Autonomous Robots
https://doi.org/10.1007/s10514-019-09856-1

Distributed assignment with limited communication for multi-robot multi-target tracking

Yoonchang Sung[1] · **Ashish Kumar Budhiraja**[1] · **Ryan K. Williams**[1] · **Pratap Tokekar**[1]

Received: 29 August 2018 / Accepted: 25 April 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

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 a bounded amount of time. We present two algorithms: (1) a greedy algorithm that is guaranteed to find a 2-approximation to the optimal (centralized) solution but 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.

Keywords Multi-robot system · Task assignment · Distributed algorithm

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 (Parker and Emmons 1997; Parker 2002; Touzet 2000; Kolling and Carpin 2007; Hönig and Ayanian 2016). 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 a 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 (2017) 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 (Fig. 1).

This is one of the several papers published in *Autonomous Robots* comprising Special Issue on Robot Communication Challenges: Real-World Problems, Systems, and Methods.

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

✉ Yoonchang Sung
yooncs8@vt.edu

Ashish Kumar Budhiraja
ashishkb@vt.edu

Ryan K. Williams
rywilli1@vt.edu

Pratap Tokekar
tokekar@vt.edu

[1] Department of Electrical and Computer Engineering, Virginia Tech, 1185 Perry Street, Blacksburg, VA 24061, USA

Published online: 17 May 2019

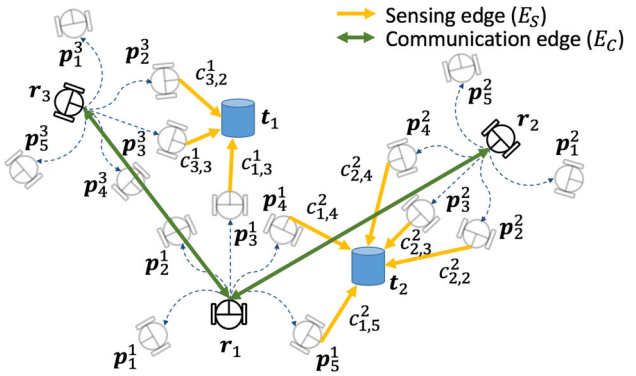


Fig. 1 Description of multi-robot task allocation for multi-target tracking. In this example, three robots ($\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$) are tracking two moving targets ($\mathbf{t}_1, \mathbf{t}_2$). Each robot has five motion primitives ($\mathbf{p}_m^i$) to choose from at each time step. c represents the cost of observing a target from a motion primitive

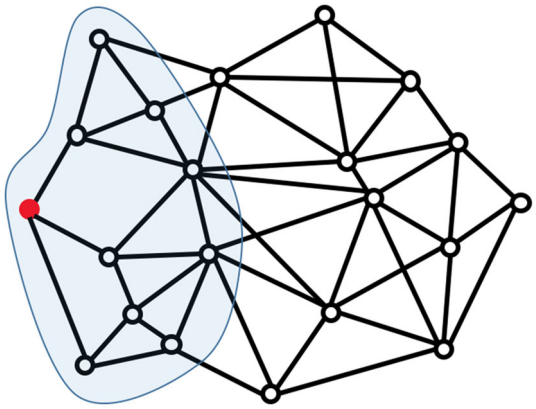


Fig. 2 Communication graph. The blue shaded region indicates a radius–2 neighborhood of the red solid node. The red solid node may be unaware of the entire communication graph topology. A local algorithm that works for the red solid node only requires local information of nodes in the blue shaded region.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

We assume that each robot has a number of motion primitives to choose from. A motion primitive is a local trajectory obtained by applying a sequence of actions (Howard et al. 2014). 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 (Suomela 2013). 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 (Suomela 2013) 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) (Young 2001).

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 (Nemhauser et al. 1978). A sequential greedy algorithm, where the robots take turns to greedily choose motion primitives, is known to yield a 2–approximation for this problem (Tokekar et al. 2014). 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.

A local algorithm (Suomela 2013) 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 (Fig. 2). The robot does not need to know information beyond its local neighborhood, thereby achieving better scalability.

Floréen et al. (2011) 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 $\mathcal{O}((1+\epsilon)(1+1/h))$ 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.

[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.

4. We demonstrate the applicability of the proposed algorithm through Gazebo simulations.

A preliminary version of this paper was presented at ICRA 2018 (Sung et al. 2018). This expanded paper extends the 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 Sect. 2. We describe the problem setup in Sect. 3. Our proposed distributed algorithms are presented in Sect. 4. We present results from representative simulations in Sect. 5 before concluding with a discussion of future work in Sect. 6.

2 Related work

A number of algorithms have been designed to improve multi-robot coordination under limited bandwidth (Yan et al. 2013; Li et al. 2017; Otte and Correll 2013, 2018; Kassir et al. 2016) and under communication range constraints (Williams and Sukhatme 2013; Vander Hook et al. 2015; Kantaros et al. 2015). This includes algorithms that enforce connectivity constraints (Kantaros and Zavlanos 2017; Williams et al. 2015), explicitly trigger when to communicate (Dimarogonas et al. 2012; Zhou and Tokekar 2018; Ge and Han 2017) and operate when connectivity is intermittent (Best et al. 2018; Guo and Zavlanos 2018). 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 (Khan et al. 2016; Robin and Lacroix 2016).

Charrow et al. (2014) 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. (2015) worked on an auction-based decentralized algorithm for cooperative path planning to track a moving target. Ahmad et al. (2017) presented a unified method of localizing robots and tracking a target that is scalable with respect to the number of robots. Zhou and Roumeliotis (2011) developed an algorithm that finds an optimal trajectory of multiple robots for the active target tracking problem. Capitan et al. (2013) 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 considered single target tracking, which may not necessarily hold in the case of tracking multiple targets in a distributed fashion.

Pimenta et al. (2009) 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. (2018) addressed the *fairness* issue for cooperative multi-robot multi-target tracking, which is achieving balanced coverage among different targets. One of the problems that we define in Sect. 3 (i.e., Problem 1) has a similar motivation. However, unlike the algorithm in Banfi et al. (2018), we are able to give a global performance guarantee. Xu et al. (2013) presented a decentralized algorithm that jointly solves the problem of assigning robots to targets and positioning robots using mixed-integer nonlinear programming. While they proved the complexity in terms of computational time and communication (i.e., the amount of data needed to be communicated), the solution quality was only evaluated empirically. Instead, we bound the solution quality as a function of the communication rounds. Furthermore, our formulation takes as input a set of discrete actions (i.e., motion primitives) that the robot must choose from, unlike the previous work.

We study a problem similar to the one termed as Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) proposed by Parker and Emmons (1997). The objective in CMOMMT is to maximize the collective time of observing targets. Parker (2002) developed a distributed algorithm for CMOMMT that computes a local force vector to find a direction vector for each robot. We empirically compare this algorithm with our proposed one and report the results in Sect. 5. Kolling and Carpin (2007) studied the behavioral CMOMMT that added a new mode (i.e., help) to the conventional track and search modes of CMOMMT. The help mode asks other robots to track a target if the target escapes from the FOV of some robot. Although our work does not allow mode changes, previous works regarding CMOMMT did not provide theoretical optimality guarantees and did not explicitly consider scenarios where the communication bandwidth is limited. Refer to Section IV(C) of a more detailed summary of CMOMMT.

In our prior work (Tokekar et al. 2014), 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 presented, possibly resulting in all-to-all communication in the worst case. Instead, in this work, we introduce a new version of the problem and also 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 (2004) and the more recent work by Korsah et al. (2013) contain detailed survey of this problem. There exists distributed algorithms with provable guarantees for different versions of this problem (Choi et al. 2009; Luo et al. 2015; Liu and Shell 2011). There also exists various multi-robot deployment strategies for task assignment under communication constraints. These constraints include limited available information (Kanakia et al. 2016), limited communication flows (Le Ny et al. 2012), and connectivity requirement (Kantaros and Zavlanos 2016). See the survey papers (Zavlanos et al. 2011; Ge et al. 2017) on these results. Le Ny et al. (2012) studied a formulation with a similar communication constraint as ours. However, their formulation assumed that the robots know which targets to track. In this paper, we tackle the challenge of simultaneously assigning robots to targets by choosing motion primitives with limited communication bandwidth which might degrade task performance when there are unreliable communication links and communication delays.

Turpin et al. (2014) proposed a distributed algorithm that assigns robots to goal locations while generating collision-free trajectories. Morgan et al. (2016) solved the assignment problem by using distributed auctions and generating collision-free trajectories by using sequential convex programming. Bandyopadhyay et al. (2017) adopted the Eulerian framework for both swarm formation control and assignment. However, these works may not be suitable for target tracking applications as the targets were assumed to be static. For more survey results about SATA, see the work by Chung et al. (2018). Recently, Otte et al. (2017) 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 (Angluin 1980; Linial 1992; Naor and Stockmeyer 1995) 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 (Hanckowiak et al. 2001), vertex cover (Åstrand et al. 2009; Åstrand and Suomela 2010), dominating set (Lenzen and Wattenhofer 2010), and set cover (Kuhn et al. 2006). Suomela (2013) gives a broad survey of local algorithms. We build on this work and adapt a local algorithm for solving SATA.

3 Problem description

Consider a scenario where multiple robots are tracking multiple mobile targets. Robots can observe targets within their FOV and predict the future states of targets. Based on predicted target states, robots decide where to move (i.e., by selecting a motion primitive) in order to keep track of targets. By discretizing time, the problem becomes one of combinatorial optimizations—choose the next position of robots based on the predicted position of the targets. Thus, we solve the SATA problem at each time step.

We define sets, R and T, to denote the collection of robot and target labels respectively: $R = \{1, \ldots, i, \ldots, |R|\}$ for robot labels and $T = \{1, \ldots, j, \ldots, |T|\}$ for target labels. Let r and t denote the set of robot states and *predicted* target states, respectively. In this paper, states are given by the positions of the robots and the targets in 2- or 3-dimensional space. However, the algorithms presented in this paper can be used for more complex states (*e.g.*, 6 degree-of-freedom pose). Here, $r(k) = \{\mathbf{r}_1(k), \ldots, \mathbf{r}_i(k), \ldots, \mathbf{r}_{|R|}(k)\}$ denotes the state of the robots at time k. $t(k) = \{\mathbf{t}_1(k), \ldots, \mathbf{t}_j(k), \ldots, \mathbf{t}_{|T|}(k)\}$ denotes the state of the targets at the next time step (i.e., at time $k+1$) predicted at time k. 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. Each robot independently obtains the predicted states, $t(k)$, by fusing its own noisy sensor measurements using, for example, a Kalman filter.

We define the labels of available motion primitives for the i-th robot as $P^i = \{1, \ldots, m, \ldots, |P^i|\}$. These labels correspond to a set of motion primitive states of the i-th robot at time k given by: $p^i(k) = \{\mathbf{p}^i_1(k), \ldots, \mathbf{p}^i_m(k), \ldots, \mathbf{p}^i_{|P^i|}(k)\}$. Note that the term *motion primitives* in this paper represents the future state of a robot at the next time step (i.e., at time $k+1$) computed at time k. We compute a set of the motion primitives a priori by discretizing the continuous control input space. This can be done by various methods such as uniform random sampling or biased sampling based on predicted target states. However, once a set of the motion primitives is obtained, the rest of the proposed algorithms (in Sect. 4) remain the same.

We define $\mathcal{RS}(\mathbf{p}^i_m(k))$ to be the set of targets that can be observed by the m-th motion primitive of i-th robot at time k. Specifically, the j-th target is said to be observable by the m-th motion primitive of a robot i, iff $\mathbf{t}_j(k) \in \mathcal{RS}(\mathbf{p}^i_m(k))$. It should be noted that only targets that were observed by robot i at time $k-1$ are candidates to be considered for time k because unobserved targets at time $k-1$ cannot be predicted

by the robot i. Note also 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.[2]

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 $\mathbf{t}_j(k) \in \mathcal{RS}(\mathbf{p}_m^i(k))$ and $\mathbf{t}_j(k) \in \mathcal{RS}(\mathbf{p}_m^l(k))$, 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 (Niehsen 2002) to merge their individual predictions of the target's state into a joint prediction 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 but 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 Fig. 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_m^i be the binary variable which represents the i-th robot selecting the m-th motion primitive. That is, $x_m^i = 1$ if a motion primitive m is selected by a robot i and 0 otherwise.[3] Since each robot can choose only one motion primitive, we have:

$$\sum_{m \in P^i} x_m^i \leq 1 \ \ \forall i \in R. \tag{1}$$

Our objective is to find x_m^i. We propose two following problems.

[2] After these assumptions, we omit the time index (i.e., k) for notational convenience.

[3] If all $x_m^i = 0$ for a robot i, then it can choose any motion primitives since the objective value will remain the same.

Problem 1 (BOTTLENECK) The objective is to select primitives such that we maximize the minimum tracking quality:

$$\underset{x_m^i}{\text{argmax}} \quad \min_{j \in T} \left(\sum_{i \in R} \sum_{m \in P^i} c_{i,m}^j x_m^i \right), \tag{2}$$

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

Here, $c_{i,m}^j$ can represent the tracking quality given by, for example, the inverse of the distance between m-th motion primitive of i-th robot and j-th target. Alternatively, $c_{i,m}$ can be binary (1 when the m-th motion primitive of robot i sees target j and 0 otherwise) making the objective function equal to maximizing the minimum number of targets tracked.

We term this as the BOTTLENECK version of SATA. In the BOTTLENECK version, 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 i-th robot is assigned to j-th target and 0 otherwise.

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

$$\sum_{i \in R} y_i^j \leq 1 \ \ \forall j \in T. \tag{3}$$

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

$$\underset{x_m^i, y_i^j}{\text{argmax}} \sum_{j \in T} \left(\sum_{i \in R} y_i^j \left(\sum_{m \in P^i} c_{i,m}^j x_m^i \right) \right), \tag{4}$$

subject to the constraints in Eqs. (1) and (3).

Both versions of the SATA problem are NP-Hard (Vazirani 2001). The WINNERTAKESALL version can be optimally solved using a Quadratic Mixed Integer Linear Programming (QMILP) in the centralized setting.[4] 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.

[4] Note that Problem 2 can also be converted into a simpler Mixed Integer Linear Programming (MILP) by linearizing the product of the binary variables in Eq. (4), which is not covered in this paper.

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. We show how the local algorithm for MPCP can be modified to solve SATA by means of a linear relaxation.

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 (Nemhauser et al. 1978). The algorithms are described in the next section.

4 Distributed algorithms

We begin by describing the local algorithm that solves the BOTTLENECK version of SATA.

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. (2011) 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 Fig. 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 $c_{i,m}^j$. 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 $\mathbf{p}_m^i \in p^i$ is associated with a variable x_m^i. The upper two layers of $\mathcal{G}$ in Fig. 3 are related with a packing problem (Eq. (4)). The lower two layers are related with the covering problem.

Lemma 1 *The* BOTTLENECK *version (Eq. (2)) can be rewritten as a linear relaxation of ILP:*

$$\begin{aligned} &\textit{maximize} \quad w \\ &\textit{subject to} \quad \sum_{m \in P^i} x_m^i \leq 1 \quad \forall i \in R \\ &\qquad \sum_{i \in R} \sum_{m \in P^i} c_{i,m}^j x_m^i \geq w \quad \forall j \in T \\ &\qquad\qquad x_m^i \geq 0 \quad \forall m \in P^i. \end{aligned} \tag{5}$$

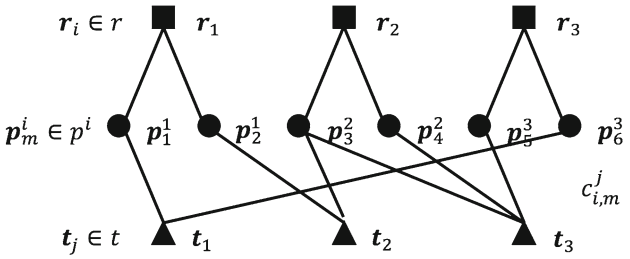


Fig. 3 One instance of a graph for MPCP when there are three robot nodes, six motion primitive nodes and three target nodes

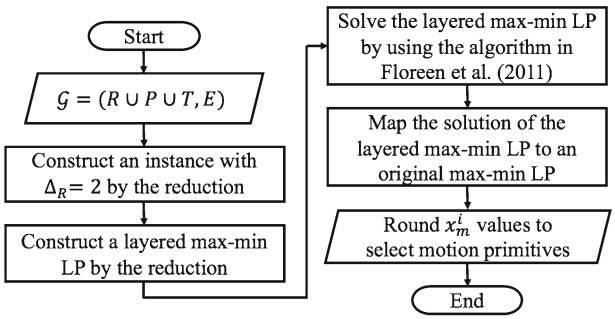


Fig. 4 Flowchart of the proposed local algorithm

The proof is given in Appendix A.

Floréen et al. (2011) presented a local algorithm to solve MPCP in Eq. (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.

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

4.1.1 Local algorithm from Floréen et al. (2011)

The local algorithm in Floréen et al. (2011) 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 Fig. 5. h is a user-defined parameter that allows to trade-off computational time with optimality. If the number of layers is set to h, then it means that a robot can communicate with another robot that is no more than h communication edges (i.e., hops) away. The layered graph breaks the symmetry that inherently exists in the original graph. This layered mechanism is specifically designed for solving MPCP and is covered in depth in Section 4 of Floréen et al. (2011). We omit the details in this paper due to limited space and redirect the readers to Section 4 of Floréen et al. (2011) for the construction of the layered 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

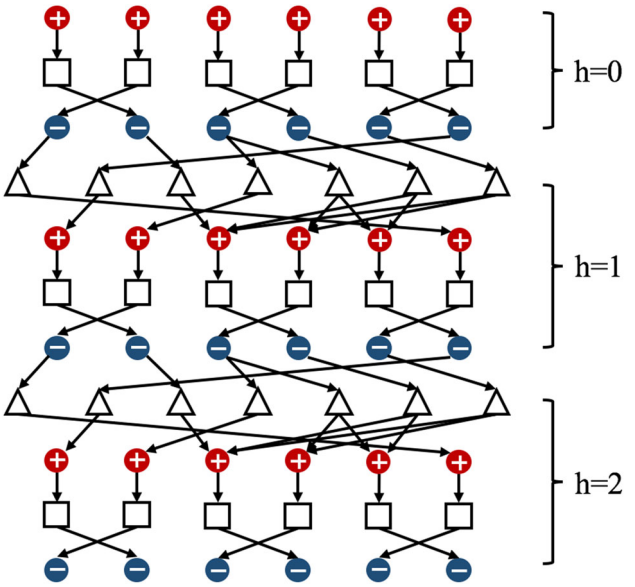


Fig. 5 Graph of the layered max-min LP with $h = 2$ that is obtained from the original graph of Fig. 3 after applying the local algorithm. The details for constructing a layered graph are given in Section 4 of Floréen et al. (2011). Each motion primitive $\mathbf{p}_m^i \in p^i$ is colored either red or blue to break the symmetry of the original graph. Squares, circles, and triangles represent robot nodes, motion primitive nodes, and target nodes, respectively, corresponding to Fig. 3

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 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 Floréen et al. (2011) for a proof. □

Note that each node in the layered graph carries out its local computation (details of the local computation for solving SATA are included in Floréen et al. (2011)). 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 Sect. 4.1.1 to a distributed SATA problem, each node and edge in a layered graph must be realized at each time step (i.e., generating a graph shown in Fig. 5 which becomes the input to the local algorithm (Floréen et al. 2011)). In our case, the only computational units are the robots. Nodes that correspond to motion primitives, $\mathbf{p}_m^i \in p^i$, 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 j is realized by a robot i satisfying $\mathbf{t}_j \in \mathcal{RS}(\mathbf{p}_m^i)$. If there are multiple robots whose motion primitives can sense the target (by Assumption 1), they can arbitrarily decide which amongst them realizes the target nodes in a constant number of communication rounds.

After applying the local algorithm of Sect. 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 Eq. (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.

Algorithm 1: Local algorithm

```
for r_i(k) ∈ r(k) do
    p^i(k) ←ComputeMotionPrimitives(r_i(k)).
    Find targets that can be sensed by m-th motion primitive of
    i-th robot (p^i_m(k)).
    Construct a h-hop communication graph.
    Apply local algorithm (Floréen et al. 2011).
    x̂^i_m ← Rounding(x^i_m).
    p^{i*}_m(k) ← Motion primitive with x̂^i_m = 1.
    ApplyAction(p^{i*}_m(k)).
    k ← k + 1.
end
```

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 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 Tokekar et al. (2014) 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).

Table 1 Solution returned by the local algorithm for the example shown in Fig. 3, with all edges' weights set to 1, as a function of h

$\mathbf{p}_m^i$	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

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 (Table 1).

The above table shows the result of applying the local algorithm to the graph in Fig. 3 when all edge weights were set to 1. 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 2 ($\mathbf{r}_2$) and the robot 3 ($\mathbf{r}_3$) will select the motion primitive 3 ($\mathbf{p}_3^2$) and the motion primitive 6 ($\mathbf{p}_6^3$), 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 1 has the same equal value on both motion primitives of its own no matter how many number of layers are used. This is because all the targets are already observed by robots 2 and 3 with higher values.

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. If the communication graph is disconnected and forms more than one connected component, the greedy algorithm can independently be applied to each connected component without modifying the algorithm. Note again that the greedy algorithm is for the WINNERTAKESALL version of SATA.

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

Algorithm 2: Greedy algorithm

```
   Input : Order of robots R.
1  Initialize w(t_j) = 0 ∀j ∈ T.
2  for i ∈ R do
3      for m ∈ P^i do
4          Compute c^j_{i,m} ∀j ∈ T.
5          w'(p^i_m) = Σ_j max{w(t_j), c^j_{i,m}}.
6      end
7      Determine x^i_m = argmax w'(p^i_m) ∀m ∈ P^i.
8      Update w(t_j) = max{w(t_j), c^j_{i,m}} ∀j ∈ T.
9  end
10 y^j_i ← 0 ∀i ∈ R, j ∈ T.
11 for j ∈ T do
12     r*_i ← argmax_{i∈R} Σ_m c^j_{i,m} x^i_m.
13     y^j_{i*} ← 1.
14 end
```

even if multiple primitives can track the same target 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 Eq. (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 SATA.*

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 (Nemhauser et al. 1978).

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 carried out four types of simulations to verify the efficacy of the proposed algorithms under the condition that the amount of time required for communication is limited.

First, we compare the performance of the greedy and local algorithms with centralized, optimal solutions. Second, we study the effect of varying the parameters (i.e., the number of layers) for the local algorithm. Third, we describe how to implement the algorithms for sequential planning over multiple horizons and evaluate their performance over time. Last, we compare the greedy algorithm with a state-of-the-art distributed tracking algorithm.

5.1 Comparisons with centralized solutions

We performed comparison studies to verify the performance of the proposed algorithms. We compared the greedy solution with the optimal, centralized QMILP solution as well as a random algorithm as a baseline for the WINNERTAKESALL version. We compared the local algorithm's solution with the optimal ILP solution as well as the LP with rounding for BOTTLENECK. For these comparisons, we assumed that there are only two primitives to choose from (making the random algorithm a powerful baseline). We later analyzed the algorithms with more primitives. We used Tomlab (2017) 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 16 GB of memory the maximum time to solve was around 3 s on a case with 150 targets. Most of our cases were solved in less than 2 s.

We randomly generated graphs similar to Fig. 3 for the comparison. To control the topology of the randomly generated graphs, we defined $\phi : \mathcal{G}_S \rightarrow \mathbb{R}$ 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 started with the upper half of the graph, connecting each robot to its two motion primitives. Then, we iterated through each of motion primitive and randomly chose a target node to create an edge. Next, we iterated through target nodes and randomly chose a motion primitive to create an edge. We also added random edges to connect disconnected components (to keep the implementation simpler). We repeated this in order to get the required graph. If we needed to increase the degree of target nodes in the graph, we created new edges to random primitives till we achieved 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., $c^j_{i,m} \in \{0, 1\}$ for all cases.

The comparative simulation results for WINNERTAKESALL-WINNERTAKESALL are presented in Fig. 6. 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. The difference between the greedy algorithm and the baseline becomes smaller as $\phi(\mathcal{G}_S)$ increases. This could be because of the fact that the baseline saturates at the maximum objective value with fewer robots as $\phi(\mathcal{G}_S)$ increases. As $\phi(\mathcal{G}_S)$, number of targets, and number of robots increase, the performance of the greedy algorithm also improves.

Figure 7 shows the comparison results for BOTTLENECK where the objective values were computed from the w term of Eq. (5). As the proposed local algorithm is a linear relaxation of the ILP formulation, we compared 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. The specific configuration of input motion primitives and target states is such that no matter what motion primitives are chosen, at least one target will be left uncovered. This means that the bottleneck objective (i.e., the optimal value of ILP) is 0. 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

We analyzed the performance of the local algorithm for different number of layers (i.e., h), as shown in Fig. 8. The LP value (without rounding) is the upper bound on the optimal solution. We observed how much the rounding sacrifices by comparing the LP with and without the rounding. In the case where h was set to 5 and 8 for both with and without the rounding, 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 time

The greedy and local algorithms find the motion primitives to be applied over a single horizon. In order to track over time, the SATA problem will need to be solved repeatedly at each time step. In this section, we describe how to address this and

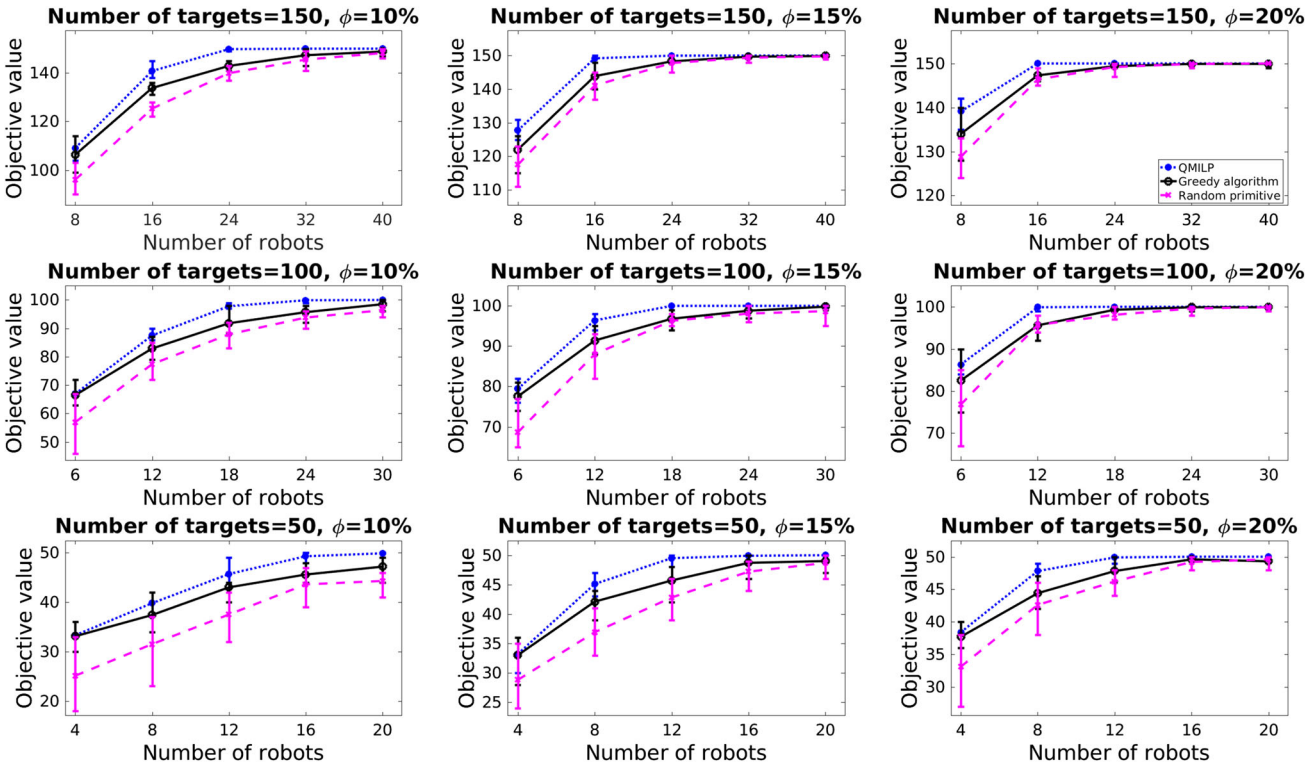


Fig. 6 Showing the comparative results of QMILP, greedy algorithm, and randomly choosing a motion primitive for WINNERTAKESALL. To generate the graphs, we varied number of robots, total number of targets, and $\phi(\mathcal{G}_S)$. We ran 100 trials for each case

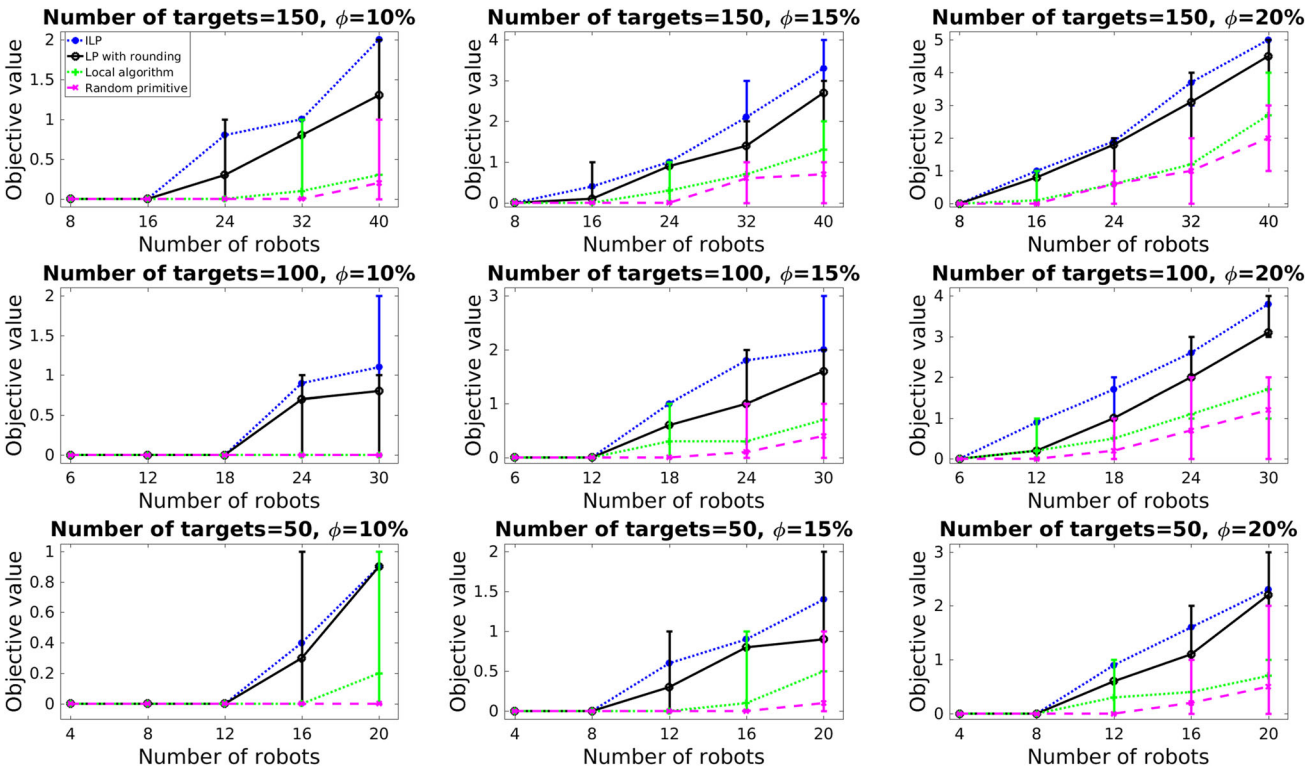
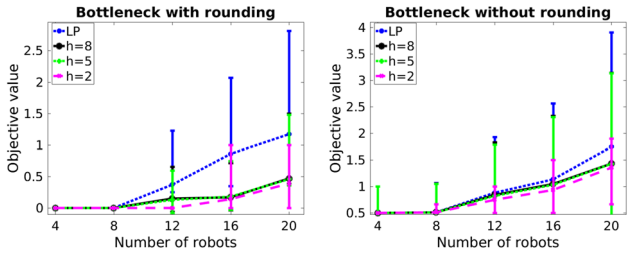


Fig. 7 Comparison simulation for the BOTTLENECK version of the ILP, LP with rounding, local algorithm and randomly choosing a motion primitive. We set h to 2 in the local algorithm, for all cases. Each case was obtained from 100 trials



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

Fig. 8 Analysis of varying the 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

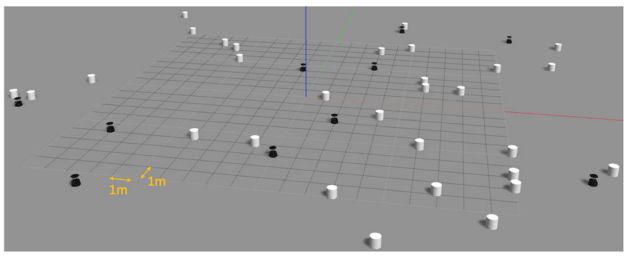


Fig. 9 Gazebo simulator showing ten robots tracking thirty randomly moving targets. We set the sensing and communication ranges to 5 m and 10 m, respectively

other challenges associated with a practical implementation. We demonstrate a realistic scenario of cooperative multi-target tracking in the Gazebo simulator using ROS (Fig. 9). A bounded environment consists of 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.

Greedy Algorithm We implemented the greedy algorithm to solve the WINNERTAKESALL variant in a fully distributed fashion. There was no centralized algorithm and each robot was a separate ROS node that only had access to the local information. Each robot had its local estimator that estimated the state of targets within its FOV. We simulated 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

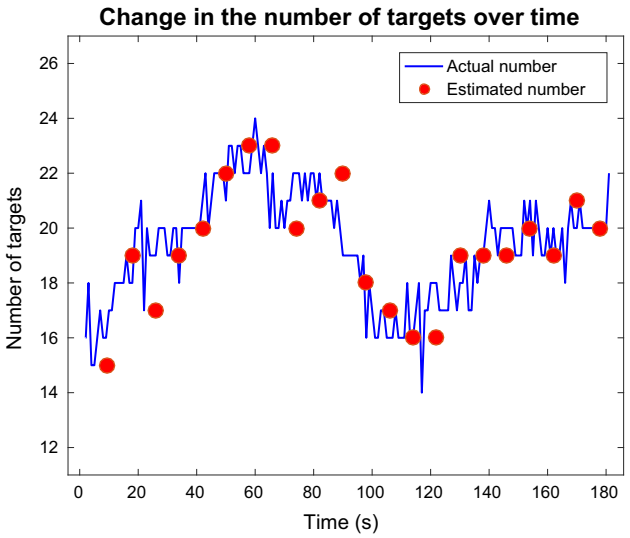


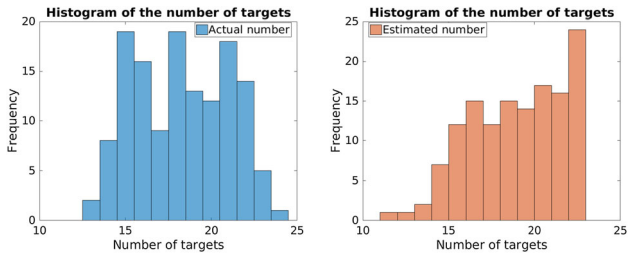
Fig. 10 Change in the number of targets over time when ten robots are tracking thirty moving targets

keeps track of the most recent $w(\mathbf{t}_j)$ vector received from a predecessor robot within 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 (Niehsen 2002).

For simulations we set the selection and execution periods times to $0.2|R|s$ and 6 s, respectively, where $|R|$ is the number of robots. Each robot made its choice after 0.2 s within the selection period. Each robot had a precomputed library of 21 motion primitives including staying in place. It should be noted that our algorithms do not require a motion primitive of stay in place. Each robot had a disk-shaped FOV. The sensing and communication ranges were set to 5 m and 10 m, respectively. We tested both the inverse of the distance and the number of targets as tracking quality (which defines $c^j_{i,m}$).

We carried out simulations using ten robots tracking thirty moving targets, as shown in Fig. 9. Initial positions of robots and targets were randomly chosen in a 30×30 m square environment. It may be possible that some targets were outside the FOV of any robots in the beginning.

Figure 10 shows the change in the number of targets over time from a randomly generated instance where the objective was to track the most number of targets. We show both the estimated number of targets 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 8 s). This is



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

Fig. 11 Histogram of the number of targets

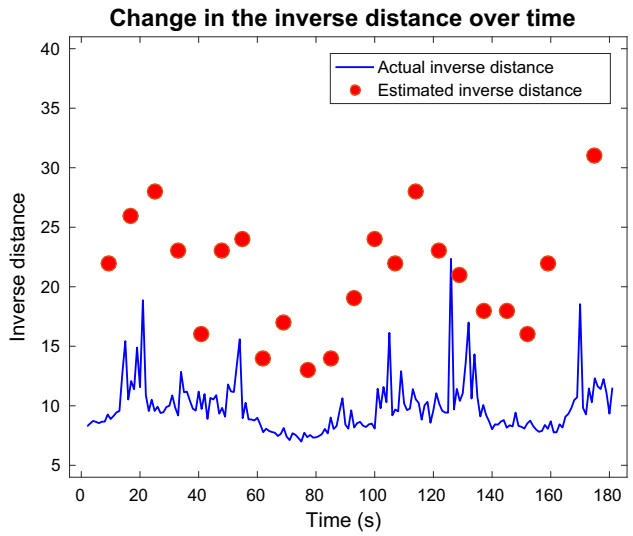
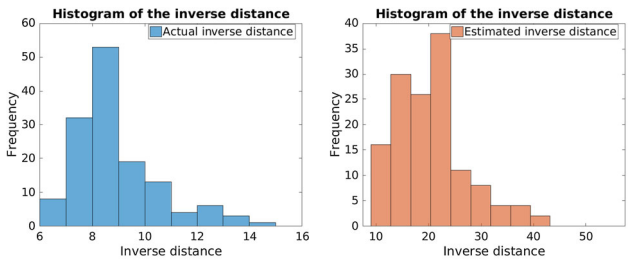


Fig. 12 Change in the inverse of the distance over time when ten robots are tracking thirty moving targets



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

Fig. 13 Histogram of the inverse distance

based on the predicted trajectory of the targets.[5] The actual number of targets was found by counting the target that is within the FOV of any robots during the execution period. Figure 11 shows the histogram of the actual and estimated number of targets for 10 trials, each lasting three minutes.

Figures 12 and 13 show the corresponding plots when the objective was to maximize the total quality of tracking

[5] Although we model linear motion for the targets, more sophisticated models for the prediction of target states can also be employed.

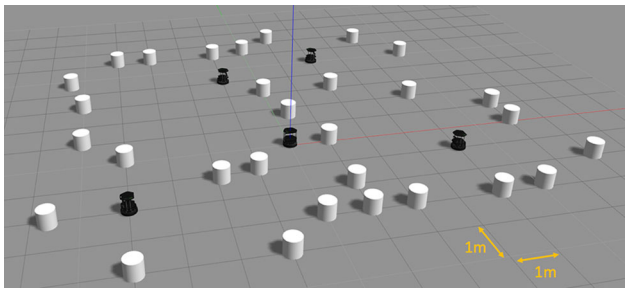


Fig. 14 Snapshot of the Gazebo simulator that shows when five robots are tracking thirty stationary and moving targets. The sensing and communication ranges were set to 3 m and 6 m, respectively

(inverse distance to the targets). Here, we saw that the estimated and the actual values differed much more than the previous case. We conjectured that this was due to the fact that the uncertainty in the motion model of the robots, targets, and measurements had 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 larger than estimated.

Local algorithm We also implemented the proposed local algorithm as shown in Fig. 14. Five mobile robots were deployed to track thirty targets (a subset of which were mobile) with a FOV of 3 m on the xy plane. For each robot two motion primitives were used: one was to remain in the same position and the other one was randomly generated between $-30°$ and $30°$ of the robot's heading traveling randomly up to $1m$.

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 was employed to choose motion primitives that maximize the minimum number of targets being observed by any robots.

Figure 15 shows the resultant trajectories of robots and targets obtained from the simulation. Figure 16 presents the (total/average) 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 to the optimal path.

5.4 Comparison of the greedy algorithm with other CMOMMT algorithm

We compared the greedy algorithm with an algorithm proposed by Parker Parker (2002) following the CMOMMT approach. This algorithm addresses the same objective as the WINNERTAKESALL. Parker's algorithm computes a local force vector for all robots (attraction by nearby targets and repul-

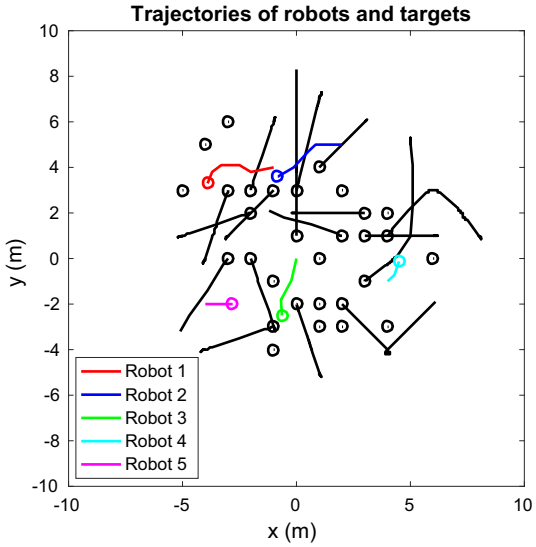
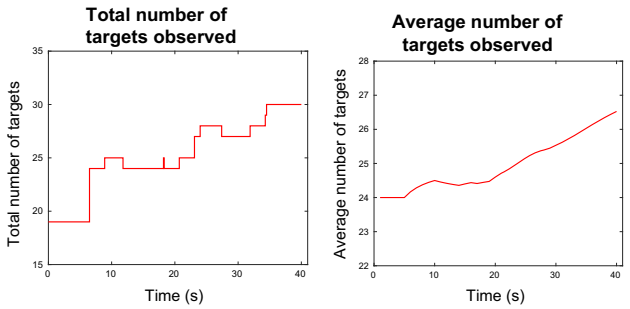


Fig. 15 Plot of trajectories of robots and targets applying the local algorithm to the simulation given in Fig. 14. Black lines represent trajectories of thirty targets. ○ denotes the end position of trajectories. The algorithm was performed for 40 s



(a) Total number of targets observed.

(b) Average number of targets observed over time.

Fig. 16 Change in the total and average number of targets being observed by any robots over time

sion by other nearby robots). It does not require any central unit to determine their motion primitives and considers limited sensing and communication ranges, similar to this paper. Parker's algorithm determines the moving direction of robots by using the local force vector and moves the robots along this direction until they meet the available maneuverability at each time step. However, no theoretic guarantee with respect to the optimal solution was provided by this algorithm.

We created an environment of 200 × 200 m square for comparison using MATLAB. The robots can move $10m$ per time step while the targets can move 5 m per time step and randomly changed their direction every 25 time steps. If the targets met the boundary of the environment, they picked a random direction that kept them within the environment. In each instance, robots and targets were randomly located initially. The sensing and communication ranges were set to 40 m and 80 m, respectively.

We empirically studied two cases: the first is to evaluate the objective value of the proposed greedy algorithm and Parker's algorithm for the same problem instance at a given time step; and the second is to apply the two algorithms over 200 time steps starting from the same configuration.

When both algorithms were applied to the same problem setup (Fig. 17a), the objective values for both algorithms increased as the number of targets increased. Nevertheless, the greedy algorithm outperformed Parker's algorithm. This can be attributed to the fact that Parker's algorithm computes the local force vector based on a heuristic (get closer to the targets) but does not explicitly optimize the objective function of `WINNERTAKESALL`. In Fig. 17b, c, similar results can be seen when both algorithms generate different trajectories for robots after 200 time steps. The comparison measure used in Fig. 17c is the average of the objective value over time, first proposed by Parker (2002). These empirical simulations show the superior performance of the greedy algorithm over the existing method.

In summary, we find that our algorithms perform comparably with centralized, optimal algorithms and outperform the baseline algorithm. We also find that greedy algorithm has better performance than the decentralized algorithm from Parker (2002). In theory, the performance bound for the local algorithm worsens as h, the amount of communication available, decreases. However, in practice, we find that the local algorithm does not require a large number of layers to yield good performance, which reduces the computational and communication burden.

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. (2011) 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 both versions of SATA over multiple horizons. In principle, we can replace each motion primitive input with a longer horizon trajectory and plan for multiple time steps (say, H time steps). However, this comes at the expense of increased number of trajectories $|P^i|^H$ to choose from which will result in increased computational time. Further-

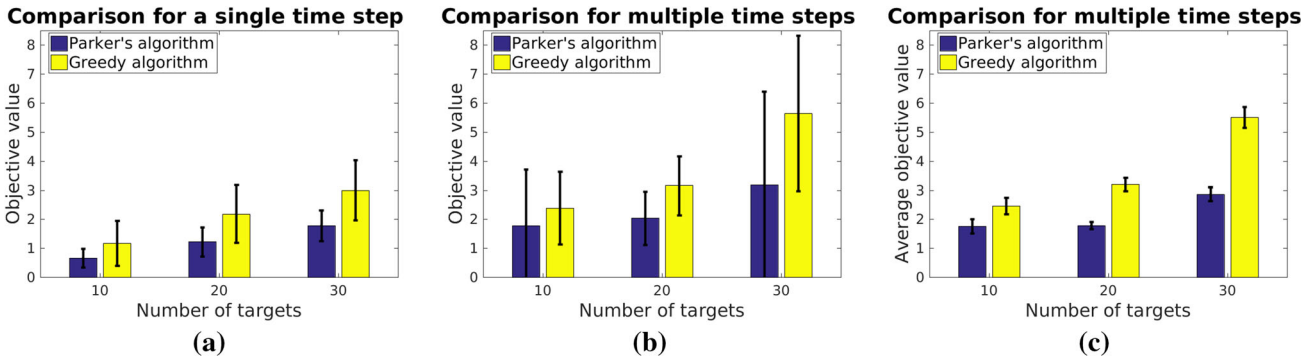


Fig. 17 Comparison with the Parker's algorithm Parker (2002). **a** 200 instances were run. **b** 200 time steps were run. **c** 200 time steps were run to compare the metric proposed by Parker (2002). We used 10 robots for all cases. We ran 10 trials for **b** and **c**. Bar graphs show the mean and standard deviation for different number of targets (10, 20 and 30 targets)

more, planning for a longer horizon will require prediction of targets' states far in the future which can lead to poorer tracking performance. We are also working on implementing the resulting algorithms on actual aerial robotic systems to carry out real-world experimentation.

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

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:

$$\begin{aligned} &\max_{x_m^i} \min_{j\in T} \left(\sum_{i\in R} \sum_{m\in P^i} c_{i,m}^j x_m^i \right) \\ &\text{subject to} \quad \sum_{m\in P^i} x_m^i \le 1 \quad \forall i \in R \\ &\qquad\qquad x_m^i \ge 0 \quad \forall m \in P^i. \end{aligned} \tag{7}$$

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

$$w^* = \min_{j\in T} \left(\sum_{i\in R} \sum_{m\in P^i} c_{i,m}^j {x_m^i}^* \right). \tag{8}$$

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

B Proof of lemma 2

Considering $c_{i,m}^j$, 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:

$$w(\mathbf{t}_j) \triangleq \max\{c_{i,m}^j | x_m^i = 1,\ \forall i \in R, m \in P^i\}. \tag{9}$$

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

$$\begin{aligned} \sum_{j\in T} w(\mathbf{t}_j) &= \sum_{j\in T} \max\{c_{i,m}^j | x_m^i = 1,\ \forall i \in R, m \in P^i\} \\ &= \sum_{j\in T} \Big(\sum_{i\in R} y_i^j \Big(\sum_{m\in P^i} c_{i,m}^j x_m^i \Big) \Big). \end{aligned} \tag{10}$$

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_i^j chooses the maximum value of $\sum_{m\in P^i} c_{i,m}^j x_m^i$ among all robots, which is shown in lines 10–14 of Algorithm 2. Therefore, the last equation is equal to the inner term of Eq. (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 ($\mathbf{r}_i$) having two motion primitives ($\mathbf{p}_m^i$) 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.

	$\mathbf{p}_1^1, \mathbf{p}_1^2$	$\mathbf{p}_2^1, \mathbf{p}_2^2$
$\mathbf{r}_1$	$\mathbf{t}_1$	$\varnothing$
$\mathbf{r}_2$	$\varnothing$	$\mathbf{t}_2$

Let's apply the BOTTLENECK version of greedy algorithm to this case. Since the objective of the BOTTLENECK variant is to maximize the minimum tracking quality, the robot 1 ($\mathbf{r}_1$) chooses motion primitive 2 ($\mathbf{p}_2^1$) because choosing motion primitive 1 ($\mathbf{p}_1^1$) gives the value of 1 while choosing motion primitive 2 ($\mathbf{p}_2^1$) gives the value of 0. For the same reason, the robot 2 ($\mathbf{r}_2$) chooses motion primitive 1 ($\mathbf{p}_1^2$). This gives the total value of 0, whereas the optimal solution is 2 as the first robot and second robot choose motion primitive 1 ($\mathbf{p}_1^1$) and motion primitive 2 ($\mathbf{p}_2^2$), respectively. The similar case is reproducible with a larger number of robots, motion primitives, and targets. Thus, the simple greedy performs arbitrarily badly for the BOTTLENECK variant.

References

Ahmad, A., Lawless, G., & Lima, P. (2017). An online scalable approach to unified multirobot cooperative localization and object tracking. *IEEE Transactions on Robotics*, *33*(5), 1184–1199.

Angluin, D. (1980) Local and global properties in networks of processors. In *Proceedings of the twelfth annual ACM symposium on theory of computing*. ACM, (pp. 82–93).

Åstrand, M., & Suomela, J. (2010) 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, (pp. 294–302).

Åstrand, M., Floréen, P., Polishchuk, V., Rybicki, J., Suomela, J., & Uitto, J. (2009) A local 2-approximation algorithm for the vertex cover problem. In *International symposium on distributed computing*. Springer (pp. 191–205).

Bandyopadhyay, S., Chung, S.-J., & Hadaegh, F. Y. (2017). Probabilistic and distributed control of a large-scale swarm of autonomous agents. *IEEE Transactions on Robotics*, *33*(5), 1103–1123.

Banfi, J., Guzzi, J., Amigoni, F., Flushing, E. F., Giusti, A., Gambardella, L., & Di Caro, G. A. (2018) An integer linear programming model for fair multitarget tracking in cooperative multirobot systems. *Autonomous Robots*, pp. 1–16.

Best, G., Forrai, M., Mettu, R. R., & Fitch, R. (2018). Planning-aware communication for decentralised multi-robot coordination. In *Proceedings of the international conference on robotics and automation*, Brisbane, Australia, (Vol. 21).

Capitan, J., Spaan, M. T., Merino, L., & Ollero, A. (2013). Decentralized multi-robot cooperation with auctioned pomdps. *The International Journal of Robotics Research*, *32*(6), 650–671.

Charrow, B., Kumar, V., & Michael, N. (2014). Approximate representations for multi-robot control policies that maximize mutual information. *Autonomous Robots*, *37*(4), 383–400.

Choi, H.-L., Brunet, L., & How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, *25*(4), 912–926.

Chung, S.-J., Paranjape, A., Dames, P., Shen, S., & Kumar, V. (2018). A Survey on Aerial Swarm Robotics. *IEEE Transactions on Robotics*.

Dimarogonas, D. V., Frazzoli, E., & Johansson, K. H. (2012). Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, *57*(5), 1291–1297.

Floréen, P., Hassinen, M., Kaasinen, J., Kaski, P., Musto, T., & Suomela, J. (2011). Local approximability of max-min and min-max linear programs. *Theory of Computing Systems*, *49*(4), 672–697.

Ge, X., & Han, Q.-L. (2017). Distributed formation control of networked multi-agent systems using a dynamic event-triggered communication mechanism. *IEEE Transactions on Industrial Electronics*, *64*(10), 8118–8127.

Gerkey, B. P., & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, *23*(9), 939–954.

Ge, X., Yang, F., & Han, Q.-L. (2017). Distributed networked control systems: A brief overview. *Information Sciences*, *380*, 117–131.

Gharesifard, B. & Smith, S. L. (2017). Distributed submodular maximization with limited information. In *IEEE transactions on control of network systems*.

Guo, M., & Zavlanos, M. M. (2018). Multirobot data gathering under buffer constraints and intermittent communication. *IEEE transactions on robotics*.

Hanckowiak, M., Karonski, M., & Panconesi, A. (2001). On the distributed complexity of computing maximal matchings. *SIAM Journal on Discrete Mathematics*, *15*(1), 41–57.

Hönig, W., & Ayanian, N. (2016) Dynamic multi-target coverage with robotic cameras. In *IEEE RSJ International conference on intelligent robots and systems (IROS)* (pp. 1871–1878).

Howard, T., Pivtoraiko, M., Knepper, R. A., & Kelly, A. (2014). Model-predictive motion planning: Several key developments for autonomous mobile robots. *IEEE Robotics and Automation Magazine*, *21*(1), 64–73.

Kanakia, A., Touri, B., & Correll, N. (2016). Modeling multi-robot task allocation with limited information as global game. *Swarm Intelligence*, *10*(2), 147–160.

Kantaros, Y., Thanou, M., & Tzes, A. (2015). Distributed coverage control for concave areas by a heterogeneous robot-swarm with visibility sensing constraints. *Automatica*, *53*, 195–207.

Kantaros, Y., & Zavlanos, M. M. (2016). Global planning for multi-robot communication networks in complex environments. *IEEE Transactions on Robotics*, *32*(5), 1045–1061.

Kantaros, Y., & Zavlanos, M. M. (2017). Distributed intermittent connectivity control of mobile robot networks. *IEEE Transactions on Automatic Control*, *62*(7), 3109–3121.

Kassir, A., Fitch, R., & Sukkarieh, S. (2016) Communication-efficient motion coordination and data fusion in information gathering teams. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, (pp. 5258–5265).

Khan, A., Rinner, B., & Cavallaro, A. (2016) Cooperative robots to observe moving targets: Review, *IEEE transactions on cybernetics*.

Kolling, A., & Carpin, S. (2007). Cooperative observation of multiple moving targets: an algorithm and its formalization. *The International Journal of Robotics Research*, *26*(9), 935–953.

Korsah, G. A., Stentz, A., & Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, *32*(12), 1495–1512.

Kuhn, F., Moscibroda, T., & Wattenhofer, R. (2006) The price of being near-sighted. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, (pp. 980–989).

Le Ny, J., Ribeiro, A., & Pappas, G. J. (2012). Adaptive communication-constrained deployment of unmanned vehicle systems. *IEEE Journal on Selected Areas in Communications*, *30*(5), 923–934.

Lenzen, C., & Wattenhofer, R. (2010) Minimum dominating set approximation in graphs of bounded arboricity. In *International symposium on distributed computing*. Springer, (pp. 510–524).

Li, H., Chen, G., Huang, T., & Dong, Z. (2017). High-performance consensus control in networked systems with limited bandwidth communication and time-varying directed topologies. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(5), 1043–1054.

Linial, N. (1992). Locality in distributed graph algorithms. *SIAM Journal on Computing*, *21*(1), 193–201.

Liu, L., & Shell, D. A. (2011). Assessing optimal assignment under uncertainty: An interval-based algorithm. *The International Journal of Robotics Research*, *30*(7), 936–953.

Luo, L., Chakraborty, N., & Sycara, K. (2015). Distributed algorithms for multirobot task assignment with task deadline constraints. *IEEE Transactions on Automation Science and Engineering*, *12*(3), 876–888.

Morgan, D., Subramanian, G. P., Chung, S.-J., & Hadaegh, F. Y. (2016). Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, *35*(10), 1261–1285.

Naor, M., & Stockmeyer, L. (1995). What can be computed locally? *SIAM Journal on Computing*, *24*(6), 1259–1277.

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions–1. *Mathematical programming*, *14*(1), 265–294.

Niehsen, W. (2002) Information fusion based on fast covariance intersection filtering. In *Proceedings of the fifth international conference on information fusion, 2002*, vol. 2. IEEE, (pp. 901–904).

Otte, M., & Correll, N. (2013). Any-com multi-robot path-planning: Maximizing collaboration for variable bandwidth. In A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. A. Hsieh, L. E. Parker, & K. Støy (Eds.), *Distributed autonomous robotic systems* (pp. 161–173), Springer.

Otte, M., Kuhlman, M., & Sofge, D. (2017) Multi-robot task allocation with auctions in harsh communication environments. In *International symposium on multi-robot and multi-agent systems (MRS) 2017*. IEEE, (pp. 32–39).

Otte, M., & Correll, N. (2018). Dynamic teams of robots as ad hoc distributed computers: Reducing the complexity of multi-robot motion planning via subspace selection. *Autonomous Robots*, *42*(8), 1691–1713.

Parker, L.E., & Emmons, B. A. (1997) Cooperative multi-robot observation of multiple moving targets. In *Proceedings IEEE International conference on robotics and automation*, vol. 3 (pp. 2082–2089).

Parker, L. E. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous robots*, *12*(3), 231–255.

Pimenta, L. C., Schwager, M., Lindsey, Q., Kumar, V., Rus, D., Mesquita, R. C., & Pereira, G. A. (2009). Simultaneous coverage and tracking (scat) of moving targets with robot networks. In G. S. Chirikjian, H. Choset, M. Morales, & T. Murphey (Eds.), *Algorithmic foundation of robotics VIII* (pp. 85–99). Springer.

Robin, C., & Lacroix, S. (2016). Multi-robot target detection and tracking: Taxonomy and survey. *Autonomous Robots*, *40*(4), 729–760.

Sung, Y., Budhiraja, A. K., Williams, R. K., & Tokekar, P. (2018) Distributed simultaneous action and target assignment for multi-robot multi-target tracking. In *2018 IEEE International conference on robotics and automation (ICRA)* (pp. 1–9).

Suomela, J. (2013). Survey of local algorithms. *ACM Computing Surveys (CSUR)*, *45*(2), 24.

Tokekar, P., Isler, V., & Franchi, A. (2014) Multi-target visual tracking with aerial robots. In *2014 IEEE RSJ International conference on intelligent robots and systems* (pp. 3067–3072).

Tomlab: Optimization environment large-scale optimization in matlab. http://tomopt.com/docs/quickguide/quickguide006.php, Accessed 3 Jan 2017.

Touzet, C. F. (2000). Robot awareness in cooperative mobile robot learning. *Autonomous Robots*, *8*(1), 87–97.

Turpin, M., Michael, N., & Kumar, V. (2014). Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, *33*(1), 98–112.

Vander Hook, J., Tokekar, P., & Isler, V. (2015). Algorithms for cooperative active localization of static targets with mobile bearing sensors under communication constraints. *IEEE Transactions on Robotics*, *31*(4), 864–876.

Vazirani, V. (2001). *Approximation algorithms*. Berlin: Springer.

Williams, R. K., Gasparri, A., Sukhatme, G. S., & Ulivi, G. (2015) Global connectivity control for spatially interacting multi-robot systems with unicycle kinematics. In *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, (pp. 1255–1261).

Williams, R. K., & Sukhatme, G. S. (2013). Constrained interaction and coordination in proximity-limited multiagent systems. *IEEE Transactions on Robotics*, *29*(4), 930–944.

Xu, Z., Fitch, R., Underwood, J., & Sukkarieh, S. (2013). Decentralized coordinated tracking with mixed discrete-continuous decisions. *Journal of Field Robotics*, *30*(5), 717–740.

Yan, Z., Jouandeau, N., & Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, *10*(12), 399.

Young, N. E. (2001) Sequential and parallel algorithms for mixed packing and covering. In *Proceedings 42nd IEEE symposium on foundations of computer science* (pp. 538–546).

Yu, H., Meier, K., Argyle, M., & Beard, R. W. (2015). Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles. *IEEE/ASME Transactions on Mechatronics*, *20*(2), 541–552.

Zavlanos, M. M., Egerstedt, M. B., & Pappas, G. J. (2011). Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, *99*(9), 1525–1540.

Zhou, K., Roumeliotis, S. I., et al. (2011). Multirobot active target tracking with combinations of relative observations. *IEEE Transactions on Robotics*, *27*(4), 678–695.

Zhou, L., & Tokekar, P. (2018). Active target tracking with self-triggered communications in multi-robot teams. *IEEE Transactions on Automation Science and Engineering*, *99*, 1–12.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Yoonchang Sung received the B.S. and M.S. degrees from the Department of Mechanical Engineering, Korea University, Seoul, Korea, in 2011 and 2013, respectively. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA. His main research interests include algorithmic robotics, path planning, computational geometry, and cooperative robotics.

Ashish Kumar Budhiraja received the B.S. and M.S. degrees from IIT Bombay, India and Virginia Tech, Blacksburg, VA, USA, respectively. He is currently working at Apple.

Ryan K. Williams is an Assistant Professor in the Department of Electrical and Computer Engineering at Virginia Polytechnic Institute and State University. He received the BS degree in computer engineering from Virginia Polytechnic Institute and State University and the Ph.D. degree in electrical engineering from the University of Southern California, in 2005 and 2014, respectively. His research interests include control, cooperation, and intelligence in distributed multi-node systems, topological methods in cooperative phenomena, and distributed algorithms for optimization, estimation, inference, and learning. He received the Viterbi Fellowship, and has been featured by various news outlets, including the L.A. Times, and has a patent pending for his work on high-speed AUVs. He is a member of the IEEE.

Pratap Tokekar is an Assistant Professor in the Department of Electrical and Computer Engineering at Virginia Tech. Previously, he was a Postdoctoral Researcher at the GRASP lab of University of Pennsylvania. He obtained his Ph.D. in Computer Science from the University of Minnesota in 2014 and Bachelor of Technology degree in Electronics and Telecommunication from College of Engineering Pune, India in 2008. He is a recipient of the NSF CISE Research Initiation Initiative award. His research interests include algorithmic and field robotics, and cyber physical systems, and their applications to precision agriculture and environmental monitoring.