Mobile Agent Rendezvous in a Ring

Evangelos Kranakis* Danny Krizanc† Nicola Santoro* Cindy Sawchuk*

Abstract

In the rendezvous search problem, two mobile agents must move along the n nodes of a network so as to minimize the time required to meet or rendezvous. When the mobile agents are identical and the network is anonymous, however, the resulting symmetry can make the problem impossible to solve. Symmetry is typically broken by having the mobile agents run either a randomized algorithm or different deterministic algorithms. We investigate the use of identical tokens to break symmetry so that the two mobile agents can run the same deterministic algorithm. After deriving the explicit conditions under which identical tokens can be used to break symmetry on the n node ring, we derive the lower and upper bounds for the time and memory complexity of the rendezvous search problem with various parameter sets. While these results suggest a possible tradeoff between the mobile agents' memory and the time complexity of the rendezvous search problem, we prove that this tradeoff is limited.

1 Introduction

The rendezvous search problem for mobile agents is a search optimization problem based on the following question:

> How should two mobile agents move along the n nodes of a network in order to minimize the time required to meet or rendezvous?

Yu and Yung[23] note that the activities of mobile agents may lead them to networks where, due to faulty nodes, incompatibile naming conventions, or a refusal to provide IDs, the host nodes cannot be uniquely and globally identified. Such problems increase the difficulty of the rendezvous search problem. For example, when the nodes in a network are uniquely numbered, the rendezvous search problem is trivial since identical mobile agents can rendezvous at the node with the lowest identification number. The problem is more difficult if the nodes of the network do not have unique identifiers and becomes even harder if there exists no globally fixed orientation or sense of direction [14].

Algorithm designers typically break symmetry in the rendezvous search problem either by using randomized algorithms or by having the mobile agents use different deterministic algorithms. (See [1] to [8], [10] to [12], [15], and [17] to [23]). In fact, Yu and Yung [23] prove that the rendezvous search problem cannot be solved on a general graph if the mobile agents use the same deterministic algorithm. While Baston and Gal [11] mark the starting points of the players, i.e., mobile agents, they still rely on randomization or different deterministic algorithms when solving the rendezvous search problem.

Rather than using randomized algorithms or different deterministic algorithms for various mobile agents, we want to explore how symmetry can be broken with identical tokens. While these tokens are similar to the marked starting points of Baston and Gal [11], we assume that the mobile agents participating in the rendezvous search problem all run the same deterministic algorithm.

1.1 The Network Model

Our model consists of two identical mobile agents that are $d \leq \frac{n}{2}$ nodes apart in an anonymous, synchronous, and possibly oriented n node ring. An oriented ring is defined as a ring where all the mobile agents on nodes in the ring share a common orientation, i.e., they agree on the direction considered to be clockwise. A given node requires only enough memory to host a token and at most two mobile agents. Each mobile agent, MA, owns a single identical stationary token, i.e., the tokens are indistinguishable and, once they are positioned in the ring, they cannot be moved. 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. The MAs follow the **same** deterministic algorithm and begin execution at the same time.

Memory permitting, a MA can count the number of nodes visited, 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, the number of nodes to the nearest token, 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 the MAs either meet on a network node or simultaneously cross the same network link while moving in opposite directions. While we are aware of the problems that arise from the latter definition of rendezvous, we defer the study of these problems to a later date. Note that when the number of nodes in the network is odd, rendezvous occurs on a node but when the number of nodes is even, rendezvous occurs on a network link.

We assume that rendezvous occurs as soon as possible after the MAs begin travelling in opposite directions, i.e., one MA moves clockwise around the ring, while the other MA moves counterclockwise.

Known			Lower/Upper	Memory
n	d	$oriented$	Bound	Required
Y	Y	Y	$3d/2$	$O(\log d)$
Y	Y	N	$3d/2, 5d/2$	$O(\log d)$
Y	N	Y	$(n+d)/2$	$O(\log n)$
Y	N	N	$(n+d)/2$	$O(\log n)$
N	Y	Y	$3d/2$	$O(\log d)$
N	Y	N	$3d/2, 5d/2$	$O(\log d)$
N	N	Y	$n+(d/2)$	$O(\log n)$
N	N	N	$n+(d/2)$	$O(\log n)$

Table 1. Time Complexity for Rendezvous Search Problem with Two Mobile Agents at Distance $d < \frac{n}{2}$

1.2 Our Contribution

In this paper, we study the rendezvous search problem for two identical MAs in an anonymous, synchronous, and possibly oriented n node ring. Rather than using randomized algorithms or different deterministic algorithms to break symmetry, we use identical stationary tokens to break symmetry so that the MAs can run the same deterministic algorithm.

For example, consider an anonymous, synchronous and oriented ring with $n = 5$ nodes where two identical MAs are located $d = 1$ node apart. If the MAs simultaneously run the same deterministic algorithm, they will not rendezvous since they move in the same direction, at the same speed, and at the same time. Rendezvous is guaranteed, however, if each MA has an identical stationary token that it leaves at its respective starting node. The algorithm directs each MA to walk until a token is reached, reverse direction, and continue walking. Rendezvous will occur after each MA has walked at most three nodes.

Consider another example where $n = 4$ is even and $d = 2$. Identical tokens cannot be used, in the manner described above, to break symmetry in this case and thus rendezvous is impossible. Our first result is the generalization of this example, i.e., when $d = \frac{n}{2}$ it is impossible to construct an algorithm that uses identical stationary tokens to break symmetry and thus solve the rendezvous search problem.

Having established when identical stationary tokens **cannot** be used to break symmetry, we then consider the rendezvous problem where $d < \frac{n}{2}$. Each MA places its identical stationary token on its respective starting node in the ring and begins to walk around the ring at the speed of one node per unit of time. The MAs know that the network is a ring but they do not necessarily know d, n, whether $d < \frac{n}{2}$, or the orientation of the ring.

In Table 1, we show the upper and lower bounds on the time complexity necessary for a rendezvous between two mobile agents that use identical stationary tokens and know that $d < \frac{n}{2}$. If only one value appears in the *Lower/Upper Bound* column, then the bound is tight. Otherwise the lower bound precedes the upper bound. While the calculation of the lower bounds assumes that the MAs have unbounded memory, the calculation of the upper bounds assumes the amount of memory listed in the last column of Table 1.

As Table 1 indicates, if the MAs know d, then the lower bound on the time complexity of the rendezvous search problem is $\frac{3d}{2}$. If d is unknown but n is known, the lower bound on the time complexity increases to $\frac{n+d}{2}$. If both d and n are unknown, however, then the lower bound on the time complexity increases to $n + \frac{d}{2}$.

To calculate the upper bounds on the time complexity, we present four solutions to the rendezvous search problem when $d < \frac{n}{2}$ and identical stationary tokens are used to break symmetry. These solutions, with one exception, prove that the lower bounds in Table 1 are tight. The exception occurs when d is known but the orientation of the ring is unknown. The lower bound on the time complexity is $\frac{3d}{2}$ but the corresponding upper bound is $\frac{5d}{2}$.

While the results in Table 1 assume that the MAs know $d < \frac{n}{2}$, the solutions can be easily changed so that the MAs stop if $d = \frac{n}{2}$. It merely requires that each MA have memory $O(\log n)$. The last row in Table 1 represents the case where the MAs do not know n, d, or the orientation of the ring. The solution used to calculate the upper bound requires $O(\log n)$ memory and $\frac{5n}{4}$ time. In Algorithm 1, we present another solution for the case where the MAs do not know n, d, or the orientation of the ring, but this solution requires $O(\log \log n)$ memory and $O(\frac{n \log n}{\log \log n})$ time. While Algorithm 1 suggests that the MAs' memory complexity can be reduced at the expense of a solution's time complexity, we prove that if the MAs do not know n, d, or the orientation of the ring, then an algorithm that solves the rendezvous search problem when $d < \frac{n}{2}$ and stops otherwise requires $\Theta(\log\log n)$ memory.

This result, plus the upper bounds stated earlier, implies that solving the rendezvous search problem when the MAs have only $O(1)$ memory requires a change in the model. Suppose that the tokens are no longer stationary, i.e., anytime a MA encounters a token, it can move the token to an adjacent node. Algorithm 2, which requires only $O(1)$ memory, successfully uses identical but movable tokens to break symmetry and solve the rendezvous search problem. The resulting time complexity approaches $O(n)$ as d approaches 1, and approaches $O(n^4)$ as d approaches $\frac{n}{2}$.

1.3 Outline of the Paper

In section 2, we prove that when $d = \frac{n}{2}$, two MAs cannot use identical stationary tokens to break symmetry in an anonymous, synchronous, and possibly oriented ring. In section 3, we prove lower and upper bounds for the rendezvous search problem when the MAs know that $d < \frac{n}{2}$ and use identical stationary tokens to break symmetry. The bounds show the impact of the agents' knowledge of n and d, as well as the orientation of the ring, on the search time. In section 4, we present an algorithm that again uses identical stationary tokens but does so with $O(\log\log n)$ memory and runs in time $O(\frac{n\log n}{\log\log n})$. The solutions in sections 3 and 4 imply that there is a tradeoff between the MAs' memory and the resulting time complexity of the rendezvous search problem. In section 5, however, we prove that this tradeoff is limited since, when the MAs do not know n, d, or the orientation of the ring, they need $\Theta(\log\log n)$ memory for an algorithm that stops if a rendezvous is impossible and otherwise achieves a rendezvous.

The results of section 5 imply that finding a $O(1)$ memory solution for the rendezvous search problem requires a change in the model. In section 6, we allow a MA to move a token anytime it resides on the same node as that token. We present a $O(1)$ memory algorithm that uses these movable tokens to solve the rendezvous search problem when the MAs do not know n, d, or the orientation of the ring. The paper concludes with a summary and a short discussion of open problems.

2 The Feasibility of Rendezvous

Before we construct algorithms that use identical stationary tokens to break symmetry in the rendezvous search problem, we need to identify the conditions under which identical stationary tokens can be used to that end. In Theorem 1 below, we prove that when $d = \frac{n}{2}$, the MAs **cannot** use identical stationary tokens to break symmetry and thus, if they simultaneously run the same deterministic algorithm, they will not rendezvous.

Let the atomic operations in an algorithm be restricted to "release the token", "move one node in a given direction", and "do not move". An algorithm may also contain two constructs. A *do-until* loop instructs the MAs to perform a finite sequence of operations until a given state occurs, while an *if-then* statement checks if a given condition holds and, if so, instructs the MAs to perform a finite sequence of operations. The following lemma is needed in the proof of Theorem 1.

Lemma 1 *Assume that the MAs are initially $d > 0$ nodes apart on the network. If the MAs simultaneously run the same deterministic algorithm and that algorithm is restricted to a fixed, finite sequence of atomic operations, i.e., there are no* do-until *loops or* if-then *statements, then the distance, d, between the MAs will not change.*

Lemma 1 implies that the inclusion of a *do-until* loop or an *if-then* statement in an algorithm is a necessary condition for a rendezvous. Let s_0^i denote the ith MA's state at time 0. The ith MA's history at time t is defined as the sequence $H_t^i = s_0^i, s_1^i, ..., s_t^i$. The boolean value of the conditional statement in a *do-until* loop or an *if-then* statement evaluated at time t is a function of the history of the given MA at time $t-1$. Two MAs cannot evaluate the same conditional statement at time t and receive different boolean values unless their histories differ at time $t-1$.

Theorem 1 *Consider two identical MAs in an anonymous, synchronous n node ring. The MAs are initially $d = \frac{n}{2}$ nodes apart on the network. To begin an algorithm, the MAs place identical tokens on their respective starting nodes and these tokens remain in place for the rest of the algorithm. If the two MAs simultaneously run the same deterministic algorithm, they will not rendezvous.*

Sketch of the **Proof** of Theorem 1. Suppose that the two MAs rendezvous at time t. Consider the first time, $t^* < t$, that $d \neq \frac{n}{2}$. At time t^*, d changes because one MA moved and the other did not, or because both MAs moved but in opposite directions. The MAs must have evaluated conditional statements from a *do-until* loop or an *if-then* statement and received different boolean values. This implies that the histories of the two MAs differed at time $t^* - 1$. Let time $t' \leq t^*$ denote the first time that the histories of the MAs differ. If the histories of the MAs first differ at time t', then the states of the two MAs first differ at time t'. This can happen only if a conditional statement evaluated at time $t' - 1$ yielded different values for the two MAs. This implies, however, that the histories of the MAs differed at time $t' - 1$ and contradicts the fact that t' was the first time that the histories differed. This proves Theorem 1. ■

3 The Time Complexity of Rendezvous

While Theorem 1 proves that identical stationary tokens cannot be used to break symmetry when $d = \frac{n}{2}$, Theorems 2 and 3 below provide the lower and upper bounds for the time complexity of the rendezvous search problem when symmetry can be broken using identical stationary tokens, i.e., $d < \frac{n}{2}$.

Theorem 2 *Consider two identical MAs at distance $d < \frac{n}{2}$ from each other in an anonymous, synchronous, n node ring. The MAs know that $d < \frac{n}{2}$ and that the network is a ring. The MAs are assumed to have unbounded memory. When the MAs simultaneously run the same deterministic algorithm for the rendezvous search problem, the lower bounds stated in Table 1 hold.*

Proof of Theorem 2. Consider the case where the MAs know d. The two MAs begin the rendezvous search problem with the same state. After a MA travels d nodes in given direction from its starting node, it has new information because it either finds a token or learns that the token is in the other direction from the starting node. The MAs are still at least d nodes apart, however, so a rendezvous will require that the MAs travel at least another $\frac{d}{2}$ nodes. Since the MAs must travel at least $\frac{3d}{2}$ nodes for a rendezvous to occur and moving to an adjacent node takes one unit of time, a rendezvous requires at least $\frac{3d}{2}$ units of time when d is known. This proves the lower bound for all cases where d is known.

Now assume that the MAs do not know d but they know n, the number of nodes in the ring. One of the MAs may find a token after travelling d nodes in a given direction. The other MA, however, will not have any new information until it travels $\frac{n}{2}$ nodes in the same direction. At that time, the MAs will still be at least d nodes apart. Given that the MAs must travel at least $\frac{n+d}{2}$ nodes for a rendezvous to occur, and moving to an adjacent node takes one unit of time, then a rendezvous requires at least $\frac{n+d}{2}$ units of time when n is known and d is unknown. This proves the lower bound for all cases where n is known and d is unknown.

In the remaining cases, the MAs know neither n nor d. A MA can count m, the number of nodes from its starting node to the first token, but it cannot tell if $m = d$ or $m = n - d$ until it has travelled around the ring and counted n. When the MAs return to their respective starting nodes, they are still d nodes apart. Given that the MAs must travel at least $n + \frac{d}{2}$ nodes for a rendezvous to occur, and moving to an adjacent node takes one unit of time, then a rendezvous requires at least $n + \frac{d}{2}$ units of time when n and d are unknown. This proves the lower bound for all cases where n and d are unknown. ∎

The upper bounds for the time complexities stated in Table 1 are proven in Theorem 3. They are derived by constructing a solution that ensures a rendezvous under the given conditions, e.g., when d and the orientation of the ring are known but n is not necessarily known. While the calculation of the lower bounds in Theorem 2 assumes that the MAs have unbounded memory, the memory requirements for the upper bounds are those of the corresponding solution.

Theorem 3 *Consider two identical MAs at distance $d < \frac{n}{2}$ from each other in an anonymous, synchronous n node ring. The MAs know that $d < \frac{n}{2}$ and that the network is a ring. When the MAs simultaneously run the same deterministic algorithm for the rendezvous search problem, the upper bounds stated in Table 1 hold. In particular, with the exception of the case where d is known and the ring is not oriented, the lower bounds of Theorem 2 are tight.*

Proof of Theorem 3. The following four mutually exclusive cases exhaustively describe the subsets of knowledge that a MA may possess about a ring, i.e., n, d, or the ring's orientation.

Case 1: Each mobile agent has memory $O(\log d)$, knows d and the ring's orientation, but may or may not know n. With d and the orientation known, one of the mobile agents is guaranteed to find a token in d steps.

Algorithm for Case 1
1. Release the token.
2. Begin walking around the ring in a counter-clockwise direction.
3. If a token is found within d steps, continue walking in the same direction.
4. Otherwise, if no token is found by d steps, reverse direction and continue walking.

The mobile agent that finds a token at d steps continues walking in the same direction. The other mobile agent reverse its direction and continues walking. Rendezvous will occur in time $\frac{3d}{2}$.

Case 2: Each mobile agent has memory $O(\log d)$, knows d, does not know the ring's orientation, and may or may not know n. Given d, a mobile agent needs to walk at most d steps before deciding whether to reverse direction.

Algorithm for Case 2
1. Release the token.
2. Choose a direction and begin walking around the ring.
3. If a token is found within d steps, continue walking in the same direction.
4. Otherwise, if no token is found by d steps, reverse direction and continue walking.

In the worst case, $n > 3d$ and the two mobile agents choose opposite directions in step 2 of the algorithm such that they are walking on the $long$ side of the ring. They each walk d steps without reaching a token, so they then reverse direction and walk until rendezvous occurs. At time $2d$, each mobile agent is back at its original position, i.e., where it released its token in step 1, so the mobile agents will rendezvous in an additional $\frac{d}{2}$ steps. As a result, the algorithm ensures a rendezvous in time $\frac{5d}{2}$.

Case 3: Each mobile agent has memory $O(\log n)$, knows n, does not know d and may or may not know the orientation. Given n, a mobile agent walks at most $\frac{n}{2}$ steps before deciding whether to reverse direction.

Algorithm for Case 3

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. If a token is found within $\frac{n}{2}$ steps, continue walking in the same direction.
4. Otherwise, reverse direction at $\frac{n}{2}$ steps and continue walking.

In the worst case, the two mobile agents choose the same direction in step 2 of the algorithm. One of the mobile agents walks $n/2$ steps, finds no token, and thus reverses direction and continues walking. The other mobile agent walks d steps, finds a token, maintains the current direction, and continues walking. As a result, rendezvous occurs in time $\frac{n+d}{2}$.

Case 4: Each mobile agent has memory $O(\log n)$ and knows the ring's orientation, but does not know n or d.

Algorithm for Case 4

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. Count the number of steps to the first token, δ_1, and continue walking.
4. Count the number of steps to the second token, δ_2.
/*The mobile agent is back at its starting node.*/
5. If $\delta_1 < \delta_2$, continue walking in the same direction.
6. Otherwise, reverse direction and continue walking.

In the worst case, the two mobile agents choose the same direction in step 2 of the algorithm and thus rendezvous will occur in time $n + \frac{d}{2}$. ■

All of the algorithms discussed so far require either $O(\log d)$ or $O(\log n)$ memory. In the *special case* where d is known to be less than or equal to $\frac{n}{3}$, there exists an algorithm that achieves rendezvous with memory $O(1)$.

Theorem 4 *Consider two identical MAs at distance d from each other in an anonymous synchronous n node ring. In this* special case*, the MAs know that $d \leq \frac{n}{3}$ and that the network is a ring, but do not know n, d, nor the orientation of the ring. Let each MA have memory $O(1)$. The time complexity of the rendezvous search problem in this case is $\frac{n+d}{2}$.*

Proof of Theorem 4. Each mobile agent has memory $O(1)$, knows that $d \leq \frac{n}{3}$, i.e., d is at most *one third* of n, but does not know n, d, or the orientation.

Algorithm for $O(1)$ memory case

1. Release the token.
2. Choose a direction and begin walking around the ring.
3. At the first token, reverse direction and continue walking.
4. At the second token, reverse direction and continue walking.

In the worst case, the MAs will travel in the same direction. One MA will find a token after travelling d nodes in the given direction, and will reverse direction and continue to travel. At time $2d$, this MA will have returned to its original position. The other MA, however, has not yet reached this token since $d \leq \frac{1}{3}$. The two MA will rendezvous in an additional $\frac{n-3d}{2}$ steps. Thus the two MA will rendezvous in time $\frac{n+d}{2}$. ■

While the preceding solutions assume that the MAs know $d < \frac{n}{2}$, the solutions can be easily changed so that the MAs stop if $d = \frac{n}{2}$. Suppose that in all cases, the MAs have $O(\log n)$ memory. A MAs can count the number of steps taken after it makes a decision whether or not to change direction. If the intertoken distance is $d = \frac{n}{2}$, then the MA can stop as rendezvous is impossible. Otherwise, the MA proceeds as before.

4 Memory Tradeoff

With $O(\log n)$ memory, the MAs can detect if $d = \frac{n}{2}$ and act appropriately, i.e., stop if $d = \frac{n}{2}$ and rendezvous otherwise. If the MAs did not know n, d, or the orientation of the ring, the time complexity of the solution is $n + \frac{d}{2}$. When the knowledge of the MAs is unchanged but their memory is reduced to $O(\log \log n)$, the following algorithm can detect if $d = \frac{n}{2}$ and act appropriately. The time complexity of the algorithm is $O(\frac{n \log n}{\log \log n})$, which suggests that the MAs' memory size can be reduced at the expense of a solution's time complexity.

Algorithm 1

Let $p_1, \ldots, p_k$ denote the first k prime numbers such that $\prod_{i=1}^{k} p_i > n$.

1. Release the token.
2. Set $m = p_1$.
3. Choose a direction and begin travelling around the ring.
4. Count the number of steps, $\bmod\ m$, to the first token, δ_1, and continue walking.
5. Count the number of steps, $\bmod\ m$, to the second token, δ_2. /* The MA is back at its starting node. */
6. If $\delta_1 \bmod m = \delta_2 \bmod m$,
   If $m = p_k$, stop. *Rendezvous is not possible.*
   If $m < p_k$, set $m = p_{i+1}$ and repeat from step 4.
7. If $\delta_1 \bmod m < \delta_2 \bmod m$, continue travelling in the same direction.
8. Else, reverse direction and continue travelling.

/* If step 7 or 8 is executed, rendezvous occurs in another $\frac{d}{2}$ steps.*/

Theorem 5 *Consider two identical MAs with memory $O(\log\log n)$ that are distance d from each other in an anonymous, synchronous n node ring. The MAs do not know n, d, the orientation of the ring, or if $d = \frac{n}{2}$. The MAs simultaneously run the same deterministic algorithm. In the rendezvous search problem for two such MAs, Algorithm 1 correctly detects if $d = \frac{n}{2}$ and acts appropriately, i.e., stops if $d = \frac{n}{2}$ and rendezvous otherwise. The time complexity of the algorithm is $O(\frac{n \log n}{\log\log n})$.*

Proof of Theorem 5. The Chinese Remainder Theorem implies that if $d \equiv (n-d) \bmod p_i$ for all $p_i, i = 1, \ldots, k$, then $d \equiv (n-d) \bmod \prod_{i=1}^{k} p_i$. Let the p_i be the first k prime numbers such that $\prod_{i=1}^{k} p_i > n$. The algorithm checks each p_i in turn to see if $d \equiv (n-d) \bmod p_i$. If the statement is true for all p_i, then $d = n - d = \frac{n}{2}$ and the algorithm stops since rendezvous is impossible. If $d \not\equiv (n-d) \mathrm{mod}\ p_i$ for some p_i, however, then $d < \frac{n}{2}$. The first time the algorithm discovers such a p_i, one of the MAs reverses its direction and a rendezvous occurs $\frac{d}{2}$ steps later.

The worst case occurs when $d = \frac{n}{2}$ since all k of the p_i values have to be checked. The resulting run time is $O(kn)$, but we need to determine the value of k.

Consider the smallest k such that $\prod_{i=1}^{k} p_i > n$. This implies that $\prod_{i=1}^{k-1} p_i \leq n$ and thus $\prod_{i=1}^{k} p_i \leq n * p_k \leq n^2$. As a result, $\prod_{i=1}^{k} p_i \in O(n)$.

Let $\pi(m)$ denote the number of prime numbers less than or equal to m. Apostol [9] states that for any integer m, $\frac{m}{6 \log m} \leq \pi(m) \leq \frac{8m}{\log m}$. Since p_i is prime by definition, $\pi(p_i) = i$ for all i so that $\frac{p_i}{6 \log p_i} \leq \pi(p_i) = i \leq \frac{8 p_i}{\log p_i}$ and thus $p_i \geq \frac{i \log p_i}{8}$. Taking the product over all i and using Stirling's approximation [13] results in

$$\prod_{i=1}^{k} p_i \geq \prod_{i=1}^{k} \frac{i \log p_i}{8} = k! 8^{-k} \prod_{i=1}^{k} \log p_i \geq k! 8^{-k} \geq 2^{\Omega(k \log k)} \tag{1}$$

The logarithm of equation 1, given that $\prod_{i=1}^{k} p_i \in O(n)$, indicates that $O(\log n) \geq k \log k$, so a valid value for k must satisfy $k \log k \leq c \log n$.

Let $k = (\log n)^{1-\epsilon}$. Substituting for k yields $(1-\epsilon)\log\log n \leq c(\log n)^{\epsilon}$ so that, given a constant $c > 0$, a corresponding value of ϵ can be calculated such that $(\log n)^{1-\epsilon}$ is a valid value for k. The maximum value for k is then at least $(\log n)^{1-\epsilon}$. Substituting $(\log n)^{1-\epsilon}$ for k in $k \log k \leq c \log n$ yields $k \log(\log n)^{1-\epsilon} \leq c \log n$ such that $k \leq \left(\frac{c}{1-\epsilon}\right) \frac{\log n}{\log\log n}$ and thus $k \in O(\frac{\log n}{\log\log n})$. As a result, the time complexity of the algorithm, $O(kn)$, is $O(\frac{n \log n}{\log\log n})$. ■

5 Limits to the Memory Tradeoff

While the discussion in section 4 implies that the MAs' memory size can be reduced at the expense of a solution's time complexity, the following theorem proves that Algorithm 1 achieves the lower bound of memory required to solve the rendezvous search problem under the given model.

Theorem 6 *Consider two identical MAs that are distance d from each other in an anonymous, synchronous n node ring. The MAs do not know n, d, the orientation of the ring, or if $d = \frac{n}{2}$. The MAs simultaneously run the same deterministic algorithm. To begin an algorithm, each MA places an identical token on its respective starting node and these tokens remain in place for the rest of the algorithm. Any algorithm that correctly detects if $d = \frac{n}{2}$ and acts appropriately, i.e., stops if $d = \frac{n}{2}$ and rendezvous otherwise, requires that each MA have at least m bits of memory, where m is $\Omega(\log\log n)$.*

Proof of Theorem 6. Consider a unidirectional ring of n nodes where $k = 2$ MAs are placed $d = \frac{n}{2}$ nodes apart. Each MA has m bits of memory. Without loss of generality, label one of the MAs as MA_1 and the other as MA_2. Let Γ denote the non-empty collection of algorithms that correctly solve the rendezvous search problem for two MAs with m bits of memory each, i.e., every algorithm in Γ stops if $d = \frac{n}{2}$ and otherwise ensures a rendezvous. Let A denote an arbitrary algorithm in Γ.

Construct the following sequence of bits for each node in the ring. Let the first m bits in the sequence for a given node be the m bits stored in the memory of the first MA to visit that node. Every time that a MA visits a node, the m bits stored in the MA's memory at that time are appended to the sequence. Let r denote the number of complete trips around the ring completed by a MA in algorithm A. Given a unidirectional ring, the two MAs will complete the same number of rounds when $d = \frac{n}{2}$, otherwise one MA would pass the other and rendezvous would occur. After r rounds of algorithm A, the resulting sequence of bits for a given node will contain at least krm bits, where $k = 2$ is the number of MAs. The $2rm$ bits stored at a given node represent the histories, i.e., the interleaved sequence of states, of the MAs at that node. In a ring with n nodes where n is sufficiently large, i.e., $n >> 2rm$, two such constructed sequences of bits will be identical, i.e., two distinct nodes will have the same sequence of $2rm$ bits. If n is large enough, these two nodes will not be separated by a token.

Let $node_a$ and $node_b$ denote the two nodes with identical bit sequences, where $node_a$ precedes $node_b$ in the unidirectional ring. By the construction of the bit sequences, MA_1 is in the same state at $node_a$ as it is at $node_b$ for *each* of the r complete rounds of the algorithm. The same is true for MA_2. If all of the nodes from $node_a$ up to, but not including, $node_b$ are removed from the ring,the behaviour of MA_1 and MA_2 at $node_b$, and any other remaining node, is unchanged. This implies that the behaviour of the two MAs, under the given algorithm A, is the same in the original case where $d = \frac{n}{2}$ as in the case where nodes are deleted such that $d < \frac{n}{2}$. This contradicts the assumption that algorithm A will correctly stop without a rendezvous when $d = \frac{n}{2}$ and ensure a rendezvous otherwise.

Algorithm A is an arbitrary algorithm chosen from Γ, the collection of algorithms that correctly solve the rendezvous search problem for two MAs with m bits of memory each. The algorithm fails because $n >> 2rm$. We want to find the value of m, i.e., the amount of memory available to a MA, such that for a given n, $n \leq 2rm$ holds and thus algorithm A and any other algorithm arbitrarily chosen from Γ will succeed.

In r rounds, the two MAs will deposit a total of $2rm$ bits at each of the n nodes. These $2rm$ bit sequences represent 2^{2rm} different strings, so n must exceed $c_0 2^{2rm}$, where c_0 is a non-zero positive integer constant, if any of the strings are to be repeated at different nodes, i.e., as in $node_a$ and $node_b$ above.

Consider r, the number of rounds of the arbitrary algorithm A. We claim that $r \leq 2^{km}$ where $k = 2$. Suppose not. If $r > 2^{2m}$, then both MAs will eventually return to their respective starting nodes with the same bits in memory as when they started.

With $r \leq 2^{2m}$, then $c_0 2^{2rm} \leq c_0 2^{m2^{m+1}}$ and $n \geq c_0 2^{m2^{m+1}}$ will cause the algorithm A to fail. Thus each MA needs m bits of memory such that $n < c_0 2^{m2^{m+1}}$ in order to guarantee the correctness of algorithm A. This implies that $\log n < m2^{m+1}$ so that $\log \log n < 2m + \log m + 1$. Thus the correctness of an arbitrary algorithm in Γ is guaranteed if each MA has at least m memory bits where $m \in \Omega(\log \log n)$. ■

Together, Theorem 5 and Theorem 6 imply that $\Theta(\log \log n)$ memory bits are required to ensure the correctness of an algorithm for the rendezvous search problem with two identical MAs on an anonymous, synchronous n node ring.

6 Constant Memory

The preceding theorem indicates that if the MAs have $O(1)$ memory and do not know n, d, or the orientation of the ring, then the rendezvous search problem cannot be solved in the given model. We need to change the model if we want to solve the rendezvous search problem in such cases. Suppose we change the model so that, upon meeting a token, a MA can move that token to an adjacent node. We prove, in Theorem 7 below, that Algorithm 2 can use such movable tokens to solve the rendezvous search problem with memory $O(1)$.

Algorithm 2 (The Movable Tokens Algorithm)
/* Assume memory $O(1)$, and n, d, and orientation are unknown. */
1. Release the token.
2. Choose a direction and begin walking.
3. Upon finding a token, reverse direction.
4. Move the token one node in the new direction.
5. Continue walking in the new direction.
6. Repeat from step 3 until rendezvous occurs.

Theorem 7 *Consider two identical MAs with memory $O(1)$ at a distance d apart in an anonymous, synchronous, n node ring. The MAs do not know n, d, or the orientation of the ring. If, upon reaching a token, a MA can move the token to an adjacent node, then Algorithm 2 solves the rendezvous search problem in time $T \in O\left(d^2 \left(\frac{d^2}{n-2d} + n\right)\right)$. Thus, as d approaches 1, T approaches $O(n)$ and, as d approaches $\frac{n}{2}$, T approaches $O(n^4)$.*

7 Conclusion

After proving that identical stationary tokens cannot be used to break symmetry in the given model when $d = \frac{n}{2}$, we derive the lower and upper bounds on the time complexity of the rendezvous search problem when the mobile agents successfully use such tokens to break symmetry. We then present an algorithm that suggests there is a tradeoff between the mobile agents' memory size and the time complexity of the rendezvous search problem. We prove that this tradeoff is limited, however, by showing that $\Theta(\log \log n)$ memory enables the mobile agents to stop when $d = \frac{n}{2}$ and rendezvous otherwise. As a result, constructing a $O(1)$ memory algorithm that solves the rendezvous search problem requires a change in the model. Suppose that the identical tokens are no longer stationary, i.e., upon meeting a token, a MA can move that token to an adjacent node. We present a $O(1)$ memory algorithm that uses such tokens to break symmetry and solve the rendezvous problem.

Our discussion of lower and upper bounds on the time complexity of the rendezvous search problem contains an open problem. If the MAs know d but do not know the orientation of the ring, then the lower bound for the time complexity of the rendezvous search problem is $\frac{3d}{2}$ but the upper bound is $\frac{5d}{2}$. In future research, it would be interesting to study how changes in the model affect the time complexity of the rendezvous search problem. For example, one might study a network topology that differs from the ring or the case where each mobile agent has more than one token.

Acknowledgements

The authors wish to thank Paola Flocchini for many useful conversations. 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, Asymmetric Rendezvous on the Circle, Dynamics and Control, 10, pp. 33-45, 2000.

[2] S. Alpern, Rendezvous Search: A Personal Perspective, LSE Research Report, CDAM-2000-05, London School of Economics, 2000.

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

[4] S. Alpern and J.V. Howard, Alternating Search at Two Locations, LSE OR Working Paper, 99.30, 1999.

[5] S. Alpern and D. Reyniers, The Rendezvous and Coordinated Search Problems, Proceedings of the 33rd Conference on Decision and Control, Lake Buena Vista, FL, December 1994.

[6] E.J. Anderson and S. Essegaier, Rendezvous Search on the Line with Indistinguishable Players, SIAM Journal of Control and Optimization, 33, pp. 1637-1642, 1995.

[7] E.J. Anderson and S. Fekete, Two-dimensional Rendezvous Search, Operations Research, 49, pp.107-188, 2001.

[8] E.J. Anderson and R.R.Weber, The Rendezvous Problem on Discrete Locations, Journal of Applied Probability, 28, pp. 839-851, 1990.

[9] T.M. Apostol, Introduction to Analytical Number Theory, Springer Verlag, 1997.

[10] V. Baston and S. Gal, Rendezvous on the Line When the Players' Initial Distance is Given by an Unknown Probability Distribution, SIAM Journal of Control and Optimization, 36, No.6, pp. 1880-1889, 1998.

[11] 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.

[12] E. Chester and R. Tutuncu, Rendezvous Search on the Labeled Line, *Old title*: Rendezvous Search on Finite Domains, Preprint, Department of Mathematical Sciences, Carnegie Mellon University, 2001 (revised 2002).

[13] T. Cormen, C. Leiserson, and R. Rivest, Introduction to Algorithms, MIT Press, Cambridge, Massachusetts, 1992.

[14] P. Flocchini, B. Mans, and N. Santoro, Sense of Direction in Distributed Computing, Theoretical Computer Science, 291, No. 1, pp. 29-53,2003.

[15] J. Howard, Rendezvous Search on the Interval and Circle, Operations Research, 47, No.4, pp. 550-558, 1999.

[16] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk, The Rendezvous Search Problem with More Than Two Mobile Agents, preprint, 2002.

[17] W.S. Lim, Rendezvous Search on the Line with Three Players, LSE CDAM Research Report, LSE-MDS-81, 1997.

[18] W.S. Lim and S. Alpern, Minimax Rendezvous Search on the Line, Journal of Control and Optimization, 34, pp.1650-1665, 1996.

[19] W.S. Lim and A. Beck and S. Alpern, Rendezvous Search on the Line with More Than Two Players, Operations Research, 45, pp.357-364, 1997.

[20] M. Pikounis and L.C. Thomas, Many Player Rendezvous Search: Stick Together or Split and Meet?, Working Paper 98/7, University of Edinburgh, Management School, 1998.

[21] T. C. Schelling, The Strategy of Conflict, Harvard University Press, Cambridge, MA, 1960.

[22] L.C. Thomas amd P.B. Hulme, Searching for Targets Who Want to be Found, Journal of the Operations Research Society, 48, Issue 1, pp. 44-50, 1997.

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