Multi-robot Symmetric Rendezvous Search on the Line

Deniz Ozsoyeller[1] and Pratap Tokekar[2]

Abstract—We study the Symmetric Rendezvous Search Problem for a multi-robot system. There are $n > 2$ robots arbitrarily located on a line. Their goal is to meet somewhere on the line as quickly as possible. The robots do not know the initial location of any of the other robots or their own positions on the line. The symmetric version of the problem requires the robots to execute the same search strategy to achieve rendezvous. Therefore, we solve the problem in an online fashion with a randomized strategy. In this paper, we present a symmetric rendezvous algorithm which achieves a constant competitive ratio for the total distance traveled by the robots. We validate our theoretical results through simulations.

Index Terms—Multi-Robot Systems, Distributed Robot Systems, Path Planning for Multiple Mobile Robots or Agents, Motion and Path Planning.

I. INTRODUCTION

THERE are various examples of the rendezvous search in practice: two friends, separated in a shopping mall, and want to meet again; a group of parachutists want to meet after landing in a large field; a rescue helicopter looking for a lost hiker in the desert [1]. The common challenges are that the searchers are unaware of the location of the others and no common meeting point has been decided a priori. As such, the searchers need to execute an online rendezvous search to meet at a common location as quickly as possible.

In this paper, we study the robotic version of rendezvous search problem: a team of robots whose locations are unknown to each other should meet somewhere in the environment in the least time possible. The robots have limited sensing capabilities and are operating in a large environment. The robots can only communicate when they are in close proximity of each other; i.e., the communication is only possible when the robots meet each other. There are many reasons why the robots may need to rendezvous. For example, in a scenario where the robots need share information with the others urgently, a rendezvous algorithm would lead to better coordinated planning [2].

There are two primary versions of the rendezvous search problem, depending on whether or not the robots can meet in advance of the search to agree on the strategies that they will execute. In *asymmetric* rendezvous search, the robots can meet in advance and choose distinct roles for each robot. For example, one robot can wait while the other carries out the search. This is different from *symmetric* rendezvous search, where the robots execute the same strategy, since they do not have a chance to agree on their roles. In this version, it is not necessary to implement a different strategy on each robot; thus, this makes it more appealing for programming large multi-robot systems.

In this paper, we study the symmetric version of the rendezvous search problem with $n > 2$ robots that are arbitrarily located on a line. For example, robots may be deployed in a linear environment such as a road, a corridor, a river, or a tree row. We consider a scenario where the robots are unaware of their own position along the line or the positions of any of the other agents. In fact, we consider the even more challenging scenario where even the initial distance between any pair of robots is unknown. We also do not assume that the robots know the directions leading to the other robots (i.e., a robot does not know whether the other robots are to its left, or right, or both). Each robot can only keep track of their own positions relative to their own starting positions. In the absence of any prior knowledge and global information, we propose an online strategy to be followed by all the robots.

Note that if the identical robots that initially face the same direction start executing the same pure (deterministic) strategy simultaneously, then the distance between the adjacent robots on the line will always remain the same, thus the robots will never meet. For this reason, the solutions need to consider mixed strategies that involve randomization. Therefore, our strategy involves making random choices in the directions to move to break the symmetry of the search.

We analyze its performance using the notion of competitive ratio. The competitive ratio of an online algorithm is the worst-case ratio of the cost of the solution (i.e., distance traveled by the robots) found by the online algorithm to the cost of an optimal offline solution. For omniscient robots, the optimal offline algorithm would be to move toward the midpoint of the line segment with endpoints at the positions of the leftmost and rightmost robot. If an online algorithm has a constant competitive ratio, then it means it performs competitively with respect to omniscient robots even in the absence of prior knowledge.

The rendezvous search problem has been extensively studied in literature for linear search environments with the focus mostly on two-players rendezvous [3]–[12]. The studies on multi-player rendezvous [10], [13], [14] assume that the initial distances between the robots are known and the robots are placed equidistant apart. For the symmetric version, it is also assumed that when two players meet, they may exchange any information known to them at the time. We do not make any of these assumptions in our study. Moreover, our approach is distributed instead of centralized [15]. Our main contribution is to study the symmetric version of the multi-robot rendezvous

[1]Deniz Ozsoyeller is with the Department of Software Engineering, Yaşar University, Izmir, Turkey `deniz.ozsoyeller@yasar.edu.tr`

[2]Pratap Tokekar is with the Department of Computer Science, University of Maryland, College Park, MD, USA `tokekar@umd.edu`

problem at an unknown initial distance between any pair of robots. We present an online rendezvous algorithm which yields a constant competitive ratio.

II. RELATED WORK

The rendezvous search problem generalizes the *linear search problem* in which a single searcher tries to find a stationary object located at an unknown initial distance. This is also known as the *lost cow problem* and was originally solved in [16]. That solution was rediscovered in [17]. In the formulation of the lost cow problem, a near-sighted cow searches for the only gate in a long, straight fence. This gate is located at an unknown initial distance d, possibly to the left or right of the cow. The LostCow solution consists of the cow alternately searching to its left and then to its right starting from its initial location and doubling the distance it travels in each round until it finds the gate. This algorithm has a competitive ratio of 9 which is the best possible performance for a deterministic online algorithm. Beck and Newman [16] showed that introducing some randomness reduces this competitive ratio to 4.591. The same result was also provided by Kao et al. [18]. The linear search problem is also studied with two [19] or more agents [20] some of which are faulty [21], [22]. Chrobak et al. [20] extended the lost cow problem to k cows (termed Mobile Entities) and showed that, independent of the number of cows, the group search cannot be better than the LostCow algorithm's performance.

Two-player rendezvous search problem on the line has been well studied for the symmetric [3]–[8] and the asymmetric [7], [9]–[12], [23] versions. In the symmetric version, the initial distance between the players may be known [3]–[6], [9] or unknown [7], [8], [24]. Baston and Gal [7] considered the case where the initial distance $2d$ between the players is drawn from an unknown distribution with only its expected value known to be $E[2d] = \mu$. They provided a symmetric algorithm with expected meeting time 13.325μ and competitive ratio 26.650. In our previous work [8], we improved this result by providing a 17.686 distance-competitive and 24.843 time-competitive symmetric rendezvous algorithm for two robots at an unknown initial distance. In this paper, we extend this study to $n > 2$ robots. Unlike the two-robot case, in multi-robot-case, the search does not end when only two robots meet. Thus, this makes the efficient coordination of the robots more challenging.

The solutions for multi-player linear rendezvous search problem in the literature consider that the distance between each pair of adjacent robots is known [4], [10], [13], [14], [25], or the players are given some information about their past actions or probabilistic moves [26], or use a centralized approach [15] in which the route for each robot, that is aware of its global position, is assigned by a central authority. In contrast to these solutions, we propose a distributed algorithm to solve the problem without the knowledge of the initial distance between any pair of robots, the robot's own position, or the motion pattern of any other robot. Lim et al. [14] studied the rendezvous of $m \leq n$ blind, unit speed players. The authors showed that $R_{3,2}^a$ is 47/48 and $R_{n,n}^s$ is

asymptotic to $n/2$, where $R_{n,m}^a$ and $R_{n,m}^s$ denote the least expected rendezvous time of m players for the asymmetric and symmetric strategy, respectively. Prior to this study, Lim and Alpern focused on the asymmetric version of the same problem and minimizing the maximum time to rendezvous rather than the expected time [13]. The asymmetric value of the n-player minimax rendezvous time M_n has an upper bound $n/2 + (n/\log n) + o(n/\log n)$. Gal [10] presented a simpler strategy for the problem in [13] and showed that the worst case meeting time has an asymptotic behavior of $n/2 + O(\log n)$. The rendezvous of only three agents is considered in [4], [25]. Baston [4] proposed a strategy that has a maximum rendezvous time of 3.5. Alpern and Lim [25] showed that Baston's strategy is the only strategy to achieve this result.

III. PROBLEM FORMULATION

We consider $n > 2$ identical robots that are placed arbitrarily on a line. The initial distance between every pair of robots is unknown. The robots are unaware of their own initial positions and the initial positions of the others. We assume that each robot knows the number n. Rendezvous can occur anywhere on the line, since no rendezvous location is fixed in advance. We study the synchronous case in which we assume that the robots have synchronized clocks, thus, they start executing the algorithm at the same time.

Let x_j denote the initial position of robot R_j in order from left to right, where $j \in 1, ..., n$. Let d be initial distance between the leftmost robot R_1 and the rightmost robot R_n. Without loss of generality, R_1 is located at $x = 0$ and R_n is located at $x = d$. Based on the initial configuration of the robots, R_1 and R_n are called the boundary robots and the robots that are located between R_1 and R_n are called the internal robots. It is easy to see that each boundary robot has one neighbor while each internal robot has two neighbors.

We focus on minimizing the distance competitive ratio of our algorithm. Due to using symmetric strategies, the expected distance traveled is identical for all robots. Here the expectation is taken over the randomization in the strategy and over the number of robots. Therefore, we only analyze it for one robot.

IV. MULTI-ROBOT SYMMETRIC RENDEZVOUS ALGORITHM

In this section, we present the distributed online algorithm $\mathcal{MSR}$ for multi-robot rendezvous. Since the search is on the line, we maintain the invariant that the robots never cross paths. This implies that once the boundary robots (R_1 and R_n) meet, then the algorithm terminates, since all robots would meet. Therefore, the main idea in $\mathcal{MSR}$ is to find out if any robot pair can act as boundary robots and meet the rest of the robots to achieve rendezvous.

We assign a non-negative sequence $f_{-1}, f_0, f_1, f_2, \ldots$ to each robot, where $f_{-1} = 0$ and $f_i = r^i$, for $i \geq 0$. Robots use the same expansion radius $r > 1$ which is fixed throughout the algorithm. We derive the optimized value of r in the proof of Theorem 1. The algorithm proceeds in rounds indexed by integers $i \geq 0$. Each round consists of two consecutive phases:

phase-1 and phase-2. The first direction the robot performs the search in round i refers to phase-1. We denote this direction by dir_1. In phase-2 of round i, the robot searches the opposite direction of dir_1, which is denoted by $\overline{dir_1}$. Let the initial distance between R_1 and R_n be $d = r^{k+\delta}$, where $\delta \in (0, 1]$ and $k \in \mathbb{Z}^+$. Note that δ and k are unknown to the algorithm.

Our online algorithm is a hybrid of two search modes: randomized search and deterministic search. The robot executing $\mathcal{MSR}$ starts by performing a randomized search to break the symmetry but switches to a deterministic search once it meets a robot. A robot is called *single* if it has not met any other robot yet. Thus, at the beginning of the algorithm, all robots are single. As long as the robot is single, it executes the randomized search mode in which it flips a coin at the beginning of each round to decide its itinerary. If it tosses heads, then it moves right in phase-1 and left in phase-2; otherwise it moves left in phase-1 and right in phase-2. Rendezvous can never occur if the coin flips of all robots in each round after the start come up the same. This also implies that the robots will always be executing the randomized mode of the algorithm. The randomized mode adopts the symmetric rendezvous algorithm SR presented in our previous work for two robots [8].

When the single robot meets another single robot, it switches to the deterministic search mode. In this mode, the robot does not randomize its direction at the beginning of a round. Instead, it always starts the round in its deterministic direction dir_d. If the single robot meets another single robot in phase-1 of round i, then $dir_d = \overline{dir_1}$. However, if the single robot meets another single robot in phase-2 of round i, then $dir_d = dir_1$. Once the robot is in the deterministic search mode, its search becomes similar to the one in the *LostCow* algorithm, with the difference that in our algorithm, the robot uses the first direction it moves in a round (phase-1) to search for any undiscovered mobile robots instead of a stationary object while it uses the second direction it moves (phase-2) to see whether the rest of the robots are gathered.

We now describe the motion pattern of robot R_j. In round i, R_j starts at one of $x = x_j \pm f_{2i-1}$. If the robot is to first search the right direction in round i, then it moves right to $x = x_j + f_{2i}$ in phase-1 and left to $x = x_j - f_{2i+1}$ in phase-2. If the robot is to first search the left direction in round i, then it moves left to $x = x_j - f_{2i}$ in phase-1 and right to $x = x_j + f_{2i+1}$ in phase-2. Assuming that R_j does not meet any robot during round i, its possible ending position is $x = x_j \pm f_{2i+1}$. This is also the initial position for round $i+1$. The possible itineraries of R_j based on its initial position in round i are shown in Fig. 1.

In our algorithm, the robot can also be in one the following additional states beyond the initial *single* state: $boun_t$ and *internal*. We determine these states based on the fact that an internal robot can meet another robot in both directions since it has two neighbors while a boundary robot can meet another robot in only one direction since it has only one neighbor. However, in $\mathcal{MSR}$, although the robot in *internal* state is the actual internal robot in the initial configuration, the robot in $boun_t$ state only acts as one but does not have to be the actual boundary robot. Thus, we refer $boun_t$ as temporary boundary.
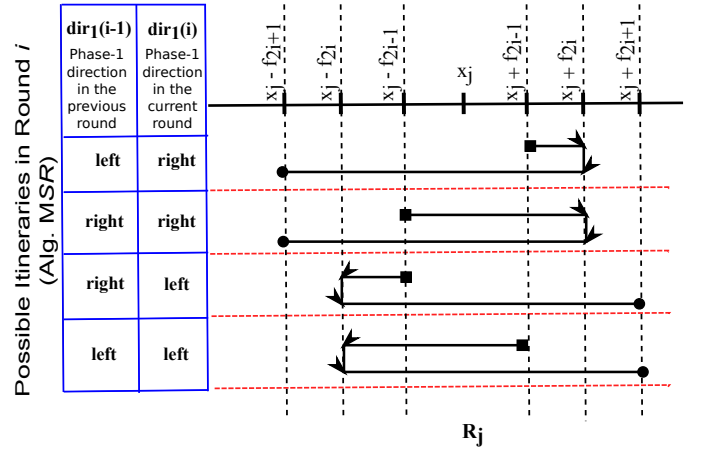

Fig. 1: The possible itineraries of robot R_j executing algorithm $\mathcal{MSR}$. The arrows show the directions that the robot is moving in round i. The small square and circle icons show the beginning and end of a motion, respectively.
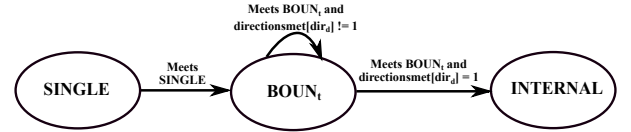

Fig. 2: Finite state machine representation of $\mathcal{MSR}$.

Next, we introduce the behaviours of the robot that meets another robot in a round. When single robots R_j and $R_{j'}$ meet in round i, they both change their states to $boun_t$. Recall that this is the last round before both robots switch to the deterministic search mode. Starting from round $i+1$, the $boun_t$ robot R_j no longer flips a coin but always starts a round in dir_d. The robot in this state continues the search until it meets another $boun_t$ robot in dir_d. When this happens, it changes its state to internal and waits until it sticks to a $boun_t$ robot. A sequence of consecutive internal robots is always surrounded by $boun_t$ robots which carry out the search in both directions. This also implies that both neighbors of an internal robot can be internal, or one of its neighbors can be internal while the other is $boun_t$. Thus, a single robot can never encounter an internal robot. It is easy to see that the robots R_1 and R_n can never be internal since they each have only one neighbor, hence can meet the other only in one direction. A $boun_t$ robot collects any single robot moving towards it. In other words, the single robot always sticks to and moves along with the $boun_t$ robot that it meets. Therefore, the neighbors of a robot at the initial configuration can change when the robot changes its state to $boun_t$. For example, consider the consecutive robots R_{j-1}, R_j, and R_{j+1} at the beginning. Although R_{j+1} is not initially adjacent to R_{j-1}, if R_j sticks to R_{j+1}, then R_{j+1} becomes adjacent to R_{j-1}. The pseudocode of $\mathcal{MSR}$ is presented in Algorithm 1 and Algorithm 2, and its finite state representation is shown in Fig. 2.

Let $D_{i,1}(D_{i,2})$ denote the distance traveled in phase-1(phase-2) of round i. If the direction in phase-1 of this round differs from the previous round, then $D_{i,1} = f_{2i} - f_{2i-1}$.

Otherwise, $D_{i,1} = f_{2i} + f_{2i-1}$. Regardless of the direction in phase-1, $D_{i,2} = f_{2i} + f_{2i+1}$. Thus, the distance traveled (the length of an itinerary) in round i is either $D_i = f_{2i+1} + 2f_{2i} - f_{2i-1}$ or $D_i = f_{2i+1} + 2f_{2i} + f_{2i-1}$. The maximum total time required for round i is $T_i = T_{i,1} + T_{i,2}$, where $T_{i,1} = f_{2i} + f_{2i-1}$ and $T_{i,2} = f_{2i} + f_{2i+1}$.

In the synchronous case, robots start each phase of a round at the same time. This is managed by introducing waiting times in $\mathcal{MSR}$ algorithm. The waiting time in a phase is determined by the following cases: (1) The robot does not meet another robot. (2) The robot meets another robot but continues moving without waiting at the met location, which occurs when a $boun_t$ robot meets a single or an internal robot. (3) The robot meets another robot and waits at the met location till the end of the phase, which occurs when two $boun_t$ or two single robots meet. If case-1 or case-2 occurs, then $W_{i,1} = T_{i,1} - D_{i,1}$ and $W_{i,2} = T_{i,2} - D_{i,2}$. Let $D_{i,1,m}(D_{i,2,m})$ denote the distance traveled by the robot until it meets another robot in phase-1(phase-2) of round i. Since any robot pair meets before they reach their ending positions in the phase, we have $D_{i,1,m}(D_{i,2,m}) < D_{i,1}(D_{i,2})$. If case-3 occurs, then $W_{i,1} = T_{i,1} - D_{i,1,m}$ and $W_{i,2} = T_{i,2} - D_{i,2,m}$.

An example execution of $\mathcal{MSR}$ when $d = 50$, $r = 1.28$, and $n = 10$ is presented in Fig. 3. Each robot has unit speed. Fig. 3 (Top) shows the positions of the robots until rendezvous. The small circles on the plot depict the meeting time of the robots. The rendezvous occurs after 7 rounds. Fig. 3 (Bottom) shows the robot meetings and the changes in their states during a round. Initially, all robots are in single (S) state. In phase-1 of round 2, R_4 and R_5 meet and change their states to $boun_t$ (B). This is followed, in the next phase, by the meetings of two pairs of single robots: R_0 and R_1, and R_8 and R_9, which also change their states to $boun_t$. In round 3, single robots R_3, R_6, R_7 meet and stick to $boun_t$ robots R_4, R_5, and R_8, respectively. The ID of the robot that sticks to another robot is shown below this robot's ID. In round 4, single R_2 meets $boun_t$ R_1 and sticks to it. Next, we see that the $boun_t$ robots R_1 and R_4 meet in their deterministic direction and become internal robots. The $boun_t$ robots R_5 and R_8 also meet in their deterministic direction and become internal robots. In phase-2 of round 5, R_0 meets and collects the waiting internal robots R_1 and R_4, while R_9 meets and collects the waiting internal robots R_8 and R_5. Finally, the rendezvous occurs when robots R_0 and R_9 meet.

V. Analysis of Algorithm $\mathcal{MSR}$

In this section, we find an upper bound on the expected distance traveled by the robot and compare it with the optimal offline solution. Note that since our strategy is symmetric, the expected distance traveled is identical for all robots. In the offline setting, the robots know their positions on the line and the initial distance between the boundary robots. Thus, the solution would be for them to move toward $x = d/2$. Our main result is presented in Theorem 1.

Let α be the first round that any robot can travel far enough to reach the meeting location of any pair of robot. We divide the execution of 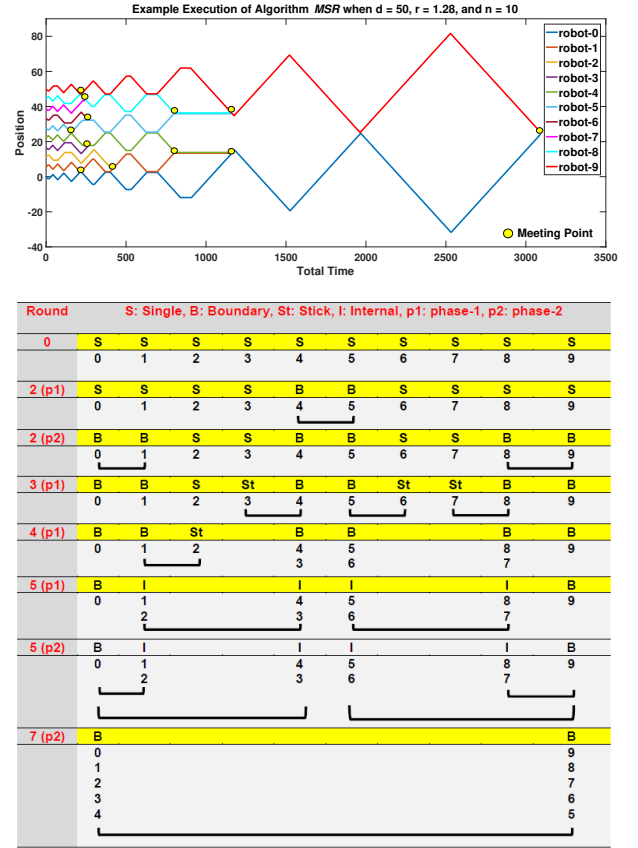the analysis into two stages. *Stage-1* consists of rounds $0 \leq i \leq \alpha - 1$. We ignore the possibility of rendezvous occurring in this stage and bound the probability of rendezvous not occurring. *Stage-2* consists of rounds $i \geq \alpha$ in which the rendezvous can occur. In Lemma 2, we show that if any pair of robots meet in Stage-2, then all robots meet. Rendezvous cannot occur in this stage as long as the coin flips of all robots come up the same, when all robots are still in the randomized search mode of $\mathcal{MSR}$. Thus, we compute the expected distance traveled using an infinite sum.

Let S_i denote the event that at least one adjacent single robot pair p meets in round i. Here p can be any adjacent robot pair. S_i will be our proxy event for rendezvous. Let M_i^j be the event that R_j initially moves right and $\overline{M}_i^j$ be the event that R_j initially moves left in round i. The adjacent robots R_j and R_{j+1} can meet if event $E_1 = M_i^j \wedge \overline{M}_i^{j+1}$ or $E_2 = \overline{M}_i^j \wedge M_i^{j+1}$ occurs. The probability that a single robot meets an adjacent single or $boun_t$ robot in round i is $\mathbb{P}[E_1 \vee E_2] = 1/2$. On the other hand, R_j and R_{j+1} cannot meet when they initially move in the same direction; that is if event $E_3 = M_i^j \wedge M_i^{j+1}$ or $E_4 = \overline{M}_i^j \wedge \overline{M}_i^{j+1}$ occurs. Thus, we have

$$\mathbb{P}[S_i] = \begin{cases} 0 & \text{for } i \leq \alpha - 1, \\ \frac{2^n - 2}{2^n} & \text{for } i \geq \alpha. \end{cases}$$

Next, we calculate the value of round α in terms of k and r. To upper bound the expected distance traveled, we consider the furthest possible meeting location x_m for p. Assuming

Fig. 3: Example execution of algorithm $\mathcal{MSR}$.

Algorithm 1: $\mathcal{MSR}$, Multi-robot symmetric rendezvous search algorithm.

```
1  global
   //  S:SINGLE;  B:BOUNDARYt;  I:INTERNAL;  N:NOROBOT
2     state = S
3     meets = N
4     stick = 0
5     firstmeeting = 0
6     direction = 1
7     directionsmet[2] = {0}
8     RIGHT = 1, LEFT = -1
9     HEAD = 0, TAIL = 1
10 end global
11
12 i = 0 // i denotes the current round.
13 phase = 1
14 continues = true
15 while Rendezvous() ≠ true do
16    if state == S then
17       cointoss = Random(HEAD, TAIL)
18       direction = RIGHT
19       if cointoss == TAIL then
20          direction = LEFT
21       end
22    end
23    if stick ≠ 1 then
24       if state ≠ I then
25          phase = 1
26          continues = MoveTo(direction * f(2i), direction,
                          phase, i)
27          if continues == true then
28             phase = 2
29             continues = MoveTo(-1 * direction * f(2i + 1),
                             -1 * direction, phase, i)
30             if continues == true then
31                i = i + 1
32             end
33          end
34       end
35       if state == I then
36          WaitInternal()
37       end
38    end
39    if stick == 1 then
40       StickToRobot()
41    end
42 end
```

Algorithm 2: MoveTo(*destination*, *currdirection*, *phase*, *round*)

```
1  dir = 0
2  prevpos = GetCurrentPos()
3
4  if currdirection == LEFT then
5     dir = 1
6  end
7  if firstmeeting == 1 then
8     direction = currdirection
9     firstmeeting = 0
10 end
11
12 while Reach(destination) ≠ true do
13    meets = CheckState()
14    if (state == S or state == B) and meets == N then
15       Move(currdirection)
16    else if state == S and meets == B then
17       directionsmet[dir] = 1
18       stick = 1
19       return false
20    else if state == S and meets == S then
21       state = B
22       directionsmet[dir] = 1
23       firstmeeting = 1
24       break
25    else if state == B and (meets == I or meets == S) then
26       CollectAndMove(currdirection, destination)
27    else if state == B and meets == B then
28       if directionsmet[dir] ≠ 1 then
29          directionsmet[dir] = 1
30          state = I
31          return false
32       else if directionsmet[dir] == 1 and Total() != n then
33          break
34       else if directionsmet[dir] == 1 and Total() == n then
35          rendezvous = 1
36          return false
37       end
38    end
39 end
40
41 currpos = GetCurrentPos()
42 D_round,phase = DistanceTrav(prevpos, currpos)
43 W_round,phase = T_round,phase - D_round,phase.
44 Wait(W_round,phase)
45 return true
```

that one of the robots in p can be one of the actual boundary robots, either $x_m \in (d, d + f_{2i-1})$ or $x_m \in (-f_{2i-1}, 0)$. Let this boundary robot be R_n. Then, robot R_j is able to reach x_m if $x_j + f_{2i+1} \geq d + f_{2i-1}$. In the worst case, $R_j = R_1$, so $x_j = x_1 = 0$. Therefore, we have

$$f_{2i+1} \geq d + f_{2i-1}, \tag{1}$$

which holds for $\alpha \geq k/2 + 1 - \log_{r^2} r^2 - 1$. Henceforth, $\alpha = k/2 + i^*$, where $i^* = 1 + \lceil |\log_{r^2} r^2 - 1| \rceil$.

Let A_i denote the event that the algorithm is still active in round i. We assume that Algorithm $\mathcal{MSR}$ is always active when $i < \alpha$, thus $\mathbb{P}[A_{i<\alpha}] = 1$. In Lemma 2, we show that rendezvous cannot occur in round $i \geq \alpha$ only if all robots' coin flips come up the same. Therefore, the probability that the algorithm is still active in round $i \geq \alpha$ is $\mathbb{P}[A_{i\geq\alpha}] = \mathbb{P}[\overline{S_0} \wedge \cdots \wedge \overline{S_{i-1}}]$. Thus, we have

$$\mathbb{P}[A_i] = \begin{cases} 1 & 0 \leq i \leq \alpha - 1, \\ \left(\frac{2}{2^n}\right)^{i-\alpha+1} & \alpha \leq i. \end{cases} \tag{2}$$

We continue by analyzing the distance traveled in Stage-1 and Stage-2. We start with the computation of the expected distance traveled during Stage-1 which encompasses the rounds $i < \alpha$.

Lemma 1. *The expected distance traveled during Stage-1 satisfies*

$$\sum_{i=0}^{\alpha-1} \mathbb{E}[D_i \mid A_i]\mathbb{P}[A_i] < r^{k+2i^*}\left(\frac{r + 2 + r^{-1}}{r^2 - 1}\right). \tag{3}$$

Proof. Since we consider that rendezvous cannot occur in this stage, $\mathbb{P}[A_i] = 1$. The distance traveled by a robot in round i is $D_i \leq f_{2i+1} + 2f_{2i} + f_{2i-1}$. Therefore,

$$\sum_{i=0}^{\alpha-1} \mathbb{E}[D_i \mid A_i]\mathbb{P}[A_i] < \sum_{i=0}^{\alpha-1} (f_{2i+1} + 2f_{2i} + f_{2i-1}) \cdot 1$$
$$< r^{k+2i^*}\left(\frac{r + 2 + r^{-1}}{r^2 - 1}\right).$$

$\square$

We now compute the expected distance traveled for all rounds $i \geq \alpha$. Unlike Stage-1, rendezvous occurs in this stage with nonzero probability. In the next lemma, we show that the meeting of any pair of single robots always results in rendezvous in the next round.

Lemma 2. *Given that event S_i occurs in round $i \geq \alpha$, then the robots always achieve rendezvous in round $i + 1$.*
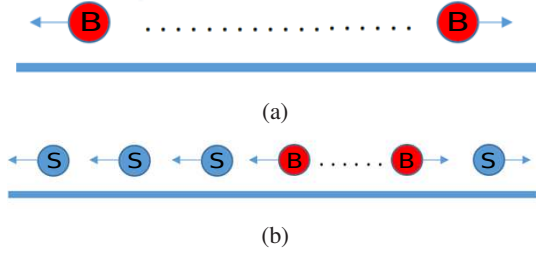
Fig. 4: Two possible configurations at the end of phase-1 of round $i + 1$, given that event S_i occurs in round i, that is at least one adjacent single robot pair p meets in round i. B refers to the boun_t robot and S refers to the single robot.

Proof. Consider that all robots are single at the beginning of round i. (1) ensures that any adjacent single robot pair can meet if event $E_1 \vee E_2$ occurs between them in round $i \geq \alpha$ and thus can become a boun_t robot pair. The actual boundary robots can never be internal. Moreover, each sequence of consecutive internal robots is always located between two boun_t robots which move in opposite deterministic directions. Therefore, it is guaranteed that once there is a boun_t robot pair in round i, then there will always be at least one active boun_t robot pair until rendezvous.

First, we show that the rendezvous may not occur in round i if this is the first round that there is a boun_t robot pair p. In $\mathcal{MSR}$, if pair p is formed in phase-1, then both robots in p wait till the end of this phase, and move away from each other in phase-2. Thus, rendezvous cannot occur in this round. Following an unsuccessful round i, the robots will start round $i+1$. Now, consider that we have at least one boun_t robot pair p at the beginning of round $i + 1$. At the end of phase-1 of round $i + 1$, the robots can have two possible configurations. In the first configuration, which is shown in Fig. 4(a), the rest of the robots are between the leftmost boun_t robot R_j and the rightmost boun_t robot $R_{j'}$. This can occur in two ways. The first way is when R_j and $R_{j'}$ are the actual boundary robots. The second way is when R_j and $R_{j'}$ are not the actual boundary robots but have already met any single robot in their deterministic direction. That is, any robot that started to the left of R_j and right of $R_{j'}$ would have already met R_j and $R_{j'}$. Recall that a single robot sticks to the boun_t robot that it meets. In phase-2 of the first configuration, R_j and $R_{j'}$ move towards each other while collecting any robot between them, which would be already in internal state, thus the rendezvous occurs.

In the second configuration shown in Fig. 4(b), there can only be single robots in the deterministic directions of R_j and $R_{j'}$. In this case, the single robots on the left side of R_j move in the same direction as R_j, while the single robots on the right side of $R_{j'}$ move in the same direction as $R_{j'}$. Thus, the rendezvous cannot occur in phase-1. However, in phase-2, R_j and $R_{j'}$ move towards each other while collecting any robots between them and start waiting before starting the next round. Meanwhile, the single robots move toward the waiting boun_t robots R_j and $R_{j'}$ and meet them, thus the rendezvous occurs. (1) also ensures that while the boun_t robots are waiting for the

end of phase-2 at their meeting location, a single robot moving towards this location would be able to meet them. $\qquad\square$

Lemma 3. *The expected distance traveled during Stage-2 satisfies*

$$\sum_{i=\alpha}^{\infty} \left[\left(\mathbb{E}[D_i \mid \overline{S}_i]\mathbb{P}[\overline{S}_i]\right) + \left(\mathbb{E}[D_i \mid S_i]\mathbb{P}[S_i]\right) \right] \mathbb{P}[A_i]$$
$$< r^{k+2i^*} \left(\frac{0.75r^3 + 1.5r^2 + 1.75r + r^{-1} + 2}{4 - r^2} \right). \quad (4)$$

Proof. If S_i does not occur in round i, then the expected distance traveled in round i is

$$\mathbb{E}[D_i \mid \overline{S}_i] \leq f_{2i+1} + 2f_{2i} + f_{2i-1}. \quad (5)$$

By Lemma 2, event S_i in round i results in rendezvous in round $i + 1$. Therefore, the distance traveled when S_i occurs is equal to the distance traveled in rounds i and $i+1$. Hence, we have

$$\mathbb{E}[D_i \mid S_i] < (f_{2i+1} + 2f_{2i} + f_{2i-1}) +$$
$$(f_{2(i+1)+1} + 2f_{2(i+1)} + f_{2(i+1)-1})$$
$$< r^{2i} \left(r^3 + 2r^2 + 2r + r^{-1} + 2 \right). \quad (6)$$

The expected distance traveled during Stage-2 is the sum of the expected distance traveled in unsuccessful rounds and the expected distance traveled in the (final) successful round, which is given by

$$\sum_{i=\alpha}^{\infty} \left[\left(\mathbb{E}[D_i \mid \overline{S}_i]\mathbb{P}[\overline{S}_i]\right) + \left(\mathbb{E}[D_i \mid S_i]\mathbb{P}[S_i]\right) \right] \mathbb{P}[A_i]. \quad (7)$$

First, we compute the expected distance traveled in successful rounds during Stage-2 using (2) and (6), which is bounded by

$$\sum_{i=\alpha}^{\infty} \left(\mathbb{E}[D_i \mid S_i]\mathbb{P}[S_i]\right) \mathbb{P}[A_i]$$
$$< 0.75 r^{k+2i^*} \left(\frac{r^3 + 2r^2 + 2r + r^{-1} + 2}{4 - r^2} \right) \text{ for } n > 2.$$
$$(8)$$

Next, we compute the expected total distance traveled during unsuccessful Stage-2 rounds using (2) and (5), which is bounded by

$$\sum_{i=\alpha}^{\infty} \left(\mathbb{E}[D_i \mid \overline{S}_i]\mathbb{P}[\overline{S}_i]\right) \mathbb{P}[A_i]$$
$$< 0.25 r^{k+2i^*} \left(\frac{r + 2 + r^{-1}}{4 - r^2} \right) \text{ for } n > 2. \quad (9)$$

Therefore, we conclude that (7) is bounded by the sum of (8) and (9). $\qquad\square$

Theorem 1. *For the choice of $r = 1.28$, $\mathcal{MSR}$ algorithm has the competitive ratio of 54.732 and the competitive ratio of 34.154 as $n \to \infty$.*

Proof. The expected distance traveled is obtained by adding the expressions in equations (3) and (4). Since the initial distance between the leftmost and rightmost robot on the line is $d = r^{k+\delta}$, we first replace each occurrence of r^k with

$dr^{-\delta}$ and n with 3. Then, we divide it by $d/2$, which is the cost of the optimal offline solution. This expression is maximized at $\delta = 0$, and the coefficient of d in it is minimized when $r = 1.28$. As a result, we guarantee the competitive ratio of 54.732. As $n \to \infty$, the distance traveled in Stage-2 approaches to 0; thus the competitive ratio of our algorithm approaches to 34.154. □

VI. SIMULATIONS

In this section, we study the performance of our algorithm in simulations and verify the bound obtained in Theorem 1. We implemented a multithreaded system using Java, where each robot is represented as a separate thread that executes the Algorithm $\mathcal{MSR}$. Varying the parameters d, n, and r, we report the following: average distance competitive ratio, average number of rounds, average total time, average time competitive ratio, and average distance traveled. Each result is averaged over 100 trials, where each trial uses the maximum of the respective data among n robots.

Fig. 5 (a) and (b) show the average distance competitive ratio as a function of n an d and as a function of n and r, respectively. This ratio is obtained by dividing the maximum distance traveled by $d/2$, where d is the respective initial distance value tested in the simulations. In plot (a), n is varied between 4 and 64 while d is varied between 50 and 150. The robots are distributed uniformly random on the line. We observe that the average distance competitive ratio is higher when n is small and stays constant as n and d increases. We further see that this ratio is smaller than our theoretically proved upper bound. Recall that the optimal r value obtained in Theorem 1 is 1.28. In plot (b), this value is shown with double stars. With respect to the change in n and when d is fixed, we can see that $\mathcal{MSR}$ performs the best for $r = 1.28$.

Next, in Fig. 5 (c) and (d), we investigate the average number of rounds to achieve rendezvous with respect to the change in n and d (Plot (c)), also with respect to the change in n and r (Plot (d)). The average of total number of rounds stays constant as n increases and decreases as r increases. Since the number of rounds is proportional to the distance the traveled by the robots, in plot (c), we see that the average number of rounds increases as d increases, but stays constant with respect to the change in n.

The total time required for rendezvous is equal to the sum of the total distance traveled and total waiting time. Fig. 5 (e) and (f) show the results of the simulations for the average total time with respect to the change in n. Plot (e) reports the results for various d when $r = 1.28$, whereas plot (f) reports the results for various r values when $d = 75$. As expected, the total time increases as d increases and stays constant as n increases. In plot (f), we observe that $r = 1.28$ outperforms the other r values as n increases. Figure 5 (g) shows the time competitive ratio with respect to the change in n and d. Comparing the distance and time ratios in Figure 5 (a) and (g), we see that the distance competitive ratio is less than time competitive ratio.

Fig. 5 (i) provides the average distance traveled for various n values with respect to the change in d. As in the average total time results, we see that the average distance traveled increases as d increases and stays constant as n increases for $r = 1.28$. In Fig. 6, we compare the effect of equidistant and uniformly random initial placements of the robots on the performance of our algorithm. For various n values and $d = 75$, the average distance traveled is smaller when the robots are randomly located on the line compared to the other case. The performances of $\mathcal{MSR}$ in these two cases approach each other as n increases.

VII. CONCLUSION

This paper addresses the symmetric rendezvous search problem with multiple robots on the line. In our problem formulation, the initial distance between any pair of robots is unknown to the robots. Moreover, the robots do not know their own positions and the positions of each other. For this problem, we introduced an online algorithm whose competitive ratio is 57.732 which in the asymptotic case becomes 34.154. The algorithm presented here can be extended to the asynchronous case. In the asynchronous case, the robots do not need to start at the same time and use waiting time within the rounds. Our immediate future work is to present this extension and provide its theoretical performance guarantees and simulations.

ACKNOWLEDGEMENT

REFERENCES

[1] L. Thomas and P. Hulme, "Searching for targets who want to be found," *Journal of the Operational Research Society*, vol. 48, no. 1, pp. 44–50, 1997.

[2] M. Meghjani and G. Dudek, "Multi-robot exploration and rendezvous on graphs," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5270–5276.

[3] E. J. Anderson and S. Essegaier, "Rendezvous search on the line with indistinguishable players," *SIAM Journal on Control and Optimization*, vol. 33, no. 6, pp. 1637–1642, 1995.

[4] V. Baston, "Note: Two rendezvous search problems on the line," *Naval Research Logistics*, vol. 46, no. 3, pp. 335–340, 1999.

[5] Q. Han, D. Du, J. Vera, and L. F. Zuluaga, "Improved bounds for the symmetric rendezvous value on the line," *Operations research*, vol. 56, no. 3, pp. 772–782, 2008.

[6] G. Stachowiak, "Asynchronous deterministic rendezvous on the line," in *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 2009, pp. 497–508.

[7] V. Baston and S. Gal, "Rendezvous on the line when the players' initial distance is given by an unknown probability distribution," *SIAM Journal on Control and Optimization*, vol. 36, no. 6, pp. 1880–1889, 1998.

[8] D. Ozsoyeller, A. Beveridge, and V. Isler, "Symmetric rendezvous search on the line with an unknown initial distance," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1366–1379, 2013.

[9] S. Alpern and S. Gal, "Rendezvous search on the line with distinguishable players," *SIAM Journal on Control and Optimization*, vol. 33, no. 4, pp. 1270–1276, 1995.

[10] S. Gal, "Rendezvous search on the line," *Operations Research*, vol. 47, no. 6, pp. 974–976, 1999.

[11] S. Alpern and A. Beck, "Asymmetric rendezvous on the line is a double linear search problem," *Mathematics of Operations Research*, vol. 24, no. 3, pp. 604–618, 1999.

[12] ——, "Pure strategy asymmetric rendezvous on the line with an unknown initial distance," *Operations Research*, vol. 48, no. 3, pp. 498–501, 2000.

[13] W. S. Lim and S. Alpern, "Minimax rendezvous on the line," *SIAM Journal on Control and Optimization*, vol. 34, no. 5, pp. 1650–1665, 1996.
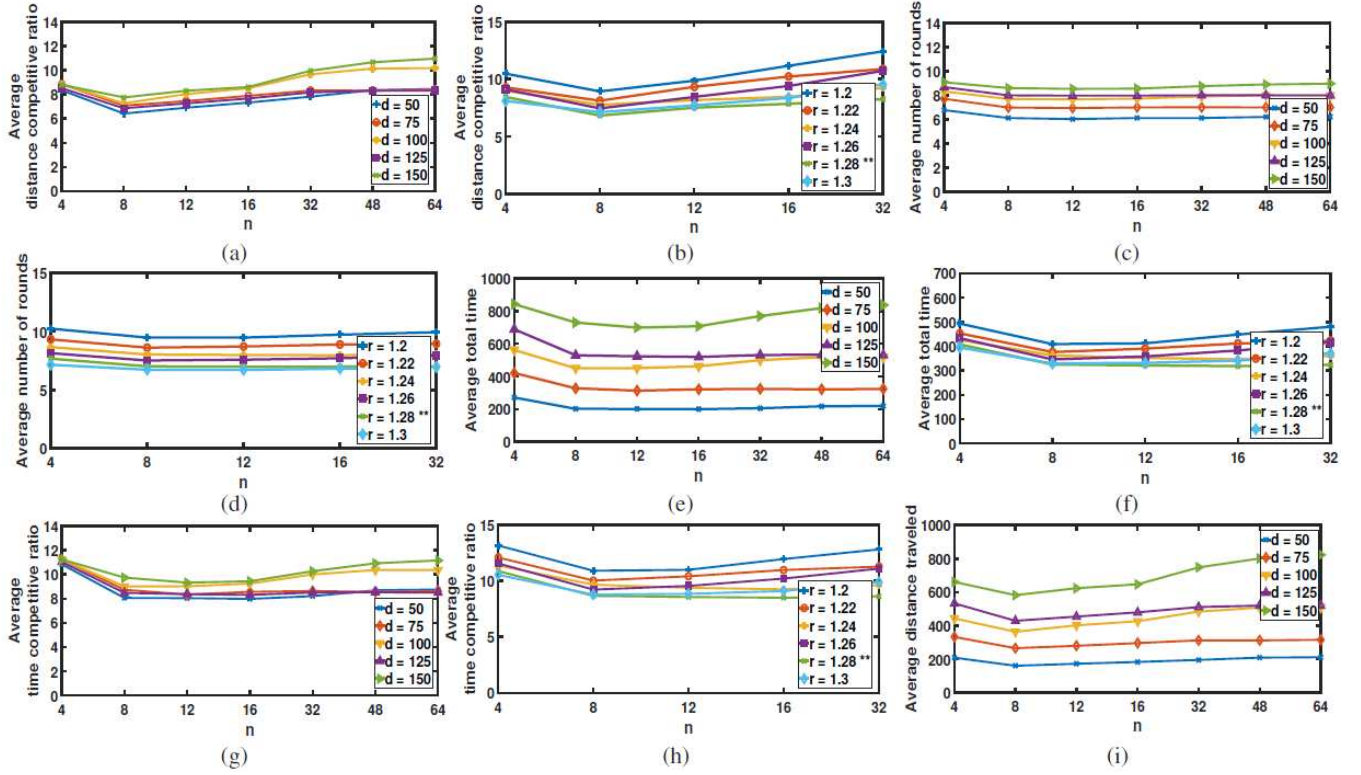
Fig. 5: (a) and (b) Average distance competitive ratio. (a) $r = 1.28$. (b) $d = 75$. (c) and (d) Average of total number of rounds. (c) $r = 1.28$. (d) $d = 75$. (e) and (f) Average total time. (e) $r = 1.28$. (f) $d = 75$. (g) and (h) Average time competitive ratio. (g) $r = 1.28$. (h) $d = 75$. (i) Average distance traveled. $r = 1.28$.
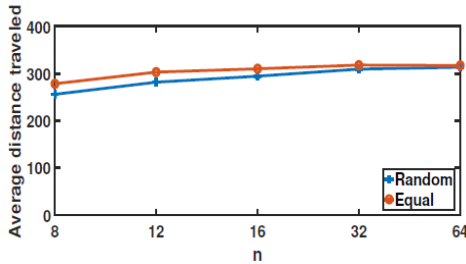

Fig. 6: Average distance traveled. Random vs Equal initial distance, $d = 75$.

[14] W. Lim, S. Alpern, and A. Beck, "Rendezvous search on the line with more than two players," *Operations Research*, vol. 45, no. 3, pp. 357–364, 1997.
[15] H. Chuangpishit, J. Czyzowicz, E. Kranakis, and D. Krizanc, "Rendezvous on a line by location-aware robots despite the presence of byzantine faults," in *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*. Springer, 2017, pp. 70–83.
[16] A. Beck and D. Newman, "Yet more on the linear search problem," *Israel journal of mathematics*, vol. 8, no. 4, pp. 419–429, 1970.
[17] R. A. Baezayates, J. C. Culberson, and G. J. Rawlins, "Searching in the plane," *Information and computation*, vol. 106, no. 2, pp. 234–252, 1993.
[18] M.-Y. Kao, J. H. Reif, and S. R. Tate, "Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem," *Information and Computation*, vol. 131, no. 1, pp. 63–79, 1996.
[19] E. Bampas, J. Czyzowicz, L. Gasieniec, D. Ilcinkas, R. Klasing, T. Kociumaka, and D. Pajak, "Linear search by a pair of distinct-speed robots," *Algorithmica*, vol. 81, no. 1, pp. 317–342, 2019.
[20] M. Chrobak, L. Gasieniec, T. Gorry, and R. Martin, "Group search on the line," in *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 2015, pp. 164–176.
[21] J. Czyzowicz, E. Kranakis, D. Krizanc, L. Narayanan, and J. Opatrny, "Search on a line with faulty robots," *Distributed Computing*, vol. 32, no. 6, pp. 493–504, 2019.
[22] A. Kupavskii and E. Welzl, "Lower bounds for searching robots, some faulty," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, 2018, pp. 447–453.
[23] P. Leone, J. Buwaya, and S. Alpern, "Search-and-rescue rendezvous," *European Journal of Operational Research*, 2021.
[24] J. Czyzowicz, R. Killick, and E. Kranakis, "Linear rendezvous with asymmetric clocks," in *22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, 2018.
[25] S. Alpern and W. S. Lim, "Rendezvous of three agents on the line," *Naval Research Logistics (NRL)*, vol. 49, no. 3, pp. 244–255, 2002.
[26] M. A. A. El-Hadidy, "Study on the three players' linear rendezvous search problem," *International Journal of Operational Research*, vol. 33, no. 3, pp. 297–314, 2018.