Project Paper for 6.854

Members: Seungkook Yun (yunsk@mit.edu)
Sooho Park (dreamneo@mit.edu)

Distributed Online Matching Algorithm For Multi-Path Planning of Mobile Robots

1. Introduction

Currently, we are working on mobile robots which move only on a special structure: trusses (See Fig. 2). Each robot has two grippers and three joints for 3-d motion (See Fig.1). This robot is expected to move on the trusses and help construction of the structure in future. We need to develop a collaboration scheme for lots of robots on the same structure. Note that a robot can only move on the specific points on the trusses and every robot has the identical structure and functions. Also, it must occupy at least one point to support itself.

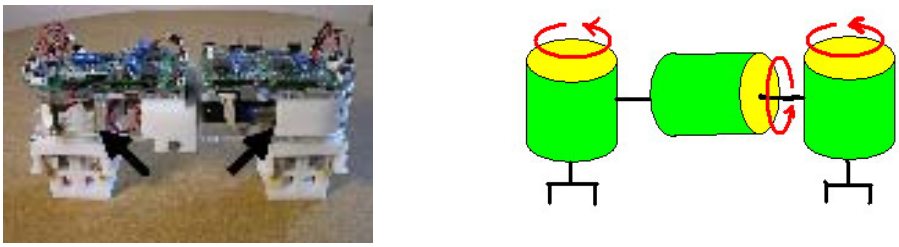


Fig.1 the mobile robot and its structure: 3-joints and 2 grippers

Fig.2 (a) truss structure and a robot
(b) example of path planning of a single robot: numbers denote nodes on the trusses

1.1 Problem Statement

Our task is to deploy robots to request points on the trusses by minimum number of total moves. It is to minimize the energy usage. For this, we need a collision-free min-cost path planning algorithm for multi-robots since a robot can't move through the other robot. The trusses can be modeled by a graph, in which nodes are the specific points where a gripper of the robot can grasp and edges denote whether a robot can move between pairs of nodes. We will assume that the graph structure, the target points, and the initial position of robots are given, and also that the robots are collaborating.

The real challenge is that we want a distributed algorithm. That is, we do not want a centralized controller which knows everything and controls all the robots. Instead, a robot has limited communication and sensing area which is modeled by number of edges from the node occupied by the robot, and it must decide which way to go with the limited (sensed) information.

1.2 Our Approach and Results

In this paper, we show that the problem can be solved by a perfect matching between initial nodes and target nodes. The matching uses the fact that the robots are *identical*. This is not an offline matching problem, but closer to the online matching. However, this problem is different from the online matching in that a robot does not know how other robots are matched to the requests. From now, we call the problem 'distributed online matching'. In the online matching, it is proven that the lower bound of the competitive ratio of any deterministic algorithm is *(2k-1)*. The permutation algorithm [1] achieves this bound.

In our work, a distributed matching algorithm is devised, using a greedy and the permutation algorithm. We prove that it has $\frac{1}{2}(3k^2 - k)$-competitive ratio for the case where each robot is deployed one by one, and $(2k^2 - k)$-competitive ratio for the simultaneous deployment. We also give a discussion about randomization, and conjecture that the randomization might not be helpful.

1.3 Clarification of the problem

We clarify our problem as follows:

- The truss structure is modeled by an undirected graph *G,* where nodes are points where a robot can grasp and there is a positive cost on each edge.
- Naturally, triangular inequality holds for *G*
- We model each robot as a point on the graph *G*
- The number of the robots is *k*, and all robots are *identical*
- The sensing range of robots is assumed to be one, i.e. a robot can communicate with other robot if and only if they are adjacent.
- When the communication is possible, the robots can share all information they have.
- Initially, each robot is located in r_i, and *R* is a set of all the initial nodes.

- All the target nodes are given to each robot. T is a set of the target nodes. t_j denotes each target node. Note that j is not related with index of robots. T can be overlapped with R.
- The goal is for the robots to reach all target nodes with minimum cost.

This paper is organized as follows: Section 2 describes the best offline algorithm and how it can be viewed as the matching problem. In Section 3, we propose the distributed online matching algorithm and show how it works. We give discussions about the bound in section 4. We will give some modification to the algorithm at Section 5 to improve the practicality of the algorithm.

2. Optimal offline algorithm

In this section, we discuss about the optimum solution of the centralized offline case where a central controller knows everything on the graph and control each robot. We also show that we can solve the problem with a perfect matching algorithm.

2.1 Optimal solution for the centralized offline case

This problem can be reduced to a min-cost disjoint path problem with vertex capacities since at any time each vertex can be occupied by only one robot. We solve this problem as follows:

- Expand the graph G to reflect the vertex capacities (as we solved in the problem set). Note that vertex capacity is one for all vertices.
- Find disjoint paths of the expanded graph. In our case, we should use min-cost max flow rather than just max flow algorithm. This yields the min cost paths without any collision.

2.2 Can this be solved as a matching problem?

If a robot does not have its physical size, any perfect matching between R and T will give a feasible solution, and each robot can move on the shortest path to its matched target node. Moreover, the min-cost perfect matching is, of course, the optimum solution. However, in a real system, this set of the shortest paths may raise two critical problems: the collision where robots' path cross and the deadlock where paths form a cycle in the graph. Now we propose an intuitive selection criterion to avoid these two issues.

2.2.1 Selection criteria: *to exchange identities*

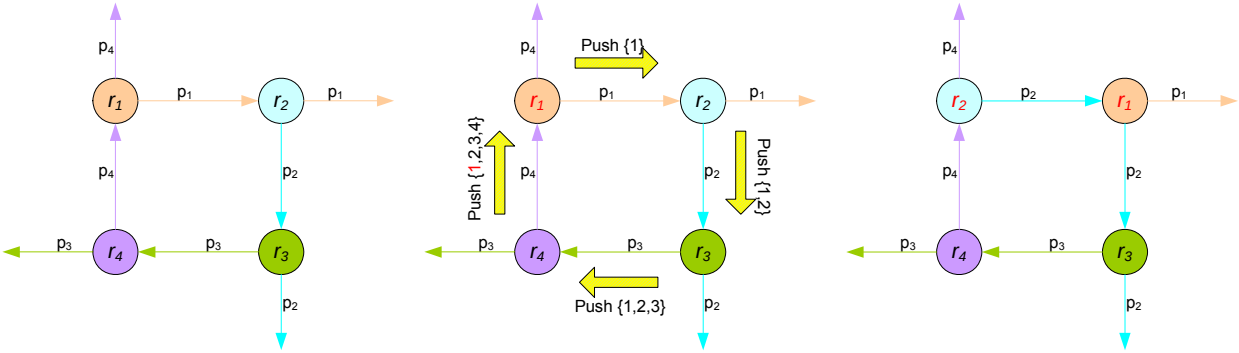
When two paths cross, simply changing identities, which means to exchange all information they have and the action they are executing now, will prevent the collision as shown in Fig. 3. On the other case when adjacent robots shares the same edge but their paths do not cross, let the higher ID robots go first and the other robots wait until robots with the higher ID pass the edge.

(a) (b)

Fig. 3 exchanging identities (a) two paths are crossing (b) after exchanging IDs

2.2.2 Selection criteria: *to break a deadlock*

A deadlock is the status in which some robots can not move even though any paths of the robots are not crossed as shown in Fig. 4. There are four robots $\{r_1, r_2, r_3, r_4\}$ with each path $\{p_1, p_2, p_3, p_4\}$, which form a rectangular cycle in Fig. 4(a). To prevent this deadlock, we introduce a communication protocol *push*. A robot send a *push* with its ID to another robot which is blocking its way, and it also includes IDs of the robots which are pushing the robot behind. Note that in a cycle, every robot can communicate each other since they are adjacent by definition of the cycle. If a robot finds that it is pushing itself as shown in Fig. 4(b), it is in a cycle. In that case, a robot with the lowest ID forces the blocking robot to exchange IDs until a robot in the *push* protocol does not block it anymore. After this forced exchange, the cycle will be broken as shown in Fig. 4(c). Therefore, we do not need to worry about the deadlock with this selection criterion.



(a) (b) (c)

Fig. 4 How to break a deadlock

(a) a deadlock (a cycle) happens (b) communication protocol for preventing the deadlock: *push*

(c) the cycle is broken by exchanging identities.

The last question is whether the process will be terminated. The answer is definitely yes. Whenever at least one robot is moving, the total distance between the robots and the target nodes is strictly decreasing because there is no deadlock. Note that the selection criteria never increase the distance. Therefore, if each robot has a *distinct* target node, which means a perfect matching, they will eventually reach one of them. In a distributed system, as we will see soon, more than two robot can initially have the same target node. We will make each robot have the distinct target in the next section.

3. Distributed matching algorithm

Here we propose our algorithm: a distributed greedy permutation in which each robot goes to the nearest target node and when there is collision, the collided robots are reconfigured by spreading them into the local optimal vertices. We show that the algorithm has $O(k^2)$ competitive ratio for both sequential and simultaneous deployment of the robots.

3.1 A simple greedy algorithm is bad

Let's consider an intuitive greedy algorithm where each robot successively finds the nearest target node. A robot chooses the nearest target from its current position as the first target node. If the target node has been occupied, it will choose the nearest target node from that target node. Now, see Fig.5 where *(k-1)* robots have occupied *(k-1)* target nodes and only one robot is finding its target by the greedy algorithm. If the left edge has $(1+\varepsilon)$ cost, where ε is a positive number, the robot will continuously go to the right node and finally return to t_k after visiting all the right nodes. The cost is (2^k-1), while the optimal cost is only $(1+\varepsilon)$. Therefore, in this case, the competitive ratio is (2^k-1) when ε is small. In the appendix, we show that (2^k-1) is the tight upper bound of the simple greedy algorithm.

Fig.5 A bad example for a simple greedy algorithm

3.2 Basic concepts

We start with an important lemma in [1] about the minimal weight partial matching M_i which is defined as the set of edges that form the minimal weight partial perfect matching between the subset $\{r_1, r_2, \cdots, r_i\}$ and subset of T with a minimal number of edges in M_i- M_{i-1}. Define T_i as the subset of T, vertices of T which are incident on M_i.

Lemma 3.1 *The cost of the M_i 's form a monotonically non-decreasing sequence.*

Lemma 3.2 *For each i, the set difference T_i-T_{i-1} contains exactly one vertex.*

The proof of two lemmas are given in [1]

We extend this partial matching to distributed case. Let R_j be the subset of R, consisted of robots which have already collided at some target node with at least one other robot in the group. Let M_{R_j} be the min-cost matching of the sub-group R_j. Note that M_{R_j} is sequentially constructed as a new robot collides with some robot of R_j. Therefore, Lemma 3.1 and 3.2 hold for M_{R_j}. Now, we prove an important lemma about local min-cost matching of two disjoints subsets of R.

Lemma 3.3 $\forall R_i, R_j \ (i \neq j, R_i \bigcap R_j = \phi),\ M_{R_i+R_j} = M_{R_i} + M_{R_j}$

Proof. Assume that there exists $M_{R_i+R_j}$ which has less cost than $M_{R_i} + M_{R_j}$. Then we can find the better partial matching than M_{R_i} and M_{R_j} by making subsets of edges which are only incident on R_i and R_j. This is a contradiction. Therefore the cost of $M_{R_i} + M_{R_j}$ is the same as $M_{R_i} + M_{R_j}$. Besides, when we add M_{R_j} to M_{R_i}, augmented number of the edges are minimal by the definition of M_{R_j}.

3.3 Distributed matching by imitating partial optimal matching and using communication

For simplicity, consider a case where each robot is deployed sequentially, that is, a robot starts to move only after its predecessor reaches its final target node. We will deal with a general case later. The algorithm works as follows:

Algorithm

Let a robot r_i move to its nearest target node

1:

If *the target is empty*

Make a new group R_i

else *(it must have collided with a robot in some group R_j and it may be already a member of another group R_l)*

Get a new target node by comparison with the partial optimal matching $M_{R_j+R_l}$

Let every other robots in $R_l + R_j$ ***know the new robot and its target node by visiting them***

Move r_i to the new target node

End if

2:

Repeat 1 until r_i reaches empty target node

At the start, a robot behaves greedily. If the target node is already occupied, we add the robot to the group to which the collided robot belongs to. Then, we compute a new target node which is incident on the partial optimal matching of the group including the new member and which is not in the target nodes of the old group. There will be exactly one target node to visit according to Lemma 3.2. One thing we need to do is to let every robot in the group know which robots are the members of the group and which their target nodes are. Otherwise, one robot can be a member of multiple groups because each member of a group may have the different information about the group to which it belongs, and that leads branches of the group. We call this process *communication* from now. The other case is that a robot may collide with another group after joining one group. The algorithm works as the same: find a new target node by comparison with $M_{R_j+R_i}$, do *communication* with all members, and go to the new target node. We will provide some modifications at section 5 to decrease this communication

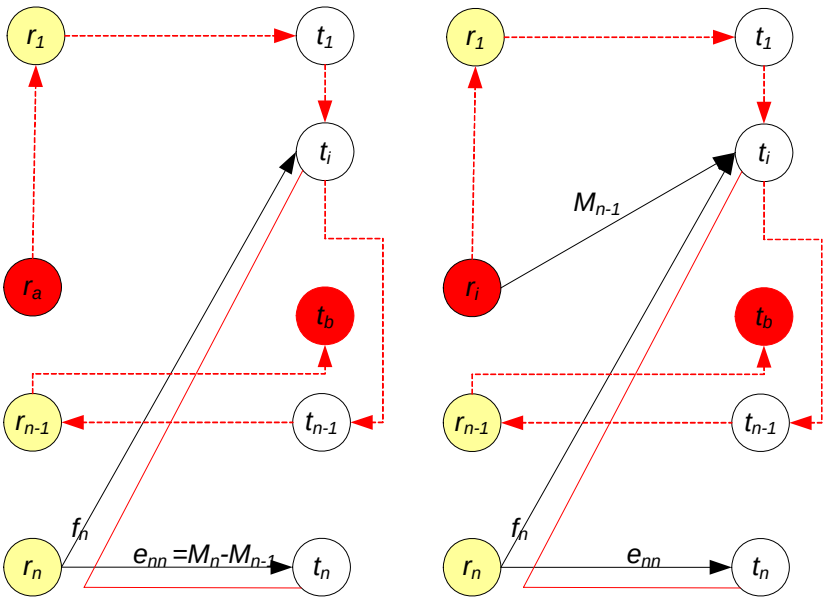
cost, but a simple worst case analysis shows that the competitive ratio is still $O(k^2)$. Also, we present a lower bound for the number of collisions, which is $O(k^2)$ and an empirical proof showing that up to $\frac{k^2}{2}$ collisions happen in the worst case without communication.

Now we show how much the proposed algorithm costs. For the cost of the initial greedy approach, we introduce a set of edges F_{R_j} which is a collection of the nearest edges of R_j.

Lemma 3.4 $\forall R_j, \text{cost}(F_{R_j}) \le \text{cost}(M_{R_j})$

Proof. Assume that total m-robots are in a group R_j. WLOG, we can reorder the robots in order of the deployment. Let $f_i \ (i = 1,..,n)$ be the cost of each robot's first approach edge. Since the cost of the incident edge on r_i in M_{R_j} can not be less than f_i, $\sum^{m} \text{cost}(f_i) \le \text{cost}(M_{R_j})$.

From now we use M as the cost of the set M



(a) (b)

Fig. 6 matching with a new robot in a group: (a) a full traversing path of the robots in the old group (b) a full traversing path of the new robot

Lemma 3.5 *$\forall R_j$, there exists a path which start at any initial node r_i, traverse every initial node and target node at least once, and ends at an arbitrary target node t_i. We call it a full-traversing path. The cost of the path is less than 3 times of sum of all M_{R_j} from 1 robot to current n-robots, which is denoted by $(3n-2)M_{R_j}\big|_{\#robot=n}$ where $M_{R_j}\big|_{\#robot=n}$ means n robots has been deployed in the group.*

Proof. We prove the lemma by induction. When a number of robots in R_j is 1, it is obviously true. Assume the lemma holds for a group of *n-1* robots, the optimum matching of which is denoted by $M_{R_j}\big|_{\#robot=n-1}$. When the *n*-th robot r_n comes in with its final target node t_n (Note that firstly this robot must collide with a robot in R_j), there are two cases in the new optimum $M_{R_j}\big|_{\#robot=n}$: r_n matches t_n (See Fig 6) or some t_m in the old R_j with *n-1* robots (Fig. 7). f_n is the first approach edge to the nearest target node.

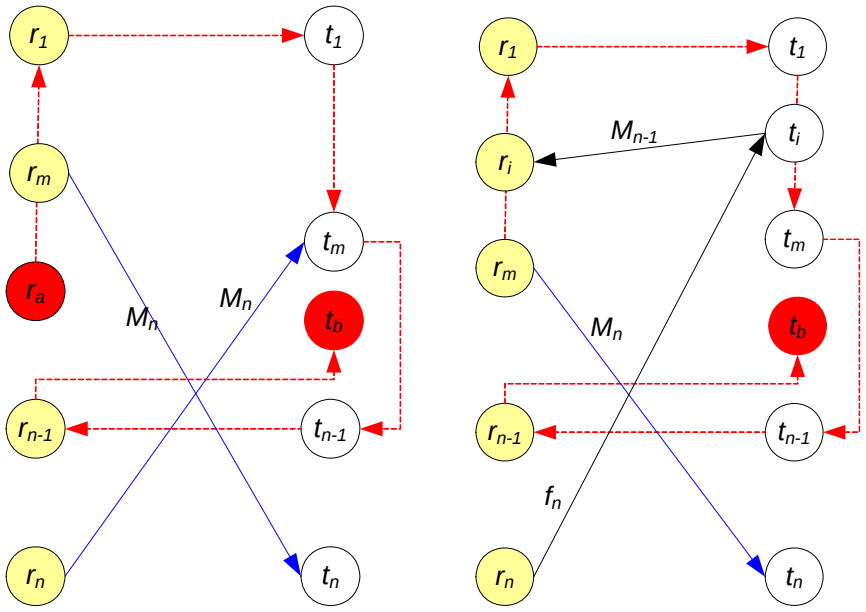


Fig. 7 matching with a new robot in a group: (a) a full traversing path of the robots in the old group (b) a full traversing path of the new robot

Firstly look at the former in Fig. 6(a). Suppose there is a full traversing path from an arbitrary robot to any target node, denoted by r_a and t_b in $R_j\big|_{\#robot=n-1}$, which is the red dotted line in Fig 6(a), we can make a new full traversing path by only adding the reciprocating partial path for the new target node, which are f_n and e_{nn} drawn as a red solid line. Note that $M_{R_j}\big|_{\#robot=n-1} + f_n \le M_{R_j}\big|_{\#robot=n}$, because any matching cost with a new robot should be no less than the nearest edge cost. Also, when r_n matches t_n,

$$e_{nn} = M_{R_j}\big|_{\#robot=n} - M_{R_j}\big|_{\#robot=n-1}$$

, because we do not change any matching in the old R_j but add the new matching from r_n to t_n. Therefore,

$$\begin{aligned} 2f_n + 2e_{nn} &\le 2(M_{R_j}\big|_{\#robot=n} - M_{R_j}\big|_{\#robot=n-1}) + 2(M_{R_j}\big|_{\#robot=n} - M_{R_j}\big|_{\#robot=n-1})) \\ &\le 4(M_{R_j}\big|_{\#robot=n} - M_{R_j}\big|_{\#robot=n-1}) \end{aligned}$$

.

On the other hand, for a full-traversing path from r_n, which was not $R_j\big|_{\#robot=n-1}$, we use r_i, the former match of

t_i in $M_{R_j}\big|_{\#robot=n-1}$ which is the target node directed by f_n . We can make a full-traversing path from r_i to any target node in $\{t_1, t_2, \cdots, t_{n-1}\}$ by the assumption. We can easily add t_n to the path by using e_{nn}. In this case, we need the following path:

$$f_n + 2e_{nn} + \{r_i \to t_i\} \le M_{R_j}\big|_{\#robot=n} - M_{R_j}\big|_{\#robot=n-1} + 2(M_{R_j}\big|_{\#robot=n} - M_{R_j}\big|_{\#robot=n-1}) + M_{R_j}\big|_{\#robot=n-1}$$
$$\le 3M_{R_j}\big|_{\#robot=n} - 2M_{R_j}\big|_{\#robot=n-1}$$

Secondly, when t_n matches another robot r_m in $M_{R_j}\big|_{\#robot=n}$, r_n must match one of the target nodes, t_m, in the old group as shown in Fig 7. As we did, we can make a new full traversal path from any r_a and t_b in $R_j\big|_{\#robot=n-1}$ by adding $\{r_m \to t_n\}$ and $\{r_n \to t_m\}$ two times in the new optimal matching in $R_j\big|_{\#robot=n}$.

$$2(\{r_m \to t_n\} + \{r_n \to t_m\}) \le 2M_{R_j}\big|_{\#robot=n}$$

If we start at r_n, we use the full traversal path from r_i , which was matched by the target t_i in the old group. and add the reciprocal path of $\{r_m \to t_n\}$. That costs:

$$f_n + \{r_i \to t_i\} + 2\{r_m \to t_n\} \le M_{R_j}\big|_{\#robot=n} - M_{R_j}\big|_{\#robot=n-1} + M_{R_j}\big|_{\#robot=n-1} + 2M_{R_j}\big|_{\#robot=n}$$
$$\le 3M_{R_j}\big|_{\#robot=n}$$

Therefore, the worst adding cost among all cases in M_{R_j} is bounded by $3M_{R_j}\big|_{\#robot=n}$, which directs:

$$P_n \le P_{n-1} + 3M_{R_j}\big|_{\#robot=n}$$
$$P_n \le [3(n-1)-2]M_{R_j}\big|_{\#robot=n-1} + 3M_{R_j}\big|_{\#robot=n}$$
$$\le [3(n-1)-2]M_{R_j}\big|_{\#robot=n} + 3M_{R_j}\big|_{\#robot=n}$$
$$= (3n-2)M_{R_j}$$

Lemma 3.6 *The cost of constructing a group* $R_j\big|_{\#robot=n}$ *by sequentially adding one robot is bounded by* $\frac{1}{2}(3n^2 - n)M_{R_j}\big|_{\#robot=n}$

Proof. Examine each step of adding a new robot to the group. By the proposed algorithm, a robot r_i start moving with finding the nearest target node which belongs to the target nodes of $R_j\big|_{\#robot=i-1}$. Then it has to traverse all the other members in $R_j\big|_{\#robot=i-1}$ for *communication* and finally reaches the final target point. Note that this is a path which starts from r_i, visits all the target nodes of $R_j\big|_{\#robot=i-1}$, and arrives t_i. By Lemma3.5, we can find the path with the bounded cost. Therefore, the whole cost is:

$$
\begin{aligned}
COST &\le \sum_{\#robot}^{n} (3i-2) M_{R_j}\big|_{\#robot=i} \\
&\le \sum_{\#robot}^{n} (3i-2) M_{R_j}\big|_{\#robot=n} \\
&\le M_{R_j}\big|_{\#robot=n} \sum_{\#robot}^{n} (3i-2) \\
&\le (\frac{3}{2}n^2 - \frac{1}{2}n) M_{R_j}
\end{aligned}
$$

Lemma 3.7 *When merging two groups of* $R_j\big|_{\#robot=n}$ *and* $R_l\big|_{\#robot=m}$ *into a new group* $R_j\big|_{\#robot=m+n}$, *where both groups are only constructed by adding a single robot. The whole cost including the merge and the construction of two groups are bounded by* $\frac{1}{2}[3(n+m)^2-(n+m)]M_{R_j+R_l}$.

Proof. WLOG, assume that the last robot of R_l hits a robot in R_j. Then the cost of only merging is to traverse all robots in $R_l + R_j$ from the hit point to the final target point in $M_{R_j+R_l}$. Note that only r_m needs to reconfigure itself by Lemma3.3. Therefore the merging cost is bounded by $[3(n+m)-2]M_{R_j+R_l}$, and total cost including the constructions is:

$$
\begin{aligned}
COST &\le \frac{1}{2}(3n^2 - n) M_{R_j}\big|_{\#robot=n} + \frac{1}{2}(3m^2 - m) M_{R_l}\big|_{\#robot=m} + [3(n+m)-2] M_{R_j+R_l} \\
&\le \frac{1}{2}(3n^2 - n) M_{R_j}\big|_{\#robot=n} + \frac{1}{2}(3m^2 - m)(M_{R_j+R_l} - M_{R_j}\big|_{\#robot=n}) + [3(n+m)-2] M_{R_j+R_l} \\
&\le [\frac{1}{2}(3m^2 - m) + 3(n+m) - 2] M_{R_j+R_l}
\end{aligned}
$$

We used 3.3 and assumed $m \ge n$ WLOG. And,

$$
\begin{aligned}
&\frac{1}{2}[3(n+m)^2-(n+m)] - [\frac{1}{2}(3m^2 - m) + 3(n+m) - 2] \\
&= \frac{1}{2}(3n^2 + 6nm - 7n - 6m + 4) \\
&= \frac{1}{2}[3(n^2 - 2n + 1) + 6nm - 6m - n + 1] \\
&= \frac{1}{2}[3(n-1)^2 + (6m-1)(n-1)] \ge 0
\end{aligned}
$$

Therefore, the Lemma is true.

Lemma 3.8 *When merging two groups of* $R_j\big|_{\#robot=n}$ *and* $R_l\big|_{\#robot=m}$ *into a new group* $R_j\big|_{\#robot=m+n}$. *The whole cost including the merge and the construction of two groups are bounded by* $[3(n+m)-2]M_{R_j+R_l}$.

Proof. By Lemma3.7, we find that the construction cost including the merge is the same as making a group by sequentially adding a single robot. Therefore, Proof for Lemma3.7 holds for the general merge.

Lemma 3.9 ***The proposed distributed algorithm for the sequential deployment has*** $\frac{1}{2}(3k^2-k)$ ***competitive ratio.***

Proof. After all robots are deployed to the target nodes by the proposed algorithm, there may exist a several independent (non-collided) groups. The worst case is the only one group remains because the total construction cost of merging is always bigger than the separated construction cost of multi groups by Lemma3.7. At the last merge, total number of robots must be *k*, therefore, total cost is no greater than $\frac{1}{2}(3k^2-k)M_R$. Note that $\text{cost}(M_R)$ is the optimum.

Lemma 3.10 *The proposed distributed algorithm for the simultaneous deployment gives* $(2k^2-k)$ *competitive ratio.*

Proof. Unlike the online matching, the deployment of robots takes a physical time. Therefore, the simultaneous deployment may cause a case that a robot collides with a group which is reconfiguring now. To eliminate this confusion, we set a primary robot in each group which firstly constructed the group. When a new robot collides with any robot of the group, we force it to visit the primary robot first, then let it do the same process – to visit all target nodes in the group and to settle in the new target node - as a robot does in the sequential deployment. Now we can handle the reconfigurations in order of visiting the primary robots. Note that we can always reconfigure one by one in order of visiting the primary robot, by Lemma3.2. This visiting is shown to take another $f_n + M_{R_j}\big|_{\#robot=n-1} \le M_{R_j}\big|_{\#robot=n}$ by induction like in Fig. 6. Therefore the bound in Lemma 3.5 is modified into $(4n-3)M_{R_j}$. And this consequentially yields total $(2k^2-k)M_{R_j}$.

4. Discussions

4.1 Is the bound tight?

In our problem, the communication cost is dominant. If it is not necessary, that is, a group member can share the information after the first collision by any method, we achieve *O(k)* competitive ratio as the online matching does. On the other hand, if there exists a way to reduce the cost of the *communication* or a good algorithm without the communication although it makes the problem more hard to analyze, we may reduce the order of the competitive ratio. Anyhow, it seems hard to prove the tight bound while the online matching is easily proven to have *(2k-1)* bound for any deterministic algorithm [1]. We provide a proof that this bound may be tight in the next section.

4.2 O(k^2) collisions without communication

We suspected that the number of collision between robots should be decreased to reduce the cost. Without communication, robot only shares information when there is collision. We will prove a lower bound on the worst case number of collisions first. It was hard to analyze the number of collisions between robots in the worst case,

so we implemented the brute-force search algorithm to see the empirical result.

First, we prove that at least $O(k^2)$ collisions happens in the worst case. Let's consider the sequential deployment case. When a new robot collides with a settled one, they share the information so that the new robot won't visit a target node where it knows some robot is occupying. See Fig. 8. We can represent a robot as a node and information as a link and a number. The number on the robot node represents the number of other robots' location that the robot knows. The link represent that the two robots connected by a link collided directly once. The robots at higher position in the graph knows less than the ones below. In a bad case, a new robot collides with the robot with the least information and consecutively collides with the robots that it has no information about. The robot doesn't know about the robot which is 2 nodes away. So, when there are k robots settled down and a new robot is deployed, there can be $\frac{k}{2}$ collisions for even k, and $\frac{k+1}{2}$ collisions for odd k. Therefore, if we deploy k robots sequentially, we get $O(k^2)$ collisions. Also, a brute-force search for the worst case gave us $\left(\frac{k}{2}\right)^2$ collisions for k robots.

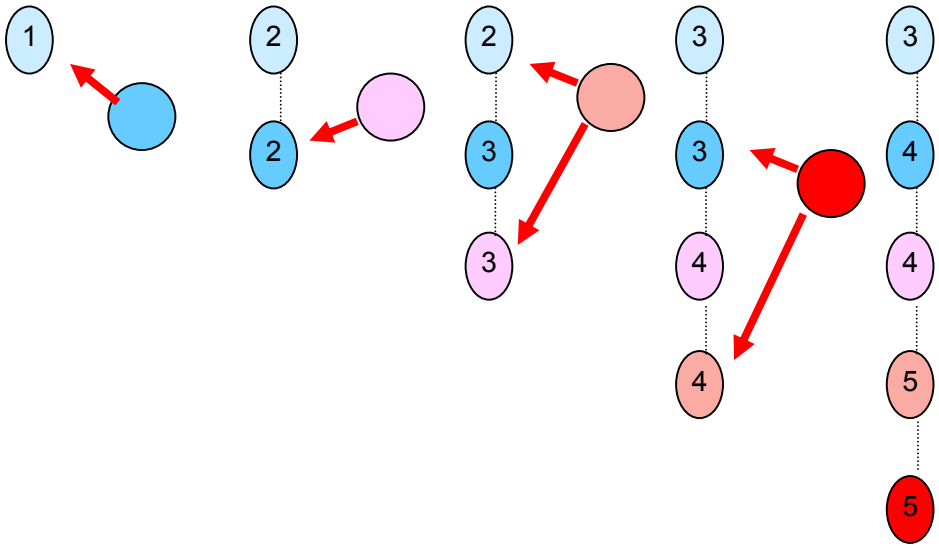


Fig. 8 collisions between robots for the sequential deployment of robots: It divides into two cases where the number of robots is odd, and even.

4.3 Could randomized algorithm help?

As far as we have searched, there are only a few results on the randomized online matching. The best competitive ratio of the randomized one[2] is $O(\log^3 k)$ while a deterministic is $O(k)$. They use a special graph, HST tree, rather than a generic graph. And it is shown that a generic graph can be modified into a HST tree. We did not try this approach to our problem. However, we conjecture that a randomized algorithm would not work well in our case because it is much harder to narrow the probability – the choices. For instance, for a uniform metric graph where nodes are fully connected and every edge cost is 1, the proposed randomized online

matching algorithm - which finds the nearest one and if there is tie (multi nodes with the same cost), it randomly select one – gives *O(logk)* expected competitive ratio while a greedy one does *O(k)*. However, in our case, the same algorithm also yields *O(k)*, which is the same order as that of a greedy one. Although one example can not tell everything, we guess this is a hint that a randomized algorithm might not work better mainly because of the required *communication-equivalent-cost.*

5. Improving the practicality of the algorithm

We came up with two modifications to decrease the number of collisions, which can be used as a combination as well. However, a simple worst case analysis shows that the methods don't improve the competitive ratio more than a constant factor improvement, though it will definitely work better in practice.

5.1 Communication only after a new robot finds the empty node

The proposed algorithm given at section 3.3 uses communication everytime when there is a collision between a new robot and a group. It leads to multiple communications if the new robot collides with another group again. We can easily modify this so that there will be communication only after the new robot finds its place. In this case, the robot will merge several groups at one time. It surely decreases the communication cost, the number of collisions between robots. However, the competitive ratio is still $O(k^2)$.

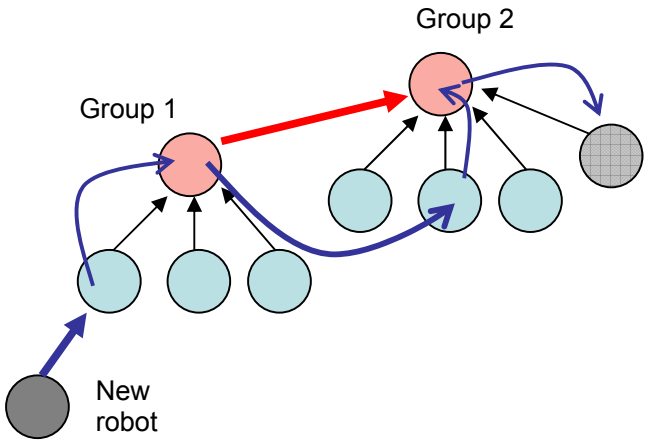


Fig. 9 a traversal of a new robot

5.2 Using a primary robot for each group and union-set data structure

The other method is to use a primary robot for each group so that a new robot collides with the group only needs to communicate with the primary robot. We will also use the union-set data structure to manage the merges between groups. Now, each group is a set with a primary robot as the set-representative. If a new robot collides with a member of a group, it will go to the primary robot of the group and get a target node. The primary robot knows the group members and its target nodes so that it can calculate the partial optimal matching. When groups are merged, the primary robot of one group will send new robots to the other group's primary robot in the future. It's similar to union-set data structure.

Fig. 9 shows a possible traversal of a new robot when there are two groups of robots (each composed of 4 members and 1 primary robot). It collides with a robot in the first group and visits the primary robot. The primary robot sends the new robot to a target node, which is actually occupied by a robot in the second group. It goes to the primary robot of the second group. Now, the two groups need to be merged. It can be done by setting the primary robot in one group to send new robots to other group's primary robot in the future or actually updating the links of each robot. We didn't analyze the costs fully. However, a simple analysis showed that it doesn't improve the competitive ratio since the path to the primary robot could include all the members in the group so that the number of collision will be similar.

6. Conclusion

In our paper, we propose a new distributed matching algorithm for path-planning algorithm of mobile robots. We show that this problem can be seen as a perfect matching by some selection criteria. It is proven that the algorithm has $O(k^2)$ competitive ratio while a simple greedy gives an exponential one.

References

[1] B. Kalayanasundaram and K. Pruhs. Online Weighted Matching, *Journal of Algorithms 14(3)*, 1993

[2] A. Meyerson, A. Nanavati, and L. Poplawski, Randomized Online Algorithms for Minimum Metric Bipartite Matching, *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006

Appendix. A simple greedy algorithm for our problem.

This algorithm works as a robot continuously find the nearest neighbor target node.

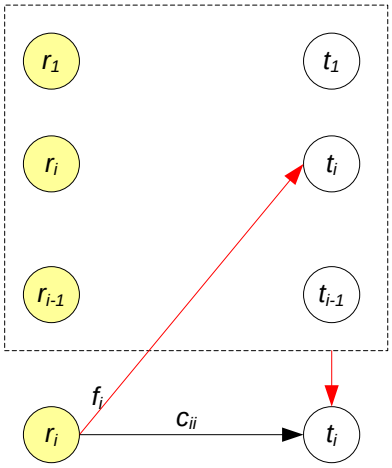


Fig.10 a simple greedy algorithm

Lemma. A simple greedy has 2^k-*1* competitive ratio.

Proof. By induction with a number of the robots. When *#robot*=1, it is obvious. Say that this holds for *i-1* robots.

That is, $SG_{i-1} \le (2^{i-1}-1)M_{i-1}$

There are two cases: If ri does not collide anything after its deployment, then

$$\begin{aligned} SG_i &\le SG_{i-1} + c_{ii} \\ &\le (2^{i-1}-1)M_{i-1} + M_i \\ &\le 2^{i-1}M_i \\ &< (2^i\text{-}1)M_i \end{aligned}$$

In the second case, the maximum number of collision is i-1. The total adding cost to SG_{i-1} is bounded as follows:

$$\begin{aligned} \text{cost} &\le c_{ii} + (c_{ii} + f_1) + 2(c_{ii} + f_1) + \cdots + 2^{i-3}(c_{ii} + f_1) \\ &\le 2^{i-2}(c_{ii} + f_1) \end{aligned}$$

This cost is maximized when $f_1 = c_{ii}$, and in that case $f_1 = c_{ii} \le \text{cost}(M_i)$.

Therefore, the induction holds for *i* as follows:

$$\begin{aligned} SG_i &\le SG_{i-1} + 2^{i-1}M_i \\ &\le (2^{i-1}-1)M_{i-1} + 2^{i-1}M_i \\ &\le (2^i\text{-}1)M_i \end{aligned}$$