

Rendezvous Search with Mobile Agents: How good is the Advice?

E. Kranakis
Carleton University
School of Computer Science
Ottawa, Ontario, K1S 5B6, Canada

May 2, 2012

- Origins of Rendezvous
- Focusing on the Domain: Searching with...
 - Games
 - Uncertainty
 - Waiting
- Focusing on the Agent:
 - Computing Model
 - Possibility/Impossibility
 - Deterministic/Probabilistic
 - Synchronous/Asynchronous
- Rendezvous in...
 - Ring
 - Torus

Rendezvous Problem

- Rendezvous Problem:

Two or more entities move along a domain of communication channels until they meet. Channels at intersections are chosen according to a decision process.

- Entities: people, particles, agents, etc
- Domain: mall, airport, square, etc
- Channels: streets, sidewalks, edges, etc
- Process: deterministic, random, with(out) advice, etc

Which algorithm/strategy should they choose in order to meet as soon as possible?

Origins: Pure Form

- The “pure form” of randomized rendezvous problem has its origins to George Pólya¹ who enjoyed taking walks in the forest while thinking about mathematics.

“One day while out on his walk he encountered one of his students strolling with his fiancée. Somewhat later their paths crossed again and even later he encountered them once again”

...and like any mathematician would do...

“this caused him to wonder how likely it was that walking randomly through paths in the woods, one would encounter others similarly engaged”

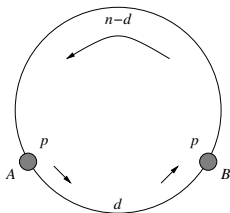
...thus leading to his seminal work.²

¹Alexanderson. The random walks of George Pólya. MAA, 2000.

²Pólya. Über eine aufgabe betreffend die irrfahrt im strassennetz. Math. Annalen, 84:149–160, 1921.

Convergence of Two Random Walks

- Polya's Rendezvous can be interpreted as follows:
 - 1 **Repeat until** other agent is present:
 - 2 **If** heads move right **else** move left



- **Theorem** Consider an n node ring. Two agents with $O(1)$ memory, starting at even distance $d \leq n/2$ can rendezvous in expected $\frac{d}{2}(n-d)$ steps.
- Can be analyzed using the martingale approach.

Convergence of Two Random Walks (1/4)

- Let E_d be the expected time for two agents starting at an (even) distance d on a ring of (even) size n to rendezvous using the above algorithm.
- It is easy to see that $E_0 = 0$, and

$$E_{n/2} = 1 + (1/2)E_{n/2} + (1/2)E_{n/2-2}.$$

- The latter equation gives rise to the recurrence

$$E_{n/2} = 2 + E_{n/2-2}. \quad (1)$$

Convergence of Two Random Walks (2/4)

- One of the following three cases may occur.
 - ① The two mobile agents make a single step and either move in the same direction with probability $1/2$, or
 - ② in opposite direction either towards each other with probability $1/4$, or
 - ③ away from each other with probability $1/4$.
- From this we derive the identity

$$E_d = 1 + (1/2)E_d + (1/4)E_{d-2} + (1/4)E_{d+2}, \quad (2)$$

for $d = 2, 4, \dots, n/2 - 2$.

- Substituting $d + 2$ for d in Identity 2 and solving the resulting equation in terms of E_d we derive that for $d \geq 4$,

$$E_d = 2E_{d-2} - E_{d-4} - 4. \quad (3)$$

- The initial condition $E_0 = 0$ and Identity 3 yield $E_4 = 2E_2 - 4$.

Convergence of Two Random Walks (3/4)

- More generally, we can prove the following identity for $2d \leq n/2$,

$$E_{2d} = dE_2 - 2d(d-1). \quad (4)$$

We prove by induction that there are sequences a_d, b_d such that

$$E_{2d} = a_d E_2 - 4b_d.$$

- Indeed,

$$\begin{aligned} E_{2d} &= 2E_{2d-2} - E_{2d-4} - 4 \\ &= 2(a_{d-1}E_2 - 4b_{d-1}) - (a_{d-2}E_2 - 4b_{d-2}) - 4 \\ &= (2a_{d-1} - a_{d-2})E_2 - 4(2b_{d-1} - b_{d-2} + 1), \end{aligned}$$

giving the recurrences $a_d = 2a_{d-1} - a_{d-2}$ and $b_d = 2b_{d-1} - b_{d-2} + 1$ with initial conditions $a_0 = b_0 = 0, a_1 = 1, b_1 = 0$.

Convergence of Two Random Walks (4/4)

- Solving the recurrences we obtain easily that $a_d = d$ and $b_d = -\frac{1}{2}d + \frac{1}{2}d^2$, which proves Identity 4.
- To derive a formula for E_{2d} , it remains to compute E_2 . Identity 4 yields the values

$$\begin{aligned}E_{n/2} &= \frac{n}{4}E_2 - 2\frac{n}{4}\left(\frac{n}{4} - 1\right) \\E_{n/2-2} &= \left(\frac{n}{4} - 1\right)E_2 - 2\left(\frac{n}{4} - 1\right)\left(\frac{n}{4} - 2\right),\end{aligned}$$

which when substituted into Identity 1 shows that $E_2 = n - 2$.

- Finally, substituting this last value into Identity 4 we derive

$$E_{2d} = d(n - 2d). \quad (5)$$

- Above algorithm translates into a finite automaton with a constant number of states

Open Problem(s)

- Is there an analogue of this result for k agents?
 - Is there a closed form formula?
 - There are different ways to accomplish rendezvous!

Rendezvous is Useful in P2P Computing

- JXTA 2:
 - introduces the concept of a rendezvous super-peer network, greatly improving scalability by reducing propagation traffic,
 - implements the shared resource distributed index (SRDI) within the rendezvous super-peer network, creating a loosely consistent, fault-resilient, distributed hash table.³
- iChat:
 - eliminates the need for using servers in the cloud to communicate.

³Gong, JXTA: A network programming environment, Internet Computing, IEEE, 5(3), 88–95, 2001.

Origins: Game Theoretic Form

- The Nobel laureate T. C. Schelling is generally credited with introducing the “game theoretic form” of the rendezvous problem in his technical report⁴

“if chess is the standard example of zero-sum game, charades⁵ may typify the game of pure coordination; if pursuit epitomizes the zero-sum game, rendezvous may do the same for the coordination game”

concerning the strategy of conflict.

⁴Schelling, Prospectus for Reorientation of Game Theory, Rand Corporation, 1958, P-1491, 17 Sep 1958 (page 4).

⁵An acting game in which one player acts out a word or phrase, often by miming similar-sounding words, and the other players guess the word or phrase.

Rendezvous Search Games

- A *search space* is usually represented by a graph $G = (V, E)$.
 - A *searcher* starts from a given vertex.
 - A *hider* (mobile or otherwise) chooses its hiding point independently of the searcher and occupies vertices.
- Neither searcher nor hider have any a priori knowledge of the movement or location of the other until they are some critical distance r apart, also called the *discovery radius*.
- Search problems are two-person, zero-sum games with associated strategies available to the searcher and receiver.⁶
- Cost of game specified by $c : \mathcal{S} \times \mathcal{H} \rightarrow \mathcal{R} : (S, H) \rightarrow c(S, H)$ representing the *loss* (or effort) by searcher S when using trajectory S while the hider uses trajectory H .
- Since the game is zero-sum, $c(S, H)$ also represents the *gain* of the hider.

⁶S. Alpern and S. Gal. The theory of search games and rendezvous. Kluwer Academic Publishers, 2003.

Example

- Consider the *uniform* mixed hiding strategy: hider is immobile but chooses the hiding vertex H randomly with the uniform distribution among the vertices of a graph with n vertices.
- Assume discovery radius satisfies $r = 0$.
- Searcher discovers at most one new vertex per time unit we have $\Pr[T \leq t] \leq \min\{1, \frac{t}{n}\}$, where T is the capture time.
- It follows that the expected capture time satisfies

$$E[T] = \sum_t \Pr[T > t] \geq \sum_t \max\left\{0, 1 - \frac{t}{n}\right\} = \frac{n+1}{2}.$$

- Is this tight?
- In a mixed search strategy searcher selects a permutation $\sigma \in S_n$ at random with the uniform distribution.

Optimality of Example in K_n

- For a given σ, H the capture time $c(\sigma, H)$ is equal to the smallest t such that $\sigma(t) = H$.
- Given t, H , there are exactly $(n-1)!$ permutations $\sigma \in S_n$ such that $\sigma(t) = H$.

$$\sum_{H=1}^n \sum_{\sigma \in S_n} c(\sigma, H) = \sum_{H=1}^n \sum_{t=1}^n \sum_{\substack{\sigma \in S_n \\ \sigma(t)=H}} t = \frac{n(n+1)}{2} \cdot (n-1)!$$

- It follows that the expected capture time is

$$E[T] = \frac{1}{n!} \cdot \frac{1}{n} \cdot \sum_{H=1}^n \sum_{\sigma \in S_n} c(S, H) = \frac{n+1}{2}.$$

Problem much more difficult in arbitrary graphs!

Searching with Uncertainty

- A mobile agent in a network wants to locate an *item* available at one node in the network.
- Each of the nodes of the network maintains a database indicating the first edge on a shortest path to the items of interest.
- Uncertainty arises from the fact that inaccuracies may occur in the database for reasons such as the movement of items, out-of-date information, etc.
- Item occupies an unknown location but information about its whereabouts can be obtained by querying the nodes of the network.
- How does the agent find the requested item while traversing the minimum number of edges?⁷

⁷Kranakis, Krizanc, Searching with Uncertainty. In SIROCCO 1999, 194-203, Carleton Scientific, 1999.

Algorithm in the Ring

- Consider a ring of n nodes.
- **Idea:** Consult visited nodes and take majority of advice.
- Leads directly to the following algorithm parameterized by k .
Algorithm: Search in a given direction for k steps.

1. Let dir be the direction given by the initial node.
2. **for** $j = 1$ **to** k or until item is found **do**
3. move in direction dir .
4. **if** majority of processors agree with dir ,
 continue until item is found.
5. **else** reverse direction;
 continue until item is found.

Analysis (1/3)

- Search proceeds for k steps and direction is chosen according to principle of maximum likelihood:

direction agreed by the majority of nodes along this path of length k .

- Probability direction is correct is

$$p_k := \sum_{i \geq \lceil k/2 \rceil} \binom{k}{i} p^i (1-p)^{k-i}$$

- Let X be the random variable that counts the number of steps traversed by the algorithm.

Analysis (2/3)

- Let d be the distance of the initial node to the node containing the desired item.
- If decision of the direction made by the algorithm is correct then $X = d$ or $X = d + 2k$ depending on whether or not the algorithm reversed direction after k steps.
- Similarly, if the decision made was incorrect then $X = n - d$ or $X = n - d + 2k$ depending on whether or not the algorithm reversed direction after k steps.
- It follows that with probability at least p_k the number of steps traversed by the algorithm does not exceed $d + 2k$.

Analysis (3/3)

- Observe that $p_k = \Pr[S_k \geq k/2]$ and using Chernoff-Hoeffding bounds (setting $\mu = kp$ and $\delta = 1 - \frac{1}{2p}$) we derive that

$$\begin{aligned} p_k &= \Pr[S_k \geq (1 - \delta)\mu] \\ &= 1 - \Pr[S_k < (1 - \delta)\mu] \\ &> 1 - e^{-\mu\delta^2/2}, \end{aligned}$$

where S_k is the sum of k independent and identically distributed random variables X_1, \dots, X_k such that $\Pr[X_i = 1] = p$, for all i .

- Choosing $k \geq c \frac{2p}{(p-1/2)^2} \ln n$, the probability that the algorithm requires less than $d + 2k$ steps is $\geq 1 - n^{-c}$ for $c > 0$.

More on Searching with Advice

- Extensions of the model in other settings are quite interesting
 - In complete graphs.⁸
 - In networks with liars.⁹
 - Random walks with advice.¹⁰
 - Using clocks to alternate between phases of ignoring and following advice.¹¹
 - ... and there is a lot more!
- We'll return to the theme of advice!

⁸Kirousis, Kranakis, Krizanc, Stamatiou, Locating information with uncertainty in fully interconnected networks, DC 283–296, 2000.

⁹Hanusse, Kranakis, Krizanc, Searching with mobile agents in networks with liars, DAM 137(1): 69–85, 2004.

¹⁰Hanusse, Kavvadias, Kranakis, Krizanc, Memoryless search algorithms in a network with faulty advice, TCS 402(2): 190–198, 2008.

¹¹Hanusse, Ilcinkas, Kosowski, Nisse, Locating a target with an agent guided by unreliable local advice: how to beat the random walk when you have a clock?, PODC, 355–364, 2010.

Searching with Advice

- A *mobile agent* is searching for an item stored at a node t of a network, without prior knowledge of its exact location.
- Each node of the network has a database that will answer queries of the form

"How do I find t ?"

by responding with the first edge on a shortest path to t .

- Some nodes, called *liars*, give bad advice.
- How do we search for the item?

Search Algorithm with given parameter $q > 1/2$

- 1 Agent arrives at a node of degree Δ and if it discovers the token it halts. Otherwise, it asks the node for advice.
- 2 Node responds by pointing to one of the edges incident to it.
- 3 Agent then flips a biased coin and with probability q it follows the advice.
 - That is, it moves to the adjacent node which is the other endpoint of the edge.
 - If it decides not to follow the advice (an event with probability $1 - q$), it selects uniformly another edge among the remaining $\Delta - 1$ incident edges and follows that edge.
- 4 Above steps repeated at the new node until the token is found.

Searching with Advice

- **Theorem** Let G be any network of maximal degree Δ with k liars in which the distance between the initial node s and the token t is d . Then the expected number of steps of a mobile agent to reach t is less than $d \left(1 + \frac{6}{r-1}\right) + \frac{6r^{k+3}}{(r-1)^3}$ where $r = (\Delta - 1) \frac{q}{1-q}$.
- **Theorem** Assume that in K_n the number of liars is $k = cn$, where c is a constant, $0 < c < 1$. Then the expected number of steps to reach the token is $\frac{1}{(q-qc)(1-q)} + O\left(\frac{1}{n}\right)$.
- Also possible to analyze the torus in this setting.

Open Problems

- Very little known on
 - Tight bounds for all types of braphs.
 - Analysis of preferential attachment models.

Domain/Behavior/Advice

- Domain:
 - modelled by a graph
 - processors are nodes
 - communication links are edges
 - sense of direction (consistent edge labelling clockwise/counterclockwise)
 - etc
- Behavior:
 - identical (anonymous)
 - synchronous/Asynchronous
 - etc
- Advice:
 - token

Emphasizing the Agent!

What is a Mobile Agent?

- (American Heritage Dictionary): An agent is “one that has the power or authority to act or represent another” or “the means by which something is done or caused”.
- (Shoham 1997):
A software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes
- (Bradsaw 1997):
An agent that inhabits an environment with other agents and processes is expected to be able to communicate and cooperate with them, and perhaps move from place to place in doing so

Mobile Agents Model: Advice as a Token

- **Deterministic** finite automata $A = \langle S, \delta \rangle$:
 - 1 S set of states including start state and halt state
 - 2 δ transition function $\delta : S \times P \rightarrow S \times M$
 - $P = \{\text{present, notpresent}\}$ (presence of token at a node)
 - $M = \{+1, 0, -1\}$ (movement of token towards)
- **Probabilistic** finite automata $A = \langle S, \delta \rangle$:
 - 1 S set of states including start state and halt state
 - 2 δ transition function $\delta : S \times P \rightarrow S \times M$
 - $P = \{\text{present, notpresent}\}$ (presence of token at a node)
 - $M = \{+1, 0, -1\}$ (movement of token towards)
 - 3 $C = \{H, T\}$ result of fair coin toss

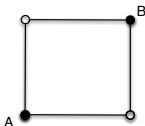
Mobile agents are essentially mobile (communicating) processes moving on vertices of a network.

Rendezvous

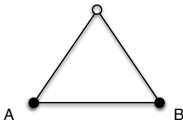
- Two agents are said to rendezvous if after some finite time they occupy the same node at the same time.
- Interested in
 - measuring:
 - How long it takes (number of synchronous time steps)
 - The memory the agents require (proportional to $\log(|S|)$, where S is the state space)
 - testing
 - Possibility/Impossibility of rendezvous.
- Rendezvous may not be possible due to parity considerations. We (sometimes) assume as part of the model that
 - agents can meet at an edge.

Is Rendezvous always Possible?

- Impossible to rendezvous for two synchronous, identical MAs even with tokens



- Impossible to rendezvous for two synchronous, identical MAs without tokens.



However, rendezvous is guaranteed if each MA has an identical stationary token that it leaves at its respective starting node! Why?

- Ring
 - Deterministic Oriented
 - k Mobile Agents
 - Randomized
- Torus
 - Deterministic Oriented
 - Deterministic Unoriented

$k \geq 2$ Mobile Agents in a Ring of Size n

- Several trade-offs possible for k mobile agents¹²

Memory	Time
$O(k \log n)$	$O(n)$
$O(\log n)$	$O(kn)$
$O(k \log \log n)$	$O(\frac{n \log n}{\log \log n})$
$O(\log n)$	$O(n)$
$O(\log k)$	$O(n)$
$O(\log k)$	$O(n \log k)$

- Lets look at $O(k \log \log n)$ memory bound when $k = 2$.
- p_1, \dots, p_r are the first r prime numbers s.t. $\prod_{i=1}^r p_i > n$.

¹²Flocchini, Kranakis, Krizanc, Santoro, Sawchuk, Multiple Mobile Agent Rendezvous in a Ring. LATIN 2004, pp. 599-608.

Algorithm for $k = 2$

1. Release the token. /*Let d be the inter-token distance*/
2. Set $m = p_1$.
3. Choose a direction and begin travelling around the ring.
4. Count the number of steps, mod m , to the first token, δ_1 , and continue walking.
5. Count the number of steps, mod 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 $m = p_r$, stop. /* Rendezvous is not possible. */
 If $m < p_r$, set $m = p_{i+1}$ and repeat from step 4.
8. If $\delta_1 \bmod m < \delta_2 \bmod m$, continue travelling in same direction.
9. Else, reverse direction and continue travelling.
/* If step 8 or 9 is executed, rendezvous occurs
in another $\frac{d}{2}$ steps.*/

Algorithm for $k = 2$: Analysis

- By Chinese Remainder Theorem if $d \equiv (n - d) \pmod{p_i}$ for all $i = 1, \dots, r$, then $d \equiv (n - d) \pmod{\prod_{i=1}^r p_i}$.
- Algorithm checks $d \equiv (n - d) \pmod{p_i}$, for each 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 $d \not\equiv (n - d) \pmod{p_i}$ for some p_i , however, then $d < \frac{n}{2}$.
- However,

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

- Therefore $r \in O\left(\frac{\log n}{\log \log n}\right)$.
- The time complexity of the algorithm, $O(rn)$, is $O\left(\frac{n \log n}{\log \log n}\right)$.

Rendezvous with Failing Tokens

- Several trade-offs possible for rendezvous.¹³
- n is the size of the ring and k is the number of agents.

<i>Tokens Fail</i>	<i>Knowledge</i>	<i>Time</i>	<i>Memory</i>
Never	n or k	$O(n)$ $O(n \log k)$	$O(\log n)$ $O(\log k)$
Upon release	n k	$O(n)$ $O(k n)$	$O(k \log n)$ $O(k \log n)$
Anytime	n k	$O(k n)$ $O(k^2 n)$	$O(k \log n)$ $O(k \log n)$

- Lets look at the case of failure upon release with k known and assuming that $\gcd(k', n) = 1$, for all $k' \leq k$

¹³Flocchini, Kranakis, Krizanc, Luccio, Santoro, Sawchuk, Mobile Agents Rendezvous When Tokens Fail. SIROCCO 2004.

Algorithm

- 1 Release the token at the starting node.
- 2 Choose a direction and start walking.
- 3 Compute the sequence of $3k$ inter-token distances i.e., $S = d_1, d_2, \dots, d_{3k}$.
- 4 Let S^R be the reverse of S .
- 5 Find the shortest aperiodic subsequence Q that starts with the first element of S^R and is repeated such that $S^R = Q^\gamma + d_\gamma, \dots, d_1$ where $\gamma < |Q|$.
- 6 Move to MeetingPoint(Q)

Algorithm: Analysis (1/2)

- Let $m = k - f$ be the number of tokens that do not fail.
- Let $A = \delta_1, \dots, \delta_m$ be the sequence of the m inter-token distances that exist after the f tokens have failed.
- Let $S(a)$ be the sequence of $3k$ inter-token distances calculated by a given agent a in step 3 of the Algorithm.
- Let $S^R(a)$ be the reverse of $S(a)$ and A^R be the reverse of A .
- For all agents, the $3k$ inter-token distances are of the form

$$S^R(a) = (A^R)^\rho + d_\gamma, \dots, d_1 = (\delta_m, \dots, \delta_1)^\rho + d_\gamma, \dots, d_1. \quad (6)$$

where $(A^R)^\rho$ is the concatenation of ρ copies of the aperiodic subsequence A^R , $+$ is the concatenation operator, and d_γ, \dots, d_1 is a subsequence such that $\gamma < m$.

- Thus, there exists at least one aperiodic subsequence, namely A^R , that satisfies equation 6. Note that A^R is aperiodic

Algorithm: Analysis (2/2)

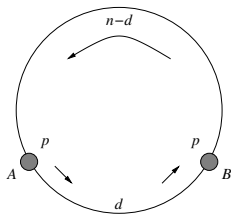
- If A^R is the shortest subsequence that satisfies Equation 6, the agents discover A^R in step 5 of Algorithm. Otherwise, the agents discover a shorter aperiodic subsequence, Q , that satisfies Equation 6.
- The subsequence discovered in step 5 of Algorithm is unique. This implies that all the agents identify the same rendezvous node in the remaining steps of Algorithm and rendezvous occurs.
- Calculating S , the sequence of $3k$ inter-token distances requires $O(k \log n)$ memory and requires $O(kn)$ time. Identifying the appropriate subsequence in step 5, determining the rendezvous node, and walking to the rendezvous node takes $O(kn)$ time, so the overall time requirement is $O(kn)$.

Open Problems

- Tight bounds not known.
- Effect of multiple tokens not well understood.

Flickering Tokens

- Each agent has a *flickering token* which it releases at its starting position.¹⁴
- When A (respectively, B) is traversing the node occupied by its token it will see the token with probability p and will not see it with probability $1 - p$.



¹⁴Kirousis, Kranakis, Krizanc, Rendezvous with Flickering Tokens, 2010.

Flickering Tokens Algorithm

- The mobile agents run the following algorithm:

Double Token Algorithm (DTA):

1. Leave token at starting position;
2. Repeat until rendezvous;
3. Choose a direction and walk on the nodes of the ring;
4. **If** you see token **then** change direction;
5. **else** stay in same direction.

Flickering Tokens

- **Theorem:** The expected time until rendezvous for DTA is at most

$$\frac{n}{4p(1-p)} + \frac{n}{4} \quad \text{if the initial MA distance is } d = n/2,$$
$$\frac{1+p(1-p)}{p(1-p)}(n-d) \quad \text{if the initial MA distance is } d < n/2.$$

Open Problems

- Tighten the constants.
- Is there any interesting Time/Memory tradeoff?
- Is there any advantage in introducing “Waiting Search” for flickering tokens on a ring?
- Similar tradeoffs for other networks?

Randomized Rendezvous on a Ring

- Polynomial time rendezvous is possible on any network using the theory of random walks.
- Therefore two agents on an n node ring can rendezvous in $O(n^2)$ time with a random walk.
- The number of states required is $O(1)$ - essentially memoryless
- Can we do better using more memory?

Algorithm	Time	Memory	RandBits
RandWalk	$O(n^2)$	$O(1)$	$O(n^2)$
RandWalk with Tokens	$O(n)$	$O(1)$	$O(1)$
Coin Half-Tour	$O(n)$	$O(\log n)$	$O(1)$
Approximate Counting	$O(n)$	$O(\log \log n)$	$O(n)$

Random Walk with Tokens (1/2)

- How can tokens and randomization be combined?
- Two MAs have memory $O(1)$ and know neither n nor d .
- Algorithm: **Random Walk with Tokens**
 - 1 Release the token.
 - 2 Set count = 0.
 - 3 Choose a direction and walk until a token is reached.
 - 4 At the token, set count = count + 1.
 - 5 **If** count mod 2 = 0, change direction with probability $0 \leq p \leq 1$.
 - 6 Otherwise, maintain the same direction.
 - 7 Walk to the next token.
 - 8 **Repeat** from step 4 until rendezvous occurs.

Random Walk with Tokens (2/2)

- The expected number of rounds is

$$\sum_{i=1}^{\infty} (2p(1-p))(1-2p(1-p))^{i-1} i = \frac{1}{(2p(1-p))}.$$

All but the final round take time n .

- Once the MAs are travelling in opposite directions on the ring, the expected time to rendezvous is $\frac{n}{2}$. Thus the total expected time to rendezvous is

$$n \left(\frac{1}{2p(1-p)} - 1 \right) + \frac{n}{2} = \frac{n(1-p(1-p))}{2p(1-p)}$$

which is $O(n)$. The expected time is minimized when $p = \frac{1}{2}$.

- Theorem** Two agents with $O(1)$ memory each, starting at an even distance $d \leq n/2$ on an even n node synchronous and unoriented ring and carrying one stationary token each can rendezvous in $O(n)$ expected time.

Randomized Rendezvous

Consider an oriented ring of size n and two mobile agents A, B at distance d from each other (see Figure 1). Further assume that $d \leq n - d$.

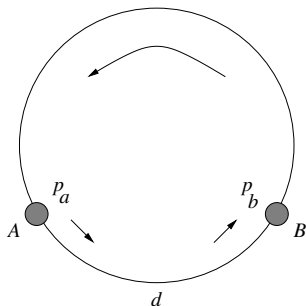


Figure: Two mobile agents at distance d in an oriented ring with n nodes.

If you Know the Distance

Assume the two MAs know the distance d . The precise algorithm for A is as follows (a similar algorithm is executed for B).

Algorithm for mobile agent A :

1. Repeat until rendezvous.
2. A with probability p_a takes $\lceil d/2 \rceil$ steps to the right; if you do not meet other MA go back to original position;
3. A with probability $1 - p_a$ takes $\lceil d/2 \rceil$ steps to the left; if you do not meet other MA go back to original position;

Let X be the random variable that counts the number of trials until success. This is a geometric distribution with probability of success $p_a(1 - p_b)$. It follows that the algorithm requires $O(\log d)$ memory, $O\left(\frac{1}{p(1-p)}\right)$ random bits, and time $O\left(\frac{d}{p(1-p)}\right)$.

Incremental Step Algorithm

Assume the two MAs do not know the distance d .

Algorithm for mobile agent A :

1. For $r = 1$ until rendezvous do;
2. A with probability p_a takes r steps to the right;
if you do not meet other MA go back to original position;
3. A with probability $1 - p_a$ takes r steps to the left;
if you do not meet other MA go back to original position;

Let X be the random variable that counts the number of steps until rendezvous. At the k -th trial it takes time $2k$ for the mobile agents to explore and return to their original position, in which case the total running time is 2 multiplied by

$$1 + 2 + \dots + \left(\lceil d/2 \rceil + \left\lceil \frac{1}{p_a(1-p_b)} \right\rceil \right).$$

Expected # of random bits is $O(n)$, the expected running time is $O\left(\left(d + \frac{1}{p_a(1-p_b)}\right)^2\right)$, and the expected memory $O(\log n)$.

Coin Half Tour Algorithm (Alpern)

Repeat the following until rendezvous occurs:

- ① Flip a fair coin;
 - ① if H then $D = \textit{left}$,
 - ② else $D = \textit{right}$.
 - ② Proceed in direction D for $n/2$ steps.
- Probability of success on each phase equals $1/2$ implies expected time to rendezvous is $O(n)$
 - Requires an automaton with $n/2$ states, i.e., $\Theta(\log n)$ memory

Is this optimal?

Can we achieve linear time rendezvous with less memory?

Randomized Rendezvous Algorithm on the Ring

Theorem: $\Theta(\log \log n)$ bits of memory are necessary and sufficient to achieve rendezvous in linear time on an n node ring.

- Repeat the following until rendezvous occurs:¹⁵
 - 1 Flip a fair coin;
 - 1 if H then $D = \text{left}$,
 - 2 else $D = \text{right}$.
 - 2 Proceed in direction D flipping coin at each step until t H s in a row observed.
- Essentially Coin Half Tour where half tour is replaced by a walk of expected length 2^t .
- Can be implemented with $2t$ states.

Let m iterations of repeat loop in algorithm be called a *round*.

¹⁵Kranakis, Krizanc, Morin, Randomized Rendez-Vous with Limited Memory. ACM TALG, 2011.

Martingales

- Q_1, Q_2, \dots is a martingale wrt X_1, X_2, \dots if for all i ,
 $E[|Q_i|] < \infty$ and $Q_i = E[Q_{i+1} | X_1, \dots, X_i]$
- If Q_1, Q_2, \dots is a martingale with respect to X_1, X_2, \dots and T is a stopping time for X_1, X_2, \dots then $E[Q_T] = E[Q_1]$ provided that at least one of the following holds:
 - 1 Q_i is uniformly bounded for all $i \leq T$,
 - 2 T is bounded, or
 - 3 $E[T] < \infty$ and $\exists M < \infty$ s.t. $E[|Q_{i+1} - Q_i| | X_1, \dots, X_i] < M$
- For example,

$$Q_i = \left(\sum_{j=1}^i (X_j - E[X]) \right)^2 - i \cdot \text{Var}(X)$$

is a martingale with respect to $X = X_1, X_2, \dots$ (i.i.d.s), and

$$\text{var} \left(\sum_{i=1}^T X_i \right) = E[T] \text{var}(X)$$

Upper Bound Analysis

- We show, with constant probability, distance travelled by an agent in a round is at least $\beta 2^t \sqrt{m}$ for some constant $\beta > 0$.
- Set $m = \left\lceil \frac{n^2}{\beta^2 2^{2t}} \right\rceil \leq 1 + \frac{n^2}{\beta^2 2^{2t}}$. For any given round, the probability the agents will rendezvous is constant.
- Using the above plus the observation that the rounds are independent and the sum of their lengths form a martingale

$$Q_i = \left(\sum_{j=1}^i (X_j - E[X]) \right)^2 - i \cdot \text{Var}(X)$$

with $X = X_1, X_2, \dots$ i.i.d., for which we can take as a stopping time the first round in which the agents rendezvous we get:

- **Theorem:** Algorithm achieves rendezvous on an n -node ring in expected time $O(n^2/2^t + 2^t)$ and uses at most $2t$ states.

Lower Bound Analysis

- A finite automata is *well-behaved* if its corresponding graph is strongly-connected. For well-behaved automata we define a *round* as the steps between consecutive visits to the start state. A well-behaved automata is *unbiased* if the expected distance it travels in a round is O .
- For any t state automata that achieves rendezvous in expected R steps there exists a well-behaved unbiased automata with $2t$ states that achieves rendezvous in less than expected $2R$ steps.
- The expected rendezvous time for any t -state well-behaved unbiased automata is $\Omega(n^2/2^t)$. From the above we conclude:
- **Theorem:** Any $t/2$ state rendezvous algorithm has expected rendezvous time $\Omega(n^2/2^t)$.

Open Questions

- Case of multiple agents in the ring.
- Other topologies.
- Random walks with advice.

Some Subtle Observations

- Deterministic rendezvous may not have a solution, e.g., symmetric cases
- Deterministic rendezvous has a solution when n is odd.

Question

Can the agents determine whether or not the rendezvous problem has a solution?

Token/Memory/Time Tradeoffs

- \mathcal{RV} : *rendezvous without detection*
Agents know that the rendezvous problem has a solution either for the given system configuration or regardless of the system configuration and they just want to accomplish rendezvous at a node of the ring in, say, minimum number of steps
- \mathcal{RVD} : *rendezvous with detection*
We are also interested in the *halting problem* for rendezvous. I.e., an algorithm that detects feasibility of a solution for all starting positions after a *finite number of steps* (usually dependent either on their distance or the size of the network).

Algorithm for Rendezvous with Detection (RVD)

Algorithm Two-Tokens.

- ① Drop 1st token at your home base and 2nd token to node located to the right.
- ② **repeat**
- ③ Travel right and move every second token you meet one position to the right.
- ④ **until** agent detects two tokens on top of each other.
- ⑤ **if** two tokens are found on top of each other go around and check if other two tokens are also on top of each other.
- ⑥ **if** yes then rendezvous is not possible **else** agent waits at last position.
- ⑦ **endif**
- ⑧ **endif**

Rendezvous with Detection (\mathcal{RVD})

- **Theorem:** \mathcal{RVD} is solvable in a unidirectional ring for two mobile agents with constant memory and two indistinguishable tokens each, in time $O(n^2)$.
- **Theorem:** Neither the rendezvous problem (\mathcal{RV}) nor rendezvous with detection (\mathcal{RVD}) is solvable for two identical mobile agents having constant memory and one token each in a unidirectional ring.
- **Theorem:** Rendezvous with detection (\mathcal{RVD}) is not solvable for two identical mobile agents having constant memory and one token each in a bidirectional ring.
- **Theorem:** The rendezvous problem (\mathcal{RV}) for two mobile agents having constant memory and two tokens each require $\Omega(n^2)$ time in a bidirectional ring of size n .

Rendezvous with Detection

- Time bounds for two mobile agents with constant memory to detect if rendezvous is possible (\mathcal{RVD}) and to rendezvous when input is asymmetric (\mathcal{RV}) on an n node synchronous uni-, bi-directional ring with one or two tokens.¹⁶

Conditions		Time Required for	
# Tokens	# Directions	\mathcal{RVD}	\mathcal{RV}
1	1	∞	∞
1	2	∞	$\Theta(n^2)$
2	1	$\Theta(n^2)$	$\Theta(n^2)$
2	2	$\Theta(n^2)$	$\Theta(n^2)$

¹⁶Cyzowicz, Dobrev, Kranakis, Krizanc, The Power of Tokens: Rendezvous and Symmetry Detection for two Mobile Agents in a Ring. SOFSEM 2008.

Multiple Tokens (1/2)

- **Algorithm**

- 1 Just go around and every time increase the base by t ;
- 2 Skip first token of the base, then walk until the second token is found, pick it up and keep picking up until you have $t - 1$ tokens in hand¹⁷, and then just lay them down one after another.
- 3 Stop the whole thing and verify if you find the next base while laying the tokens of your base.


- **Theorem:** The rendezvous problem (\mathcal{RV}) for two mobile agents having constant memory and t tokens each requires $\Omega(n^2/t)$ time in an unidirectional ring of size n . Moreover, there is an algorithm achieving this bound.

¹⁷if you cannot count to t , just have the tokens next to each other, the first empty space means end-of-base

Multiple Tokens (2/2)

Consider a synchronous ring with n nodes and two mobile agents located at two nodes of a bidirectional ring.¹⁸

- **Theorem:** Rendezvous with detection (\mathcal{RVD}) is solvable for two mobile agents having $t \geq 3$ tokens and $O(\log t)$ bits of memory each in time $O(mn)$, where m is the smallest integer such that $\binom{m-1}{t-2} \geq n - 1$.
- **Theorem:** Rendezvous with detection (\mathcal{RVD}) is solvable for two mobile agents having $t > 2$ tokens and memory $O(\log t)$ each, in time $O(n^{\frac{t-1}{t-2}} t)$ in a bidirectional ring. Moreover, if $t = \log n$ then it takes time $O(n \log n)$.

¹⁸Cyzowicz, Dobrev, Kranakis, Krizanc, The Power of Tokens: Rendezvous and Symmetry Detection for two Mobile Agents in a Ring. SOFSEM 2008. 

Open Problems

- We are lacking general non-trivial lower bounds for $t \geq 3$ tokens.
- Can we derive sharp upper and lower bounds for t tokens?
- Another problem concerns whether t tokens are really more powerful than $t - 1$ tokens.
- It would also be interesting to look at rendezvous with detection for more than two mobile agents, and also consider the case where no synchrony is assumed.

Oriented $n \times n$ Torus

Rendezvous in an Oriented Torus¹⁹

# Tokens	Token-Type	Memory	Time
$O(1)$	Unmovable	$o(\log n)$	∞
1	Movable	$o(\log n)$	∞
1	Unmovable	$O(\log n)$	\mathcal{RVD}
2	Movable	$O(1)$	\mathcal{RVD}

¹⁹Kranakis, Krizanc, Markou, Mobile Agent Rendezvous in a Synchronous Torus. In DAM 159 (2011) 896-923.

Open Problems

- Are the results tight?
- What about the
 - multidimensional torus?
 - hypercube?
- What is the Memory/# Tokens Tradeoff for RVD ?