Survey of Distributed Optimization Algorithms in Multi-Agent Robotics: Challenges and Open Problems

Sreejeet Maity [*]

Abstract

The domain of robotics has undergone notable transformations propelled by the evolution of distributed optimization, offering innovative solutions to address challenges inherent in multi-robot systems. It stands out as a versatile framework, providing tailored algorithms for a broad class of pivotal problems of fundamental interest. An exploration of open research challenges in distributed optimization, particularly within the context of multi-agent robotics, enriches scholarly and industrial discussions, underscoring its enduring relevance in applied fields. Despite the mature state of distributed optimization, literature specifically dedicated to its application in multi-robot scenarios remains relatively scarce. This paper undertakes a comprehensive investigation into the realm of distributed optimization applied to multi-robot systems.

The survey meticulously navigates through three primary classes of optimization algorithms in the field of robotics: distributed first-order methods(DFO), distributed sequential convex programming methods, and alternating direction methods of multipliers (ADMM). The distinctive focus is directed towards fully distributed methods, obviating the need for centralized coordination or computation. The foundational structure of each algorithm class is explicated, accentuating essential variations devised to address associated limitations. Additionally, the paper scrutinizes the practical implications of critical assumptions inherent in these algorithms, delineating the classes of robotics problems for which they exhibit optimal suitability.With the overarching objective of propelling the application of distributed optimization in robotics, the paper asserts the seamless alignment of canonical multi-robot problems with the distributed optimization framework. Examples include multi-robot simultaneous localization and planning (SLAM), multi-robot target tracking, and multi-robot task assignment problems. Offering a comprehensive description of each class's fundamental structure and presenting representative algorithms within, the paper culminates in a simulation case study involving multiple agents. Furthermore, in the practical deployment of multi-robot systems, the vulnerability of communication channels or decentralized broadcasters to malicious modifications introduces a critical dimension to system dynamics. Consequently, ensuring robustness against adversarial attacks emerges as a paramount consideration for the real-time deployment of robotic swarms in mission-critical scenarios. This paper aims to discuss this extremely practical yet often-ignored assumption to alleviate the issues in the popular vanilla algorithms. In this way, a short discussion on the robustness of these algorithms should follow each algorithm section.To sum up, this project endeavors to present a meticulous survey of the application of distributed optimization in addressing both adversarial and non-adversarial settings. A simulation case study in the `PyGame` robotic environment, where multiple drones collaboratively track a ground vehicle, serves as a practical demonstration. This study not only facilitates an examination of the efficacy of various optimization algorithms in the collaborative tracking task but also contributes valuable insights to advance our understanding of the practical implications and performance metrics in the realm of multi-robot systems.

[*]Sreejeet Maity is with the Department of Electrical Engineering at North Carolina State University. Email: `smaity2@ncsu.edu`.

1 Introduction

Distributed optimization, a foundational discipline in mathematical optimization, has recently emerged as a transformative approach in addressing intricate coordination and collaboration challenges within multi-robot systems. This first part of the paper endeavors to elucidate the foundational aspects of distributed optimization, delineating its framework, essential components, and versatile applicability across a spectrum of multi-robot challenges. Simultaneously, it sets the stage for the subsequent section, which embarks on a comprehensive survey of existing distributed optimization methods. This survey not only showcases their diverse applications in multi-robot scenarios but also outlines unresolved research challenges, thereby paving the way for continued advancements in the field.

In the distributed optimization framework, robots function as computation nodes, collaboratively optimizing a shared decision variable while adhering to individual local objective and constraint functions. The joint objective function aggregates the individual robots' local objectives, and constraints amalgamate their respective local constraints. This decentralized approach empowers each robot to optimize locally, sharing only partial knowledge of the joint optimization problem with its immediate neighbors through a one-hop communication network. This model aligns with prevalent connectivity scenarios in robotics problems, deliberately excluding algorithms requiring multi-hop communication or a hub-spoke network topology for enhanced practicality and real-world applicability.

A distinctive feature of distributed optimization in multi-robot systems is the inherent preservation of privacy. Each robot possesses knowledge solely of its local objective and constraint functions, ensuring the confidentiality of sensitive information. This privacy-preserving property is crucial in scenarios involving collaborative motion planning or cooperative estimation, fostering trust among robotic agents and facilitating the deployment of these algorithms in real-world applications.

The applications of distributed optimization in multi-robot systems are both varied and extensive, encompassing simultaneous localization and mapping (SLAM), target tracking, task assignment, collaborative trajectory planning, and multi-robot learning. The paper offers comprehensive insights into methodologies for reformulating these problems into forms amenable to distributed optimization algorithms, emphasizing the flexibility and efficacy of optimization-based approaches in addressing diverse challenges within the field of robotics.

The survey categorizes distributed optimization algorithms into three main classes: distributed first-order methods, distributed sequential convex programming, and the alternating direction method of multipliers (ADMM). Each class is characterized by a distinct mathematical technique, and representative algorithms within each category are presented to elucidate the breadth and depth of approaches available to researchers and practitioners in the field.

Distributed first-order methods leverage local gradient information to optimize the joint objective function, making them well-suited for scenarios where the joint objective is smooth and convex. Distributed sequential convex programming methods come into play in non-convex scenarios, approximating the problem iteratively with convex subproblems. The Alternating Direction Method of Multipliers (ADMM) stands out as a powerful technique, decomposing problems into smaller sub-problems and showcasing flexibility and efficiency in multi-robot systems.

The tracking problem is formulated as a distributed optimization task, wherein each agent optimizes its trajectory to collectively track ground vehicles. The simulator is employed for implementation, demonstrating the efficacy of the distributed optimization algorithm in real-time decision-making. The simulation showcases the convergence of drones to a common solution for optimal ground vehicle tracking, addressing challenges such as real-time decision-making, dynamic environment changes, and communication constraints.

Building on the groundwork laid in the first part of this two-part series, the study underscores the potential of distributed optimization algorithms in solving multi-robot problems formulated as a sum of local objective functions subject to a union of local constraint functions. The paper explores the robustness and privacy-preserving properties of distributed optimization, emphasizing its relevance in the realm of robotics. Acknowledging that distributed optimization is an underutilized approach in robotics, this survey identifies various research opportunities in this space.

As the second part of a series on distributed optimization for multi-robot systems, this survey delves into relevant algorithms, and their applicability to distinct classes of robotics problems, and addresses the practical implications and challenges associated with their implementation on robotics platforms. The survey categorizes distributed optimization algorithms into three broad classes: Distributed First-Order Algorithms (DFO), Distributed Sequential Convex Programming, and Distributed Alternating Direction Method of Multipliers (ADMM) extensions.

`Distributed First-Order Algorithms (DFO)`: DFO methods leverage averaging local gradients computed by each computational node to perform an approximate gradient descent update. This class includes sub-categories such as distributed (sub)-gradient descent, distributed gradient tracking, distributed stochastic gradient descent, and distributed dual averaging algorithms. Despite their commonality in utilizing a consensus procedure for achieving robot agreement on a common solution, DFO algorithms exhibit versatility in handling dynamic communication networks and limited computation resources in robotics problems.

`Distributed Sequential Convex Programming`: This category employs Sequential Convex Optimization techniques, a common approach in centralized optimization, to minimize a sequence of convex approximations to the original non-convex problem. Distributed Sequential Convex Programming methods use consensus techniques to construct convex approximations of the joint objective function. While some algorithms within this class may not be suitable for problems with dynamic communication networks, others, like the NEXT family of algorithms, demonstrate flexibility in optimizing non-convex objective functions, specifically designed for such scenarios.

`Alternating Direction Method of Multipliers (ADMM)`: ADMM, a prominent class of algorithms, works by minimizing the augmented Lagrangian of the optimization problem using alternating updates to the primal and dual variables. The method is naturally amenable to constrained problems. While the original ADMM requires a central computation hub, Consensus ADMM (C-ADMM) and other distributed variants have been developed to address the unique characteristics of robotics problems, including uni-directional communication networks and limited communication bandwidth. Several existing surveys on distributed optimization focus on diverse applications such as distributed power systems, big-data problems, and game theory. While others broadly address distributed first-order optimization methods or non-convex optimization, our survey stands out by specifically targeting applications of distributed optimization to multi-robot problems. By doing so, we highlight the practical implications of the assumptions made by many distributed optimization algorithms, providing a condensed taxonomic overview of useful methods for these applications.

In conclusion, this paper serves as an indispensable resource for robotics practitioners and researchers alike, offering a foundational understanding of distributed optimization in multi-robot systems. The insights provided not only pave the way for further explorations of the series, which delves deeper into existing distributed optimization methods and their applications but also stimulate further exploration of open research challenges. This fosters continued advancements in the dynamic intersection of distributed optimization and multi-robot systems, addressing real-world challenges and pushing the boundaries of what is possible in the realm of collaborative robotics. By categorizing algorithms and addressing their practical implications, the paper not only offers insights into the current state of the field but also identifies avenues for future research in the application of distributed optimization to robotics problems.

2 Notation and Preliminaries

In this section, we introduce the notational conventions employed in this scholarly work and furnish precise definitions for mathematical concepts integral to the discourse on distributed optimization algorithms.

The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is denoted as ∇f, and its Hessian is represented as $\nabla^2 f$. The vector comprised entirely of unit elements is denoted as $\mathbf{1}_n$, where n signifies the dimensionality of the vector.

We expound upon pertinent notions regarding the connectivity of graphs:

`Definition 1` : An undirected graph $\mathcal{G}$ is deemed connected if a path exists between every pair of vertices (i, j), where $i, j \in \mathcal{V}$. It is important to note that such a path may traverse intermediary vertices within $\mathcal{G}$.

`Definition 2 (Connectivity of a Directed Graph)`: For a directed graph $\mathcal{G}$, strong connectivity is asserted if a directed path is present between every pair of vertices (i, j), where $i, j \in \mathcal{V}$. Additionally, weak connectivity is affirmed if the underlying undirected graph is connected. The underlying undirected graph $\mathcal{G}_u$ of a directed graph $\mathcal{G}$ corresponds to a graph sharing the same vertex set as $\mathcal{G}$ and an edge set derived by considering each edge in $\mathcal{G}$ as a bidirectional edge. Consequently, every strongly connected directed graph is weakly connected, though the reverse does not hold.

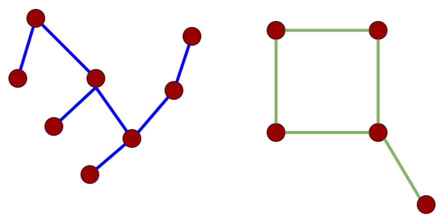

Figure 1: Agent Connectivity depicted by Undirected Graph

`Definition 3 (Stochastic Matrix)`: A non-negative matrix $W \in \mathbb{R}^{n \times n}$ is denoted a row-stochastic matrix if $W\mathbf{1}_n = \mathbf{1}_n$, implying that the sum of all elements in each row of the matrix equals one. Similarly, W is referred to as a column-stochastic matrix if $\mathbf{1}_n^\top W = \mathbf{1}_n^\top$. For a doubly-stochastic matrix W, both conditions $W\mathbf{1}_n = \mathbf{1}_n$ and $\mathbf{1}_n^\top W = \mathbf{1}_n^\top$ hold.

In the realm of distributed optimization within multi-robot systems, the orchestration of communication and computation steps among robots is pivotal for the joint minimization of an objective function. This study concentrates on scenarios where the exchange of information among robots adheres to the topology of an underlying distributed communication graph, potentially subject to temporal variations. This communication graph, denoted as $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$, comprises vertices $\mathcal{V}(t) = \{1, \ldots, N\}$ and edges $\mathcal{E}(t) \subseteq \mathcal{V}(t) \times \mathcal{V}(t)$, facilitating pairwise communication. For undirected graphs, the set of neighbors of robot i is denoted as $\mathcal{N}_i(t)$. In the context of directed graphs, the in-neighbors of robot i, capable of transmitting information to i, are denoted as $\mathcal{N}_i^+(t)$, and the out-neighbors, capable of receiving information from i, are denoted as $\mathcal{N}_i^-(t)$.

3 Problem Formulation

In the context of distributed optimization, we contemplate a scenario wherein each robotic entity $i \in \mathcal{V}$ possesses access to its individualized objective function $f_i : \mathbb{R}^n \to \mathbb{R}$. It is noteworthy that the local objective function of each robot is exclusively known to that particular robot, and there exists no knowledge regarding the local objective functions of other robots within the system. This formulation finds relevance in various robotics applications, where the local objective functions are contingent upon data gathered locally by each robot, typically in the form of sensor measurements. In addition to the exclusive knowledge of their respective local objective functions, it is presumed that constraints about the optimization variable are solely discerned at the local level. The robots collaboratively strive to address a collective optimization problem characterized by a separable objective function. Each component of this objective function is exclusively known to a single robot, encapsulating the essence of the problem at hand.

$$\begin{aligned} \min_{x} & \sum_{i \in \mathcal{V}} f_i(x_i) \\ \text{s.t} \quad & g_i(x) = 0 \, \forall i \in \mathcal{V} \\ & h_i(x) \leq 0 \, \forall i \in \mathcal{V} \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ signifies the collective optimization variable, with $g_i(x)$ representing the equality constraint function specific to robot i, and $h_i(x)$ denoting its corresponding inequality constraint function. It is essential to observe that not every robot is obligated to possess a local constraint function. In instances where a robot lacks such a function, the pertinent constraint functions are deliberately excluded from the overarching optimization problem.

In the domain of robotics, a confluence of challenges, including privacy considerations and constraints imposed by limited computational and communication resources, impedes the seamless exchange of local objective and constraint functions among robots. Consequently, our attention is directed towards the exploration of distributed algorithms tailored to empower each robot to ascertain the optimal solution to the collective problem while abstaining from the sharing of its local problem data and functions.

Within the purview of these distributed algorithms, each robot maintains a proprietary copy of the optimization variable, with x_i designated as the local optimization variable for robot i. Distributed optimization algorithms are instrumental in navigating a reformulation of the optimization problem, as follows:

$$\begin{aligned} \min_{\{x_i, \forall i \in \mathcal{V}\}} & \sum_{i \in \mathcal{V}} f_i(x_i) \\ \text{s.t} \quad & x_i = x_j \;\; \forall (i,j) \in \varepsilon \\ & g_i(x_i) = 0 \, \forall i \in \mathcal{V} \\ & h_i(x_i) \leq 0 \, \forall i \in \mathcal{V} \end{aligned} \tag{2}$$

The imposition of the consensus constraints, denoted as $x_i = x_j$ for all $(i,j) \in \varepsilon$, defines a pivotal element within the reformulation. This constraint, applicable universally across robot pairs, contributes to the elucidation of the problem at hand. Assuming a connected communication graph for undirected graphs and weak connectivity for directed graphs, the optimal cost within this reformulation mirrors that of the original problem. Furthermore, the minimizing arguments x_i^* derived from this reformulation align seamlessly with the minimizing argument x^* of the original problem, a correspondence that extends across all robots indexed by $i = 1, \ldots, n$.

Within the landscape of distributed optimization, several canonical robotics challenges find a natural representation. This section delves into the contemplation of five overarching problem categories amenable to resolution through distributed optimization methodologies. These categories encompass multi-robot Simultaneous Localization and Mapping (SLAM), multi-robot target tracking, multi-robot task assignment, collaborative planning, and multi-robot learning. It is noteworthy that, although the employment of an optimization-based approach for certain challenges may not be immediately apparent, we demonstrate that these problems readily lend themselves to formulation as distributed optimization challenges. This formulation is achieved through the introduction of auxiliary optimization variables, coupled with the incorporation of a judicious set of consensus constraints.

3.1 Multi-Robot Simultaneous Localization and Mapping (SLAM)

In the domain of multi-robot Simultaneous Localization and Mapping (SLAM) problems, a collective of robotic entities endeavors to iteratively refine their estimations of position and orientation, commonly referred to as pose, within a unified representation of the surrounding environment. Adopting a comprehensive landmark-based SLAM methodology entails the optimization of both the map features and the poses of the individual robots as follows:

$$\begin{aligned} \min_{x,m} \quad & \sum_{i=1}^{N} \sum_{t=0}^{T-1} \frac{1}{2} \|\bar{\mathbf{z}}^{i,t}(\mathbf{x}_{i,t}, \mathbf{x}_{i,t+1}) - \hat{\mathbf{z}}_{i,t+1}\|^2_{\Omega_{i,t}} \\ & + \sum_{i=1}^{N} \sum_{k=1}^{M} \frac{1}{2} \|\tilde{\mathbf{z}}_i^k(\mathbf{x}_i, \mathbf{m}_k) - \mathbf{z_o}_i^k\|^2_{\Lambda_{i,t}}, \end{aligned} \tag{3}$$

In a scenario involving N robots and M map features evolving over a temporal span of $T+1$ timesteps, the anticipated relative poses $\bar{\mathbf{z}}^{i,t}$ are contingent upon two consecutive poses of robot i, determined from the robot's odometry measurements.$\mathbf{z_o}_i^k$ denotes the k_{th} obstacle detected by the i_{th} agent in any instant. Simultaneously, the expected relative pose $\tilde{\mathbf{z}}_i^k$ is influenced by the pose of robot i and the position of map feature k. To consolidate the variables of interest, we introduce concatenation, defining $\mathbf{x}_i = [\mathbf{x}_{i,0}^\top, \mathbf{x}_{i,1}^\top, \ldots, \mathbf{x}_{i,T}^\top]^\top$, $\mathbf{x} = [\mathbf{x}_1^\top, \mathbf{x}_2^\top, \ldots, \mathbf{x}_N^\top]^\top$, and $\mathbf{m} = [\mathbf{m}_1^\top, \mathbf{m}_2^\top, \ldots, \mathbf{m}_M^\top]^\top$.

The components of the objective function incorporate error terms, each scaled by the corresponding information matrices Ω_i^t and Λ_i^t, which are associated with the measurements gathered by robot i. While the initial subset of terms within the optimization problem's objective function exhibits separability across individual robots, the subsequent subset of terms lacks this property. Consequently, a reformulation of the optimization problem becomes imperative to facilitate compatibility with distributed optimization algorithms. The non-separability of the objective function emanates from the inherent interdependence between the map features and the robot poses. To establish separability in the objective function, a strategic approach involves introducing localized replicas of variables corresponding to each feature. This introduces an associated set of consensus constraints, specifically in the form of equality constraints, aimed at preserving equivalence between the resulting problem and the original formulation. The resulting problem takes the form

$$\begin{aligned} \min_{x,\hat{m}_i} \quad & \sum_{i=1}^{N}\sum_{t=0}^{T-1}\frac{1}{2}\|\bar{\mathbf{z}}^{i,t}(\mathbf{x}_{i,t},\mathbf{x}_{i,t+1})-\hat{\mathbf{z}}_{i,t+1}\|^2_{\Omega_{i,t}} \\ & +\sum_{i=1}^{N}\sum_{k=1}^{M}\frac{1}{2}\|\tilde{\mathbf{z}}_i^k(\mathbf{x}_i,\mathbf{m}_k)-\mathbf{z_o}_i^k\|^2_{\Lambda_{i,t}}, \end{aligned} \tag{4}$$

where robot i maintains $\hat{\mathbf{m}}_i$, its local copy of the map $\mathbf{m}$. The problem is separable among the robots; in other words, its objective function can be expressed in the form

$$f(\mathbf{x},\hat{\mathbf{m}}_1,\hat{\mathbf{m}}_2,\ldots,\hat{\mathbf{m}}_N)=\sum_{i=1}^{N}f_i(\mathbf{x}_i,\hat{\mathbf{m}}_i),$$

where

$$f_i(\mathbf{x}_i,\hat{\mathbf{m}}_i)=\sum_{t=0}^{T}\frac{1}{2}\|\bar{\mathbf{z}}^{i,t}(\mathbf{x}_{i,t},\mathbf{x}_{i,t+1})-\hat{\mathbf{z}}_{i,t+1}\|^2_{\Omega_i^t}+\sum_{k=1}^{M}\frac{1}{2}\|\tilde{\mathbf{z}}_i^k(\mathbf{x}_i,\hat{\mathbf{m}}_{i,k})-\mathbf{z_o}_i^k\|^2_{\Lambda_i^t}.$$

Analogously, the bundle adjustment problem lends itself to interpretation, wherein the map features encapsulate the scene geometry, while the robot poses encompass the optical attributes of their respective cameras. However, when applied in unstructured environments, a noteworthy challenge arises in ensuring consensus among multiple robots regarding the labeling of map landmarks. As an alternative strategy, pose graph optimization emerges, circumventing the explicit estimation of the map and instead relying on relative pose measurements derived from shared observations of environmental features. From this vantage point, the framework of multi-robot SLAM encompasses a direct phase, wherein robots process raw sensor measurements to generate relative pose measurements, and a subsequent "back-end" phase, wherein optimal robot poses are determined based on the aforementioned relative pose measurements. The SLAM back-end is conventionally formulated as a pose graph optimization problem, wherein edges connecting nodes signify noisy relative pose estimates, originating from raw sensor measurements and obtained during the front-end processing. This pose graph optimization problem inherently possesses a separable structure, rendering it amenable to distributed optimization techniques. The formulation is expressed as follows:

$$\min_{\{(R_i,\tau_i)\}_{i=1}^n} \quad \sum_{(i,j)\in\varepsilon}\omega_{ij}^2\|\mathbf{R}_j-\mathbf{R}_i\tilde{\mathbf{R}}_{ij}\|_F^2+\frac{\omega_{ij}}{2}\|\tau_j-\tau_i-\mathbf{R}_i\tilde{\tau}_{ij}\|_2^2, \tag{5}$$

In the realm of robotic pose estimation, the pose graph optimization problem unfolds its intricacies, with $\mathbf{R}_i$ and τ_i embodying the rotation matrix and translation vector characterizing the pose of robot i. Concurrently, $\tilde{\mathbf{R}}_{ij}$ and $\tilde{\tau}_{ij}$ stand as estimations of the aforementioned rotation matrix and translation vector.

The crux of the pose graph optimization problem lies in the minimization of errors between anticipated relative poses, extrapolated from the estimated poses, and the actual measured relative poses. This minimization endeavor spans all edges encapsulated within the graph, a pursuit that encapsulates the essence of the optimization process.

In tandem, the front-end component of Simultaneous Localization and Mapping (SLAM) introduces an additional layer of complexity, with its compatibility with distributed optimization techniques representing an open avenue of scholarly exploration. Consequently, the application of distributed optimization algorithms to the graph-based SLAM problem, as articulated in Equation (7), unveils a promising prospect for advancing the frontiers of research in robotic perception and localization.

3.2 Multi-Robot Task Assignment

Within the multi-robot task assignment problem, the overarching objective is to discern an optimal assignment of N robots to a set of M tasks, with the primary aim of minimizing the cumulative cost incurred in completing the designated tasks. The ascendancy of optimization-based methodologies in this domain is underscored by their adaptability in accommodating diverse constraints and encapsulating preferences, rendering them a versatile and general-purpose approach.

The task assignment conundrum is conventionally abstracted as a weighted bipartite graph, wherein each edge symbolizes the cost associated with assigning a robot to a specific task. The resultant optimization problem is typically cast in the framework of integer optimization. However, prevailing methodologies often navigate a relaxation of this problem. The relaxed optimization problem assumes the following form:

$$\begin{aligned} &\min_{x} \sum_{i=1}^{N} c_i^T x_i \\ \text{s.t} \quad &\sum_{i=1}^{N} (0 \leq x_i \leq 1) = 1 \\ &\sum_{i=1}^{M} (0 \leq x_i \leq 1) = 1 \end{aligned} \tag{6}$$

Here, x_i is the optimization variable representing the task assignment for robot i, and c_i is the cost vector associated with robot i.

To make the problem amenable to distributed optimization, we can assign a local copy of x to each robot:

$$\begin{aligned} &\min_{\hat{x}} \sum_{i=1}^{N} c_i^T \hat{x}_{i,i} \\ \text{s.t} \quad &\sum_{i=1}^{N} \hat{x}_{i,i} = 1_M \\ &1_M^T \hat{x}_{i,i} = 1 \\ &0 \leq \hat{x}_i \leq 1 \\ \hat{x}_i = \hat{x}_j \quad &\forall (i,j) \in \varepsilon \end{aligned} \tag{7}$$

Here, $\hat{x}_{i,i}$ is the local copy of x for robot i and $\hat{x} = [\hat{x}_0,\ \hat{x}_1,\ \hat{x}_2,\ \dots\ \hat{x}_n]$

3.3 Multi-Robot Target Tracking

In the domain of multi-robot target tracking, a consortium of robots engages in the acquisition of measurements pertaining to a designated agent of interest, commonly referred to as a target. The collective objective is to collaboratively infer and estimate the trajectory of this target. Instances of multi-robot target tracking manifest across a spectrum of robotics applications, spanning environmental monitoring and surveillance to pivotal domains such as autonomous driving. In the latter context, the ascertained trajectory of the target assumes significance for scene prediction, thereby contributing to the facilitation of secure and autonomous operational paradigms.

Illustrated in Figure 2 is a representation of the multi-robot target tracking problem, wherein a quartet of quadrotors conducts noisy observations of a ground vehicle flagged as the target. Each distinctive cone in the figure delineates the region within which an individual quadrotor can observe the target, accounting for the finite measurement range of the sensors equipped on the quadrotor platforms.

The formalization of multi-robot target tracking problems often adopts the framework of Maximum A Posteriori (MAP) optimization. In this context, the robots endeavor to compute an estimate that maximizes the posterior distribution characterizing the trajectory of the target, given the comprehensive set of observations made by all participating robots. This optimization problem assumes a structured form when a model capturing the dynamics of the target, denoted by $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$, is at the disposal of the robots. The ensuing optimization problem is articulated as follows:

$$\min_{x} \quad \sum_{t=0}^{T} \frac{1}{2} \| x_{t+1} - g(x_t) \|_{\Omega_t}^2 + \sum_{i=1}^{N} \sum_{t=0}^{T} \frac{1}{2} \| y_{i,t} - h_i(x_t) \|_{\Lambda_i,t}^2 \tag{8}$$

where $x_t \in \mathbb{R}^n$ denotes the pose of the target at time t and $y_{i,t} \in \mathbb{R}^m$ denotes robot i's observation of the target at time t, over a duration of $T+1$ timesteps. We represent the trajectory of the target with $x = [x_0^\top, x_1^\top, \ldots, x_T^\top]^\top$. While the first term in the objective function corresponds to the error between the estimated state of the target at a subsequent timestep and its expected state based on a model of its dynamics, the second term corresponds to the error between the observations collected by each robot and the expected measurement computed from the estimated state of the target, where the function $h_i : \mathbb{R}^n \rightarrow \mathbb{R}^m$ denotes the measurement model of robot i. Further, the information matrices $\Omega_t \in \mathbb{R}^{n \times n}$ and $\Lambda_{i,t} \in \mathbb{R}^{m \times m}$ for the dynamics and measurement models, respectively, weight the contribution of each term in the objective function appropriately, reflecting prior confidence in the dynamics and measurement models.

The MAP optimization problem in (10) is not separable; hence, not amenable to distributed optimization in its current form, due to coupling in the objective function arising from x. Nonetheless, we can arrive at a separable optimization problem through a fairly straightforward reformulation. We can assign a local copy of x to each robot, with $\hat{x}_i$ denoting robot i's local copy of x. The reformulated problem becomes

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{N} \sum_{t=0}^{T} \frac{1}{2N} \| \hat{x}_{i,t+1} - g(\hat{x}_{i,t}) \|_{\Omega_t}^2 + \sum_{i=1}^{N} \sum_{t=0}^{T} \frac{1}{2} \| y_{i,t} - h_i(\hat{x}_{i,t}) \|_{\Lambda_i,t}^2 \\ \text{s.t} \quad & \hat{x}_i = \hat{x}_j \quad \forall (i,j) \in \varepsilon, \end{aligned} \tag{9}$$

where $\hat{x} = [\hat{x}_1^\top, \hat{x}_2^\top, \ldots, \hat{x}_N^\top]^\top$. Subsequent to this reformulation, the application of distributed optimization algorithms becomes viable for computing an estimate of the target's trajectory, as delineated in Equation (11).

The concurrent multi-robot control problem entails the computation of a sequence of control inputs, guiding a group of robots to either track a predefined reference trajectory or fulfill specific tasks, such as collaborative object manipulation. Illustrated in Figure 4 is an instance of a collaborative manipulation scenario involving three quadrotors collectively moving an object. The dashed line signifies the reference trajectory for manipulating the load.

Extensive research has been dedicated to collaborative multi-robot planning, control, and manipulation problems, yielding a diverse array of methodologies tailored to address these challenges. Notably, receding horizon or model predictive control (MPC) approaches have garnered significant attention due to their adeptness in encoding intricate problem constraints and objectives. Within the framework of MPC approaches, these multi-robot problems are cast as optimization problems over a finite time duration at each timestep. The resultant optimization problem is systematically resolved to yield a sequence of control inputs spanning the specified time horizon. However, only the initial control input is implemented by each robot at the current timestep. Subsequently, at the next timestep, a fresh optimization problem is formulated, culminating in the computation of a new sequence of control inputs to derive a new control input for that timestep. This iterative process persists until the successful completion of the designated task. At time t, the associated MPC optimization problem has the form

$$\begin{aligned} \min_{x,u} \quad & \sum_{i=1}^{N} f_i(x,u) \\ \text{subject to} \quad & g(x,u) = 0 \\ & h(x,u) \leq 0 \\ & x_{i,0} = \bar{x}_i \quad \forall i \in V, \end{aligned} \tag{10}$$

In the formulation of the multi-robot control problem, the state trajectory of robot i is denoted by $x_i \in \mathbb{R}^{n_i}$, its control input trajectory is represented by $u_i \in \mathbb{R}^{m_i}$, and the concatenated states and control inputs are defined as $x = [x_1^\top, x_2^\top, \ldots, x_N^\top]^\top$ and $u = [u_1^\top, u_2^\top, \ldots, u_N^\top]^\top$, respectively. The objective function $f_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ for robot i assumes a quadratic form, expressed as

$$f_i(x,u) = (x_i - \tilde{x}_i)^\top Q_i (x_i - \tilde{x}_i) + (u_i - \tilde{u}_i)^\top R_i (u_i - \tilde{u}_i),$$

where $\tilde{x}_i$ and $\tilde{u}_i$ denote the reference state and control input trajectory, $Q_i \in \mathbb{R}^{n_i \times n_i}$ and $R_i \in \mathbb{R}^{m_i \times m_i}$ are associated weight matrices, $n = \sum_{i=1}^{N} n_i$, and $m = \sum_{i=1}^{N} m_i$. The dynamics function of the robots is encapsulated in $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, which may also encode other equality constraints. Inequality constraints, encompassing collision-avoidance and various state or control input feasibility constraints, are expressed through $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^l$. Additionally, the first state variable of each agent adheres to the constraint of being equal to its initial state, denoted by $\bar{x}_i$. In each iteration of the Model Predictive Control (MPC) optimization problem presented in Equation (14), the initial state $\bar{x}_i$ of robot i is set to its current state at that particular timestep.

It is noteworthy that the MPC optimization problem in Equation (14) is not inherently separable, contingent upon the nature of the equality and inequality constraints. However, a separable format can always be achieved through the introduction of localized copies of the optimization variables that exhibit coupling in the original formulation (14). The functions g and h also possess the capacity to encode complementarity constraints, particularly relevant in scenarios involving manipulation and locomotion challenges that necessitate the establishment and dissolution of rigid body contacts. In cases where the optimization variables are intricately intertwined in both the objective function and the equality and inequality constraints of (14), a judicious reformulation assumes the following structure:

$$\begin{aligned} \min_{\hat{x},\hat{u}} \quad & \sum_{i=1}^{N} f_i(\hat{x}_i, \hat{u}_i) \\ \text{subject to} \quad & g(\hat{x}_i, \hat{u}_i) = 0 \quad \forall i \in \mathcal{V} \\ & h(\hat{x}_i, \hat{u}_i) \leq 0 \quad \forall i \in \mathcal{V} \\ & \phi_i(\hat{x}_i) = \bar{x}_i \quad \forall i \in \mathcal{V} \\ & \hat{x}_i = \hat{x}_j \quad \forall (i,j) \in \varepsilon, \end{aligned} \tag{11}$$

where the function ϕ_i outputs the first state variable corresponding to robot i, given the input $\hat{x}_i$ which denotes robot i's local copy of x. Similarly, $\hat{u}_i$ denotes robot i's local copy of u, with $\hat{x} = [\hat{x}_1^\top, \hat{x}_2^\top, \ldots, \hat{x}_N^\top]^\top$ and $\hat{u} = [\hat{u}_1^\top, \hat{u}_2^\top, \ldots, \hat{u}_N^\top]^\top$.

4 Main Results/Algorithms

From the relevant papers, in this section, we categorize the distributed optimization algorithms into three main classes: Distributed First-Order Methods, Distributed Sequential Convex Programming, and the Alternating Direction Method of Multipliers. Each of them is briefly overviewed through a representative distributed algorithm. The sequel discussion focuses on the separable optimization problem, excluding local equality and inequality constraint functions, denoted as g_i and h_i. In the future, we hope to seek further in more advanced distributed optimization algorithms that include these constraint functions.

A. Distributed First-Order Algorithms

In this section, we explore Distributed First-Order Methods. Initially, this paper focuses on excluding local equality and inequality constraint functions (g_i and h_i). In a subsequent installment of this series, these constraints are incorporated, presenting a survey of more advanced distributed optimization algorithms.

Before diving into specific algorithms, we first explore the common framework shared by all these approaches. Each algorithm progresses through discrete iterations ($k = 0, 1, \ldots$) until convergence, involving both a communication and a computation step in each iteration. Apart from the assumption that each robot evaluates its local objective function f_i independently, we distinguish between "internal" variables $\mathcal{P}_i^{(k)}$ computed by the robot at each iteration k and "communicated" variables $\mathcal{Q}_i^{(k)}$ shared with its neighbors. This reduces individuality and helps us analyze the system in a distributed paradigm. The algorithms also incorporate parameters $\mathcal{R}_i^{(k)}$, typically requiring pre-assigned coordination among all robots.

In the realm of distributed optimization, the collective goal of all robots is to minimize the joint objective function of relevance, while attaining consensus on a common set of optimizing variables. Two perspectives on achieving consensus are distinguished in this paper: distributed first-order methods and distributed sequential convex programming methods enforce consensus constraints implicitly, whereas the alternating direction method of multipliers enforces these constraints explicitly. A foundational assumption within this paper pertains to the perfect sharing and updating of local positional coordinates among agents. Existing literature has demonstrated that this assumption may be challenged in scenarios involving an intruder. Specifically, an intruder has the potential to influence a subset of agents, denoted as ϵ, compelling them to disseminate malicious information to their neighbors. In instances where this malicious information exhibits structural characteristics, distinctive properties about the steady-state behavior of agents can be discerned with uniqueness.

Keeping the attacked case aside, let us first decipher the scenario where all the updates and information exchanges are accurate up to an extent. It is pretty well known that gradient descent methods, widely applied in solving various optimization problems, only require computing the gradient, i.e., the first derivative of the objective and constraint functions. The updates to the optimization variable follow the form:

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f\left(x^{(k)}\right)$$

where $\alpha^{(k)}$ denotes a diminishing step-size, and $\nabla f\left(x^{(k)}\right)$ is the gradient of the objective function given by:

$$\nabla f(x) = \sum_{i \in \mathcal{V}} \nabla f_i(x)$$

The computation of $\nabla f(x)$ entails acquiring knowledge of the objective function from all robots, thereby mandating the aggregation of this information at a central node. Given the focused exploration of distributed optimization algorithms in this context, the conventional method for computing $\nabla f(x)$ loses its validity. Consequently, it becomes imperative to devise an alternative procedure. Distributed First-Order (DFO) algorithms emerge as a valuable resource, introducing alternative update schemes that surmount this challenge by empowering each robot to exclusively leverage its local gradients. Concurrently, communication with neighbors facilitates the attainment of consensus on a shared solution. DFO methods manifest themselves as two overarching subclasses: Adapt-Then-Combine (ATC) methods and Combine-Then-Adapt (CTA) methods, distinguished by the relative sequencing of communication and computation procedures. In the first method, each robot updates its local optimization variable using its gradient before combining its local variable with that of its neighbors, eliminating the need to locally aggregate the individual gradients to a central server to compute the system gradient :

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \left(x_j^{(k)} - \alpha^{(k)} y_j^{(k)} \right)$$

In the context presented, $x_j^{(k)} \in \mathbb{R}^n$ signifies the local variable of the neighboring robot j, and $y_j^{(k)}$ represents the local gradient $y_j^{(k)} = \nabla f_j \left(x_j^{(k)} \right)$. The parameter w_{ij} pertains to the weights associated with the communication network in consideration. As a relatively 'non-related' part of our primary investigation, our proposed simulation undertook an exhaustive exploration across all classes of underlying network topology, to discern inherent mathematical connections between said topology and the convergence time.

Back to the discussion, CTA methods, in contrast, involve combining each robot's local variable with that of its neighbors before incorporating its local gradient:

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} - \alpha^{(k)} y_i^{(k)}$$

The canonical distributed subgradient method is a specific case where $y_i^{(k)} = \partial f_i \left(x_i^{(k)} \right)$, necessitating the use of a diminishing step-size.Algorithm 1 encapsulates the procedural intricacies of the distributed gradient tracking method in the algorithm `DIGing`. Given that the algorithm updates itself directly based on shared information, any perturbation or adversarial incursion capable of distorting this information inevitably impacts the algorithm's output. Consequently, it becomes imperative to devise robust iterations of these algorithms before their application in real-world scenarios.

Sophisticated gradient tracking methods, such as `DIGing`, employ an estimate of the average gradient computed through dynamic average consensus:

$$y_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} y_j^{(k)} + \left[\nabla f_i \left(x_i^{(k+1)} \right) - \nabla f_i \left(x_i^{(k)} \right) \right]$$

sizes method does not require a diminishing step-size, and at initialization, all robots select a common step-size. Furthermore, robot i initializes its local variables with $x_i^{(0)} \in \mathbb{R}^n$ and $y_i^{(0)} = \partial f_i \left(x_i^{(0)} \right)$. Notably, many DFO methods impose additional restrictions on the weighting matrix;

for instance, DIGing requires a doubly-stochastic weighting matrix for undirected communication networks.

Algorithm 1 `DIGing`

```
Initialization: $k \leftarrow 0, x_i^{(0)} \in \mathbb{R}^n, y_i^{(0)} = \nabla f_i\left(x_i^{(0)}\right)$
Internal variables: $\mathcal{P}_i^{(k)} = \emptyset$
Communicated variables: $\mathcal{Q}_i^{(k)} = \left(x_i^{(k)}, y_i^k\right)$
Parameters: $\mathcal{R}_i^{(k)} = (\alpha, w_i)$
for $i \in \mathcal{V}$ do in parallel
    Communicate $\mathcal{Q}_i^{(k)}$ to all $j \in \mathcal{N}_i$
    Receive $\mathcal{Q}_j^{(k)}$ from all $j \in \mathcal{N}_i$
$(x_i^{(k+1)}, y_i^{(k+1)}) = \left(\sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} x_j^{(k)} - \alpha y_i^{(k)}, \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} y_j^{(k)} + \nabla f_i\left(x_i^{(k+1)}\right) - \nabla f_i\left(x_i^{(k)}\right)\right)$
$k \leftarrow k + 1$
end for
```

Further, robot i initializes its local variables with $x_i^{(0)} \in \mathbb{R}^n$ and $y_i^{(0)} = \partial f_i\left(x_i^{(0)}\right)$. Algorithm 1 summarizes the update procedures in the distributed gradient tracking method `DIGing`. Many DFO methods impose additional restrictions on the weighting matrix. For example, DIGing requires a doubly stochastic weighting matrix for undirected communication networks.

B. Distributed Sequential Convex Programming

Sequential Convex Programming constitutes a methodology for solving optimization problems through the iterative computation of a sequence of iterates, each representing an approximation to the original problem. Newton's method serves as a quintessential illustration of a Sequential Convex Programming approach. In this context, and more broadly in quasi-Newton methods, a quadratic approximation of the objective function is undertaken at an operational point $x^{(k)}$, yielding:

$$\begin{aligned}\tilde{f}(x) =& f\left(x^{(k)}\right) + \nabla f\left(x^{(k)}\right)^\top \left(x - x^{(k)}\right) \\ &+ \frac{1}{2}\left(x - x^{(k)}\right)^\top H\left(x^{(k)}\right)\left(x - x^{(k)}\right),\end{aligned}$$

where $H(\cdot)$ represents the Hessian of the objective function, denoted as $\nabla^2 f$, or an approximation thereof. Subsequently, a solution to the quadratic program is computed:

$$x^{(k+1)} = x^{(k)} - H\left(x^{(k)}\right)^{-1} \nabla f(\tilde{x}),$$

which necessitates centralized evaluation of the gradient and Hessian of the objective function.

Distributed Sequential Programming, on the other hand, empowers individual robots to compute local estimates of the gradient and Hessian of the objective function, enabling local execution of the update procedures. Illustrating this category of distributed optimization algorithms is the NEXT algorithm. Within NEXT, each robot employs a quadratic approximation of the optimization problem as its convex surrogate model $U(\cdot)$. Specifically, each robot maintains an estimate of the average gradient of the objective function, along with an estimate of the gradient of the objective function excluding its local component. At the current iterate $x_i^{(k)}$, robot i constructs a quadratic approximation of the optimization problem. This approximation involves a minimization problem, where $\tilde{\pi}_i^{(k)}$ represents robot i's estimate of the gradient of $\sum_{j \neq i} f_j\left(x_i\right)$ at $x_i^{(k)}$, solvable locally.

Each robot then computes a weighted combination of its current iterate and the solution of this quadratic program, resulting in the procedure:

$$z_i^{(k)} = x_i^{(k)} + \alpha^{(k)}\left(\tilde{x}_i^{(k)} - x_i^{(k)}\right),$$

where $\alpha^{(k)} \in (0,1)$ denotes a diminishing step-size. Subsequently, robot i computes its next iterate by taking a weighted combination of its local estimate $z_i^{(k)}$ with that of its neighbors. This consensus-seeking procedure, given by:

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} z_j^{(k)},$$

facilitates consensus on a common solution to the original optimization problem. The weight $w_{i,j}$ must align with the underlying communication network.Additionally, robot i updates its estimates of the average gradient of the objective function, y_i, leveraging dynamic average consensus:

$$y_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} y_j^{(k)} + \left[\nabla f_i\left(x_i^{(k+1)}\right) - \nabla f_i\left(x_i^{(k)}\right)\right],$$

and updates $\tilde{\pi}_i^{(k)}$ using the procedure:

$$\tilde{\pi}_i^{(k+1)} = N \cdot y_i^{(k+1)} - \nabla f_i\left(x_i^{(k+1)}\right).$$

This decentralized approach to optimization showcases the potential for robots to collaboratively seek solutions to complex problems while maintaining localized computations.

Algorithm 2 NEXT Algorithm

```
Initialization: $k \leftarrow 0, x_i^{(0)} \in \mathbb{R}^n, y_i^{(0)} = \nabla f_i\left(x_i^{(0)}\right)$,
$\tilde{\pi}_i^{(k+1)} = N y_i^{(0)} - \nabla f_i\left(x_i^{(0)}\right)$
Internal variables: $\mathcal{P}_i = \left(x_i^{(k)}, \tilde{x}_i^{(k)}, \tilde{\pi}_i^{(k)}\right)$
Communicated variables: $\mathcal{Q}_i^{(k)} = \left(z_i^{(k)}, y_i^{(k)}\right)$
Parameters: $\mathcal{R}_i^{(k)} = \left(\alpha^{(k)}, w_i, U(\cdot), \mathcal{K}\right)$
while stopping criterion is not satisfied do
    for $i \in \mathcal{V}$ in parallel do
        $\tilde{x}_i^{(k)} = \underset{x \in \mathcal{K}}{\operatorname{argmin}} U\left(x; x_i^{(k)}, \tilde{\pi}_i^{(k)}\right)$
        $z_i^{(k)} = x_i^{(k)} + \alpha^{(k)}\left(\tilde{x}_i^{(k)} - x_i^{(k)}\right)$
    end for
    Communicate $\mathcal{Q}_i^{(k)}$ to all $j \in \mathcal{N}_i$
    Receive $\mathcal{Q}_j^{(k)}$ from all $j \in \mathcal{N}_i$
    $x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} z_j^{(k)}$
    $y_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} y_j^{(k)} + \left[\nabla f_i\left(x_i^{(k+1)}\right) - \nabla f_i\left(x_i^{(k)}\right)\right]$
    $\tilde{\pi}_i^{(k+1)} = N \cdot y_i^{(k+1)} - \nabla f_i\left(x_i^{(k+1)}\right)$
    $k \leftarrow k + 1$
end while
```

Each agent initializes its local variables with $x_i^{(0)} \in \mathbb{R}^n$, $y_i^{(0)} = \nabla f_i\left(x_i^{(0)}\right)$, and $\tilde{\pi}_i^{(k+1)} = N y_i^{(0)} - \nabla f_i\left(x_i^{(0)}\right)$, prior to executing the above update procedures. Algorithm 2 summarizes the update procedures in NEXT [31].

C. Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) falls under the category of optimization algorithms known as the method of multipliers or augmented Lagrangian methods. These methods aim to compute a primal-dual solution pair for a given optimization problem. The iterative process of the method of multipliers involves updating primal iterates as minimizers of the augmented Lagrangian, followed by dual iterates updated through dual ascent on the augmented Lagrangian. This iterative procedure continues until convergence or termination.

The augmented Lagrangian for the optimization problem in Equation (5) (considering only consensus constraints) is expressed as:

$$\begin{aligned}\mathcal{L}_a(\mathbf{x}, q) = &\sum_{i=1}^{N} f_i\left(x_i\right) \\ &+ \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} \left(q_{i,j}^\top \left(x_i - x_j\right) + \frac{\rho}{2}\left\|x_i - x_j\right\|_2^2\right),\end{aligned}$$

where $q_{i,j}$ represents a dual variable for the consensus constraints between robots i and j,

$q = \left[q_{i,j}^\top, \forall (i,j) \in \mathcal{E}\right]^\top$, and $\mathbf{x} = \left[x_1^\top, x_2^\top, \cdots, x_N^\top\right]^\top$. The parameter $\rho > 0$ is a penalty term on the violations of the consensus constraints.

While the method of multipliers computes the minimizer of the augmented Lagrangian with respect to the joint set of optimization variables, hindering distributed computation, the Alternating Direction Method of Multipliers (ADMM) performs the minimization procedure block-component-wise. This enables parallel, distributed computation of the minimization subproblem in the consensus problem. However, several ADMM algorithms may still involve some centralized computation, making them not fully-distributed in the multi-robot mesh network sense considered in this paper.

Focusing on ADMM algorithms distributed over robots in a mesh network, where each robot executes the same set of distributed steps, we examine the Consensus Alternating Direction Method of Multipliers (C-ADMM) [32] as a representative algorithm in this category. C-ADMM introduces auxiliary optimization variables into the consensus constraints in Equation (5) to facilitate fully-distributed update procedures. The primal update procedure of robot i takes the form:

$$\begin{aligned} x_i^{(k+1)} = \underset{x_i}{\operatorname{argmin}} \{ & f_i\left(x_i\right) + x_i^\top y_i^{(k)} \\ & + \rho \sum_{j \in \mathcal{N}_i} \left\| x_i - \frac{1}{2}\left(x_i^{(k)} + x_j^{(k)}\right)\right\|_2^2 \right\}, \end{aligned}$$

which only requires information locally available to robot i, including information received from its neighbors ($x_j^k, \forall j \in \mathcal{N}_i$). Consequently, this procedure can be executed locally by each agent in parallel. After communicating with its neighbors, each robot updates its local dual variable using the procedure:

$$y_i^{(k+1)} = y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \left(x_i^{(k+1)} - x_j^{(k+1)}\right),$$

where y_i denotes the composite dual variable of robot i, corresponding to the consensus constraints between robot i and its neighbors, initialized to zero. Algorithm 3 provides a summary of the update procedures in C-ADMM.

Algorithm 3 `C-ADMM Algorithm`

```
Initialization: $k \leftarrow 0, x_i^{(0)} \in \mathbb{R}^n, y_i^{(0)} = 0$
Internal variables: $\mathcal{P}_i^{(k)} = y_i^{(k)}$
Communicated variables: $\mathcal{Q}_i^{(k)} = x_i^{(k)}$
Parameters: $\mathcal{R}_i^{(k)} = \rho$
while stopping criterion is not satisfied do
    for $i \in \mathcal{V}$ in parallel do
        $x_i^{(k+1)} = \operatorname{argmin}\ x_i f_i(x_i) + x_i y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \frac{1}{2}\left(x_i - x_j\right)^2$
    end for
    Communicate $\mathcal{Q}_i^{(k)}$ to all $j \in \mathcal{N}_i$
    Receive $\mathcal{Q}_j^{(k)}$ from all $j \in \mathcal{N}_i$
    $y_i^{(k+1)} = y_i^{(k)} + \rho \sum_{j \in \mathcal{N}_i} \left(x_i^{(k+1)} - x_j^{(k+1)}\right)$
    $k \leftarrow k + 1$
end while
```

5 Simulation

In this simulation scenario, we delve into a distributed multi-drone vehicle target tracking problem, wherein interconnected robots form a communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Their collective objective is to infer the complete trajectory of a moving target. Each drone records range-limited linear measurements of the target and communicate locally with nearby drones over the undirected communication graph $\mathcal{G}$. The target's dynamics are linearly modeled as $x_{t+1} = A_t x_t + w_t$, where $x_t \in \mathbb{R}^4$ denotes the target's position and velocity at time t, A_t is the dynamics matrix, and $w_t \sim \mathcal{N}(0, Q_t)$ represents process noise.

Linear measurements are collected when the target is in close proximity, following the linear measurement model $y_{i,t} = C_{i,t} x_t + v_{i,t}$, where $y_{i,t} \in \mathbb{R}^2$ is a positional measurement, $C_{i,t}$ is the measurement matrix for drone i, and $v_{i,t} \sim \mathcal{N}(0, R_{i,t})$ is measurement noise. The initial state of the target assumes a common prior distribution $\mathcal{N}\left(\bar{x}_0, \bar{P}_0\right)$, with $\bar{x}_0 \in \mathbb{R}^4$ representing the mean and $\bar{P}_0 \in \mathbb{R}^{4\times 4}$ denoting the covariance.

The global cost function, denoted as $f(x)$, incorporates terms capturing deviations from the initial state, dynamics consistency, and measurement accuracy. The local cost function for drone i, $f_i(x)$, represents a normalized version of the global cost function.

In assessing the problem, we consider a batch solution aiming to deduce the entire trajectory of the target based on each drone's complete set of measurements. The simulation involves the comparison of distributed optimization algorithms, including C-ADMM, DIGing, and NEXT, applied to a scenario with 4 drones seeking to estimate the target's trajectory over 20 seconds or 1000 iterations. Notable differences emerge even at 500 iterations.

The comparative analysis, illustrated in Figures 2 and 3, unveils the performance characteristics of the algorithms. C-ADMM exhibits rapid convergence rates, while DIGing and NEXT converge at rates approximately five times slower, respectively, to achieve a mean squared error (MSE) below 10^{-5} in our simulation. The results emphasize the criticality of tuning for robust and efficient convergence, highlighting that algorithmic effectiveness is contingent on the specific problem instance. Despite C-ADMM's efficacy in this context, alternative algorithms may offer advantages in different scenarios. For example, C-ADMM's sensitivity to synchronization underscores the need for further research to enhance robustness in the face of real-world challenges such as synchronicity.

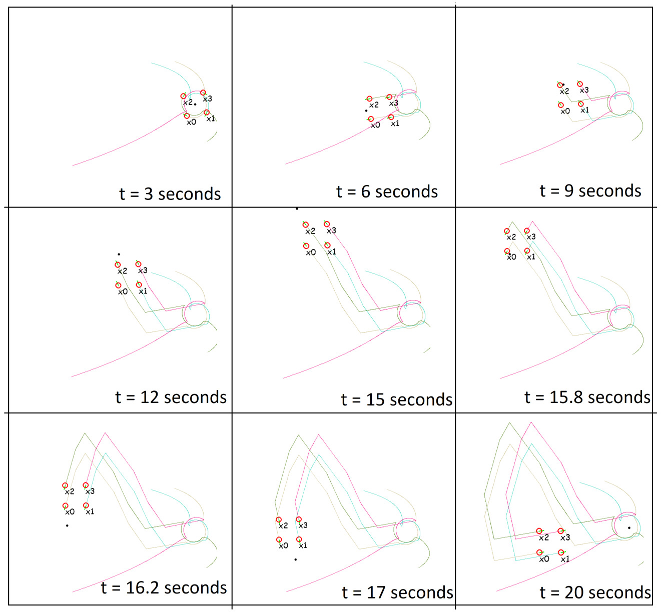

Figure 2: Four agents, represented by the color RED, operating within a fully connected communication framework, collaboratively track a target colored in BLACK efficiently within a 20-second timeframe. The three algorithms under consideration, namely DIGing, NEXT, and C-ADMM, exhibit comparable performance.

Assuming x^* is the average positional coordinate of n-agents (n=4 in our case), τ is the positional coordinate of the moving target, and δ^* is supposed to be the desired distance we want the agents to maintain while tracking a target.

$$\texttt{M.S.E} = \|x^* - \tau - \delta^*\|$$

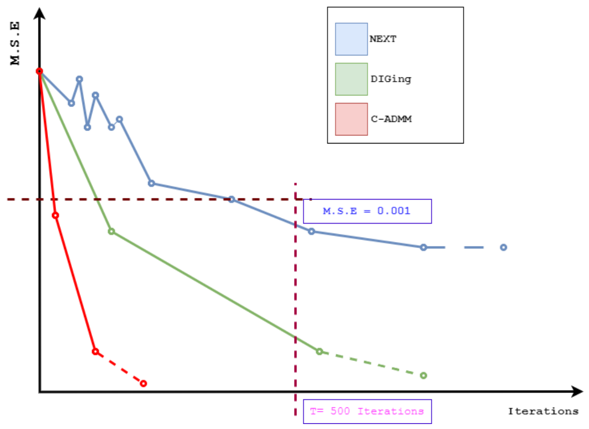

Figure 3: DIGing, NEXT, and C-ADMM are parallel executed for a more comprehensive analysis. The defined M.S.E between centroid of the four agents in RED and the tracking average is plotted concerning the number of iterations. In this specific problem of interest, we see C-ADMM outperforms the other two because of their semi-distributed nature.

6 Conclusion

In this paper, we systematically illuminate the versatile application of distributed optimization frameworks in addressing canonical challenges within multi-robot systems. Within this research domain, we discern three overarching classifications of distributed optimization algorithms: distributed first-order methods, distributed sequential convex programming methods, and the highly efficacious alternating direction method of multipliers (ADMM) (Note: This strongly depends on the type of problem we are involved with). Delving into the intricacies of each category, we illustrate the optimization strategies embedded in the algorithms, furnishing a paradigmatic algorithm for elucidation in each class. Furthermore, our exposition extends to the practical manifestation of these methodologies, featuring simulations, real-world distributed multi-drone vehicle tracking scenarios, and tangible hardware implementations. These endeavors collectively underscore the prowess and efficacy of distributed optimization algorithms in navigating the complexities inherent in multi-robot systems.

MRSs relying on wireless communication are strongly susceptible to cyber attacks. Within the domain of distributed optimization, a notable paradigm emerges where the absence of a central server necessitates local updates, restricting communication to predefined networks among agents. In this context, we delve into a scenario encompassing a network of n agents, each capable of communicating solely with its designated neighbors. Within this network, a fixed but undisclosed fraction, denoted by some fraction ϵ, pertains to agents undergoing arbitrary gradient attacks. Specifically, these agents, subject to such attacks, exhibit stochastic gradient oracles that furnish erroneous information, thereby disrupting the optimization process. The overarching objective in this scenario is to minimize the summation of local objective functions exclusively for the agents unaffected by these gradient attacks. In essence, the optimization goal is to mitigate the impact of adversarial interventions, thereby enabling the unattacked agents to collectively optimize their local objectives. This distinctive setting underscores the challenges inherent in distributed optimization when faced with adversarial influences, emphasizing the need for robust algorithms capable of navigating the complexities introduced by arbitrary gradient attacks. The following example demonstrates how an intruder can distort the corresponding algorithms by simply making a small fraction (ϵ) of agents sharing scaled information with their neighbors.

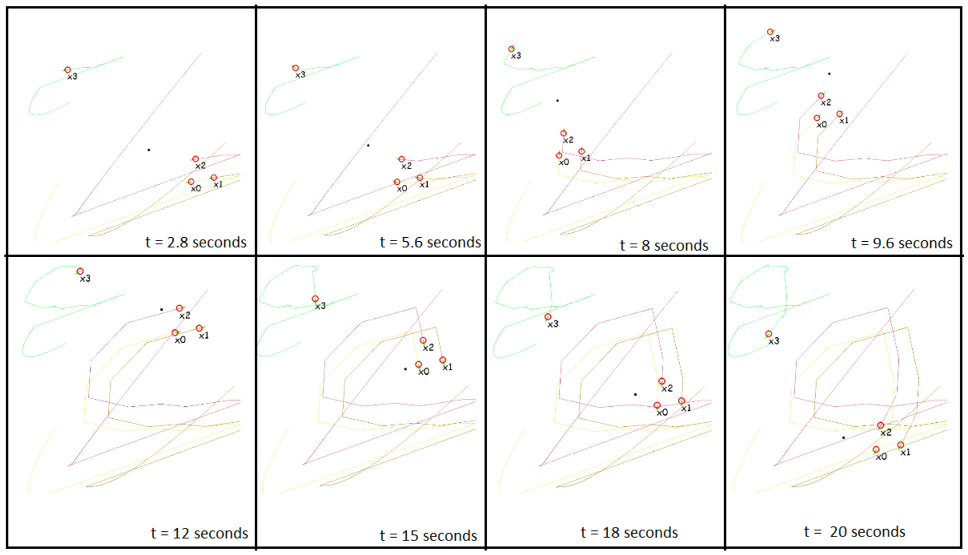

Figure 4: Four agents, represented by the color RED, operating within a simply connected communication framework, collaboratively track a target colored in BLACK efficiently within a 20-second timeframe. One of the agents is being influenced by an intruder to share scaled positional information with it's neighbor. The three algorithms under consideration, namely DIGing, NEXT, and C-ADMM, exhibit significantly depreciating performance, where the attacked agent gets separated from the non-attacked fellows, making the convergence criteria invalid.

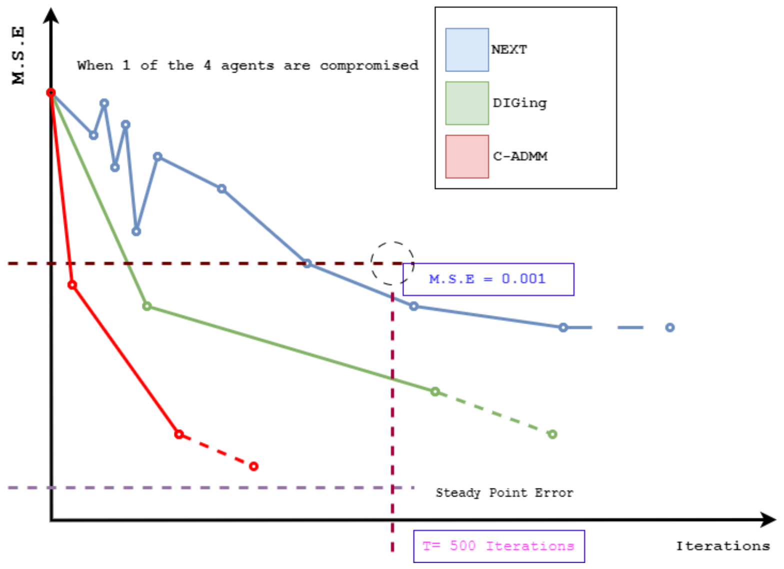

Figure 5: DIGing, NEXT, and C-ADMM are parallel executed in the attacked scenario (which causes distortion). The defined M.S.E between the distorted centroid of the four agents in RED and the tracking average is plotted concerning the number of iterations. As we can see the attack causes the M.S.E to settle down in a non-zero quantity, which warrants the development of robust versions of the proposed algorithms

7 References

[1] R. T. Rockafellar, ”Monotone operators and the proximal point algorithm,” SIAM journal on control and optimization, vol. 14, no. 5, pp. 877-898, 1976.

[2] J. N. Tsitsiklis, ”Problems in decentralized decision making and computation.” Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, Tech. Rep., 1984.

[3] O. Shorinwa, J. Yu, T. Halsted, A. Koufos, and M. Schwager, ”Distributed multi-target tracking for autonomous vehicle fleets,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 3495-3501.

[4] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, ”Multi-agent reinforcement learning via double averaging primal-dual optimization,” in Advances in Neural Information Processing Systems, 2018, pp. 9649-9660.

[5] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia, ”A messagepassing algorithm for multi-agent trajectory planning,” in Advances in neural information processing systems, 2013, pp. 521-529.

[6] L.-L. Ong, T. Bailey, H. Durrant-Whyte, and B. Upcroft, ”Decentralised particle filtering for multiple target tracking in wireless sensor networks,” in 2008 11th International Conference on Information Fusion. IEEE, 2008, pp. 1-8.

[7] T. Cieslewski and D. Scaramuzza, ”Efficient decentralized visual place recognition using a distributed inverted index,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 640-647, 2017.

[8] H. Durrant-Whyte and T. Bailey, ”Simultaneous localization and mapping: part i,” IEEE robotics & automation magazine, vol. 13, no. 2, pp. 99-110, 2006.

[9] T. Bailey and H. Durrant-Whyte, ”Simultaneous localization and mapping (slam): Part ii,” IEEE robotics & automation magazine, vol. 13, no. 3, pp. 108-117, 2006.

[10] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, ”A tutorial on graph-based slam,” IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31-43, 2010.

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

[12] V.-L. Dang, B.-S. Le, T.-T. Bui, H.-T. Huynh, and C.-K. Pham, ”A decentralized localization scheme for swarm robotics based on coordinate geometry and distributed gradient descent,” in MATEC Web of Conferences, vol. 54. EDP Sciences, 2016, p. 02002.

[13] N. A. Alwan and A. S. Mahmood, ”Distributed gradient descent localization in wireless sensor networks,” Arabian Journal for Science and Engineering, vol. 40, no. 3, pp. 893-899, 2015.

[14] M. Todescato, A. Carron, R. Carli, and L. Schenato, ”Distributed localization from relative noisy measurements: A robust gradient based approach,” in 2015 European Control Conference (ECC). IEEE, 2015, pp. 1914-1919.

[15] R. Tron and R. Vidal, ”Distributed 3-d localization of camera sensor networks from 2-d image measurements,” IEEE Transactions on Automatic Control, vol. 59, no. 12, pp. 3325-3340, 2014.

[16] A. Sarlette and R. Sepulchre, ”Consensus optimization on manifolds,” SIAM Journal on Control and Optimization, vol. 48, no. 1, pp. 56-76, 2009.

[17] K.-K. Oh and H.-S. Ahn, ”Distributed formation control based on orientation alignment and position estimation,” International Journal of Control, Automation and Systems, vol. 16, no. 3, pp. 1112-1119, 2018.

[18] H. W. Kuhn, "The hungarian method for the assignment problem," Naval research logistics quarterly, vol. 2, no. 1-2, pp. 83-97, 1955.

[19] R. N. Haksar, O. Shorinwa, P. Washington, and M. Schwager, "Consensusbased admm for task assignment in multi-robot teams," in The International Symposium of Robotics Research. Springer, 2019, pp. 35-51.

[20] L. Liu and D. A. Shell, "Optimal market-based multi-robot task allocation via strategic pricing." in Robotics: Science and Systems, vol. 9, no. 1, 2013, pp. 33-40.

[21] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in International conference on industrial, engineering and other applications of applied intelligent systems. Springer, 2010, pp. 721-730.

[22] O. Shorinwa and M. Schwager, "Distributed contact-implicit trajectory optimization for collaborative manipulation," in 2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS). IEEE, 2021, pp. 56-65.

[23] L. Ferranti, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Coordination of multiple vessels via distributed nonlinear model predictive control," in 2018 European Control Conference (ECC). IEEE, 2018, pp. 2523-2528.

[24] O. Shorinwa and M. Schwager, "Scalable collaborative manipulation with distributed trajectory planning," in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'20, vol. 1. IEEE, 2020, pp. 9108-9115.

[25] J. Yu, J. A. Vincent, and M. Schwager, "DiNNO: Distributed neural network optimization for multi-robot collaborative learning," IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 1896-1903, 2022.

[26] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in International Conference on Machine Learning. PMLR, 2018, pp. 5872-5881.

[27] Y. Zhang and M. M. Zavlanos, "Distributed off-policy actor-critic reinforcement learning with policy consensus," in 2019 IEEE 58th Conference on Decision and Control (CDC). IEEE, 2019, pp. 46744679.

[28] A. OroojlooyJadid and D. Hajinezhad, "A review of cooperative multiagent deep reinforcement learning," arXiv preprint arXiv:1908.03963, 2019.

[29] A. Nedic and A. Ozdaglar, "On the rate of convergence of distributed subgradient methods for multi-agent optimization," in IEEE Conference on Decision and Control. IEEE, 2007, pp. 4711-4716.

[30] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," SIAM Journal on Optimization, vol. 27, no. 4, pp. 2597-2633, 2017.

[31] P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," IEEE Transactions on Signal and Information Processing over Networks, vol. 2, no. 2, pp. 120-136, 2016.

[32] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," IEEE Transactions on Signal Processing, vol. 58, no. 10, pp. 5262-5276, 2010.

[33] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in 2007 46th IEEE Conference on Decision and Control. IEEE, 2007, pp 5492-5498.

[34] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," SIAM Journal on Optimization, vol. 25, no. 2, pp. 944-966, 2015.

[35] O. Shorinwa, T. Halsted, J. Yu, and M. Schwager, "Distributed Optimization Methods for Multi-Robot Systems: Part II - A Survey," 2023.