Distributed Simultaneous Action and Target Assignment for Multi-Robot Multi-Target Tracking

Yoonchang Sung, Ashish Kumar Budhiraja, Ryan K. Williams and Pratap Tokekar

***Abstract*— We study two multi-robot assignment problems for multi-target tracking. We consider distributed approaches in order to deal with limited sensing and communication ranges. We seek to simultaneously assign trajectories and targets to the robots. Our focus is on *local* algorithms that achieve performance close to the optimal algorithms with limited communication. We show how to use a local algorithm that guarantees a bounded approximate solution within $\mathcal{O}(h \log 1/\epsilon)$ communication rounds. We compare with a greedy approach that achieves a 2–approximation in as many rounds as the number of robots. Simulation results show that the local algorithm is an effective solution to the assignment problem.**

I. INTRODUCTION

We study the problem of assigning robots with limited Field-of-View (FoV) sensors to track multiple moving targets. 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 a limited communication range and bandwidth. As such, we seek assignment algorithms that rely on local information and limited, local communication with the neighboring robots. We assume that each robot has a number of motion primitives to choose from. The assignment of targets to track is therefore coupled with the selection of motion primitives for each robot. We term this as the distributed Simultaneous Action and Target Assignment (SATA) problem.

A motion primitive is a local trajectory obtained by applying a sequence of actions. We interchangeably use motion primitives to refer to the trajectories as well as the final state on them. 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 (Figure 1). Our goal is to assign motion primitives to the robots so as to track the most number of targets. This problem can be viewed as a version of set cover [1] where every target must be covered by at least one motion primitive. However, we have the additional constraint that only one motion primitive can be chosen per robot. This is called as a packing problem [1]. The combination of these two problems is called a Mixed Packing and Covering Problem (MPCP) [2].

The problem can also be formulated as a (sub)modular maximization problem subject to a partition matroid constraint [3]. A sequential greedy algorithm, where the robots take turns to greedily choose motion primitives, is known to yield a 2–approximation for this problem [4]. The sequential greedy algorithm requires at least as many communication rounds as the number of robots, which may be too slow in practice. Consequently, we focus on *local* algorithms.

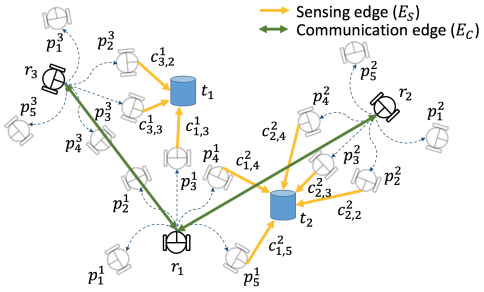


Fig. 1. Description of multi-robot task allocation for multi-target tracking. There are five motion primitives per robot.

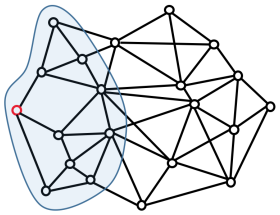


Fig. 2. Communication graph. The blue region indicates a radius–2 neighborhood of the red node. The red node can be unaware of the entire network topology. A local algorithm that works for the red node only requires a local information of nodes in the blue region. The same local algorithm is applied to all nodes in the network.

A local algorithm [1] is a constant-time distributed algorithm that is independent of the size of a network. This enables a robot to depend only on the local inputs in a fixed-radius neighborhood (Figure 2). The robot does not need to know information beyond its local neighborhood, thereby achieving better scalability. Floréen et al. [5] proposed a local algorithm to solve MPCP using max-min/min-max Linear Programming (LPs) in a distributed manner. We show how to leverage this algorithm to solve SATA.

There have been many studies [6], [7] on cooperative target tracking in both control and robotics communities. Yu et al. [8] presented an auction-based decentralized algorithm for cooperative tracking of a mobile target. Capitan et al. [9] proposed a decentralized cooperative multi-robot algorithm using auctioned partially observable Markov decision processes. The performance of decentralized data fusion was

The authors are with the Department of Electrical and Computer Engineering, Virginia Tech, USA. {yooncs8, ashishkb, rywilli1, tokekar}@vt.edu.

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

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

successfully shown under limited communication but theoretical bounds on communication rounds were not covered.

Morbidi et al. [10] presented a gradient-based control scheme for active multi-target tracking. Their focus was not on distributed control policies. Ahmad et al. [11] proposed a least squares minimization technique for cooperative multi-target tracking. However, they focused on localization, not on the multi-robot multi-target assignment. Pimenta et al. [12] adopted Voronoi partitioning to develop a distributed multi-target tracking algorithm. However, their interest lied in covering an environment coupled with multi-target tracking.

The works in [13], [14] and [15] proposed algorithms to solve simultaneous task allocation and path planning, similar to SATA. However, their approaches are centralized. In our prior work [4] we addressed the problem of selecting trajectories for robots that can track the maximum number of targets using a team of robots. No bound on the number of communication rounds was given, possibly resulting in all-to-all communication in the worst case. Instead, we explicitly bound the amount of communication.

Our contributions are as follows: (1) We show how to adapt the local algorithm for solving SATA. (2) We perform empirical comparisons with greedy and baseline centralized algorithms. (3) We demonstrate the applicability of the proposed algorithm through simulations.

II. PROBLEM DESCRIPTION

Let R and T be sets of robots and targets. $R(k) = \{\mathbf{r}_1(k), ..., \mathbf{r}_i(k), ..., \mathbf{r}_{|R|}(k)\}$ denotes the state of robots at time k and $T(k+1) = \{\mathbf{t}_1(k+1), ..., \mathbf{t}_j(k+1), ..., \mathbf{t}_{|T|}(k+1)\}$ denotes the predicted state of targets at time $k+1$. We assume that the targets can be uniquely detected and two robots know if they are observing the same target. Motion primitives of i-th robot $\mathbf{r}_i(k)$ at time k are denoted by $P^i(k) = \{\mathbf{p}^i_1(k), ..., \mathbf{p}^i_m(k), ..., \mathbf{p}^i_{|P^i|}(k)\}$. Note again that the term *motion primitives* in this paper represents the future state of a robot after the corresponding feasible control input is applied starting at time k.

We denote the sensing and communication ranges by $\mathcal{RS}$ and $\mathcal{RC}$. Predicted j-th target $\mathbf{t}_j(k+1)$ at time $k+1$ is said to be observable from m-th motion primitive of robot i iff $\mathbf{t}_j(k+1) \in \mathcal{RS}(\mathbf{p}^i_m(k))$. Likewise, α-th robot can communicate with β-th robot iff $\mathbf{r}_\alpha(k) \in \mathcal{RC}(\mathbf{r}_\beta(k))$ and $\mathbf{r}_\beta(k) \in \mathcal{RC}(\mathbf{r}_\alpha(k))$. We assume that $\mathcal{RC}(\cdot) > 2\mathcal{RS}(\cdot)$. This implies if there is a target j such that $\mathbf{t}_j(k+1) \in \mathcal{RS}(\mathbf{p}^\alpha_m(k))$ and $\mathbf{t}_j(k+1) \in \mathcal{RS}(\mathbf{p}^\beta_m(k))$, then α-th and β-th robots can communicate with each other. Therefore, neighboring robots can share their local information with each other when they observe the same targets.

We also assume that all the robots have synchronous clocks leading to synchronous rounds of communication. This is required in order to employ a greedy algorithm and local algorithm that will be covered in Section III.

Each robot must choose one of its motion primitives to maximize the tracking objective. We first show how to formulate this as an Integer Linear Program (ILP). We define two binary variables: x^i_m and y^j_i. $x^i_m = 1$ if $\mathbf{p}^i_m$ is selected by $\mathbf{r}_i$ and 0 otherwise. $y^j_i = 1$ if $\mathbf{r}_i$ is assigned to $\mathbf{t}_j$ and 0 otherwise. It follows:

$$\sum_{\mathbf{p}^i_m \in P^i} x^i_m \leq 1 \quad \forall \mathbf{r}_i \in R, \qquad \sum_{\mathbf{r}_i \in R} y^j_i \leq 1 \quad \forall \mathbf{t}_j \in T. \tag{1}$$

The objective is to assign the robots/primitives such that all targets are *equitably* covered:

$$\underset{x^i_m}{\operatorname{argmax}} \quad \min_{\mathbf{t}_j \in T} \sum_{\mathbf{r}_i \in R} \sum_{\mathbf{p}^i_m \in P^i} c^j_{i,m} x^i_m, \tag{2}$$

where $c^j_{i,m}$ denotes weights on sensing edges E_S between m-th motion primitive of i-th robot and j-th target. $c^j_{i,m}$ can represent, for example, the distance between t_j and p^i_m. Note that in case t_j and p^i_m have uncertainty associated with them, we can use the Bhattacharyya distance between the corresponding distributions to compute $c^j_{i,m}$.

Consequently, an optimal motion primitive $\mathbf{p}^{i*}_m$ for all robots can be selected based on x^i_m and y^j_i. We term this as the BOTTLENECK version of SATA.

We also define a WINNERTAKESALL variant of SATA where the objective is given by,

$$\underset{x^i_m, y^j_i}{\arg\max} \sum_{\mathbf{t}_j \in T} \left(\sum_{\mathbf{r}_i \in R} y^j_i \left(\sum_{\mathbf{p}^i_m \in P^i} c^j_{i,m} x^i_m \right) \right). \tag{3}$$

Here the goal is to maximize the quality of tracking (alternatively, number of targets that are tracked).

If we fix $c^j_{i,m} = 1$ when p^i_m can observe t_j and zero otherwise, then the objective function becomes equal to the number of targets tracked.

Both versions of the SATA problem are NP-Hard [16]. The WINNERTAKESALL version can be optimally solved using a Quadratic Mixed Integer Linear Programming (QMILP) solver 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 $\epsilon > 2$ are parameters.*

The proof follows directly from the existence of the local algorithm described in the next section. If $\triangle_R = 1$ or $\triangle_T = 1$, there exist local algorithms that give the optimal solution (c.f. Theorem 1 from [5]).

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 follows from the fact that this is a modular maximization problem subject to a partition matroid constraint [3].

III. DISTRIBUTED ALGORITHMS

A. 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 [5] 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 3). The lower part is related with the covering problem. The BOTTLENECK version (Equation 2) can be rewritten as a linear relaxation of ILP:

$$\begin{aligned} \text{maximize} \quad & w \\ \text{subject to} \quad & \sum_{\mathbf{p}_m^i \in P^i} x_m^i \leq 1 \quad \forall \mathbf{r}_i \in R \\ & \sum_{\mathbf{r}_i \in R} \sum_{\mathbf{p}_m^i \in P^i} c_{i,m}^j x_m^i \geq w \quad \forall \mathbf{t}_j \in T \\ & x_m^i \geq 0 \quad \forall \mathbf{p}_m^i \in P^i. \end{aligned} \tag{4}$$

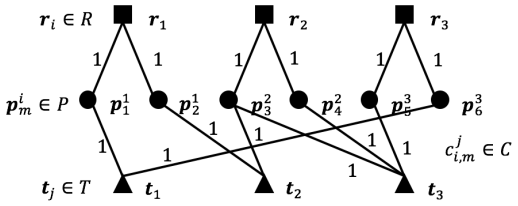


Fig. 3. One instance of a graph for MPCP when there are three robot nodes, six motion primitive nodes and three target nodes. The objective in this example is to maximize the number of targets being tracked. Hence, all weights are set to 1. In general, weights can be arbitrary values.

Floreen et al. [5] presented a local algorithm to solve MPCP in Equation 4 in a distributed fashion. We show how to adapt this to the BOTTLENECK version of SATA. An overview of our algorithm is given in Figure 4. We describe the main steps in the following.

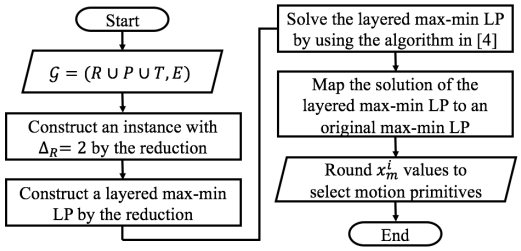


Fig. 4. Flowchart of the proposed local algorithm.

1) Local Algorithm from [5]: The local algorithm in [5] 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 (Figure 5). The details of the construction of the layered graph is given in Section 4 of [5]. h is a user-defined parameter that trades-off computational time with optimality. Layered graph breaks the symmetry that inherently exists in an original graph.

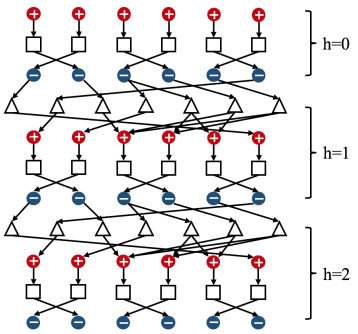


Fig. 5. Graph of the layered max-min LP with $h = 2$ that is obtained from the original graph of Figure 3 after applying the local algorithm.

Authors in [5] 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 proved that the resulting algorithm gives a constant-factor approximation ratio.

Theorem 3. *There exist 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* $\epsilon > 2$, *where* h *denotes the number of layers.*

Proof. Please refer to Corollary 4.7 from [5] for a proof. □

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. The layered graph is constructed in a local fashion without requiring any single robot to know the entire graph.

2) Realization of Local Algorithm for SATA: To apply the local algorithm to distributed SATA, 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 $\mathbf{r}_i \in R$. Nodes corresponding to the targets are also realized by the 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 who realizes the target node in a constant number of communication rounds.

After applying the local algorithm, each robot obtains x_m^i for corresponding $\mathbf{p}_m^i$. However, due to the LP relaxation, x_m^i will not necessarily be binary. 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 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.

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

As the number of layers increases, the more distinct the x_p^i 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.

$\mathbf{p}_m^i$	x_m^i	$h = 2$	$h = 10$	$h = 30$
$\mathbf{p}_1^1$	$x_1^1 =$	0.5000	0.5000	0.5000
$\mathbf{p}_2^1$	$x_2^1 =$	0.5000	0.5000	0.5000
$\mathbf{p}_3^2$	$x_3^2 =$	0.6667	0.7591	0.7855
$\mathbf{p}_4^2$	$x_4^2 =$	0.3333	0.2409	0.2145
$\mathbf{p}_5^3$	$x_5^3 =$	0.3333	0.2409	0.2145
$\mathbf{p}_6^3$	$x_6^3 =$	0.6667	0.7591	0.7855

TABLE I

SOLUTION RETURNED BY THE LOCAL ALGORITHM FOR THE EXAMPLE SHOWN IN FIGURE 3 WITH THE VARYING NUMBER OF LAYERS, h.

Algorithm 1 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 time steps without affecting the computation time significantly.

Algorithm 1: Local algorithm

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

One of the advantages of the local algorithm is that even if the communication graph is disconnected, each component of the graph can run the local algorithm independently without affecting the solution quality. The algorithm also allows for the number of robots and targets to change. Since each robot determines its neighbors at each time step, any new robots or targets will be identified and become part of the local layered graphs at the next planning timestep.

B. Greedy Algorithm

We use the greedy algorithm proposed in [4] and [17] as the baseline comparison. 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 $\boldsymbol{r}_i \in R$ **do**
  for $\boldsymbol{p}_m^i \in P^i$ **do**
    Compute $c_{i,m}^j$ $w'(\mathbf{p}_m^i) = \sum_{\mathbf{t}_j} \max\{w(\mathbf{t}_j), c_{i,m}^j\}$
  end
  Determine $x_m^i = \operatorname{argmax} w'(\mathbf{p}_m^i)\ \forall \mathbf{p}_m^i \in P^i$
  Update $w(\mathbf{t}_j) = \max\{w(\mathbf{t}_j), c_{i,m}^j\}\ \forall \mathbf{t}_j \in T$
end
$y_i^j \leftarrow 0\ \forall \mathbf{r}_i \in R,\ \mathbf{t}_j \in T$
for $\boldsymbol{t}_j \in T$ **do**
  $\mathbf{r}_i^* \leftarrow \operatorname{argmax}_{\mathbf{r}_i \in R} \sum_{\mathbf{p}_m^i} c_{i,m}^j x_m^i$
  $y_{i^*}^j \leftarrow 1$
end

As shown in Algorithm 2, the greedy algorithm runs in $|R|$ communication rounds at each time step. We define two 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.

Lemma 1. *Greedy algorithm of Algorithm 2 gives a feasible solution for the* WINNERTAKESALL *version of SATA.*

The proof is given in the appendix.

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. A centralized-equivalent approach for a complete communication runs in 2 communication rounds for receiving and sending data to neighbors. However, the local algorithm and greedy algorithm take $h \log(1/\epsilon)$ and $|R|$ communication rounds, respectively. Note that $h \ll |R|$ for most practical cases.

IV. SIMULATIONS

In this section, we evaluate empirically the performance of the local algorithm and greedy algorithm in two settings.

A. Comparison Study

We compare the proposed algorithms with the QMILP solution. The greedy algorithm and QMILP solve the WINNERTAKESALL problem and local algorithm solves the BOTTLENECK problem. However, we compare the total number of targets covered by both approaches. We used TOMLAB [18] to get the QMILP solution. TOMLAB works with MATLAB and uses IBM's CPLEX optimizer. An Intel Core i7-5500U CPU @ 2.40GHz x 4 laptop with 16 GB memory took a maximum time of around 4 seconds to solve an instance with 200 targets and average target degree of 2. Most of instances were solved in less than 2 seconds.

We randomly generated graphs similar to Figure 3 with a given average degree for comparison. We start with the upper half of the graph, connecting each robot to its two motion primitives. Then we iterate through each 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. We create new edges to random primitives till we achieve the desired degree. We generated cases by varying the degree of targets, number of targets, and number of robots using the method described above.

Figure 6 shows the minimum, maximum, and the mean number of targets covered by the local algorithm, 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 local algorithm with $h = 2$ performs comparatively to the optimal algorithm, and is always better than the baseline. In all the figures, $\Delta_R = 2$.

When the number of targets are 50 and 100 with degree 4 (Figure 6), the performance of the local algorithm does not improve as the number of robots deployed increases, which may seem counterintuitive. We conjecture that the reason behind this is the locality of the proposed algorithm. Even though more robots are used to track the same number of targets, the average degree of the target remains the same. Consequently, the communication graph for the robots becomes sparser. Since h is fixed for all cases, this implies that each robot in layered graph reaches a smaller subset of the total graph, leading to even more sub-optimal performance. One avenue of future work is to analyze this in more depth.

B. Multi-robot Multi-target Tracking Simulation

The proposed local algorithm was implemented in Gazebo (Figure 7). Five mobile robots were deployed to track thirty targets (a subset of which were mobile) with a FoV of $3m$ on the ground plane. Two motion primitives were used per robot: (1) remain in place and (2) move a random distance of up to $1m$ with a random heading between -30° and 30°.

At each time step, the local algorithm chooses motion primitives to maximize the number of targets tracked (BOTTLENECK version). We compared this with the greedy algorithm. Figure 8 shows the trajectories of robots and targets obtained from the simulation. Figure 9 compares the number of targets tracked by the local and greedy algorithms for a specific instance. Both algorithms have a sub-optimal performance guarantee, with the greedy algorithm having a better worst-case guarantee than the local one. However, in practice, both strategies perform comparably.

V. CONCLUSIONS

We present a new approach for multi-robot multi-target assignment. Our work is motivated by scenarios where the robots would like to reduce their communication while still maintaining some guarantees of tracking performance. We used the local communication framework employed by Floreen et al. [5] to leverage an algorithm that can trade-off optimality with communication complexity. We empirically evaluated this algorithm and compared it with the baseline greedy strategy. Our immediate future work is to expand the local algorithm to solve the WINNERTAKESALL version of SATA. Another extension would be to include *search primitives* for exploration, in order to handle situations where some targets fall outside the FoV of all robots.

REFERENCES

[1] J. Suomela, "Survey of local algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 24, 2013.

[2] 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.

[3] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsi," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[4] 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.

[5] 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.

[6] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, 2016.

[7] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[8] 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.

[9] 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.

[10] F. Morbidi and G. L. Mariottini, "Active target tracking and cooperative localization for teams of aerial vehicles," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 5, pp. 1694–1707, 2013.

[11] A. Ahmad, G. D. Tipaldi, P. Lima, and W. Burgard, "Cooperative robot localization and target tracking based on least squares minimization," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5696–5701.

[12] 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.

[13] A. Kulatunga, D. Liu, G. Dissanayake, and S. Siyambalapitiya, "Ant colony optimization based simultaneous task allocation and path planning of autonomous vehicles," in *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*. IEEE, 2006, pp. 1–6.

[14] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating uavs," in *Cooperative control: models, applications and algorithms*. Springer, 2003, pp. 23–41.

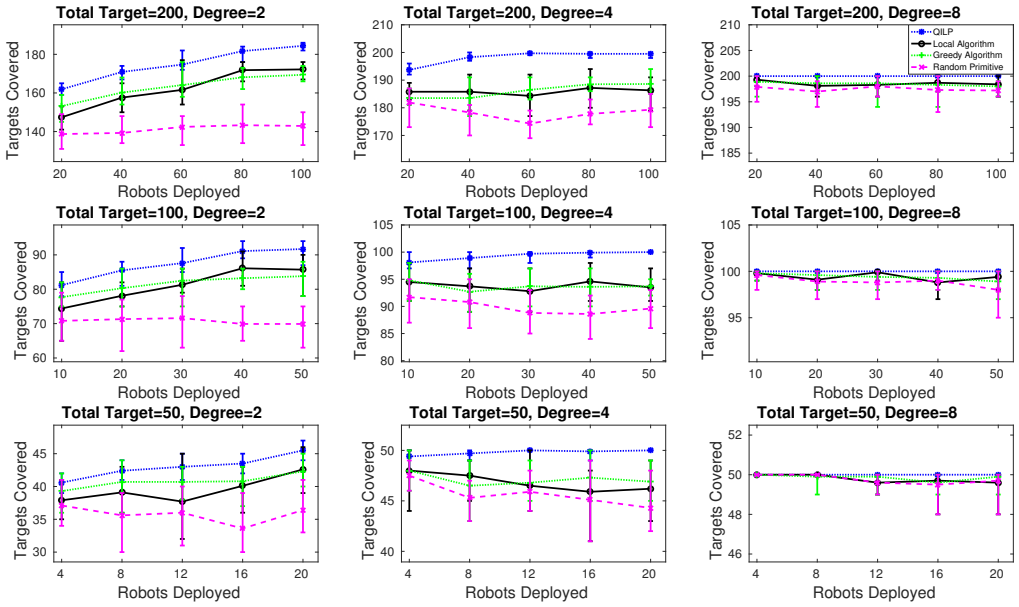


Fig. 6. Comparative results of QMILP, local algorithm with h=2, greedy algorithm, and randomly choosing a motion primitive. The plots show the mean and min/max number of targets covered from 100 trials.

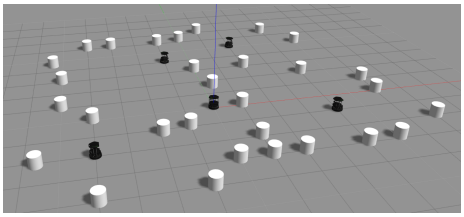
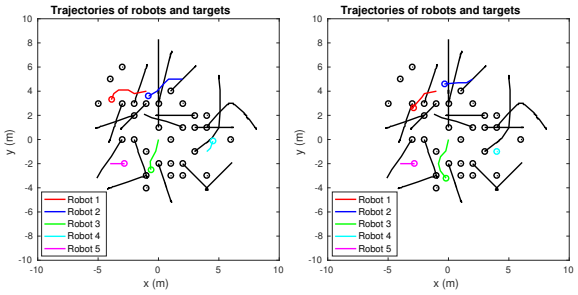


Fig. 7. Gazebo simulator showing five robots tracking thirty stationary and moving targets (please refer to the attached video).



(a) Trajectory information obtained by the local algorithm.

(b) Trajectory information obtained by the greedy algorithm.

Fig. 8. Plot of trajectories of robots and targets applying both local and greedy algorithms to the simulation given in Figure 7. Black lines represent trajectories of thirty targets. ○ denotes the end position of trajectories. Both algorithms were performed for 40 seconds on the same target trajectories.

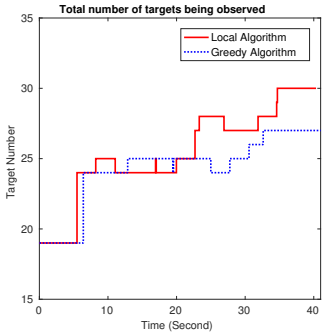


Fig. 9. Comparison in the total number of targets being observed by any robots between local and greedy algorithms.

[15] Y. Eun and H. Bang, "Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithm," *Journal of aircraft*, vol. 46, no. 1, pp. 338–343, 2009.

[16] V. Vazirani, *Approximation algorithms*. Springer Publishing Company, Incorporated, 2001.

[17] P. Dames, P. Tokekar, and V. Kumar, "Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots," *Intl. Sym. Robot. Research*, 2015.

[18] "Tomlab: Optimization environment large-scale optimization in matlab," http://tomopt.com/docs/quickguide/quickguide006.php, accessed: 2017-01-03.

APPENDIX

A. Proof of Lemma 1

Let $w(\mathbf{t}_j) \triangleq \max\{c^j_{i,m} | x^i_m = 1,\ \forall i, m\}$. Therefore, the sum of quality of tracking over all targets is:

$$\begin{aligned}\sum_{\mathbf{t}_j \in T} w(\mathbf{t}_j) &= \sum_{\mathbf{t}_j \in T} \max\{c^j_{i,m} | x^i_m = 1,\ \forall i, m\} \\ &= \sum_{\mathbf{t}_j \in T} \Big(\sum_{\mathbf{r}_i \in R} \max\{ \sum_{\mathbf{p}^i_m \in P^i} c^j_{i,m} x^i_m \} \Big) \\ &= \sum_{\mathbf{t}_j \in T} \Big(\sum_{\mathbf{r}_i \in R} y^j_i \Big(\sum_{\mathbf{p}^i_m \in P^i} c^j_{i,m} x^i_m \Big) \Big).\end{aligned} \tag{5}$$

Equation 5 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 3.