabilities of the agents. Each procedure is accompanied by a proof that agents will rendezvous with one another in bounded time. We also present bounds on the uncertainty that may be present in any agent participating in bounded-time rendezvous. These minimal capabilities provide a lower bound on the rendezvous problem for agents facing uncertainty.

2 Related Work

Schelling (1960) introduced the rendezvous problem, and it has been investigated in many settings since then. In interest of space we mention only deterministic algorithms for agents without *a priori* knowledge of one another's location. See (Alpern and Gal, 2002) for a survey of randomized rendezvous algorithms.

Rendezvous has been investigated in many specialized environments, such as lines and graphs. On a line or ring, tight bounds are known and both deterministic and randomized algorithms are known to achieve them (Alpern, 1995; Alpern and Gal, 1995; Anderson and Essegaier, 1995; Marco et al., 2006). A tight $O(\log n)$ space-bound was demonstrated for deterministic rendezvous in a tree in (Fraigniaud and Pelc, 2008). Rendezvous in a graphs or networks has been extensively studied and deterministic algorithms developed for partial and full asynchrony, for indistinguishable agents in visibly-distinct starting locations, and for agents with distinct identities (Marco et al., 2006; Kowalski and Malinowski; Czyzowicz et al., 2010a,b; Dieudonné et al., 2012). None of these algorithms con-

University of Virginia Department of Computer Science Technical Report #CS-2012-04.

sider positional error.

In a geometric setting, deterministic approaches have been posed using several approaches. Rendezvous can be easily reduced to landmarks identification and ranking algorithms (Roy and Dudek, 2001). Stigmergy allows agents to modify the environment and create their own landmarks (Shiloni et al., 2009). In a noise-free environment, rational coordinates in euclidean space can be reduced to a graph (Czyzowicz et al., 2010b), giving access to a variety of graph.

Although a researchers have investigated rendezvous with errors in timing information (Marco et al., 2006; Czyzowicz et al., 2010b), none to our knowledge have provided rendezvous for agents experiencing uncertainty in position information.

3 Definitions and Notation

We consider the multi-agent bounded-time rendezvous problem for agents with uncertainty in position and timing information. We assume agents are initially **oblivious**, having no knowledge of the state of other agents. Given two or more mutually-oblivious mobile agents with limited-range sensors, bounded-time rendezvous guarantees that all agents sense one another within a provably finite time. To formalize this problem, we first present several definitions.

An **agent** is defined by a **state** $s \in S$ and a set of achievable behaviors $\mathcal{B}$[1]. A particular behavior $B \in \mathcal{B}$ defines an evolution of the state of the agent, which we could denote as a differential or difference equation (e.g., $\dot{s} = B(s)$) or as an evolution function (e.g. $s' = B(s, \Delta t)$).

We assume the existence of a **distance function** defined over agent state, $d : S \times S \to \mathbb{R}$. We require d to be commutative (i.e., $d(a, b) = d(b, a)$). We speak informally of the **position** of an agent as the portion of its state that contributes to the calculation of d. We do not in general assume that agents are capable of measuring distances.

The **capabilities** of an agent are characterized by the sensors and actuators the agent may access and the uncertainty of each. We consider the following classes of capabilities:

- **Mobility:** Agents have some knowledge of and control over their own location. We distinguish between three classes of mobility:
  - **Exact:** No uncertainty in location.
  - **Noisy:** Location error within fixed bounds.
  - **Drifting:** Location error increases as the agent moves.
- **Detection:** There is some distance r such that any two agents whose separation is no more than r are aware of one another. Agents might be aware of one another at greater distances as well.
- **Clock:** Each agent has some notion of the passage of time. We distinguish between four classes of clocks:
  - **Synchronized:** Clocks show the same time.
  - **Skewed:** Clocks progress at a shared rate.
  - **Individual:** Clocks progresses at different rates.
  - **Variable:** The rate of each clock varies over time.

We use the term **algorithm** to refer to a deterministic behavior selection process.

Definition 1 (Rendezvous)**.** *Two algorithms, A_i and A_j, **rendezvous with** one another if, for every arbitrary starting states $s_i(0)$ and $s_j(0)$, there exists a non-negative time $t^\star$, where $t^\star$ is bounded by some fixed finite function of $d(s_i(0), s_j(0))$, such that $d(s_i(t^\star), s_j(t^\star)) \leq r$.*

*The set of algorithms A is said to be a **family of rendezvous algorithms** or to have the **rendezvous property** if, for any arbitrary pair $i \neq j$, A_i and A_j rendezvous with one another.*

Our definition of rendezvous includes a worst-case time bound by requiring a bounding function on $t^\star$. Thus, by definition, all rendezvous algorithms we discuss have a finite worst-case time bound for all input configurations.

4 Necessary Capabilities

In this section we prove several lower bounds on agents and algorithms achieving rendezvous. In particular, for some but not all environments,

- Rendezvous requires each agent's algorithm be unique.
- If $|A| > 2$, rendezvous requires some kind of clock or global coordinate system.
- Rendezvous is at least as difficult as finding static targets.
- Finding static targets implies bounds on mobility drift.

We address each of these requirements individually.

[1] We use "behavior" to formalize the notion of "what an agent does;" this is not the same as a "behavior" in a behavior-based algorithm.

4.1 Individuality

Theorem 1. *It is not possible to provide rendezvous algorithms for completely indistinguishable agents.*

Agents are indistinguishable if each has identical internal parameters; if each agents actuators react to the same requests in the same way; and if any stimulus perceived by one agent is mirrored by an indistinguishable stimulus perceived by the other agent at the same time.

Proof. Suppose we have two indistinguishable agents. Then either they make decisions stochastically or deterministically.

If they make decisions stochastically then we cannot guarantee bounded-time results.

If they make decisions deterministically then they will both make the same decisions at the same time, moving in lock-step. Hence they will never change their initial separation and never meet. □

Our solutions to the rendezvous problem achieve individuality by the use of a unique identifier for each agent.

4.2 Temporality

Theorem 2. *It is not possible to provide finite-time rendezvous algorithms for more than two agents in multidimensional space without access to some estimate of the passage of time or knowledge of a shared coordinate system.*

Proof. Rendezvous of three or more agents requires at least two agents to be mobile. Consider two such mobile agents, i and j. Let $p_i = \{(s_i(t), t)\}$ be the path through spacetime traveled by agent i. and $V_j = \{(x,t) : d(x, s_j(t)) \leq r\}$ be the volume of spacetime guaranteed to be observed by agent j. For rendezvous to be guaranteed, the path must intersect the volume; that is $p_i \cap V_j \neq \emptyset$. This is illustrated in Figure 1. Note the choice of labels i and j is arbitrary; $p_i \cap V_j \neq \emptyset \Leftrightarrow p_j \cap V_i \neq \emptyset$.

If agents do not have any estimate of the passage of time then the t coordinate of each agent's motion cannot be controlled. in particular, p_i might be $\{(s_i(t), f(t))\}$ for any monotonically-increasing function f. Rendezvous without clocks is guaranteed only if it is guaranteed for every f. That is,

$$\big(f(x) > f(y) \Rightarrow x > y\big) \Rightarrow \left(\exists t : d\Big(s_i(t), s_j\big(f(t)\big)\Big) \leq r\right)$$

This condition is satisfied only if the first part of each agent's trajectory is within r of the reverse of the first part of the other agent's trajectory. Reversals of this sort are only possible with a shared coordinate system.

Hence, rendezvous requires either an estimate of time or a shared coordinate system. □

Czyzowicz et al. (2010b) present an algorithm guaranteeing finite-time rendezvous without a clock using repeated guessing to try all possible locations and coordinate systems of other agents. As we are interested in positioning uncertainty, their repeated return and retry approach is not suitable for our work.

The above proof does not specify the accuracy of clocks, but does imply that clock must exist in some form. Our solutions handle some limited clock inaccuracy, providing a sufficient upper bound. We are unaware of any lower bound on clock accuracy.

4.3 Search

Before presenting a proof that search is required we first must define what we mean by search.

Definition 2 (Search)**.** *A algorithm, S is a **search algorithm** if, for every arbitrary starting state $s(0)$ and objective location p, there exists a non-negative time $t^\star$, where $t^\star$ is upper bounded by some fixed finite function of $d(s(0), p)$, such that $d(s(t^\star), p) \leq r$.*

Search algorithms can also be characterized as algorithms that rendezvous with stationary agents.

Theorem 3. *For some but not all classes of environments, for any family of rendezvous algorithms A there exists a search algorithm S that be executed with the same capabilities as algorithms in A and that requires asymptotically no more space or time than the most expensive $A_i \in A$.*

Theorem 3 implies that search is no harder than rendezvous; we use this to support our building rendezvous algorithms off of search algorithms.

To establish this "some but not all" property, we give two example environments where rendezvous does imply search and one where it does not.

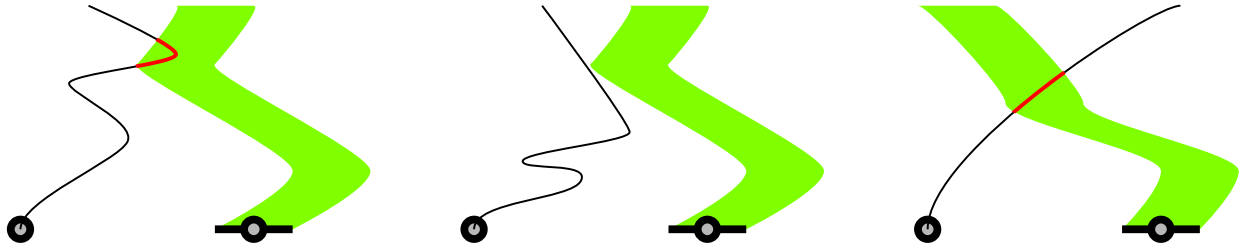
Lemma 3.1 (Featureless vector spaces). *Assume $\mathcal{B}$ is closed under addition and subtraction, d is an induced norm, and that the agents do not receive environmental inputs such as GPS signals. Then $S = A_i - A_j$, for any arbitrary $A_i, A_j \in A$, is a search algorithm.*

Proof. S may be computed by simulating A_i and A_j with addition and subtraction, as follows. Given

$$\forall s_i(0), s_j(0) \;\; \exists t^\star \;\; d\big(s_i(0)+\Delta s_i(t^\star), s_j(0)+\Delta s_j(t^\star)\big) \leq r$$

we have

$$\forall s_i(0), s_j(0) \;\; \exists t^\star \;\; d\big(s_i(0)+\Delta s_i(t^\star)-\Delta s_j(t^\star), s_j(0)\big) \leq r$$



(a) (b) (c)

Figure 1: Illustration of the proof to Theorem 2. The black line is p_i, the green area V_j. In (a) and (c) the heavier red line is $p_i \cap V_j$. The rendezvous in (a) is prevented in (b) by clock error. In (c) no clock error can cause the agents not to rendezvous.

which is the definition of S being a search algorithm. □

Lemma 3.2 (Unconstrained graphs). *Assume agents move along edges of a connected graph and can determine its structure only locally. For any two algorithms in A, at least one is a search algorithm.*

Proof. Assume two agents are executing a rendezvous algorithm but neither is executing a search algorithm. Then there exists some graph where each agent has some node it will not visit. Because the graph structure is known only locally, neither agent can know how much of the graph is inaccessible without traversing its excluded node(s). In particular, the only path between the two agents might include a subpath of length at least r accessible only through nodes each agent will not traverse. But this situation contradicts our assumption of rendezvous. Thus, at least one of agent is executing a search algorithm. □

Lemma 3.3 (Friendly features). *Assume agents can access a function defined throughout the environment which contains a single unique maximum. Then there is a rendezvous algorithm that requires less space than any search algorithm.*

Globally-defined single-maximum functions are often found in the real world. For example, altitude has a single maximum in many areas. Humans install exit signs which define the gradient of a similar function.

Proof. Searching general environments requires at least logarithmic space, either to store where the agent is or how long it has been searching. However, with the global function, rendezvous requires only a local hill-climbing routine, which requires only constant space. □

4.4 Limited Drift

Theorem 4 (Drift)**.** *To search any area of sufficient size in a d-dimensional Euclidean environment requires drift to accumulate no more quickly than $\Theta(\sqrt[d]{x})$, where x is the distance travelled.*

Proof. The length of every path an agent searching area R could follow is in $O(R)$. In Euclidean geometry, R extends at least $\Theta(\sqrt[d]{R})$ in some direction.

Assume the searching agent has drift in $\omega(\sqrt[d]{x})$. Then by the definition of the asymptotic class ω, the error in position ϵ resulting from searching R, for sufficiently large R, is greater than $\sqrt[d]{R}$. To overcome that error would require searching an additional area extending at least ϵ in all directions of the possibly-erroneous position. That secondary search area is of size at least ϵ^d, which is larger than the initial search space. Searching that would bring in even more error, and so on *ad infinitum*, meaning the search would never terminate. But nontermination contradicts the fact that search algorithms terminate in finite time. Hence, there is no search algorithm for agents whose drift is in $\omega(\sqrt[d]{s})$. □

Because drift is limited for search, by Theorem 3 it is also limited for rendezvous.

5 Parameterized Algorithm Families

In the following section of this paper we present techniques that achieve rendezvous for several different classes of agents. These algorithms are intended to create upper bounds on uncertainty and timing information to compliment the lower bounds proven in the previous section, and to provide bounds on the runtime of rendezvous algorithms in various circumstances. As such, they are written with only worst-case performance in mind. Allowing two moving agents to

Listing 1 $\mathcal{R}_{p,q}$: Rendezvous without drift

```
1. Repeat until rendezvous achieved:
2.    Repeat p times:
3.       For each bit x in the key:
4.          If x = 1: search and return q times
5.          Otherwise: wait for q time steps
6.    Double the time step
```

Listing 2 $\mathcal{D}_q$: Rendezvous with drift

```
1. Repeat until rendezvous achieved:
2.    For each bit x in the key:
3.       If x = 1: search and return q times
4.       Otherwise: wait for q time steps
5.       Increase the time step to double the search
   radius
```

rendezvous, allowing agents to move within a specified area during the "wait" cycle, and related modifications to reduce average-case runtime and tailor the algorithms to particular agent objectives are possible, but not considered further in this work.

While the correctness and runtime of each technique is separately addressed, all techniques are instantiations of two parameterized families of algorithms, which we present in Listing 1 and Listing 2 as $\mathcal{R}_{p,q}$ and $\mathcal{D}_q$. In addition to the integer parameters p and q, the listings assume the existence of a search subroutine (see Theorem 3), a b'-bit binary **key**, and a modifiable **time step**. Each "search and return" lasts one time step and involves executing the search subroutine to explore as large an area as possible before returning to a point as close to the agent's original location as mobility error allows.

Both $\mathcal{R}_{p,q}$ and $\mathcal{D}_q$ involve an interleaved pattern of searching and waiting, with the ordering of the two actions dictated by the key. We refer to the period of time in which an agent is either searching or waiting as a **bit**. We refer to the period of time in which an agent's time step has a single value as a **cycle**. Rendezvous between two agents may occur at any time, but we only prove it occurs when one agent is waiting at the time the other agent searches its waiting location. Proofs of rendezvous are developed in the following section. Several of those proofs depend on various qualities of the key.

We use numeric identifiers to ensure each agent behaves individually (see Theorem 1). We assume these identifiers are pulled from a set of b-bit binary numbers for some known b, where 2^b is an upper bond on the number of agents. From that b-bit identifier we derive each agent's b'-bit key. We consider two properties that a key might possess in addition to being unique: having a particular number of leading zeros or being shift-free.

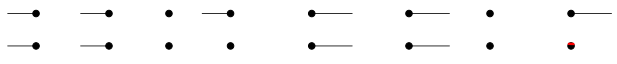


Figure 2: Rendezvous of agents without uncertainty with keys 1101 and 1100 using $\mathcal{R}_{1,1}$. The initial time step is too small; after doubling the timestep the agents rendezvous in the first distinct bit of their keys.

Definition 3 (Shift-free)**.** *Let x, y, k, and b be non-negative integers, x and y be less than 2^b, and k be less than b. The following are all defined with respect to b:*

*The k-bit **circular shift** of x w.r.t. b is*
$$S_k^b(x) \triangleq (2^k x \mod b) + \lfloor 2^{k-b} x \rfloor.$$

*x and y are **shift-similar** w.r.t b if $\exists k : S_k^b(x) = y$.*

*x is **shift-minimal** w.r.t. b if $\forall k : S_k^b(x) \geq x$*

*A set of b-bit numbers is **shift-free** if no two elements of the set are shift-similar w.r.t. b.*

If all keys belong to a shift-free set, we call them shift-free keys. Without loss of generality, we assume all shift-free keys are shift-minimal.

The largest shift-free set with respect to b contains more than $\lfloor 2^b \div b \rfloor$ elements. Thus, there is some mapping from b-bit identifiers to $\lceil b + \log_2(b) + 1 \rceil$-bit shift-free keys. As a much simpler mapping, if x is a b-bit identifier then $2x + 1$ is a $(2b + 1)$-bit shift-free key.

We now consider particular algorithm families that achieve rendezvous for agents with various capabilities.

5.1 Rendezvous without uncertainty

The simplest rendezvous we consider works for synchronized agents with exact mobility. This situation establishes a baseline off of which our other algorithms may be understood.

Theorem 5. *Consider two agents that begin executing $\mathcal{R}_{1,1}$ at the same time with the same time step and unique keys. The agents will rendezvous with each other in less than $4b'$ times the time required for one agent to search the location of the other or within b' times the initial time step, whichever is more.*

Proof. Let t_0 be the initial time step and T be the time step required for a search by one agent to find the initial location of the other agent. If the time step t_0 is not initially at least T, the time step will be in

Figure 3: Illustration of doubling proof. A's T_a cycle is in red; the $2T_b$ cycles are in blue.

$[T, 2T)$ within $(t_0 + 2t_0 + 4t_0 + \cdots + S)b' < 2b'S$, for $S < T$.

Once the time step is at least T, rendezvous will occur within $b'T$. Because each key is unique, the keys must differ in some bit. Thus, there must be some time during which one agent is searching (the active bit of its key is `1`) while the other agent is waiting (its active bit is `0`). During bit, the agent with the `1` bit will find the agent with the `0` bit, resulting in rendezvous.

Final runtime is bounded by $2b'T$ to achieve $t \in [T, 2T)$ and another $2b't$ to perform b' searches at that timestep. □

5.2 Rendezvous for skewed clocks

Suppose the agents have skewed clocks, exact mobility, and a search subroutine that views the same areas at the same time of each search. Because some distinct numbers are not distinguishable if not synchronized (e.g., `01` and `10`), we use shift-free keys. We also use $\mathcal{R}_{2,1}$, iterating through the key twice per cycle, to ensure full-key overlaps. $\mathcal{R}_{2,1}$ with shift-free keys results in rendezvous within $8b'$ times the time required for one agent to search the location of the other.

To demonstrate this technique achieves rendezvous we first present the following lemma.

Lemma 6.1 (Doubling). *Consider two agents, A and B, with cycle durations $T_a < T_b$ differing by a power of two at the beginning of B's cycle. At least half of their $2T_b$ cycles overlap.*

Proof. Call the start of B's cycle $t = 0$. Let $\alpha \in [0, 1)$ be the portion of A's T_a-duration cycle that is completed prior to $t = 0$. A's cycle reaches $2T_b$ at $\hat{t} = (1 - \alpha)T_a + 2T_a + 4T_a + \cdots + T_b$. Observe that $T_b < \hat{t} < 2T_b$, so $\hat{t}$ must fall within the first half of B's $2T_b$ cycle $[T_b, 3T_b]$.

This proof is illustrated in Figure 3. □

Theorem 6. *Consider two agents with accurate, asynchronous clocks and a repeatable search algorithm. $\mathcal{R}_{2,1}$ achieves rendezvous within two doublings of the initially-longer time step or the end of the first cycle where the search areas are sufficiently large.*

Proof. Because the keys are shift-free and because each agent searches the same location at the same time within each `1` bit, the agents will rendezvous with each other if they overlap for at least half a cycle and their searches are large enough to find one another.

By Lemma 6.1, both agents will have overlapped for at least half a cycle by the end of the second doubling of the longer time step. Because they will continue to overlap for at least half a cycle every cycle thereafter, if that first overlap didn't have large enough searches then the first search that is large enough will result in rendezvous. □

5.3 Rendezvous for noise or nondeterministic search

If the search subroutine is non-deterministic and/or the mobility suffers from noise then we cannot rely on passing a target's waiting location at the same time within each bit. This situation is not a concern if an entire search is guaranteed to overlap a waiting period, as in the synchronized case. For skewed clocks and mobility noise, agents utilize $\mathcal{R}_{2,2}$ with shift-free keys to perform two complete searches per bit. Searching twice per bit ensures that at least one entire search occurs within the other agent's waiting period.

$\mathcal{R}_{2,2}$ takes twice as long as $\mathcal{R}_{2,1}$, discussed in the previous section, meaning rendezvous is achieved within $16b'$ times the time required to search.

5.4 Rendezvous for drift

If after searching d in all directions the agent's sense of location is off by strictly less than $\frac{1}{8}d$, rendezvous is achieved by executing $\mathcal{D}_2$ with unique keys. The $\mathcal{D}$ algorithms differ from the $\mathcal{R}$ algorithms by doubling the *search radius* after every bit instead of doubling the time step after every cycle. To double the radius in a n-dimensional Euclidean environment, multiply the time step by 2^n.

Theorem 7. *Consider two agents with unique keys executing $\mathcal{D}_2$. If the agents have skewed clocks and mobility drift strictly less than $\frac{1}{8}d$, where d is the radius searched, then the agents are guaranteed to rendezvous in finite time.*

Searching twice per bit ensures that a full search overlaps a full wait, just as it did for noisy mobility. Doubling the search radius every bit keeps agents from drifting apart more rapidly than they expand their search, as demonstrated in the following proof.

Proof. Let the drift be bounded by md, for some $m < \frac{1}{8}$. Let the initial separation be d_0 and the initial search explore x in every direction. The first

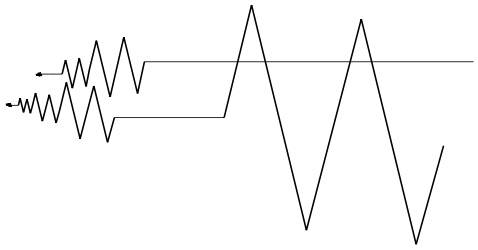


Figure 4: Example of 1D agents rendezvous with drift and keys 1100 and 1101.

bit can cause each agent to drift $2mx$, meaning $d_1 \leq d_0 + 4mx$. The next bit explores $2x$ and might lead to $d_2 \leq d_0 + 4mx + 8mx$. After the ith bit we have searches of radius $2^i x$ and separations of $d_i \leq d_0 + 4mx(1 + 2 + 4 + \cdots + i) < d_0 + 8mx2^i$. Since $8m < 1$, the search radius will eventually overtake the separation. □

This proof assumes all bits are `1`. Somewhat larger drifts should be permissible because drift does not accumulate during `0` bits and at least one key has at least one `0` bit.

The time required for rendezvous with drift is not simply expressed. The search radius might double as many as $\lceil \frac{\log d_0 - \log x}{-\log 8m} \rceil$ times before the search areas overlap, and then another b' times before the keys differ in a bit. The time required for each doubled search radius depends on the environment and search algorithm. For some environments, the runtime may be exponential in b', in the amount of drift, and in the initial separation.

5.5 Rendezvous for variable clocks

When agents have variable clocks, some time steps might be briefer than others. If some bits are expanded while others are contracted in duration it might be possible to make even shift-free keys never overlap `0` and `1` bits. The worst case pair of keys is $2^a - 1$ and $2^{a+1} - 1$, where $a = \lfloor \frac{1}{2} b' \rfloor$, because those keys require the least stretching to remove overlap.

Consider clocks whose unit of time varies within $[x, y]$. If x coincides with the longer run of bits and y with the shorter, then the minimal `1/0` overlap's duration is $(a+1)x - ay$. When $\frac{x}{y} > \frac{a}{a+1}$ we can guarantee rendezvous by having each agent search $q = \lfloor \frac{y}{x - a(y-x)} \rfloor$ times during each `1` bit, or $q = \lfloor \frac{2y}{x - a(y-x)} \rfloor$ times if the agents also experience mobility noise. We use this q to define the family of algorithms, $\mathcal{R}_{2,q}$.

Rendezvous for variable clocks takes at most $8qb'$ times longer than would searching for a static target at the same separation.

The algorithm for variable clocks does not need to change to account for noise as it already provides a full-search overlap. To handle drift less than $\frac{1}{4q} d$ we can use $\mathcal{D}_q$, doubling the search radius after every bit.

We expect that larger clock variability can be handled by additional restrictions on keys or by using a pattern of searches and waits within each bit; developing these algorithms is a subject of current research.

5.6 Rendezvous for individual clocks

If each agent has a consistent clock, but it runs at a different pace that other agents—or, equivalently, if each agents' initial time step is random—we require each shift-free key to have at least $\lceil b' \div 2 \rceil$ leading `0` bits. This restriction on keys arises from the following lemma and theorem.

Lemma 8.1. *For two agents, regardless of initial cycle durations, repeated doublings ensures the cycle durations differ by no more than* $1 : \sqrt{2}$ *for at least one quarter of each agent's cycle for every cycle after the* n*th, for some finite* n*.*

Proof. Label the initial cycle durations x and y and find real s and integer k such that $1 \leq s \leq \sqrt{2}$ and $sx = 2^k y$.

Assume that agent X finishes its x cycle at $0 < t_x \leq x$ and Y its y cycle at $0 < t_y \leq y$. Then X's cycle lasts for $2^n x$ in time window $t_x - 2x + [2^n, 2^{n+1}]x$. Because $y = 2^{-k} sx$, Y's cycle lasts for $2^n sx$ in time window $t_y - 2^{1-k} sx + [2^n, 2^{n+1}]sx$. We wish to show that, for sufficiently large n, these windows overlap by at least a quarter.

We define a constant $r = \left(\frac{t_x - t_y}{x} + 2^{k-1} s - 2 \right)$. From the bounds on t_x and t_y we can easily show that $|r| < 2^{|k|+1}$. This bound will be used to show that, for $n \geq |k| + 5$, each agent must overlap by at least one quarter of a cycle.

Observe that $[2^n, 2^{n+1}] = 2^n \beta$ where $\beta \in [1, 2]$. We find the β for which X lies within Y's cycle:

$$r + \beta s^{-1} 2^n \in [2^n, 2^{n+1}].$$

We consider the $\frac{1}{4}$-cycle range $\beta \in [\frac{13}{8}, \frac{15}{8}]$

$$\frac{(-2^{|k|+1}, 2^{|k|+1})}{2^n} + \left[\frac{13}{8}, \frac{15}{8} \right] \left[\frac{1}{\sqrt{2}}, 1 \right] \subseteq [1, 2]$$

which is satisfied for $n \geq |k| + 5$.

For Y overlapping X we have

$$-r + \beta s 2^n \in [2^n, 2^{n+1}].$$

which, over the $\frac{1}{4}$-cycle range $\beta \in [\frac{17}{16}, \frac{21}{16}]$,

$$\frac{(-2^{|k|+1}, 2^{|k|+1})}{2^n} + \left[\frac{17}{16}, \frac{21}{16}\right]\left[1, \sqrt{2}\right] \subseteq [1, 2]$$

is satisfied for $n \geq |k| + 5$. □

Theorem 8. *Let each b'-bit key have $\lceil \frac{b'}{2} \rceil$ leading 0 bits. Then $\mathcal{R}_{8,2}$ results in rendezvous for any initial time steps.*

Proof. By Lemma 8.1, each pair of agents will eventually overlap by two full cycles with relative time step durations $1 : s$, for $1 \leq s \leq \sqrt{2}$. In particular, the overlap must include the low-order bit of each key twice.

If $s < \frac{b'}{b'-1}$ then one pass through the key results in less than a full bit change in overlap. By the shift-free property of keys and the fact we search twice per bit, at least one full search of one agent must lie within a wait of the other agent.

Conversely, if $\frac{b'}{b'-1} \leq s$, the low-order bit of one key must lie more than half within the run of 0s at the high order of the other key. Because keys are shift-minimal, at least one agent searches during that bit. □

The maximal time required for rendezvous for variable clocks is $64b'$ times the time required for static search if the cycle durations reach $1 : s$ difference before the search areas are large enough to find other agents. If the cycles starts large enough for agents to find one another, it can take a lot longer: one of the cycles doubles $\left\lceil \max\{\frac{x}{y}, \frac{y}{x}\} \right\rceil + 5$ times before time step synchronization sufficient to guarantee rendezvous.

6 Cooperative Rendezvous

Rendezvous is significantly easier when groups of agents cooperate to locate other groups of agents. Cooperation allows one agent to wait while the others search, removing the need for keys and repeated searching of the same area. The division of labor can be performed without communication utilizing emergent formation algorithms (Dieudonné et al., 2008; Défago and Konagaya, 2002; Moshtagh et al., 2009). The group initially forms a circle around a single central agent; each agent then searches only its Voronoi region (plus any padding necessary to counter noise and drift). Rendezvous then reduces to the problem of searching for a static objective in an unknown environment and sharing the results of searching with other agents in the group.

7 Conclusion

We investigated rendezvous: the problem of having agents find one another in an unknown, unbounded environment with neither prior knowledge of one another nor any form of communication. We demonstrated lower bounds on the capabilities required to perform rendezvous: agents must be able to exhaustively search the environment, must possess position information and/or mobility with sub-linear drift, must have some means of tracking the passage of time, and must each execute a unique algorithm or possesses unique parameters or input.

We also demonstrated upper bounds in the form of algorithms proven to achieve rendezvous under a variety of circumstances. Our algorithms that handle mobility drift can handle up to a constant factor of the upper bound on allowable uncertainty. While we have no upper bound on clock uncertainty, our algorithms demonstrate that clocks need not be synchronized, need not progress at the same pace, and can vary in pace without preventing rendezvous. We also showed that agents with noisy but drift-free mobility, such as GPS, and approximately-correct unsynchronized clocks can locate one another in time $O(T \log_2 n)$, where n is the number of agents and T is the time it would take to search out a static objective in the same area.

While we have made significant strides in placing bounds on the time complexity and sensitivity to uncertainty in the rendezvous with other agents, there is significant room for tighter bounds. We observe informally that achieving rendezvous with less precise sensors appears to require more time; the mathematical bounds on this trade-off remain open. In addition to these open problems, there are several extensions that might be considered, such as how groups of mutually-aware agents can achieve rendezvous with other groups more rapidly than can individual agents, how agents can benefit from long-distance communication, and how agents, having located one another, can co-locate into hives without the need to trust the other agents.

We anticipate our results can be extended in variety of other areas. For example, the theoretic benefit of a particular sensor technology might be demonstrated by demonstrating algorithms that utilize it to outperform our lower bounds. An approach similar to ours might be used to provide guaranteed bounds for agents with limited capabilities performing tasks other than rendezvous. We look forward to many future explorations of guaranteed algorithms for mobile agents experiencing uncertainty.

References

S. Alpern. The rendezvous search problem. *SIAM Journal of Control and Optimization*, 33(3): 673–683, 1995.

S. Alpern and S. Gal. Rendezvous search on the line with distinguishable players. *SIAM Journal on Control and Optimization*, 33:1270–1276, 1995.

S. Alpern and S. Gal. *The theory of search games and rendezvous*. International Series in Operations Research and Management Science. 2002.

E. J. Anderson and S. Essegaier. Rendezvous search on the line with indistinguishable players. *SIAM Journal on Control and Optimization*, 33:1637–1642, 1995.

J. Czyzowicz, A. Kosowski, and A. Pelc. How to meet when you forget: Log-space rendezvous in arbitrary graphs. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC 2010)*, pages 450–459, 2010a.

J. Czyzowicz, A. Labourel, and A. Pelc. How to meet asynchronously (almost) everywhere. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 22–30, 2010b.

X. Défago and A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 97–104, New York, NY, USA, 2002. ACM.

Y. Dieudonné, O. Labbani-Igbida, and F. Petit. Circle formation of weak mobile robots. *ACM Trans. Auton. Adapt. Syst.*, 3(4):1–20, 2008.

Y. Dieudonné, A. Pelc, and D. Peleg. Gathering despite mischief. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 527–534, 2012.

P. Fraigniaud and A. Pelc. Deterministic rendezvous in trees with little memory. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC 2008)*, volume 5218 of *Springer Lecture Notes in Computer Science*, pages 242–256, 2008.

D. Kowalski and A. Malinowski. How to meet in anonymous network. In *13th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2006)*, volume 4056 of *Springer Lecture Notes in Computer Science*, pages 44–58.

G. D. Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theoretical Computer Science*, (335):315–326, 2006.

N. Moshtagh, N. Michael, A. Jadbabaie, and K. Daniilidis. Vision-based, distributed control laws for motion coordination of nonholonomic robots. *Robotics, IEEE Transactions on*, 25(4):851 –860, aug. 2009. ISSN 1552-3098.

N. Roy and G. Dudek. Collaborative exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, September 2001.

T. Schelling. *The strategy of conflict*. Oxford University Press, Oxford, England, UK, 1960.

A. Shiloni, N. Agmon, and G. A. Kaminka. Of robot ants and elephants. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 81–88, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.