Multiple Mobile Agent Rendezvous in a Ring

Paola Flocchini* Evangelos Kranakis† Danny Krizanc‡ Nicola Santoro†
Cindy Sawchuk†

September 14, 2003

Abstract

We study the rendezvous search problem for $k \geq 2$ mobile agents in an n node ring. Rather than using randomized algorithms or different deterministic algorithms to break the symmetry that often arises in this problem, we investigate how the mobile agents can use identical stationary tokens to break symmetry and solve the rendezvous problem. After deriving the conditions under which identical stationary tokens can be used to break symmetry, we present several solutions to the rendezvous search problem. We derive the lower bounds of the memory required for mobile agent rendezvous and discuss the relationship between rendezvous and leader election for mobile agents.

1 Introduction

In the mobile agent rendezvous search problem, k mobile agents located on an n node network are required to meet or rendezvous. When the mobile agents or the network nodes are uniquely numbered, solving the rendezvous search problem is trivial. When the mobile agents are identical and the network nodes are anonymous, however, the resulting symmetry can make the problem difficult to solve.

Symmetry in the rendezvous search problem is typically broken by using randomized algorithms or different deterministic algorithms [2]. While Baston and Gal [5] mark the starting points of the searchers, they still rely on randomization or different deterministic algorithms. Kranakis et al [7], however, studied how two searchers, i.e., mobile agents, on an n node ring can use identical tokens to break the symmetry.

Most of the literature on the rendezvous search problem deals with the case of $k = 2$ searchers. The few exceptions include Lim, Beck, and Alpern [8], Alpern [1], Pikounis and Thomas [9], and Gal [6], but that research focuses almost exclusively on the line. In this paper, we investigate the mobile agent rendezvous search problem for $k \geq 2$ mobile agents on an n node ring, where the mobile agents use tokens to break symmetry.

*SITE, University of Ottawa, Ottawa, ON, Canada.
Email: flocchin@site.uottawa.ca
†School of Computer Science, Carleton University, Ottawa, ON, Canada.
Email: {kranakis,santoro,sawchuk}@scs.carleton.ca.
‡Department of Mathematics and Computer Science, Wesleyan University, Middletown, Connecticut 06459, USA.
Email: dkrizanc@wesleyan.edu

1.1 The Network Model

The model consists of $k \geq 2$ identical mobile agents that are located on separate nodes of an anonymous, synchronous n node ring. The mobile agents may or may not share a common orientation, i.e., agree on the direction that is clockwise. A given node requires only enough memory to host a token and at most k mobile agents. Each mobile agent, MA, owns a single identical stationary token, i.e., the tokens are indistinguishable and once they are placed on a node, they must remain in place. A token or MA at a given node is visible to all MAs on the same node, but is not visible to any other MAs. When a MA is visible, its state is also visible. The MAs follow the **same** deterministic algorithm and begin execution of that algorithm at the same time.

Memory permitting, a MA can count such things as the number of nodes visited, the number of tokens discovered, the number of MAs discovered, the number of nodes between tokens, or the total number of nodes in the network. In addition, a MA might already know the number of nodes in the network or some other network parameter and requires sufficient memory to store this information. Since the MAs are identical, they face the same limitations on their knowledge of the network. Rendezvous occurs when all the MAs meet on a network node.

An instance of the mobile agent rendezvous problem is **solvable** when the mobile agents

1. can correctly determine whether or not rendezvous is possible and then
2. rendezvous or stop as appropriate.

Solving the mobile agent rendezvous problem involves making the correct choice, i.e., stopping if rendezvous is impossible and achieving rendezvous if possible.

In this paper, we assume that the MAs always place their tokens on their respective starting nodes in the first step of any algorithm they execute. The tokens are identical and stationary so that intertoken distances that exist at the beginning of any algorithm persist throughout the algorithm.

1.2 Our Contribution

In this paper, we continue the study of the mobile agent rendezvous search problem in the ring [7]. Our model consists of $k \geq 2$ identical MAs in an anonymous, synchronous, and possibly oriented n node ring. Rather than using randomized algorithms or different deterministic algorithms, we use identical stationary tokens to break symmetry so that the MAs can run the same deterministic algorithm.

First, we prove that if neither k, the number of MAs, nor n, the number of nodes in the ring, are known then the rendezvous problem is unsolvable. For the remainder of the paper, we assume that k is known. Next, we prove that when k is known, rendezvous is possible if and only if $\mathcal{S}$, the sequence of intertoken distances, is aperiodic.

We present three algorithms that solve the rendezvous problem when k is known*. When $\mathcal{S}$, the sequence of intertoken distances between the MAs, is aperiodic, the algorithms guarantee that rendezvous occurs. However, if $\mathcal{S}$ is periodic and thus rendezvous is impossible, the algorithms guarantee that the MAs stop. The memory and time complexities for these algorithms are presented in table 1. We also present three algorithms that solve the rendezvous problem for values of k and n that satisfy various conditions such as primality. The memory and time complexities for these algorithms are also presented in table 1.

*Either k or n must be known. If n is known, then k can be determined in one traversal of the ring.

Algorithm	Memory	Time
1	$O(k \lg n)$	$O(n)$
2	$O(\lg n)$	$O(kn)$
3	$O(k \lg \lg n)$	$O(\frac{n \lg n}{\lg \lg n})$
4[§]	$O(\lg n)$	$O(n)$
5[§]	$O(\lg k)$	$O(n \lg k)$
6[§]	$O(\lg k)$	$O(n)$

Table 1: The Rendezvous Search Problem with $k \geq 2$ Mobile Agents

Kranakis et al [7] proved that solving the rendezvous problem with $k = 2$ mobile agents requires at least $O(\lg \lg n)$ memory. In this paper, we prove that solving the rendezvous problem with $k \geq 2$ mobile agents requires at least $\Omega(\lg \lg n + \lg k)$ memory.

Finally, we prove that if the MAs share a common orientation, then the rendezvous problem and the leader election problem for MAs are equivalent. A solution to the first problem can be used to derive a solution for the second problem and vice versa. If the MAs do not share a common orientation, however, then the leader election problem for MAs is strictly more complex than the rendezvous problem since a solution to the latter problem does not always imply a solution to the former problem.

1.2.1 Outline of the Paper

In section 2, we present the impossibility results for the MA rendezvous search problem with $k \geq 2$ MAs. In section 3, we prove that rendezvous is possible if and only if the sequence of intertoken distances is aperiodic. We present unconditional solutions for rendezvous in section 4 and derive the lower bounds on memory necessary for rendezvous in section 5. In section 6, we present solutions to the rendezvous problem for specific circumstances and then, in section 7, we discuss the relationship between leader election and rendezvous for mobile agents. For space considerations, most of the proofs appear in the appendix.

2 Impossibility Results

To solve the rendezvous problem, mobile agents must recognize when rendezvous is possible. As the next theorem shows, knowledge of k or n is a necessary condition for the rendezvous of identical MAs in an anonymous, synchronous ring.

Theorem 1 *When each MA in the ring knows neither n, the number of nodes, nor k, the number of mobile agents, the mobile agent rendezvous search problem is unsolvable.*

Theorem 1 indicates that knowing either k or n is a necessary condition for solving the rendezvous problem. For the remainder of the paper, we shall assume that k is known.

[§]In this algorithm, the values for k and n must satisfy various conditions such as primality.

3 Solving the Rendezvous Problem

As stated in section 1.1, we assume that the MAs always place their tokens on their respective starting nodes in the first step of any algorithm they execute. The tokens are identical and stationary so that the intertoken distances that exist at the beginning of an algorithm persist throughout the algorithm. In this section, we prove that rendezvous is possible if and only if the sequence of intertoken distances in the ring is aperiodic.

If $\mathcal{S} = d_1, \ldots, d_k$, the sequence of intertoken distances, is periodic, then $\gcd(k, n) = g > 1$ and the k MAs can be placed on an n node ring as described in Barriere et al [4]:

i) denote the nodes adjacent to node x_i as x_{i+1} and x_{i-1} where the operations on the node indices are mod n,
ii) starting at node x_1, divide the n nodes of the ring into g non-intersecting segments of n/g nodes each,
iii) the first such segment, σ_1, is comprised of the nodes x_1 to $x_{n/g}$ inclusive,
iv) arbitrarily place k/g MAs in segment σ_1 such that no node holds more than one MA,
v) place the remaining $k - (k/g)$ MAs such that if there is a MA on node x_i, $1 \leq i \leq n/g$, then there is a MA on node $x_{i+j(n/g)}$ for $1 \leq j \leq g - 1$.

As a result, each MA initially has a *sibling* MA exactly n/g nodes away in either direction.

Let the atomic operations in any rendezvous algorithm be restricted to "release the token", "move one node in a given direction", "do not move", and "increment a given variable". In addition to these atomic operations, an algorithm may contain two constructs. In a **do-until loop**, a MA performs a finite sequence of atomic operations until a given state occurs, and if a given condition holds in an **if-then statement**, a MA performs a finite sequence of atomic operations.

Lemma 1 *Assume that the MAs are initially placed as described above so that each MA is exactly n/g nodes away from its respective* siblings. *If the MAs simultaneously start to execute an identical algorithm and that algorithm is restricted to a finite sequence of atomic operations, i.e., there are no* do-until *loops or* if-then *statements, then the distance between a mobile agent and its respective* siblings *will not change.*

Lemma 2 *Assume that the MAs are initially placed as described above so that each MA is exactly n/g nodes away from its respective* siblings. *Let any MA and its two* siblings *be referred to as a* trio. *For a given MA, let* CLToToken *and* CCToToken *denote the distance to the nearest tokens in the clockwise and counterclockwise directions respectively. Initially, the MAs comprising a* trio *have identical values for* CLToToken *and identical values for* CCToToken. *If time t^* is the first time in an algorithm that at least one MA is not exactly n/g nodes away from its respective* siblings, *then t^* is also the first time that the respective values of* CLToToken *(and thus* CCToToken*) are not equal for that MA and its* siblings.

Theorem 2 *Rendezvous is guaranteed if and only if $\mathcal{S}$, the sequence of intertoken distances, is aperiodic.*

4 Unconditional Solutions

Given the results of section 3, the mobile agent rendezvous problem can be solved when the MAs can determine that $\mathcal{S}$, the sequence of intertoken distances, is aperiodic.

First, assume that each MA has $O(k \log n)$ memory. The MAs know k but do not necessarily know n or share a common orientation. Consider the following algorithm.

Algorithm 1

1. Release the token at the starting node.
2. Choose a direction and begin to walk around the ring.
3. Compute the k intertoken distances $d_1, \ldots, d_k$.
4. If $\mathcal{S} = d_1, \ldots, d_k$ is periodic, then stop. (Rendezvous is not guaranteed.)
5. Set *forward* $= d_1, \ldots, d_k$ and *reverse* $= d_k, \ldots, d_1$.
6. Let *lexi(someSequence)* denote the lexicographically maximum rotation of the sequence *someSequence*.
7. Set *forward* = *lexi(forward)* and *reverse* = *lexi(reverse)*.
8. If *forward* and *reverse* differ, then determine which of these sequences is the lexicographic maximum and rendezvous at the node where this sequence starts.
9. Else let MA_i and MA_j denote the MAs at the beginning of *forward* and *reverse* respectively.
10. If MA_i and MA_j are the same MA, then rendezvous at the node where MA_i resides.
11. If MA_i and MA_j are distinct MAs, then look at the two paths between MA_i and MA_j in the ring.
    i) If only one of the paths had an odd number of nodes, then rendezvous at the node in the midpoint of that path.
    ii) If both paths have an odd number of nodes, then
        a) if the paths differ in length, rendezvous at the midpoint of the shorter path,
        b) else compare the sequences of intertoken distances for the two paths and rendezvous at the node in the midpoint of the path that is the lexicographic maximum.
    iii) If both paths have an even number of nodes, then rendezvous at the node in the midpoint of the path that contains an odd number of MAs.

Theorem 3 *If each MA has memory $O(k \lg n)$, then the mobile agent rendezvous problem can be solved in time $O(n)$.*

If the MAs are restricted to memory $O(\lg n)$, the mobile agent rendezvous problem is still solvable.

Consider the following algorithm. In each round, a MA may become inactive and thus spend the rest of the algorithm at its starting node. If the MAs share a common orientation, then the MAs travel in the same direction, so an active MA can identify an inactive MA because the former will find the latter stopped. If the MAs do not share a common orientation, however, an active MA that meets another MA can not tell if the latter MA is inactive or merely travelling in the opposite direction. In this case, each MA needs to set a bit that indicates whether it is active or inactive.

With only $O(\lg n)$ memory, a MA needs more than one traversal of the ring to determine if $\mathcal{S}$ is aperiodic. Consider the following algorithm.

Algorithm 2

1. Release the token at the starting node.
2. Set $c = 1$.(The number of the current round.)
3. Set $active = 1$. (A bit to indicate whether the MA is active.)
4. Set $inactive = 0$. (Count the number of inactive MA.)
5. Choose a direction and begin to walk around the ring.
6. Increment *inactive* each time an inactive MA is met.
7. Compute the distance to the cth token, d_c, i.e., if $c = 1$, count the distance to the first token and if $c = 2$, count the distance to the second token, etc.
8. Continue to walk around the ring and compare d_c to each intertoken distance between c tokens.
9. If MA sees an intertoken distance d_i such that $d_i > d_c$, then the MA continues in the same direction and becomes inactive, i.e., sets $active = 0$, when it reaches its starting node.
10. If MA did not see an intertoken distance d_i such that $d_i > d_c$, then the MA remains active when it returns to its starting node.
11. If only one MA remains active, i.e., $inactive = k - 1$, then walk around the ring and arrange the rendezvous. (A MA that reaches this point is the sole active MA.)
12. If $c == 2$ and $inactive == 0$, then stop. (All the intertoken distances are equal and thus rendezvous is impossible.)
13. Else set $c = c + 1$ and $inactive = 0$.
14. Repeat from step 5.

Theorem 4 *If each MA has memory $O(\lg n)$, then the mobile agent rendezvous problem is solvable in time $O(kn)$.*

Algorithms 1 and 2 solve the mobile agent rendezvous problem when each MA has memory $O(k \lg n)$ and $\mathrm{O}(\lg n)$ respectively. It is also possible, however, to solve the mobile agent rendezvous problem when each MA has memory $O(k \lg \lg n)$.

Let $p_1, ..., p_r$ denote the first r prime numbers such that $\prod_{i=1}^{r} p_i > n$. An active MA needs to recognize if another MA is active. Without a common orientation, each MA needs to set a bit to indicate whether it is active.

Algorithm 3

1. Release token at the starting node.
2. Set active = 1.
3. Set p_r to the first prime such that $\prod_{i=1}^{r} p_i > n$.
4. Set $p_i = p_1 = 2$.
5. Set $\alpha = k$, the number of active MAs.
6. Walk around the ring and compute the intertoken distances mod p_i between the α active MAs, i.e., $d_1, ..., d_\alpha \bmod p_i$.
7. Set $forward = d_1, ..., d_\alpha \bmod p_i$.
8. Set $reverse = d_\alpha, \ldots, d_1 \bmod p_i$.
9. If *forward* is periodic, i.e., $forward = (d_1, ..., d_{\alpha/a})^a \bmod p_i$, then

i) if at start of a block $(d_1, ..., d_{\alpha/a})$, remain active.
ii) else set active $= 0$.
iii) if $p_i < p_r$, then
a) set $p_i = p_{i+1}$, $\alpha = a$, and repeat from step 6.
b) else stop, since rendezvous is impossible.

10. If *forward* is aperiodic, then let *lexi(someSequence)* denote the lexicographically maximum rotation of the sequence *someSequence*.
11. Set *forward* = *lexi(forward)* and *reverse* = *lexi(reverse)*.
12. If *forward* and *reverse* differ, then
i) determine which of these sequences is the lexicographic maximum
and rendezvous at the node where this sequence starts.
ii) else let MA_i and MA_j denote the MAs at the beginning
of *forward* and *reverse* respectively.
13. If MA_i and MA_j are the same MA, then rendezvous at the node where MA_i resides.
14. If MA_i and MA_j are distinct MAs, then do the following.
i) Look at the two paths between MA_i and MA_j in the ring.
ii) If only one of the paths had an odd number of nodes,
then rendezvous at the node in the midpoint of that path.
iii) If both paths have an odd number of nodes, then
a) if the paths differ in length, rendezvous at the
midpoint of the shorter path,
b) else compare the sequences of intertoken distances for the two paths
and rendezvous at the node in the midpoint of the path that is
the lexicographic maximum.
iv) If both paths have an even number of nodes, then rendezvous
at the node in the midpoint of the path that contains an odd number of MAs.

The following lemma is necessary for the proof of Theorem 5 below.

Lemma 3 *Consider a prime p such that $2 < p \leq r$. Assume that for all primes $p_i < p$, the sequence of distances* mod p_i *between the α_i active MAs is periodic such that $d_1, \ldots, d_{\alpha_i} \equiv (d_1, \ldots, d_{\alpha_i/a})^a \bmod p_i$ and $a \mid \alpha_i$. Let the first MA in each occurence of the block $d_1, \ldots, d_{\alpha_i/a}$ remain active, while the remaining MAs become inactive. If the sequence of distances* mod p *between the a active MAs is periodic, then the original intertoken distances can be partitioned into $t \mid k$ equal length blocks with sums $\sigma_1, \sigma_2, \ldots, \sigma_t$ that are congruent modulo all the primes $p_1, p_2, \ldots, p$.*

Proof of Lemma 3 in section 4.
Base case:
Let $p_i = p_2 = 3$. By assumption, the sequence of distances mod p_1 between the k active MAs is periodic, so that $d_1, \ldots, d_k \equiv (d_1, \ldots, d_{k/a})^a \bmod p_1$ and $a \mid k$.

The original intertoken distances are thus partitioned into a equal length blocks:

$$\begin{aligned} s^1_1 &= d_1, \ldots, d_{k/a} \\ s^1_2 &= d_{(k/a_1)+1}, \ldots, d_{2(k/a)+1} \\ &\ldots \\ s^1_a &= d_{(a-1)(k/a)}, \ldots, d_k \end{aligned}$$

Let σ^1_i denote the sum of the distances in the block s^1_i. By construction, $\sigma^1_1 \equiv \sigma^1_2 \equiv \ldots \equiv \sigma^1_a \mod p_1$, i.e., the sums of the distances mod p_1 in the respective blocks are congruent mod p_1. The MAs at the head of the respective blocks $s_1, \ldots, s_a$ remain active, so that number of active MAs becomes a.

If the sequences of distances mod p_2 between the a active MAs is periodic, then $d_1, \ldots, d_a \equiv (d_1, \ldots, d_{a/a'})^{a'} \mod p_2$ where $a' \mid a \mid k$. Thus the a distances can be partitioned into a' equal length blocks mod p_2:

$$\begin{aligned} s^2_1 &= d_1, \ldots, d_{a/a'} \\ s^2_2 &= d_{(a/a')+1}, \ldots, d_{2(a/a')} \\ &\ldots \\ s^2_{a'} &= d_{(a'-1)(a/a')+1}, \ldots, d_a \end{aligned}$$

Let σ^2_i denote the sum of the distances in the block s^2_i. By construction, $\sigma^2_1 \equiv \sigma^2_2 \equiv \ldots \equiv \sigma^2_{a'} \mod p_2$, i.e., the sums of the distances in the a' respective blocks are congruent mod p_2. Each block s^2_i, however, contains the a/a' blocks from $s^1_{(i-1)(a/a')+1}$ to $s^1_{i(a/a')}$ and thus $\sigma^2_i = \sum_{j=1}^{a/a'} \sigma^1_{(i-1)(a/a')+j}$. As shown above, however, the σ^1_i terms are congruent mod p_1 for all i and since each σ^2_j term is the sum of a/a' σ^1_i terms, then the σ^2_j terms are also congruent mod p_1.

Thus the original distances can be partioned into $a' \mid k$ equal length blocks with sums $\sigma^2_1, \ldots, \sigma^2_{a/a'}$ that are congruent modulo p_1 and p_2. This proves that the lemma holds for the base case where $p = p_2 = 3$.

Assume that the lemma holds for all primes $p < p_r$.

Let α denote the number of active MAs at the end of the round for p_{r-1}. Suppose that in the next round, the sequence of distances mod p_r between the α active MAs is periodic, i.e., $d_1, \ldots, d_\alpha \equiv (d_i, \ldots, d_{\alpha/a_r})^{a_r} \mod p_r$ where $a_r \mid \alpha$. The sequence of distances can therefore be partitioned into a_r equal length blocks mod p_r:

$$\begin{aligned} s^r_1 &= d_1, \ldots, d_{\alpha/a_r} \\ s^r_2 &= d_{(\alpha/a_r)+1}, \ldots, d_{2(\alpha/a^r)} \\ &\ldots \\ s^r_{a_r} &= d_{(a^r-1)(\alpha/a_r)+1}, \ldots, d_\alpha \end{aligned}$$

Each block s_i^r, $i = 1, \ldots, (\alpha/a_r)$ contains the α/a_r blocks $s^{r-1}_{(i-1)(\alpha/a_r)+1}$ to $s^{r-1}_{i(\alpha/a^r)}$ and thus $\sigma_i^r = \sum_{j=1}^{\alpha/a_r} \sigma^{r-1}_{(i-1)(\alpha/a_r)+j}$.

The inductive hypothesis implies that the blocks s_i^{r-1}, $i = 1, \ldots, \alpha$, partition the original intertoken distances into $\alpha \mid k$ equal length blocks with sums σ_i^{r-1}, $i = 1, \ldots, \alpha$, that are congruent for all $p \leq p_{r-1}$. Since each of the s_i^r blocks contains α/a_r of the s_i^{r-1} blocks, then the s^r blocks partition the original intertoken distances into $a_r \mid \alpha \mid k$ equal length blocks. By construction, the sums for the s^r blocks, σ_i^r, are congruent mod p_r for all i, $i = 1, \ldots, a_r$. Since each σ_i^r, however, is merely the sum of α/a_r σ_j^{r-1} terms and the inductive hypothesis implies that the σ_j^{r-1} are congruent mod p for all $p \leq p_{r-1}$, then the sums σ_i^r, $i = 1, \ldots, a_r$ are congruent mod p for all $p \leq p_r$.

Thus for p_r, the original intertoken distances can be partitioned into $a_r \mid k$ equal length blocks with sums σ_i^r, $i = 1, \ldots, a_r$, that are congruent for all $p \leq p_r$. This completes the proof of Lemma 3. ■

Theorem 5 *If each MA has memory $O(k \lg\lg n)$, then the rendezvous problem can be solved in time $O(\frac{n \lg n}{\lg\lg n})$.*

Proof of Theorem 5 in section 4.
Algorithm 3 solves the rendezvous if it stops the MAs when rendezvous is impossible and otherwise ensures a rendezvous. Suppose that for all $p_i \leq p_r$, the sequence of distances mod p_i between the active MAs is periodic. Algorithm 3 will stop the MAs in step 8 and indicate that rendezvous is impossible. Lemma 3 implies that in the last round of Algorithm 3, where $p = p_r$, the original intertoken distances can be partitioned into $a_r \mid k$ equal length blocks with sums σ_i^r, $i = 1, \ldots, a_r$, that are congruent mod p for all $p \leq p_r$. The Chinese Remainder Theorem then implies that the sums σ_i^r, $i = 1, \ldots, a_r$, are congruent mod $\prod_{i=1}^r p_i$. Since $\prod_{i=1}^r p_i > n$, then the original intertoken distances can be partitioned $a_r \mid k$ into equal length blocks with sums σ_i^r that are equal for all i, $i = 1, \ldots, a_r$. This implies, however, that $n = a_r \sigma_i^r$ for any i and thus $a_r \mid n$. Since $\gcd(k, n) = g > 1$, $\mathcal{S}$ is periodic, and the algorithm 3 correctly stops the MAs in step 8.

Algorithm 3 must also guarantee that rendezvous occurs when possible, i.e., when $\mathcal{S}$ is aperiodic. Suppose not, i.e., $\mathcal{S}$ is aperiodic but rendezvous does not occur. This implies that for all $p_i \leq p_r$, the sequences calculated in step 7 of algorithm 3 are periodic and thus all rounds of the algorithm will be executed. In the final round, where $p_i == p_r$, algorithm 3 will stop the MAs and indicate that rendezvous is impossible. By the Chinese Remainder Theorem, however, this implies that $\mathcal{S}$ is periodic and thus contradicts the fact that $\mathcal{S}$ is aperiodic. Thus algorithm 3 solves the mobile agent rendezvous problem.

If each mobile agent has memory $O(k \lg\lg n)$, then Algorithm 3 correctly determines whether rendezvous is possible and instructs the MAs to stop or rendezvous as appropriate. In the worst case, rendezvous is impossible and the MAs must complete all r rounds of Algorithm 3, where r is the smallest number of prime numbers such that $\prod_{i=1}^r p_i > n$. Each of the r rounds takes n steps so the time complexity is $O(rn)$. Kranakis et al [7] proves that $r \in O(\frac{\lg n}{\lg\lg n})$, so the time complexity of Algorithm 3 is $O(\frac{n \lg n}{\lg\lg n})$. This completes the proof of Theorem 5. ■

5 Lower Bound on Memory

In Theorem 1 of section 2, we prove that the mobile agent rendezvous problem can only be solved if either k or n is known. When $k = 2$, Kranakis et al [7] prove that $\Omega(\lg\lg n)$ memory is required to solve the rendezvous problem when k is known.

Theorem 6 *Solving the mobile agent rendezvous problem for $k > 2$ MAs requires $\Omega(\lg\lg n + \lg k)$ memory.*

6 Conditional Solutions

In the preceding two sections, we proved that the mobile agent rendezvous problem is unconditionally solvable if each MA has memory $\Omega(\lg\lg n + \lg k)$. In this section, however, we explore conditional solutions, i.e., solutions for cases where k and n satisfy various conditions such as primality. A network designer or administrator may be able to choose the values of k and n so as to meet these conditions.

The mobile agent rendezvous problem can be solved correctly whenever the sequences $\mathcal{S}_i$ are aperiodic for all i, e.g., n is prime or n is the product of two or more primes larger than k. A network designer or administrator, however, probably cannot directly dictate that $\mathcal{S}_i$ is aperiodic for all i but they are able to choose k and n. If $\gcd(k', n) = 1$, $\forall k' \leq k$, e.g., n is prime or is the product of two primes greater than k, then $\mathcal{S}_i$ is aperiodic for all i. The following algorithm assumes an oriented ring. An *active* token in unoccupied while an *inactive* token has a MA residing on it.

Algorithm 4

1. Release the token at the starting node.
2. Set *active* $= 1$.
3. Set *count* $= 0$.
4. Begin to walk around the ring in the clockwise direction.
5. Compute the intertoken distances to the next three active tokens, i.e., d_1, d_2, d_3, and increment *count* for each inactive token passed.
6. If *count* $== k - 1$, arrange rendezvous. (Only active MA remaining.)
7. If $d_2 > d_1$ and $d_2 \geq d_3$, then remain active.
8. Else become inactive, i.e., set *active* $= 0$, continue in current direction to starting node, and wait for further instructions.
9. Repeat from step 3.

Theorem 7 *When the MAs share a common orientation and $\gcd(k', n) = 1$, $\forall k' \leq k$, then the mobile agent rendezvous problem can be solved with $O(\lg n)$ memory and $O(n)$ time in an oriented ring.*

When k is prime, the ring is oriented, and $\gcd(k', n) = 1$, $\forall k' \leq k$, then a variation of algorithm 4 solves the mobile agent rendezvous problem with $O(\lg k)$ memory and $O(n)$ time.

Algorithm 5

1. Release the token at the starting node.
2. Set active $= 1$.
3. Begin to walk around the ring in a clockwise direction.

4. Execute round 1 of algorithm 4 but calculate the intertoken distances mod k. (All MAs will return to their starting nodes and those that became inactive have set active = 0.)
5. (Now execute algorithm 4 as if on a ring of size k). The distances of interest are now the number of inhabited tokens between pairs of empty tokens.)
6. Compute the number of inhabited tokens, i.e., tokens hosting inactive MAs, met on the path to the next three uninhabited tokens, i.e., m_1, m_2, m_3.
7. If $m_1 == k-1$, arrange the rendezvous. (Only one active MA left.)
8. If $m_2 > m_1$ and $m_2 \geq m_3$, then remain active.
9. Else become inactive, i.e., set *active* = 0, continue in current direction to starting node, and wait for further instructions.
10. Repeat from step 5.

Theorem 8 *When the MAs share a common orientation, k is prime, and* $\gcd(k', n) = 1$, *$\forall k' \leq k$, the mobile agent rendezvous problem can be solved with $O(\log k)$ memory and $O(n)$ time.*

The following algorithm solves the rendezvous problem when $\gcd(k', n) = 1$, $\forall k' \leq k$, but k is not prime.

Algorithm 6

1. Release the token at the starting node.
2. Set active = 1 and count = 0.
3. Begin to walk around the ring in the clockwise direction.
4. Compute the intertoken distances mod k to the next three active tokens, i.e., d_1, d_2, d_3 mod k, and increment *count* for each inactive token passed.
5. If *count* $== k-1$, arrange rendezvous. (Only active MA remaining.)
6. If $d_2 > d_1$ mod k and $d_2 \geq d_3$ mod k, then remain active.
7. Else become inactive, i.e., set *active* = 0, and wait for further instructions.
8. Repeat from step 4.

Theorem 9 *When the MAs share a common orientation and* $\gcd(k', n) = 1$, *$\forall k' \leq k$, then the mobile agent rendezvous problem can be solved with $O(\log k)$ memory and $O(n \log k)$ time.*

7 Leader Election and Rendezvous

The relationship between the rendezvous problem and the leader election problem among the k MAs depends on whether the MAs share a common orientation.

Theorem 10 *If the MAs share a common orientation, then the leader election problem among k MAs is equivalent to the rendezvous problem for those MAs. If the MAs do not share a common orientation, however, then the leader election problem is strictly more complex than the rendezvous problem.*

8 Conclusion

After proving that the mobile agent rendezvous search problem is unsolvable when both k and n are unknown, we prove that rendezvous is possible if and only if the sequence of intertoken distances is aperiodic. We then present unconditional and conditional solutions for the rendezvous problem. We derive the lower bounds on the memory required for mobile agent rendezvous and then discuss the relationship between rendezvous and leader election for mobile agents.

In future research, it would be interesting to study how changes in the model affect the complexity of the mobile agent rendezvous search problem. For example, it would be interesting to study a network topology that differs from the ring or the case where each mobile agent has more than one token.

Acknowledgements

This research is supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada) and by MITACS (Mathematics of Information Technology and Complex Systems) grants.

References

[1] S. Alpern, Rendezvous Search: A Personal Perspective, Operations Research, 50, No. 5, pp. 772-795, 2002.

[2] S. Alpern and S. Gal, The Theory of Search Games and Rendezvous, Kluwer Academic Publishers, London, 2003.

[3] L. Barriere, P. Flocchini, P. Fraigniaud, and N. Santoro, Election and Rendezvous of Anonymous Mobile Agents in Anonymous Networks with Sense of Direction, submitted, 2003.

[4] L. Barriere, P. Flocchini, P. Fraigniaud, and N. Santoro, Can We Elect If We Cannot Compare?, to appear, Proceedings of the 15th ACM Symposium on Parallel Algorithms and Architectures (SPAA'03), 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] S. Gal, Rendezvous Search on the Line, Operations Research, 47, No. 6, pp. 849-861, 1999.

[7] E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, Mobile Agent Rendezvous in the Ring, to appear, International Conference on Distributed Computing Systems (ICDCS 2003), Providence, RI, May 19 - 22, 2003.

[8] Rendezvous Search on the Line with More than Two Players, Operations Research, 45, pp. 357-364, 1997.

[9] M. Pikounis and L.C. Thomas, Many Player Rendezvous Search: Stick Together or Split and Meet? University of Edinburgh, Management School, Preprint, 1998.

[10] X. Yu and M. Yung, Agent Rendezvous: A Dynamic Symmtery-Breaking Problem, Proceedings of ICALP'96, LNCS 1099, pp. 610-621, 1996.

A Appendix

Proof of Theorem 1 in section 2.
The proof of Theorem 1 is based on the proof of Theorem 2.1 in Barriere et al [3]. Unlike our model, each network node in the Barriere et al [3] model has a whiteboard that the MAs can access with mutual exclusion.

Proof of Lemma 1 in section 3.
Consider an algorithm comprised of a single atomic operation. Any one of the atomic operations listed above will leave the distance between every pair of siblings unchanged, since either both MAs are stationary or each MA moves one node in the same direction. Assume that this is true for an algorithm that is a sequence of m atomic operations.

Consider an algorithm that is a sequence of $m+1$ atomic operations. Assuming that the distance between every pair of siblings is unchanged after m atomic operations, it will also be unchanged if the $m+1$th operation is one of "release the token", "do not move", or "increment a given variable" since the MAs are stationary under these atomic operations. The distance between every pair of siblings is also unchanged if the $m+1$th operation is "move one node in a given direction" since both MAs move one node in the same direction. Thus the distance between every pair of siblings is unchanged after $m+1$ atomic operations. This completes the proof of Lemma 1. ∎

Proof of Lemma 2 in section 3.
For all $t < t^*$, each MA is exactly n/g nodes away from its respective *siblings* and thus, given the placement of the tokens, either all the MAs of a given *trio* $\mathcal{T}$ are on separate tokens or none of the MAs in $\mathcal{T}$ are on tokens. If the MAs in $\mathcal{T}$ are on separate tokens at time $t < t^*$, then those MAs have identical values for *CLToToken* and identical values for *CCToToken*.

Suppose that at time t^*, none of the MAs in $\mathcal{T}$ are on tokens. Suppose further that there exists a time $t' < t^*$ such that t' is the first time that the *trio*'s respective values for *CLToken* differ. Choose two of the MAs, A and B, whose values for *CLToken* differ. Without loss of generality, assume that A's value for *CLToToken* is less than B's value for *CLToToken*, i.e., $CLToToken_A < CLToToken_B$. If both A and B moved $CLToToken_A$ nodes in a clockwise direction, then A would be on a node containing a token but B would not. Given the placement of the tokens, this implies that MAs A and B are not n/g nodes apart at time t' and thus contradicts the fact that t^* is the first time that at least one MA is not exactly n/g nodes from its respective *siblings*. This completes the proof of Lemma 2. ∎

Proof of Theorem 2 in section 2.
Part A: $1 \Rightarrow 2$
Suppose not, i.e., rendezvous is guaranteed but $\mathcal{S}$, the sequence of intertoken distances, is periodic. The k MAs can be placed on the ring as described in Barriere et al [4]. This is the MA arrangement that is assumed in Lemmata 1 and 2. Lemma 1 implies that if a MA begins an algorithm with its *siblings* at a distance of n/g nodes and the algorithm contains only atomic operations, then the MA will not rendezvous with its *siblings*. The inclusion of a *do-until* loop or an *if-then* statement is therefore a necessary condition for rendezvous.

The boolean value of the conditional statement in a *do-until* loop or an *if-then* statement determines, respectively, how many iterations of the loop occur or if the body of the *if-then* statement is executed. In either case, a conditional statement evaluated at time t is a function of the known history of the given MA at time $t-1$.

Let s_0^i represent the ith MA's state at time 0. The history of the ith MA at time t is defined as the sequence $H_t^i = s_0^i, s_1^i, ..., s_t^i$. If the k MAs rendezvous at time t, let $t^* < t$

denote the first time that the distance between some MA and at least one of its *siblings* is not n/g. Denote this *trio* of MAs by $\mathcal{T}$. Lemma 2 indicates that t^* is also the first time that the values of *CLToken* differ among the MAs in $\mathcal{T}$.

At time t^*, the distance between an MA in $\mathcal{T}$ and at least one of its *siblings* changes because the MA and its *sibling* executed the same conditional statements but received different Boolean values as a result. This implies that the histories of the MA and its *sibling* differed at time $t^* - 1$. Let $t' < t^*$ denote the first time that the histories of these MAs differed, i.e., the MAs' respective states differ at time t'. The states of the two MAs can differ at time t' only if conditional statements evaluated at time $t' - 1$ yielded different values. This implies, however, that the histories of the MAs differed at time $t' - 2$ and contradicts the fact that the MAs' histories first differed at time $t^* - 1$.

As a result, the distance between a MA and at least one of its *siblings* is always n/g nodes and thus rendezvous cannot occur when $\gcd(k, n) = 1$. This contradicts the fact that rendezvous is guaranteed and thus statement 1 implies statement 2.

Part B: $2 \Rightarrow 1$

Suppose not, i.e., $\mathcal{S}$, the sequence of intertoken distances, is aperiodic but rendezvous is not guaranteed. If rendezvous cannot occur, then the rotations of the sequence $\mathcal{S}$ have no lexicographic maximum. Otherwise the MA could rendezvous at the node that begins the lexicographically maximum rotation. This implies, however, that the sequence of intertoken distances is periodic. This contradicts the fact that $\mathcal{S}$, is aperiodic and thus statement 2 implies statement 1. This completes the proof of Theorem 2. ∎

Proof of Theorem 3 in section 4.

With $O(k \lg n)$ memory, each MA can calculate $\mathcal{S}$, the sequence of intertoken distances, after walking around the ring once. There is sufficient memory to determine if $\mathcal{S}$ is periodic in step 4 of Algorithm 1 and thus the MAs will stop in step 5 if rendezvous is not possible. If the MAs reach step 6 of Algorithm 1, $\mathcal{S}$ is aperiodic and thus the *forward* and *reverse* sequences in step 7 have unique lexicographically maximum rotations. Steps 8 to 11 in Algorithm 1 ensure that the respective lexicographically maximum rotations of the *forward* and *reverse* sequences can be used to identify a unique node at which rendezvous will occur. Since the calculation of the initial sequence of intertoken distances and the determination of periodicity occurs in $O(n)$ time, Algorithm 1 guarantees that the rendezvous problem is solved in time $O(n)$. This completes the proof of Theorem 3. ∎

Proof of Theorem 4 in section 4.

In round c of Algorithm 2, an active MA calculates the intertoken distance between the first c tokens and then walks around the ring comparing this distance to the corresponding distances met. If $\mathcal{S}$, the sequence of intertoken distances, is aperiodic, then in each round of the algorithm, i.e., with each traversal of the ring, at least one MA becomes inactive and at least one MA remains active. Otherwise the intertoken distances are equal. Thus after one round, if all MAs are still active, the intertoken distances are equal and rendezvous is impossible. After at most k rounds of the algorithm, one MA will remain active and that MA can arrange the rendezvous. Each round of the algorithm requires a complete traversal of the ring and thus takes time $O(n)$. With at most k rounds required, the resulting time for Algorithm 2 is $O(kn)$. This completes the proof of Theorem 4. ∎

Proof of Theorem 7 in section 6.

First we prove that algorithm 4 solves the rendezvous problem. If, for all i, $\mathcal{S}_i$ is aperiodic, then at least one MA will become inactive and at least one MA will remain active for each iteration of step 7. Thus after at most $\lg k$ rounds, one MA will remain active and can arrange the rendezvous. If $\mathcal{S}_i$ is periodic for some i, however, then there is a round in which all MAs will become inactive and so rendezvous is impossible. Thus algorithm 4 solves the mobile agent rendezvous problem.

To prove that the time complexity of algorithm 4 is $O(n)$, we show that in each round, unique ids can be assigned to the MAs such that the MA whose d_2 value is at the beginning of the lexicographically maximum rotation of $\mathcal{S}$ always remains active. This implies that an instance of the execution of algorithm 4 can be mapped to an instance of Peterson's leader election algorithm with unique ids and thus algorithm 4 requires time $O(n)$.

To prove that the required mapping exists, we use induction on the number of rounds of algorithm 4 and induction on the number of active MAs at the beginning of a round.

Base case (last round): Let z denote the number of the last round of algorithm 4. Let m_z denote the number of MAs that are active at the beginning of round z. Let $\mathcal{S}_z$ denote the sequence of intertoken distances for the m_z MAs that are active at the beginning of round z. Let MA_{max} denote the MA at the start of the lexicographically maximum rotation of $\mathcal{S}_z$

We prove via induction that unique ids can be assigned to the m_z MAs that are active at the beginning of round z such that the MA at the beginning of the lexicographically maximum rotation of $\mathcal{S}_z$ has the maximum id and remains active at the end of algorithm 4 and is elected under Peterson's leader election algorithm.

Base case ($m_z = 2$): With $m_z = 2$ active MAs, there are only two intertoken distances, d_1, d_2. Since $\mathcal{S}_z$ is aperiodic, $d_1 \neq d_2$. Without loss of generality, let $d_1 > d_2$. Assign $id = d_1$ to the MA at the head of the intertoken segment with distance d_1 and assign $id = d_2$ to the other MA.

Inductive hypothesis: Assume that unique ids can be assigned to the $m_z = q$ active MAs at the beginning of round z such that the MA at the head of the lexicographically maximum rotation of $\mathcal{S}_z$ remains active.

$m_z = q + 1$: Let MA' denote an MA that does not start the lexicographically maximum rotation of $\mathcal{S}_z$ and whose distance to the next token, d', can be decreased by one without affecting which MA starts the lexicographically maximum rotation of $\mathcal{S}_z$. Consider all the MAs except for MA_{min}. The inductive hypothesis implies that these q MAs can be assigned unique ids such that the MA at the head of the lexicographically maximum rotation of $\mathcal{S}_z$ remains active. If we double these ids and assign $id = d'$ to MA', then the last active MA in algorithm 4 will be exactly the MA that would be elected by the Peterson leader election algorithm with unique ids.

Inductive Hypothesis: Assume that for any round between the first and the last round of algorithm 4, unique ids can be assigned to the active MAs at the beginning of the round such that the last active MA in algorithm 4 will be exactly the MA that elected by the Peterson leader election algorithm with unique ids.

First round of algorithm 4: The MAs that are active at the beginning of round 2 are exactly those MAs that are active at the end of round 1. Thus the inductive hypothesis implies that the MAs that are active at the end of round 1 can be assigned unique ids and this includes the MA whose d_2 value begins the lexicographically maximum sequence of $\mathcal{S}_1$, the sequence of itertoken distances for round 1. Given these ids, we need to assign additional unique ids to the MAs that become inactive at the end of round 1. Let ID_{active} represent the set of unique ids for the MAs that remain active at the end of round 1.

Let m_1 denote the number of MAs that become inactive at the end of round 1. We prove via induction that we can assign unique ids to the m_1 MAs that are inactive at the end of round 1 such that the MA at the beginning of the lexicographically maximum rotation of $\mathcal{S}_1$ has the maximum id and thus remains active at the end of round 1 and is elected under Peterson's leader election algorithm.

Base case ($m_1 = 1$): If $min(ID_{active}) > 1$, we can assign $id = min(ID_{active}) - 1$ to the MA that becomes inactive at the end of round 1. Otherwise, increase each id in ID_{active} by one and assign $id = 1$ to the MA that becomes inactive at the end of round 1.

Inductive hypothesis: Assume that unique ids can be assigned to the $m_1 = q$ MAs that become inactive at the end of round 1 such that the MA at the head of the lexicographically maximum rotation of $\mathcal{S}_1$ has the maximum id.

$m_1 = q + 1$: Let MA' denote an MA that becomes inactive at the end of round 1 and whose distance to the next token, d', can be decreased by one without affecting which MA starts the lexicographically maximum rotation of $\mathcal{S}_1$. Consider the MAs that become inactive after round 1 excluding MA'. The inductive hypothesis implies that these q MAs can be assigned unique ids such that the MA at the head of the lexicographically maximum rotation of $\mathcal{S}_1$ has the maximum id and remains active at the end of round 1. If we double the ids assigned to all MAs, active and inactive excluding MA', and assign $id = d'$ to MA', then the MA with the maximum value for d_2 in round 1 will remain active at the end of round 1 in algorithm 4 and will be elected in the Peterson leader election algorithm with unique ids.

Thus unique ids can be assigned to all MAs at the start of algorithm 4 such that the execution of algorithm 4 can be mapped to an execution of the Peterson leader election algorithm with unique ids. As a result, algorithm 4 has the same time complexity as the Peterson leader election algorithm with unique ids, i.e., $O(n)$. This completes the proof of Theorem 7. ∎

Proof of Theorem 9 in section 6.
Since $\gcd(k', n) = 1, \forall k' \leq k$, rendezvous is possible but the MAs do not have enough memory to identify the lexicographically maximum rotation of the sequence of intertoken distances in a single traversal of the ring.

In each execution of steps 4 to 7 of algorithm 6, at least one-third of the active MAs become inactive. The fact that $\gcd(k', n) = 1$, $\forall k' \leq k$, guarantees that the sequence of intertoken distances in each round is aperiodic so that rendezvous remains possible as the number of active MAs decreases. After at most $\log k$ rounds, the only MA that is still active is the one at the start of the lexicographically maximum rotation of the sequence of intertoken distances. Each round takes time n, so algorithm 5 ensures a rendezvous in time $O(n \log k)$. This completes the proof of Theorem 9. ∎

Proof of Theorem 10 in section 7.
If the leader election problem can be solved for the model described in section 1.1, then the elected MA can travel around the ring and instruct all other MAs to meet at a particular node. Thus a solution to the leader election problem for MAs always implies that there is a solution for the corresponding mobile agent rendezvous problem.

However, a solution to the mobile agent rendezvous problem does not necessarily imply a solution to the leader election problem. Consider the case where $k = 2$ MAs reside on $n = 3$

node ring, k is known but the MAs do not share a common orientation. If the rendezvous algorithm instructs the MAs to:

1. Choose a direction and begin to walk.
2. Walk until either:
   (a) the other MA is met at a node, i.e., rendezvous occurs and the algorithm is done, or
   (b) a token is discovered.
3. If a token is discovered, change direction and continue walking until rendezvous occurs.

If the MAs choose opposite directions initially, they will rendezvous on the node that has no token in at most two steps yet, given the symmetry of the problem, one MA cannot be distinguished from the other. Their memory contents are identical since they executed exactly the same steps of the algorithm and were on comparable nodes at the same time. Thus when the MAs do not share a common orientation, the leader election problem for MAs is strictly more difficult than the corresponding rendezvous problem.

If the MAs do share a common orientation, however, then a solution to the rendezvous problem always implies a solution to the corresponding leader election problem. In algorithm 2 of section 4, the last MA that remains active organizes the rendezvous and that MA can be elected the leader. This completes the proof of Theorem 10. ■