Noname manuscript No.
(will be inserted by the editor)

Distributed Assignment with Limited Communication for Multi-Robot Multi-Target Tracking

Yoonchang Sung · Ashish Kumar Budhiraja · Ryan K. Williams · Pratap Tokekar

Received: date / Accepted: date

Abstract We study the problem of tracking multiple moving targets using a team of mobile robots. Each robot has a set of motion primitives to choose from in order to collectively maximize the number of targets tracked or the total quality of tracking. Our focus is on scenarios where communication is limited and the robots have limited time to share information with their neighbors. As a result, we seek distributed algorithms that can find solutions in bounded amount of time. We present two algorithms: (1) a greedy algorithm that is guaranteed finds a 2–approximation to the optimal (centralized) solution albeit requiring $|R|$ communication rounds in the worst-case, where $|R|$ denotes the number of robots; and (2) a *local* algorithm that finds a $\mathcal{O}\left((1+\epsilon)(1+1/h)\right)$–approximation algorithm in $\mathcal{O}(h \log 1/\epsilon)$ communication rounds. Here, h and ϵ are parameters that allow the user to trade-off the solution quality with communication time. In addition to theoretical results, we present empirical evaluation including comparisons with centralized optimal solutions.

This material is based upon work supported by the National Science Foundation under Grant No. 1637915.

Y. Sung
E-mail: yooncs8@vt.edu

A. K. Budhiraja
E-mail: ashishkb@vt.edu

R. K. Williams
E-mail: rywilli1@vt.edu

P. Tokekar
E-mail: tokekar@vt.edu

Department of Electrical & Computer Engineering, Virginia Tech, 1185 Perry Street, Blacksburg, VA 24061, USA

1 Introduction

We study the problem of assigning robots with limited Field-Of-View (FOV) sensors to track multiple moving targets. Multi-robot multi-target tracking is a well-studied topic in robotics [1–5]. We focus on scenarios where the number of robots is large and solving the problem locally rather than centrally is desirable. The robots may have limited communication range and limited bandwidth. As such, we seek assignment algorithms that rely on local information and only require limited amount of communication with neighboring robots.

Constraints on communication impose challenges for robot coordination as global information may not always be available to all the robots within the network. As a result, it may not be always possible to design algorithms that operate on local information while still ensuring global optimality. Recently, Gharesifard and Smith [6] studied how limited information due to the communication graph topology affects the global performance. Their analysis applies for the case when the robots are allowed only one round of communication with their neighbors. If the robots are allowed multiple rounds of communication, they can propagate the information across the network. Given sufficient rounds of communication, all robots will have access to global information, and therefore can essentially solve the centralized version of the problem. In this paper, we investigate the relationship between the number of communication rounds allowed for the robots and the performance guarantees. We focus on the problem of distributed multi-robot, multi-target assignment for our investigation (Figure 1).

We assume that each robot has a number of motion primitives to choose from. A motion primitive is

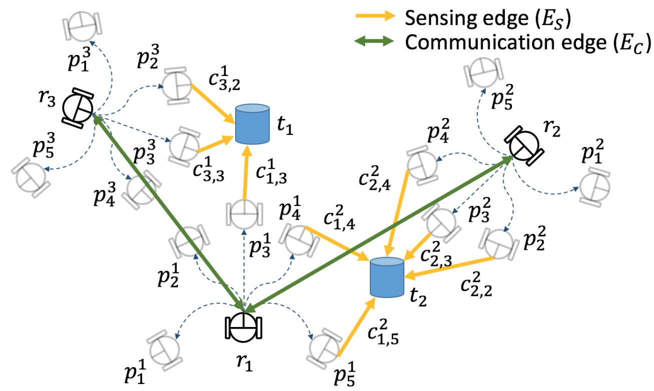


Fig. 1 Description of multi-robot task allocation for multi-target tracking. In this example, each robot has five motion primitives to choose from at each time step.

a local trajectory obtained by applying a sequence of actions [7]. A motion primitive can track a target if the target is in the FOV of the robot. The set of targets tracked by different motion primitives may be different. The assignment of targets to robots is therefore coupled with the selection of motion primitives for each robot. Our goal is to assign motion primitives to the robots so as to track the most number of targets or maximize the quality of tracking. We term this as the distributed Simultaneous Action and Target Assignment (SATA) problem.

This problem can be viewed as the dual of the set cover problem, known as the maximum (weighted) cover [8]. Every motion primitive covers some subset of the targets. Therefore, we would like to pick motion primitives that maximize the number (or weight) of covered targets. However, we have the additional constraint that only one motion primitive per robot can be chosen at each step. This implies that the relationship between a robot and the corresponding motion primitives turns out to be a packing problem [8] where only one motion primitive can be "packed" per robot. The combination of the two aforementioned problems is called a Mixed Packing and Covering Problem (MPCP) [9].

We study two versions of the problem. The first version can be formulated as a (sub)modular maximization problem subject to a partition matroid constraint [10]. A sequential greedy algorithm, where the robots take turns to greedily choose motion primitives, is known to yield a 2–approximation for this problem [11]. We evaluate the empirical performance of this algorithm by comparing it with a centralized (globally optimal solution). The drawback of the sequential greedy algorithm is that it requires at least as many communication rounds as the number of robots. This may be too slow in practice. Consequently, we study a second version of the problem for which we present a *local* algorithm whose performance degrades gracefully (and provably) as a function of the number of communication rounds.

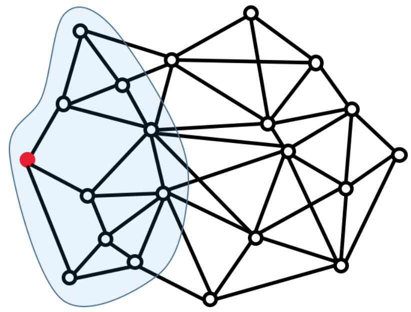


Fig. 2 Communication graph. The blue shaded region indicates a radius–2 neighborhood of the red solid node. The red solid node may be unaware of the entire communication graph topology. A local algorithm that works for the red solid node only requires local information of nodes in the blue shaded region. The same local algorithm runs on all the nodes and ensures bounded approximation guarantees on the global optimality.

A local algorithm [8] is a constant-time distributed algorithm that is independent of the size of a network. This enables a robot only to depend on local inputs in a fixed-radius neighborhood of robots (Figure 2). The robot does not need to know information beyond its local neighborhood, thereby achieving better scalability.

Floréen *et al.* [12] proposed a local algorithm to solve MPCP using max-min/min-max Linear Programming (LP) in a distributed manner. We show how to leverage this algorithm to solve SATA. This algorithm has a bounded communication complexity unlike typical distributed algorithms. Specifically, the algorithm yields a $\mathcal{O}\left((1+\epsilon)(1+1/h)\right)$ approximation to the globally optimal solution in $\mathcal{O}(h \log 1/\epsilon)$ synchronous communication rounds where h and ϵ are input parameters.[1] We verify the theoretical results through empirical evaluation.

The contributions of this paper are as follows:

1. We present two versions of the SATA problem.
2. We show how to use the greedy algorithm and adapt the local algorithm for solving the two versions of the SATA problem.
3. We perform empirical comparisons of the proposed algorithm with baseline centralized solutions.
4. We demonstrate the applicability of the proposed algorithm through Gazebo simulations.

A preliminary version of this paper was presented at ICRA 2018 [13]. This expanded paper extends the

[1] An algorithm is called a $\mathcal{O}(x)$ approximation to a maximization problem if it guarantees a solution whose value is at least $\frac{c}{x}$ of the optimal value, where c is some constant.

preliminary version with a more thorough literature survey, additional theoretical analysis, and significantly expanded empirical analysis including a description of how to implement the greedy algorithm in practice.

The rest of the paper is organized as follows. We begin by introducing the related work in Section 2. We describe the problem setup in Section 3. Our proposed distributed algorithms are presented in Section 4. We present results from representative simulations in Section 5 before concluding with a discussion of future work in Section 6.

2 Related Work

A number of algorithms have been designed to improve multi-robot coordination under limited bandwidth [14–18] and under communication range constraints [19–21]. This includes algorithms that enforce connectivity constraints [22, 23], explicitly trigger when to communicate [24–26] and operate when connectivity is intermittent [27, 28]. In this section, we focus on work that is most closely related to the SATA problem and local algorithms.

2.1 Multi-Robot Target Tracking

There have been many studies on cooperative target tracking in both control and robotics communities. We highlight some of the recent related work in this section. For a more comprehensive overview of multi-robot multi-target tracking, see the recent surveys [29, 30].

Charrow *et al.* [31] proposed approximate representations of the belief to design a control policy for multiple robots to track one mobile target. The proposed scheme, however, requires a centralized approach. Yu *et al.* [32] worked on an auction-based decentralized algorithm for cooperative path planning to track a moving target. Ahmad *et al.* [33] presented a unified method of localizing robots and tracking a target that is scalable with respect to the number of robots. Zhou and Roumeliotis [34] developed an algorithm that finds an optimal trajectory of multiple robots for the active target tracking problem. Capitan *et al.* [35] proposed a decentralized cooperative multi-robot algorithm using auctioned partially observable Markov decision processes. The performance of decentralized data fusion under limited communication was successfully shown but theoretical bounds on communication rounds were not covered. Moreover, theoretical properties presented in the above references may not necessarily to hold in the case of tracking multiple targets in a distributed fashion.

Pimenta *et al.* [36] adopted Voronoi partitioning to develop a distributed multi-target tracking algorithm. However, their objective was to cover an environment coupled with multi-target tracking. Banfi *et al.* [37] addressed the *fairness* issue for cooperative multi-robot multi-target tracking, which is achieving a balanced coverage among different targets. One of the problems that we define in Section 3 (*i.e.*, Problem 1) has a similar motivation. However, unlike the algorithm in [37], we are able to give a global performance guarantee. Xu *et al.* [38] proposed a decentralized algorithm that jointly solves assigning robots to targets and positioning robots using mixed-integer nonlinear programming. Nevertheless, all the aforementioned works do not take into account a bound on the number of communication rounds.

In our prior work [11], we addressed the problem of selecting trajectories for robots that can track the maximum number of targets using a team of robots. However, no bound on the number of communication rounds was studied, possibly resulting in all-to-all communication in the worst case. Instead, in this work we explicitly bound the amount of communication required for target assignment.

2.2 Multi-Robot Task Assignment

Multi-robot task assignment can be formulated as a discrete combinatorial optimization problem. The work by Gerkey and Matarić [39] and the more recent work by Korsah *et al.* [40] contain detailed survey of this problem. There exists distributed algorithms with provable guarantees for different versions of this problem [41–43]. There also exists various multi-robot deployment strategies for task assignment under communication constraints. These constraints include limited available information [44], limited communication flows [45], and connectivity requirement [46]. See the survey papers [47, 48] on these results.

Turpin *et al.* [49] proposed a distributed algorithm that assigns robots to goal locations while generating collision-free trajectories. Morgan *et al.* [50] solved the assignment problem by using distributed auctions and generating collision-free trajectories by using sequential convex programming. Bandyopadhyay *et al.* [51] adopted the Eulerian framework for both swarm formation control and assignment. For more survey results about SATA, see the work by Chung *et al.* [52]. Recently, Otte *et al.* [53] investigated the effect of communication quality on auction-based multi-robot task assignment. None of the above works, however, analyzed the effect of communication rounds on the solution quality, as is the focus of our work.

2.3 Local Algorithms

A local algorithm [54–56] is a distributed algorithm that is guaranteed to achieve desired objective in a finite (typically, fixed) amount of time. The typical approach is to find approximate solutions with provable (and global) performance guarantees while ensuring a bound on the communication complexity that is independent of the number of vertices in the graph. Local algorithms have been proposed for a number of graph-theoretic problems. These include, graph matching [57], vertex cover [58, 59], dominating set [60], and set cover [61]. Suomela [8] gives a broad survey of local algorithms. We build on this work and adapt a local algorithm for solving SATA.

3 Problem Description

Let R and $\hat{T}$ denote the set of robot states and predicted target states, respectively. We solve the SATA problem over a horizon of H timesteps. In the following, we will use k to denote the starting time of a planning horizon. $R(k) = \{\mathbf{r}_1(k), ..., \mathbf{r}_i(k), ..., \mathbf{r}_{|R|}(k)\}$ denotes the state of the robots at time k. $\hat{T}(k) = \{\hat{\mathbf{t}}_1(k), ..., \hat{\mathbf{t}}_j(k), ..., \hat{\mathbf{t}}_{|T|}(k)\}$ denotes the predicted state of the targets at the *end* of this planning horizon (*i.e.*, at time $k + H$). We assume that the targets can be uniquely detected and multiple robots know if they are observing the same target. Therefore, no data association is required. The predicted states, $\hat{T}$, can be obtained by fusing the noisy sensor measurements using, for example, Kalman Filter.

We denote the motion primitives of i-th robot $\mathbf{r}_i(k)$ at time k by $P^i(k) = \{\mathbf{p}^i_1(k), ..., \mathbf{p}^i_m(k), ..., \mathbf{p}^i_{|P^i|}(k)\}$. Note that the term *motion primitives* in this paper represents the future state of a robot at the end of the planning horizon (*i.e.*, at time $k + H$).

We define $\mathcal{RS}(\mathbf{p}^i_m(k))$ to be the set of targets that can be observed by the primitive $\mathbf{p}^i_m(k)$. Specifically, the j-th target is said to be observable by the m-th motion primitive of a robot i, iff $\hat{\mathbf{t}}_j(k) \in \mathcal{RS}(\mathbf{p}^i_m(k))$. Note that since $\mathcal{RS}$ is a set function, we can model complex FOV and sensing range constraints that are not necessarily restricted to 2D.

We make the following assumptions.

Assumption 1. ***(Communication Range).*** *If two robots have a motion primitive that can observe the same target, then these robots can communicate with each other. This implies if there exists a target j such that $\boldsymbol{t}_j(k) \in \mathcal{RS}(\boldsymbol{p}^i_m(k))$ and $\boldsymbol{t}_j(k) \in \mathcal{RS}(\boldsymbol{p}^l_m(k))$, then i-th and l-th robots can communicate with each other.*

Assumption 2. ***(Synchronous Communication).*** *All the robots have synchronous clocks leading to synchronous rounds of communication.*

From Assumption 1, neighboring robots can share their local information with each other when they observe the same targets. For example, robots can use techniques such as covariance intersection [62] to merge their individual predictions of the target's state into a joint prediction $\hat{T}$. This can be achieved in one round of communication when each robot simply broadcasts its own estimate of all the targets within its FOV. Note that a robot does not need to know the prediction for all the targets, only the ones that are within the FOV of one of its motion primitives. In this sense, a communication graph $\mathcal{G}_C = (R, E_C)$ can be created from a sensing graph $\mathcal{G}_S = (P \cup T, E_S)$ at each time, where E_C and E_S denote edges among robots and edges between targets and motion primitives, respectively.

As shown in Figure 1, each robot is able to compute feasible motion primitives of its own and detect multiple unique targets within the FOV. Then, the objective of the proposed problem is to choose one of the motion primitives for each robot, yielding either the best quality of tracking or the maximum number of targets tracked by the robots, depending on the application. One possible quality of tracking can be measured by the summation of all distances between selected primitives and the observed targets.

Let x^i_m be the binary variable which represents the i-th robot selecting the m-th motion primitive. That is, $x^i_m = 1$ if $\mathbf{p}^i_m$ is selected by $\mathbf{r}_i$ and 0 otherwise. Since each robot can choose only one motion primitive, we have:

$$\sum_{\mathbf{p}^i_m \in P^i} x^i_m \leq 1 \quad \forall \mathbf{r}_i \in R. \tag{1}$$

Our objective is to find x^i_m. We propose two following problems.

Problem 1 (Bottleneck). *The objective is to select primitives such that all targets are* equitably *covered:*

$$\underset{x^i_m}{\operatorname{argmax}} \quad \min_{t_j \in T} \left(\sum_{r_i \in R} \sum_{p^i_m \in P^i} c^j_{i,m} x^i_m \right), \tag{2}$$

subject to the constraints in Equation 1. Here, $c^j_{i,m}$ denotes weights on sensing edges E_S between m-th motion primitive of i-th robot and j-th target.

Here, $c^j_{i,m}$ can represent the tracking quality given by, for example, the inverse of the distance between t_j and p^i_m. Alternatively, $c^j_{i,m} \in \{0, 1\}$ making the objective function equal to the number of targets tracked.

This problem is equivalent to maximizing the minimum tracking quality. We term this as the Bottleneck version of SATA. The Bottleneck objective ensures that the resources (i.e., the robots) are diverted equally to all the targets. Multiple robots may be assigned to the same target. We also define a WinnerTakesAll variant of SATA where only one robot is assigned to a target.

We define additional binary decision variable, y_i^j. y_i^j represents the i-th robot assigned to track the j-th target. We have, $y_i^j = 1$ if $\mathbf{r}_i$ is assigned to $\mathbf{t}_j$ and 0 otherwise.

Since we restrict only one robot to be assigned to the target (unlike Bottleneck), we have:

$$\sum_{\mathbf{r}_i \in R} y_i^j \leq 1 \quad \forall \mathbf{t}_j \in T. \tag{3}$$

Problem 2 (WinnerTakesAll). *The objective is to maximize the total quality of tracking given by,*

$$\arg\max_{x_m^i, y_i^j} \sum_{\mathbf{t}_j \in T} \left(\sum_{r_i \in R} y_i^j \left(\sum_{\boldsymbol{p}_m^i \in P^i} c_{i,m}^j x_m^i \right) \right), \tag{4}$$

subject to the constraints in Equations 1 and Equation 3.

Both versions of the SATA problem are NP-Hard [63]. The WinnerTakesAll version can be optimally solved using a Quadratic Mixed Integer Linear Programming (QMILP) in the centralized setting. Our main contributions are to show how to solve both problems in a distributed manner: an LP-relaxation of the Bottleneck variant using a local algorithm; and the WinnerTakesAll variant using a greedy algorithm. The following theorems summarize the main contributions of our work.

Theorem 1 *Let $\triangle_R \geq 2$ be the maximum number of motion primitives per robot and $\triangle_T \geq 2$ be the maximum number of motion primitives that can see a target. There exists a local algorithm that finds an $\triangle_R(1+\epsilon)(1+1/h)(1-1/\triangle_T)$ approximation in $\mathcal{O}(h \log 1/\epsilon)$ synchronous communication rounds for the LP-relaxation of the* Bottleneck *version of SATA problem, where h and $\epsilon > 0$ are parameters.*

The proof follows directly from the existence of the local algorithm described in the next section.

Theorem 2 *There exists a 2–approximation greedy algorithm for the* WinnerTakesAll *version of the SATA problem for any $\epsilon > 0$ in polynomial time.*

This directly follows from the fact that the problem is a modular maximization problem subject to a partition matroid constraint [10]. The algorithms are described in the next section.

4 Distributed Algorithms

We begin by describing the local algorithm that solves Bottleneck.

4.1 Local Algorithm

In this section, we show how to solve the Bottleneck version of the SATA problem using a local algorithm. We adapt the local algorithm for solving max-min LPs given by Floréen *et al.* [12] to solve the SATA problem in a distributed manner.

Consider the tripartite, weighted, and undirected graph, $\mathcal{G} = (R \cup P \cup T, E)$ shown in Figure 3. Each edge $e \in E$ is either $e = \{\mathbf{r}_i, \mathbf{p}_m^i\}$ with weight 1 or $e = \{\mathbf{t}_j, \mathbf{p}_m^i\}$ with weight $c_{i,m}^j \in C$. The maximum degree among robot nodes $\mathbf{r}_i \in R$ is denoted by $\triangle_R$ and among target nodes $\mathbf{t}_j \in T$ is $\triangle_T$. Each motion primitive $\mathbf{p}_m^i \in P$ is associated with a variable x_m^i. The upper part of $\mathcal{G}$ in Figure 3 is related with a packing problem (Equation (4)). The lower part is related with the covering problem.

Lemma 1 *The* Bottleneck *version (Equation (2)) can be rewritten as a linear relaxation of ILP:*

$$\begin{aligned} &\textit{maximize} && w \\ &\textit{subject to} && \sum_{\boldsymbol{p}_m^i \in P^i} x_m^i \leq 1 \quad \forall \boldsymbol{r}_i \in R \\ & && \sum_{r_i \in R} \sum_{\boldsymbol{p}_m^i \in P^i} c_{i,m}^j x_m^i \geq w \quad \forall \mathbf{t}_j \in T \\ & && x_m^i \geq 0 \quad \forall \boldsymbol{p}_m^i \in P^i. \end{aligned} \tag{5}$$

The proof is given in Appendix A.

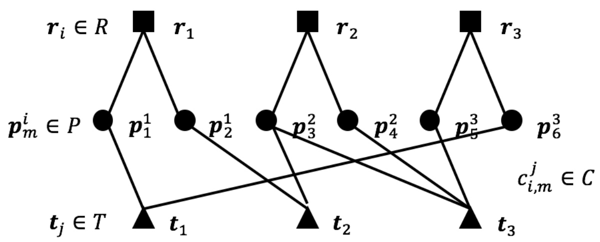


Fig. 3 One instance of a graph for MPCP when there are three robot nodes, six motion primitive nodes and three target nodes.

Floréen *et al.* [12] presented a local algorithm to solve MPCP in Equation 5 in a distributed fashion. They presented both positive and negative results for MPCP. We show how to adopt this algorithm for solving the Bottleneck version of SATA.

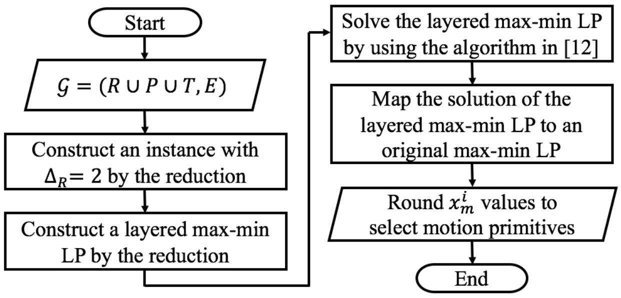


Fig. 4 Flowchart of the proposed local algorithm.

An overview of our algorithm is given in Figure 4. We describe the main steps in the following.

4.1.1 Local Algorithm from Reference [12]

The local algorithm in Reference [12] requires $\triangle_R = 2$. However, they also present a simple local technique to split nodes in the original graph with $\triangle_R > 2$ into multiple nodes making $\triangle_R = 2$. Then, a *layered* max-min LP is constructed with h layers, as shown in Figure 5. h is a user-defined parameter that allows to trade-off computational time with optimality. Layered graph breaks the symmetry that inherently exists in an original graph. The details of the construction of the layered graph is given in Section 4 of Reference [12].

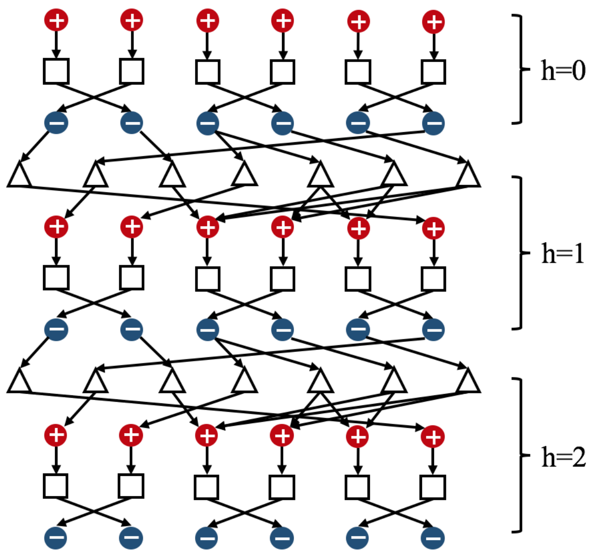


Fig. 5 Graph of the layered max-min LP with $h = 2$ that is obtained from the original graph of Figure 3 after applying the local algorithm. Each motion primitive $\mathbf{p}_m^i \in P$ is colored either red or blue to break the symmetry of the original graph.

They proposed a recursive algorithm to compute a solution of the layered max-min LP. The solution for the original max-min LP can be obtained by mapping from the solution of the layered one. The obtained solution corresponds to values of x_m^i. They prove that the resulting algorithm gives a constant-factor approximation ratio.

Theorem 3 *There exists local approximation algorithms for max-min and min-max LPs with the approximation ratio $\triangle_R(1+\epsilon)(1+1/h)(1-1/\triangle_T)$ for any $\triangle_R \geq 2$, $\triangle_T \geq 2$, and $\epsilon > 0$, where h denotes the number of layers.*

Proof. Please refer to Corollary 4.7 from Reference [12] for a proof. □

Note that each node in the layered graph carries out its local computation. Each node also receives and sends information from and to neighbors at each synchronous communication round. Constructing the layered graph is done in a local fashion without requiring any single robot to know the entire graph.

4.1.2 Realization of Local Algorithm for SATA

To apply the local algorithm of Section 4.1.1 to a distributed SATA problem, each node and edge in a layered graph must be realized at each time step. In our case, the only computational units are the robots. Nodes that correspond to motion primitives, $\mathbf{p}_m^i \in P$, can be realized by the corresponding robot $\mathbf{r}_i \in R$. Moreover, nodes corresponding to the targets must also be realized by robots. A target $\mathbf{t}_j$ is realized by a robot $\mathbf{r}_i$ satisfying $\mathbf{t}_j \in \mathcal{RS}(\mathbf{p}_m^i)$. If there are multiple robots whose motion primitives can sense the target, they can arbitrarily decide which amongst them realizes the target nodes in a constant number of communication rounds.

After applying the local algorithm of Section 4.1.1 to robots, each robot obtains x_m^i on corresponding $\mathbf{p}_m^i$ at each time. However, due to the LP relaxation, x_m^i will not necessarily be binary, as in Equation 1. For each robot we set the highest x_m^i equal to one and all others as zero. We shortly show that the resulting solution after rounding is still close to optimal in practice. Furthermore, increasing the parameter h finds solutions that are close to binary.

The following pseudo-code explains the overall scheme of each robot for a distributed SATA. We solve the SATA problem at each time step. In principle, we can replace each motion primitive with a longer horizon trajectory and plan for multiple timesteps without affecting the computation time significantly.

4.1.3 Advantages of the Local Algorithm

It is possible that there are some robots that are isolated from the others. That is, the communication graph

Algorithm 1: Local algorithm

for $r_{i,k} \in R_k$ **do**
  $\mathbf{p}^i_{m,k} \in P^i_k \leftarrow$ComputeMotionPrimitives$(\mathbf{r}_{i,k})$.
  Find targets that can be sensed by $\mathbf{p}^i_{m,k}$.
  Construct a h-hop communication graph.
  Apply local algorithm.
  $\hat{x}^i_m \leftarrow \text{Rounding}(x^i_m)$.
  $\mathbf{p}^{i*}_m \leftarrow$ Motion Primitive with $\hat{x}^i_m = 1$.
  ApplyAction$(\mathbf{p}^{i*}_m)$.
  $k \leftarrow k + 1$.
end

or the layered graph may be disconnected. However, each component of the graph can run the local algorithm independently without affecting the solution quality. Furthermore, if a robot is disconnected from the rest, then it can take a greedy approach as described in Reference [11] before they reach any other robots to communicate.

The algorithm also allows for the number of robots and targets to change over time. Since each robot determines its neighbors at each time step, any new robots or targets will be identified and become part of the time-varying local layered graphs. The robots can also be anonymous (as long as they can break the symmetry to determine which robot, amongst a set, will realize the target node, when multiple robots can observe the same target.)

The number of layers, h, directly affects the solution quality and can be set by the user. Increasing h results in better solutions at the expense of more communication. $h = 0$ is equivalent to the greedy approach where no robots communicate with each other.

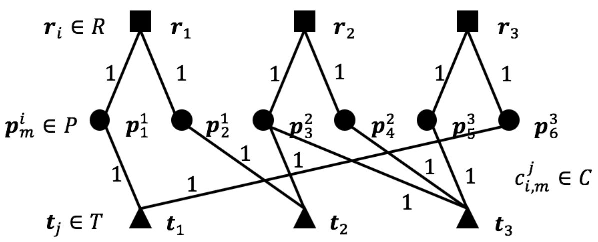


Fig. 6 Same graph of Figure 3 is used with addition of weight values for proof of concept.

The following table shows the result of applying the local algorithm to the graph in Figure 6. Three different values for h were tested: 2, 10, and 30. In all cases, $\mathbf{p}^2_3$ and $\mathbf{p}^3_6$ have larger values of x_p than other nodes. Thus, the robot $\mathbf{r}_2$ and the robot $\mathbf{r}_3$ will select $\mathbf{p}^2_3$ and $\mathbf{p}^3_6$ as motion primitives, respectively after employing a rounding technique to x_p's.

As the number of layers increases, the more distinct the x^i_p values returned by the algorithm. Another interesting observation is that robot $\mathbf{r}_1$ has the same equal value on both motion primitives of its own no matter how many number of layers is used. This is because all the targets are already observed by robots $\mathbf{r}_2$ and $\mathbf{r}_3$ with higher values.

$\mathbf{p}^i_m$	x^i_m	$h = 2$	$h = 10$	$h = 30$
$\mathbf{p}^1_1$	$x^1_1 =$	0.5000	0.5000	0.5000
$\mathbf{p}^1_2$	$x^1_2 =$	0.5000	0.5000	0.5000
$\mathbf{p}^2_3$	$x^2_3 =$	0.6667	0.7591	0.7855
$\mathbf{p}^2_4$	$x^2_4 =$	0.3333	0.2409	0.2145
$\mathbf{p}^3_5$	$x^3_5 =$	0.3333	0.2409	0.2145
$\mathbf{p}^3_6$	$x^3_6 =$	0.6667	0.7591	0.7855

Table 1 Solution returned by the local algorithm for the example shown in Figure 6 with the varying number of layers, h.

4.2 Greedy Algorithm

The greedy algorithm requires a specific ordering of the robots given in advance. The first robot greedily chooses a motion primitive that can maximize the number of targets being observed. Those observed targets are removed from the consideration. Then, the second robot makes its choice; this repeats for the rest of robots. Note again that the greedy algorithm is for the WinnerTakesAll version of SATA.

Algorithm 2: Greedy algorithm

Input : Order of robots R.
Initialize $w(\mathbf{t}_j) = 0 \ \forall \mathbf{t}_j \in T$.
for $r_i \in R$ **do**
  for $p^i_m \in P^i$ **do**
    Compute $c^j_{i,m}$.
    $w'(\mathbf{p}^i_m) = \sum_{\mathbf{t}_j} \max\{w(\mathbf{t}_j), c^j_{i,m}\}$.
  end
  Determine $x^i_m = \text{argmax}\, w'(\mathbf{p}^i_m) \ \forall \mathbf{p}^i_m \in P^i$.
  Update $w(\mathbf{t}_j) = \max\{w(\mathbf{t}_j), c^j_{i,m}\} \ \forall \mathbf{t}_j \in T$.
end
$y^j_i \leftarrow 0 \ \forall \mathbf{r}_i \in R, \ \mathbf{t}_j \in T$.
for $t_j \in T$ **do**
  $\mathbf{r}^*_i \leftarrow \text{argmax}_{\mathbf{r}_i \in R} \sum_{\mathbf{p}^i_m} c^j_{i,m} x^i_m$.
  $y^j_{i^*} \leftarrow 1$.
end

As shown in Algorithm 2, the greedy algorithm runs in $|R|$ communication rounds at each time step. We defined two additional functions: $w(\mathbf{t}_j)$ gives a quality of tracking for j-th target; and $w'(\mathbf{p}_m^i)$ gives the sum of quality of tracking over all feasible targets using m-th motion primitive of i-th robot. If, for example, $c_{i,m}^j$ is used as a distance metric, the max ensures that the quality of tracking for j-th target is only given by the distance of the nearest robot/primitive. That is, even if multiple primitives can track the same $\mathbf{t}_j$, when counting the quality we only care about the closest one. The total quality will then be the sum of qualities for each target.

The objective in Line 5 in Algorithm 2 appears, at first sight, to be different than that given in Equation 4. The following lemma, however, proves that the two objectives are equivalent.

Lemma 2 *Greedy algorithm of Algorithm 2 gives a feasible solution for the* WinnerTakesAll *version of SA-TA.*

The proof is given in Appendix B. Since the objective in Line 5 in Algorithm 2 is submodular, the resulting algorithm yields a 2–approximation to WinnerTakesAll [10].

The greedy algorithm can perform arbitrarily worse than the optimal solution if it is applied to the Bottleneck version of the problem. In Appendix C, we show an example where the greedy yields an arbitrarily bad solution for the Bottleneck version.

A centralized-equivalent approach is one where the robots all broadcast their local information until some robot has received information from all others. This robot can obtain a centralized solution to the problem. A centralized-equivalent approach for a complete $\mathcal{G}_C$ runs in 2 communication rounds for receiving and sending data to neighbors. However, the greedy algorithm and local algorithm have $|R|$ and $h\log(1/\epsilon)$ communication rounds, respectively, for a complete $\mathcal{G}_C$. Note that $h \ll |R|$ for most practical cases.

5 Simulations

We carry out three types of solutions. First, we compare the performance of the greedy and local algorithms with the centralized, optimal solutions. Next, we study the effect of varying the parameters for the local algorithm. Finally, we describe how to implement the algorithm for sequential planning over multiple horizons.

5.1 Comparisons with Centralized Solutions

We performed comparison studies to verify the performance of the proposed algorithms. We compare the greedy solution with the optimal, centralized QMILP solution as well as a random algorithm as a baseline for the WinnerTakesAll version. We compare the local algorithm's solution with the optimal ILP solution as well as the LP with rounding for Bottleneck. For these comparisons, we assume that there are only two primitives to choose from (making the random algorithm a powerful baseline). We later analyze the algorithms with more primitives. We used TOMLAB [64] to solve the QMILP and ILP problems. The toolbox works with MATLAB and uses IBM's CPLEX optimizer in the background. On a laptop with processor configuration of Intel Core i7-5500U CPU @ 2.40GHz x 4 and memory 16 GB the maximum time to solve is around 3 seconds on a case with 150 targets. Most of our cases were solved in less than 2 seconds.

We randomly generated graphs similar to Figure 3 for the comparison. To control the topology of the randomly generated graphs, we define $\phi : \mathcal{G}_S \rightarrow \Re$ to be the percentage of targets that are detected by a motion primitive. We denote the average degree of edges by $d_{avg}(\cdot)$. Therefore:

$$\phi(\mathcal{G}_S) := \frac{d_{avg}(T)}{\sum_{i=1}^{|R|}|P^i|} \times 100 = \frac{|E_S|}{\sum_{i=1}^{|R|}|P^i||T|} \times 100. \quad (6)$$

We start with the upper half of the graph, connecting each robot to its two motion primitives. Then, we iterate through each of motion primitive and randomly choose a target node to create an edge. Next, we iterate through target nodes and randomly choose a motion primitive to create an edge. We also add random edges to connect disconnected components (to keep the implementation simpler). We repeat this in order to get the required graph. If we need to increase the degree of target nodes in the graph, we create new edges to random primitives till we achieve the desired $\phi(\mathcal{G}_S)$. We generated cases by varying $\phi(\mathcal{G}_S)$, number of targets, and number of robots using the method described above. Here, the tracking quality was defined as the number of targets, *i.e.*, $c_{i,m}^j \in \{0,1\}$ for all cases.

The comparative simulation results for WinnerTakesAll are presented in Figure 7. The plots show minimum, maximum, and average of the targets covered by the greedy algorithm and QMILP running 100 random instances for every setting of the parameters. We also show the number of targets covered when choosing motion primitives randomly as a baseline. We observe that the greedy algorithm performs comparatively to the optimal algorithm, and is always better than the baseline. In all the figures, $\Delta_R = 2$, making random a relatively powerful baseline. As $\phi(\mathcal{G}_S)$, number of targets, and number of robots increase, the performance of the greedy algorithm also improves.

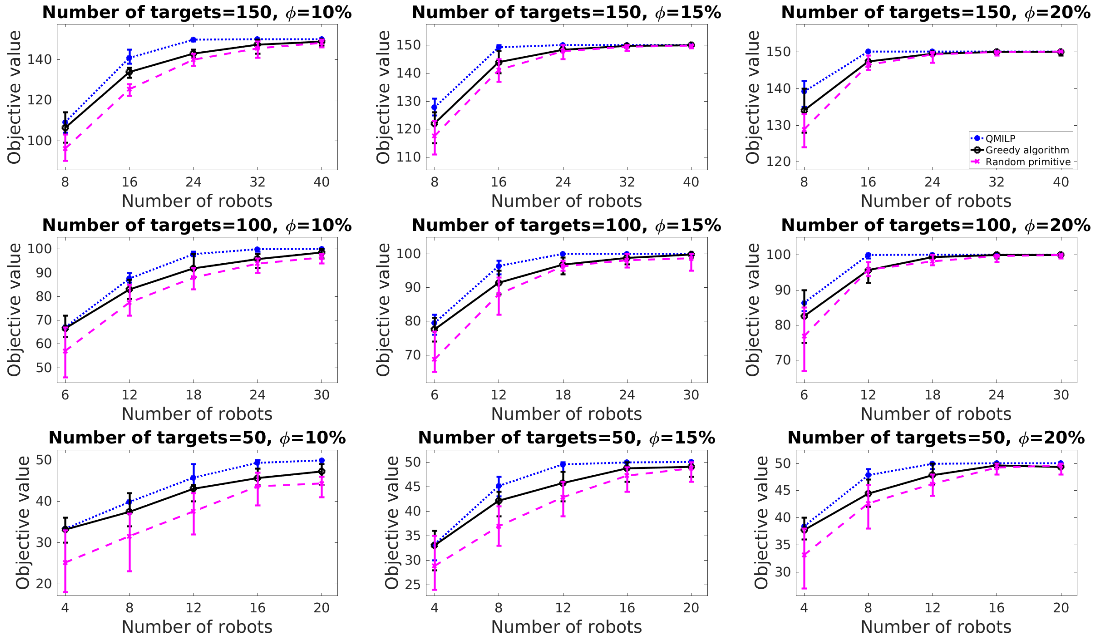


Fig. 7 Showing the comparative results of QMILP, greedy algorithm, and randomly choosing a motion primitive for WinnerTakesAll. To generate graph we vary number of robots, total number of targets, and $\phi(\mathcal{G}_S)$ We ran 100 trials for each case.

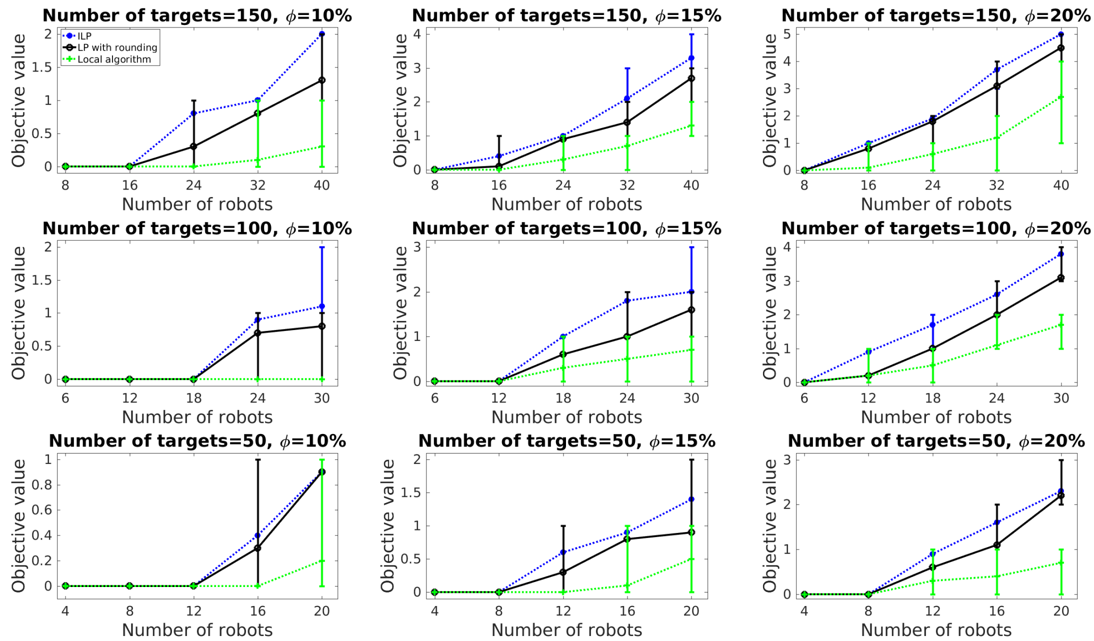
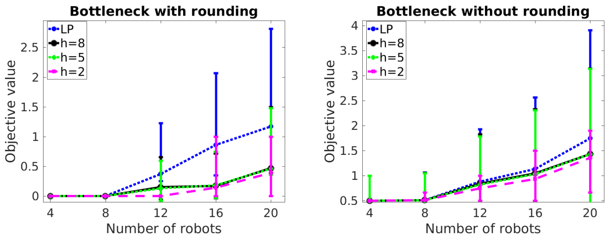


Fig. 8 Comparison simulation for the Bottleneck version of the ILP, LP with rounding and local algorithm. For the local algorithm we set h to 2 for all cases. Each case was obtained from 100 trials.

Figure 8 shows the comparison results for BOTTLENECK where the objective values were computed from the w term of Equation 5. As the proposed local algorithm is a linear relaxation of the ILP formulation, we compare the local solution with the optimal ILP solution. Note that both the ILP and LP with rounding are centralized methods. If the solution value is 0, this means that at least one target is not covered by any selected motion primitives. If the mean value is larger than 0, this implies that all targets are covered by at least one motion primitive on average. The ILP and LP with rounding outperform the local algorithm in all cases. Nevertheless, we find that the local algorithm performs comparably to the centralized methods (and far better than the theoretical bound).

5.2 Effect of h for the Local Algorithm



(a) LP with rounding. (b) LP without rounding.

Fig. 9 Analysis of varying a number of layers (h) for the local algorithm. The number of targets used is 50 and $\phi(\mathcal{G}_S) = 15\%$. We ran 100 trials for each case.

We analyzed the performance of the local algorithm for different number of layers (*i.e.*, h), as shown in Figure 9. The LP value (without rounding) is the upper bound on the optimal solution. We observe how much the rounding sacrifices by comparing the LP with and without the rounding. In the case when h is set to 5 and 8 for both with and without the rounding, however, there is no evident difference between them. This implies that h should not necessarily be large as it does not contribute to the solution quality much (as also seen in Theorem 1). In other words, the local algorithm does not require a large number of communication hops among robots, which is a powerful feature of the local algorithm.

5.3 Multi-robot Multi-target Tracking over Multiple Horizons

The greedy and local algorithms find the motion primitives to be applied over a single horizon. In practice, the motion primitives require the algorithm to be solved over multiple horizons. In this section, we describe how to address this and other challenges associated with a practical implementation. We demonstrate a realistic scenario of cooperative multi-target tracking in the Gazebo simulator using ROS (Figure 10). A bounded environment consists dynamic targets that move in a straight line and change their heading direction randomly after a certain period. The motion model is not known to the robots.

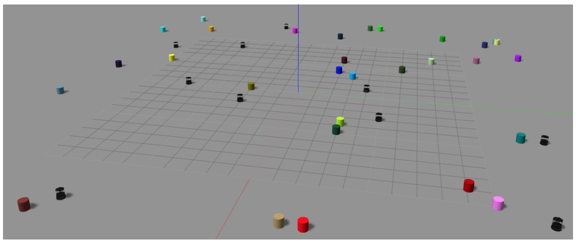


Fig. 10 Gazebo simulator showing ten robots tracking thirty randomly moving targets.

Greedy Algorithm. We implemented the greedy algorithm to solve the WINNERTAKESALL variant in a fully distributed fashion. There is no centralized algorithm and each robot is a separate ROS node that only has access to the local information. Each robot has its local estimator that estimates the state of targets within its FOV. We simulate proximity-limited communication range such that only robots that can see the same target can exchange messages with each other.

A sketch for the implementation of the greedy algorithm is as follows. Each robot has a local timer which is synchronized with the others. Each robot also knows its own ID which is also the order in which the sequential decisions are made. We partition the planning horizon into two periods. In the first *selection* period, the robots choose their own primitives sequentially using the greedy algorithm. In the second *execution* period, the robots apply their motion primitives and obtain measurements of the target.

In the selection period, a robot waits for the predecessor robots (of lower IDs) to make their selections. Every robot knows when it is its turn to select a motion primitive (since the order is fixed). Before its turn, a robot simply keeps track of the most recent $w(\mathbf{t}_j)$ vector received from a predecessor robot within the communication range. During its turn, the robot chooses its motion primitive using the greedy algorithm, and updates the $w(\mathbf{t}_j)$ vector based on its choice. It then broadcasts this updated vector to the neighbors, and waits for the selection period to end. Then, each robot applies its selected motion primitive till the end of the horizon. The process repeats after each planning horizon. The selection period can be preceded by a sensor fusion period, where the robots can execute, for example, the covariance intersection algorithm [62].

For simulations we set the selection and execution periods times to $0.2|R|s$ and $6s$, respectively, where $|R|$ is the number of robtos. Each robot makes its choice after $0.2s$ within the selection period. Each robot has a precomputed library of 21 motion primitives including staying in place (Figure 12). Each robot has a disk-shaped FOV. The sensing and communication ranges are set to $5m$ and $10m$, respectively. We tested both the inverse of the distance and the number of targets as tracking quality (which defines $c^j_{i,m}$).

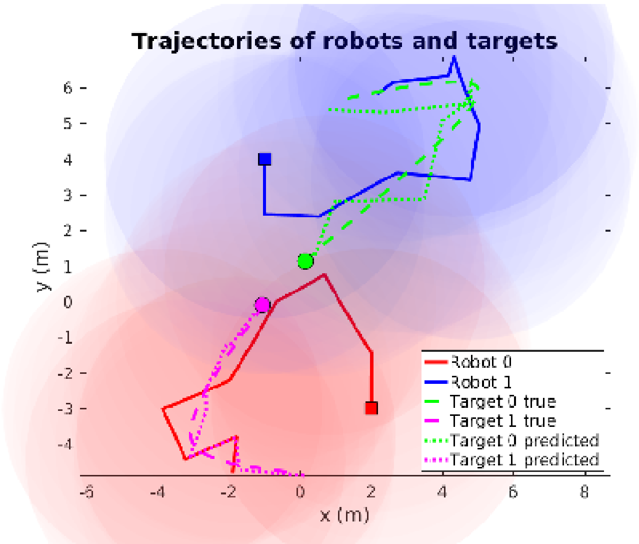


Fig. 11 Tracking performance when there are two robots and two targets moving in a bounded region. Trajectories of two targets and that of two robots are shown. In this case, *Robot* 0 tracks *Target* 1 while *Robot* 1 tracks *Target* 0. The square markers and the circular markers represent the starting positions of robots and those of targets, respectively. Transparent colored disks represent the sensing range of robots.

Figure 11 shows trajectories of robots and those of targets after applying the greedy algorithm for a horizon of 7 seconds (*i.e.*, $1s$ selection period and $6s$ execution period). In this case, the inverse distance was considered as tracking quality. Measurements are corrupted with a white Gaussian random noise. We used Kalman filter to estimate the state of targets, which is represented as green and purple dotted lines (*i.e.*, predicted target trajectories).

Figure 12 presents the result of selecting motion primitives of each robot from the sensing and communication graphs at time step 12 (which is a realization

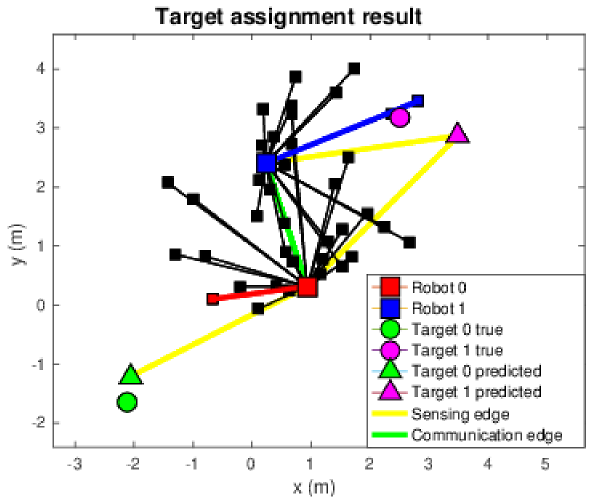


Fig. 12 Target assignment result at time step 12.

of Figure 1). The graph was obtained after both robots finished their local computations.

We carried out larger scale simulations using ten robots tracking thirty moving targets, as shown in Figure 10. Initial positions of robots and targets are randomly chosen in a $30m \times 30m$ environment. It may be possible that some targets are outside the FOV of any robots in the beginning.

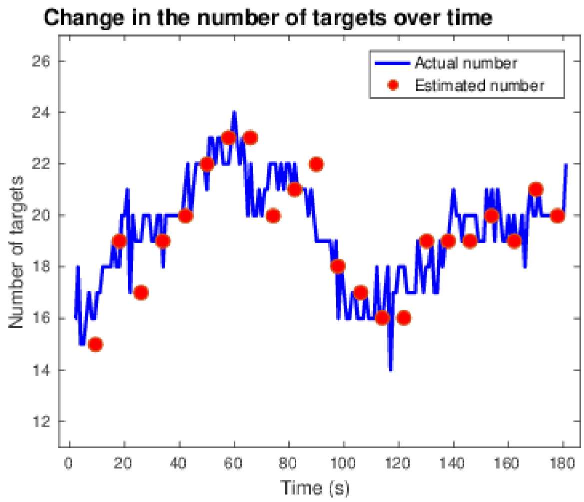
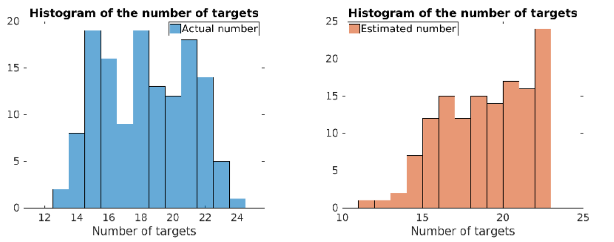


Fig. 13 Change in the number of targets over time.

Figure 13 shows the change in the number of targets over time from a randomly generated instance when the objective was to track the most number of targets. We show both the estimated number of target and the actual number of targets. The estimated number is the value of the solution found at the end of the selection period (obtained every $7s$). This is based on the predicted trajectory of the targets. The actual number of targets is found by counting the target that is within



(a) Actual number of targets. (b) Estimated number of targets.

Fig. 14 Histogram of the number of targets.

the FOV of any robots during the execution period. Figure 14 shows the histogram of the actual and estimated number of targets for 10 trials, each lasting for three minutes.

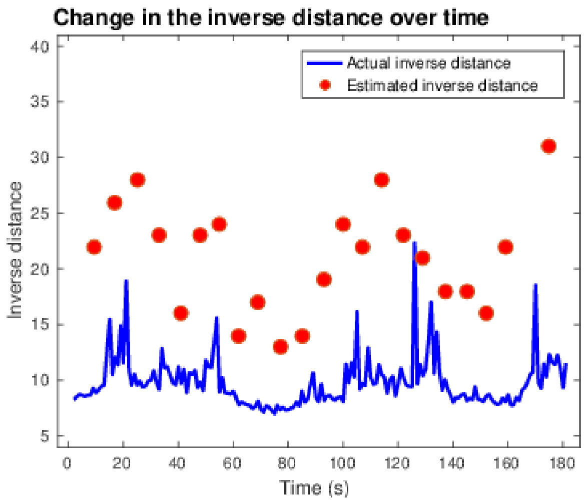
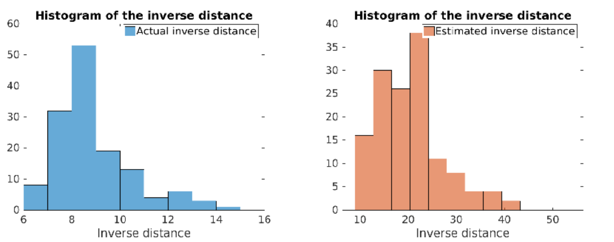


Fig. 15 Change in the inverse of the distance over time.



(a) Actual inverse distance. (b) Estimated inverse distance.

Fig. 16 Histogram of the inverse distance.

Figures 15 and 16 show the corresponding plots when the objective was to maximize the total quality of tracking (inverse distance to the targets). Here, we see that the estimated and the actual values differ much more than the previous case. We conjecture that this is due to the fact that the uncertainty in the motion model of the robots, targets, and measurements has a larger effect on the actual quality of tracking as compared to the number of targets tracked. For instance, even if the actual state of the target deviates from the predicted state, it is still likely that the target will be in the FOV. However, the actual distance between the robot and the target may be much smaller than estimated.

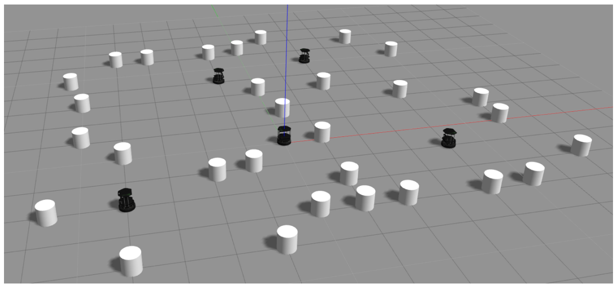


Fig. 17 Snapshot of the Gazebo simulator that shows when five robots are tracking thirty stationary and moving targets.

Local Algorithm. We also implemented the proposed local as shown in Figure 17. Five mobile robots were deployed to track thirty targets (a subset of which were mobile) with a FOV of $3m$ on the xy plane. For each robot two motion primitives were used: one is to remain in the same position and the other one is randomly generated between $-30°$ and $30°$ of the robot's heading traveling randomly up to $1m$.

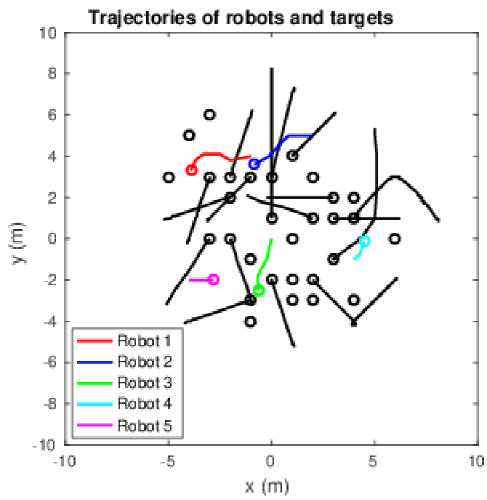


Fig. 18 Plot of trajectories of robots and targets applying the local algorithm to the simulation given in Figure 17. Black lines represent trajectories of thirty targets. ∘ denotes the end position of trajectories. The algorithm was performed for 40 seconds.

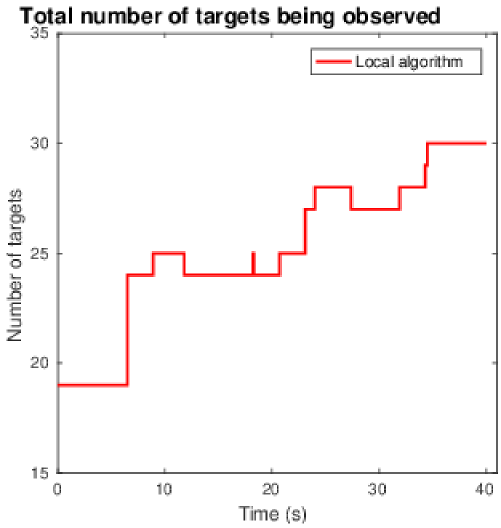


Fig. 19 Change in the total number of targets being observed by any robots over time.

The objective of this simulation is to show the performance of the proposed algorithm for the BOTTLENECK version. At each time step, the local algorithm is employed to choose motion primitives that maximize the minimum number of targets being observed by any robots.

Figure 18 shows the resultant trajectories of robots and targets obtained from the simulation. Figure 19 presents the number of targets tracked by the local algorithm for a specific instance. Although the local algorithm has a sub-optimal performance guarantee, we observe that in practice, it performs comparably.

Acknowledgement

The authors would like to thank Dr. Jukka Suomela from Aalto University for fruitful discussion.

6 Conclusion

This paper gives a new approach to solve the multi-robot multi-target assignment problem using greedy and local algorithms. Our work is motivated by scenarios where the robots would like to reduce their communication to solve the given assignment problem while at the same time maintaining some guarantees of tracking. We used powerful local communication framework employed by Floréen *et al.* [12] to leverage an algorithm that can trade-off the optimality with communication complexity. We empirically evaluated this algorithm and compared it with the baseline greedy strategies. Our immediate future work is to expand the scope of the problem to solve the WINNERTAKESALL version of SATA. We are also working on implementing the resulting algorithms on actual aerial robotic systems to carry out real-world experimentation.

References

1. L. E. Parker and B. A. Emmons, "Cooperative multi-robot observation of multiple moving targets," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3. IEEE, 1997, pp. 2082–2089.
2. L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous robots*, vol. 12, no. 3, pp. 231–255, 2002.
3. C. F. Touzet, "Robot awareness in cooperative mobile robot learning," *Autonomous Robots*, vol. 8, no. 1, pp. 87–97, 2000.
4. A. Kolling and S. Carpin, "Cooperative observation of multiple moving targets: an algorithm and its formalization," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 935–953, 2007.
5. W. Hönig and N. Ayanian, "Dynamic multi-target coverage with robotic cameras," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 1871–1878.
6. B. Gharesifard and S. L. Smith, "Distributed submodular maximization with limited information," *IEEE Transactions on Control of Network Systems*, 2017.
7. T. Howard, M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Model-predictive motion planning: several key developments for autonomous mobile robots," *IEEE Robotics & Automation Magazine*, vol. 21, no. 1, pp. 64–73, 2014.
8. J. Suomela, "Survey of local algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 24, 2013.
9. N. E. Young, "Sequential and parallel algorithms for mixed packing and covering," in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE, 2001, pp. 538–546.
10. G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—1," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.
11. P. Tokekar, V. Isler, and A. Franchi, "Multi-target visual tracking with aerial robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3067–3072.
12. P. Floréen, M. Hassinen, J. Kaasinen, P. Kaski, T. Musto, and J. Suomela, "Local approximability of max-min and min-max linear programs," *Theory of Computing Systems*, vol. 49, no. 4, pp. 672–697, 2011.
13. Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar, "Distributed simultaneous action and target assignment for multi-robot multi-target tracking," *arXiv preprint arXiv:1706.02245*, 2017.
14. Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.
15. H. Li, G. Chen, T. Huang, and Z. Dong, "High-performance consensus control in networked systems with limited bandwidth communication and time-varying directed topologies," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1043–1054, 2017.

16. M. Otte and N. Correll, "Any-com multi-robot path-planning: Maximizing collaboration for variable bandwidth," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 161–173.
17. ——, "Dynamic teams of robots as ad hoc distributed computers: reducing the complexity of multi-robot motion planning via subspace selection," *Autonomous Robots*, pp. 1–23, 2018.
18. A. Kassir, R. Fitch, and S. Sukkarieh, "Communication-efficient motion coordination and data fusion in information gathering teams," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 5258–5265.
19. R. K. Williams and G. S. Sukhatme, "Constrained interaction and coordination in proximity-limited multiagent systems," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 930–944, 2013.
20. J. Vander Hook, P. Tokekar, and V. Isler, "Algorithms for cooperative active localization of static targets with mobile bearing sensors under communication constraints," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 864–876, 2015.
21. Y. Kantaros, M. Thanou, and A. Tzes, "Distributed coverage control for concave areas by a heterogeneous robot–swarm with visibility sensing constraints," *Automatica*, vol. 53, pp. 195–207, 2015.
22. Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3109–3121, 2017.
23. R. K. Williams, A. Gasparri, G. S. Sukhatme, and G. Ulivi, "Global connectivity control for spatially interacting multi-robot systems with unicycle kinematics," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1255–1261.
24. D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1291–1297, 2012.
25. L. Zhou and P. Tokekar, "Active target tracking with self-triggered communications," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
26. X. Ge and Q.-L. Han, "Distributed formation control of networked multi-agent systems using a dynamic event-triggered communication mechanism," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 10, pp. 8118–8127, 2017.
27. G. Best, M. Forrai, R. R. Mettu, and R. Fitch, "Planning-aware communication for decentralised multi-robot coordination," in *Proceedings of the International Conference on Robotics and Automation, Brisbane, Australia*, vol. 21, 2018.
28. M. Guo and M. M. Zavlanos, "Multirobot data gathering under buffer constraints and intermittent communication," *IEEE Transactions on Robotics*, 2018.
29. A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, 2016.
30. C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.
31. B. Charrow, V. Kumar, and N. Michael, "Approximate representations for multi-robot control policies that maximize mutual information," *Autonomous Robots*, vol. 37, no. 4, pp. 383–400, 2014.
32. H. Yu, K. Meier, M. Argyle, and R. W. Beard, "Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 541–552, 2015.
33. A. Ahmad, G. Lawless, and P. Lima, "An online scalable approach to unified multirobot cooperative localization and object tracking," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1184–1199, 2017.
34. K. Zhou, S. I. Roumeliotis, *et al.*, "Multirobot active target tracking with combinations of relative observations," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 678–695, 2011.
35. J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned pomdps," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013.
36. L. C. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. Pereira, "Simultaneous coverage and tracking (scat) of moving targets with robot networks," in *Algorithmic foundation of robotics VIII*. Springer, 2009, pp. 85–99.
37. J. Banfi, J. Guzzi, F. Amigoni, E. F. Flushing, A. Giusti, L. Gambardella, and G. A. Di Caro, "An integer linear programming model for fair multitarget tracking in cooperative multirobot systems," *Autonomous Robots*, pp. 1–16, 2018.
38. Z. Xu, R. Fitch, J. Underwood, and S. Sukkarieh, "Decentralized coordinated tracking with mixed discrete–continuous decisions," *Journal of Field Robotics*, vol. 30, no. 5, pp. 717–740, 2013.
39. B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.
40. G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.
41. H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.
42. L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.
43. L. Liu and D. A. Shell, "Assessing optimal assignment under uncertainty: An interval-based algorithm," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 936–953, 2011.

44. A. Kanakia, B. Touri, and N. Correll, “Modeling multi-robot task allocation with limited information as global game,” *Swarm Intelligence*, vol. 10, no. 2, pp. 147–160, 2016.
45. J. Le Ny, A. Ribeiro, and G. J. Pappas, “Adaptive communication-constrained deployment of unmanned vehicle systems,” *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 923–934, 2012.
46. Y. Kantaros and M. M. Zavlanos, “Global planning for multi-robot communication networks in complex environments.” *IEEE Trans. Robotics*, vol. 32, no. 5, pp. 1045–1061, 2016.
47. M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, “Graph-theoretic connectivity control of mobile robot networks,” *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.
48. X. Ge, F. Yang, and Q.-L. Han, “Distributed networked control systems: A brief overview,” *Information Sciences*, vol. 380, pp. 117–131, 2017.
49. M. Turpin, N. Michael, and V. Kumar, “Capt: Concurrent assignment and planning of trajectories for multiple robots,” *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.
50. D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, “Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming,” *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.
51. S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, “Probabilistic and distributed control of a large-scale swarm of autonomous agents,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.
52. S.-J. Chung, A. Paranjape, P. Dames, S. Shen, and V. Kumar, “A Survey on Aerial Swarm Robotics,” *IEEE Transactions on Robotics*, 2018.
53. M. Otte, M. Kuhlman, and D. Sofge, “Multi-robot task allocation with auctions in harsh communication environments,” in *Multi-Robot and Multi-Agent Systems (MRS), 2017 International Symposium on*. IEEE, 2017, pp. 32–39.
54. D. Angluin, “Local and global properties in networks of processors,” in *Proceedings of the twelfth annual ACM symposium on Theory of computing*. ACM, 1980, pp. 82–93.
55. N. Linial, “Locality in distributed graph algorithms,” *SIAM Journal on Computing*, vol. 21, no. 1, pp. 193–201, 1992.
56. M. Naor and L. Stockmeyer, “What can be computed locally?” *SIAM Journal on Computing*, vol. 24, no. 6, pp. 1259–1277, 1995.
57. M. Hanckowiak, M. Karonski, and A. Panconesi, “On the distributed complexity of computing maximal matchings,” *SIAM Journal on Discrete Mathematics*, vol. 15, no. 1, pp. 41–57, 2001.
58. M. Åstrand, P. Floréen, V. Polishchuk, J. Rybicki, J. Suomela, and J. Uitto, “A local 2-approximation algorithm for the vertex cover problem,” in *International Symposium on Distributed Computing*. Springer, 2009, pp. 191–205.
59. M. Åstrand and J. Suomela, “Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks,” in *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2010, pp. 294–302.
60. C. Lenzen and R. Wattenhofer, “Minimum dominating set approximation in graphs of bounded arboricity,” in *International Symposium on Distributed Computing*. Springer, 2010, pp. 510–524.
61. F. Kuhn, T. Moscibroda, and R. Wattenhofer, “The price of being near-sighted,” in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 2006, pp. 980–989.
62. W. Niehsen, “Information fusion based on fast covariance intersection filtering,” in *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, vol. 2. IEEE, 2002, pp. 901–904.
63. V. Vazirani, *Approximation algorithms*. Springer Publishing Company, Incorporated, 2001.
64. “Tomlab: Optimization environment large-scale optimization in matlab,” http://tomopt.com/docs/quickguide/quickguide006.php, accessed: 2017-01-03.

A Proof of Lemma 1

Equation (5) of a max-min linear program is equivalent to the following max-min problem if the scalar variable w which represents the inner minimization is eliminated:

$$\begin{aligned} &\max_{x_m^i} \min_{\mathbf{t}_j \in T} \left(\sum_{\mathbf{r}_i \in R} \sum_{\mathbf{p}_m^i \in P^i} c_{i,m}^j x_m^i \right) \\ &\text{subject to} \quad \sum_{\mathbf{p}_m^i \in P^i} x_m^i \leq 1 \quad \forall \mathbf{r}_i \in R \\ &\qquad\qquad\qquad x_m^i \geq 0 \quad \forall \mathbf{p}_m^i \in P^i. \end{aligned} \tag{7}$$

From Equations (5) and (7), the following relationship is satisfied:

$$w^* = \min_{\mathbf{t}_j \in T} \left(\sum_{\mathbf{r}_i \in R} \sum_{\mathbf{p}_m^i \in P^i} c_{i,m}^j {x_m^i}^* \right). \tag{8}$$

Since Equation (2) does not require x_m^i to be a linear value, Equation (2) is equivalent to Equation (5) with additional integer constraints.

B Proof of Lemma 2

Considering $c^j_{i,m}$, which is a weight between m-th motion primitive of i-th robot and j-th target on graph $\mathcal{G}_S$, a quality of tracking ($w(\mathbf{t}_j)$) for j-th target can be defined as follows:

$$w(\mathbf{t}_j) \triangleq \max\{c^j_{i,m} | x^i_m = 1,\ \forall i, m\}. \tag{9}$$

Therefore, the sum of quality of tracking over all targets is:

$$\begin{aligned}\sum_{\mathbf{t}_j \in T} w(\mathbf{t}_j) &= \sum_{\mathbf{t}_j \in T} \max\{c^j_{i,m} | x^i_m = 1,\ \forall i, m\} \\ &= \sum_{\mathbf{t}_j \in T} \Big(\sum_{\mathbf{r}_i \in R} y^j_i \Big(\sum_{\mathbf{p}^i_m \in P^i} c^j_{i,m} x^i_m \Big)\Big).\end{aligned} \tag{10}$$

Equation (10) is obtained by taking into account the conditional term of the first equation explicitly. The last equation follows from the property that y^j_i chooses the maximum value of $\sum_{\mathbf{p}^i_m \in P^i} c^j_{i,m} x^i_m$ among all robots, which is shown in lines 10-13 of Algorithm 2. Therefore, the last equation is equal to the inner term of Equation (4).

C Greedy Performs Poorly for the `Bottleneck` Variant

We present an example of instance that shows an arbitrary poor performance of the greedy algorithm when applied to the Bottleneck variant. Consider the following case where there are two robots having two motion primitives for each and two targets. The realization of the communication and sensing graphs are as in the following table. The tracking quality in this example corresponds to the number of targets being tracked.

	p_1	p_2
r_1	t_1	$\varnothing$
r_2	$\varnothing$	t_2

Let's apply the Bottleneckversion of greedy algorithm to this case. Since the objective of the Bottleneckvariant is to maximize the minimum tracking quality, the first robot (r_1) chooses p_2 followed by the second robot (r_2) choosing p_1. This gives the value of 0, whereas the optimal solution is 2 as the first robot and second robot choose p_1 and p_2, respectively. The similar case is reproducible with a larger number of robots, motion primitives, and targets. Thus, the simple greedy performs arbitrarily bad for the Bottleneck variant.