Distributed Multi-robot trajectory Optimization (DiMOpt)

João Salvado, Masoumeh Mansouri, Federico Pecora

***Abstract*— This paper focuses in computing trajectories for multiple robots navigating on a shared space, starting and finishing on predefined configuration states, while not colliding with each other and minimizing an optimization criterion. Multi-robot trajectory optimization approaches are able to model this problem while providing optimality guarantees. However, theses methods are hampered by complex robot's kinodynamics and coupling collision constraints. Therefore, we propose a distributed multi-robot optimization (DiMOpt) algorithm responsible for tackling the issue of coupling constraints, while exploiting a single-robot l_1-penalty sequential convex programming (SCP) method which can efficiently handle non-convexities introduced by robot's kinodynamics. We compare our DiMOpt approach with a standard l_1-penalty sequential convex programming algorithm tailored to multiple robots with non-convex dynamics (mSCP). We demonstrated empirically that DiMOpt scales well for large fleets of robots while finding solutions with lower cost than mSCP. Moreover, DiMOpt is an iterative algorithm where feasible trajectories can be found before converging to an optimal solution, and results suggest the cost of such fast initial solutions are comparable to a converged solution computed via mSCP method. We also investigate how path length and robot-scenario occupancy affects the performance of DiMOpt approach.**

I. Introduction

The problem of finding dynamically feasible trajectories for a multi-robot fleet navigating in a shared environment while avoiding collisions starting and finishing on a predefined configuration has been receiving increasing attention in recent years (see related work Section II). Some decompose this overall problem into known and well studied problems such as motion planning, coordination and control, which are solved in a *loosely-coupled* manner at the cost of optimality [1]. Others, exploit efficient graph based methods on a more *tightly coupled* way, although often overlooking dynamics or by requiring costly re-engineering of environment, processes and/or robots [2]. This is a consequence of latter methods focusing in computing paths on graphs instead of dynamically feasible trajectories.

This paper considers this problem in a *tightly coupled* manner, and specifically explores trajectory optimization methods. These methods offer a powerful tool for modeling the multi-robot trajectory optimization problem with optimality guarantees [3]. Nevertheless, they are hindered by non-convexities introduced by complex robot dynamics and a combinatorial explosion of collision constraints due to coupling between robots's decision variables.

João Salvado and Federico Pecora are with the AASS Research Centre, Örebro University, <name>.<surname>@oru.se

Masoumeh Manosuri is with the School of Computer Science at the University of Birmingham, m.mansouri@bham.ac.uk

Sequential convex programming (SCP) is a standard approach that deals with non-convex functions [4], and distributed optimization methods are known to be able to handle coupling constraints [5]. More specifically, previous works have deployed SCP methods for multiple robots with linear dynamics [6], or with non-convex dynamics with single robots [7]. On the other hand, distributed optimization methods have been deployed in other contexts, for example multiple robots deciding on observable quantities of the environment and target tracking [8]. Consequentially, we propose a distributed multi-robot optimization method (DiMOpt) integrated with a single robot sequential convex programming method.

DiMOpt method is evaluated in comparison to a standard sequential convex programming method that we tailor to multiple robots with non-convex dynamics (mSCP), as described in Section IV-A. Our results suggest the DiMOpt approach scales well with the amount of robots when comparing with the mSCP approach, while finding solutions with lower cost. Interestingly, results show the DiMOpt method has the capability of finding an initial feasible solution fast (DiMOpt 1) with solution quality comparable to mSCP method. We also investigate how path length and robot-scenario occupancy affect DiMOpt performance and provide an implementation of DiMOpt to be available in https://github.com/joaosalvado/DiMOpt.

II. Related Work

The problem of multiple robots navigating in a shared space while not colliding with each other, and starting and ending on a predefined configuration has been receiving increased attention in recent years [9]. Some existing methods for multi-robot trajectory optimization exploit efficient off-the-shelf motion/trajectory planning solvers to pre-compute individual robot trajectories while avoiding collisions via scheduling [10], [1], or priority planning [11], [12]; which fundamentally neglect interplay between robots leading to sub-optimal solutions. Others exploit efficient Integer Linear Programming (ILP) and Max Flow solvers [13] suitable to the MAPF (Multi Agent Path Finding) problem [14], which scales well with the amount of robots [2],[15], but often make unrealistic assumptions on the robot's geometries and dynamics, trading optimality and reality for fast solutions as a consequence of focusing in computing paths on graphs instead of dynamically feasible trajectories.

On the other hand, we have trajectory optimization methods, where the problem is modeled as finding a set of controls that minimize a cost functional while adhering to a set of constraints [16]. Such approach makes fewer assumptions

(e.g. realistic dynamic model) with optimality guarantees, although at the cost of a poorer problem scaling. This paper proposes a trajectory optimization method. We build upon existing efficient convex optimization solvers exploited by sequential convex programming methods (SCP) [17] that are utilized to handle non-convex parts of the problem. We handle the poor problem scaling with increasing number of robots via a distributed consensus optimization method [8].

Sequential convex programming methods (SCP) have been applied in the context of multiple robots via convexification of collision constraints with linear dynamic models [6], a l_1-penalty algorithm is given in [7] as it is utilized in the popular TrajOpt ROS library for single robots. A generalization of the SCP method for optimal control problems is provided in [18] where an additional step of projection to the feasible set is introduced. Moreover, according to [19] it is possible to decouple the problem via artificially considering configurations of other robots as obstacles in case a collision is detected, thus decomposing the problem and loosing optimality.

Distributed optimization traces back to primal and dual decomposition methods that rely on the existence of sub-gradients of non-convex functions of interest. Furthermore these sub-gradients are fast to compute, which makes distributed optimization method applicable to the multi-robot context [20] [21]. More recently, Alternating Direction Method of Multipliers (ADMM) brought additional robustness in comparison with previous decomposition methods that rely on a dual ascend step with a step size that is difficult to estimate (i.e. ADMM in augmented Lagrangian form [22] relies on a factor that is both represented in the cost function and the dual step). Moreover ADMM has superior convergence [5]. In the context of multi-robot trajectory optimization the only reported example is with two robots [23], hence an evaluation is missing. The ADMM algorithm was extended/applied in consensus optimization [24] problems where multiple agents converge (i.e. reach a consensus) on quantities while having local information and communicating with each other. However, this quantities are not related to robot trajectories.

There are other articles overlapping over several of the previously mentioned fields or introduce new concepts. One of such articles introduce Signal Temporal Logic (STL) in convex optimization, allowing a richer modeling of multi-robot planning problems (e.g. defining complex tasks) [25], although scaling poorly. The work of [26] exploit efficient MAPF solvers to generate multi-robot paths utilized in the computation of traversable safe regions per robot allowing multi-robot decomposition while loosing optimality. The authors of [27] apply distributed optimization techniques to compute multi-robot paths while overlooking the complexity of dynamics in the computation of trajectories.

III. Problem Statement

A. Nomenclature

$$\begin{aligned} \text{Robots: } & r \in \{1, \dots, R\}, \\ \text{Polygons: } & p \in \{1, \dots, P\}, \\ \text{Goals: } & g \in \{1, \dots, G\}, \\ \text{Discrete time: } & k \in \{1, \dots, N\}, \\ \text{Continuous time: } & t \in [0,\, 1]. \end{aligned}$$

Matrices are defined in uppercase bold letters (e.g., matrix $\boldsymbol{M}$), vectors in lowercase bold letters (e.g., vector $\boldsymbol{v}$) and constants in non-bold uppercase (e.g., constant C). Moreover, $\boldsymbol{v}[i]$ is the i-th element of vector $\boldsymbol{v}$. Note that continuous time t is normalized as $t = \frac{k}{N}$.

B. Preliminary Definitions

In the interest of defining the Multi-robot Trajectory Optimization problem, we formulate components of this problem individually. These are the objective function and constraints modeling robot's dynamics and collisions between robots and obstacles. For explanatory reasons, we define the specific constraints that were imposed in the experiments, however DiMOpt method solves the problem under alternative constraints, e.g. different dynamic model.

1) Dynamic Model: Each robot is considered to be circle-shaped for robot-to-robot collision modeling purposes and follows a differential drive model defined as,

$$\begin{aligned} \dot{x}_r &= v_r \cos\theta_r \\ \dot{y}_r &= v_r \sin\theta_r \\ \dot{\theta}_r &= \frac{1}{2\,L_r}\omega_r, \end{aligned} \tag{1}$$

where $2\,L_r$ is the distance between right and left wheel (so L_r is the robot's circle radius). The state space pose is $\boldsymbol{x}_r = [x_r\, y_r\, \theta_r] \in SE(2)$ and the control space is $\boldsymbol{u}_r = [v_r\, \omega_r] \in \mathbb{R}^2$, where v_r stands for speed and w_r for angular velocity. Car-like models, such as this one, impose zero side slip via non-holonomic Pfaffian constraints [28] which lead to non-convex equality constraints of the general form $\dot{\boldsymbol{x}}_r = f_r(\boldsymbol{x}_r, \boldsymbol{u}_r)$. Note that, we will refer to xy-Cartesian position as $\boldsymbol{q}_r = [x_r\, y_r]$.

2) Objective Function: We optimize for the control energy of the multi-robot fleet using the following quadratic function,

$$f_0(\boldsymbol{u}) = \sum_r \boldsymbol{u}_r \boldsymbol{W} \boldsymbol{u}_r^T \;:\; \boldsymbol{W} = \begin{bmatrix} w_v & 0 \\ 0 & w_\omega \end{bmatrix} \tag{2}$$

where $W \succ 0$ is a diagonal positive-definite matrix with associated penalty weights for speed w_v and angular velocity w_ω.

3) Transcription Method: Our solution transforms the continuous time dynamics into their discrete time equivalent. To this end, we deploy a direct multiple shooting transcription method [29], [30], i.e. we go from $\boldsymbol{x}(t)$ to $\boldsymbol{x}[k]$ for $k = \{1, \dots, N\}$. More specifically, we start by approximating the dynamic model equations for each k-knot via the first-order Taylor's expansion,

$$\widetilde{\boldsymbol{x}}_r[k] = f_r(\boldsymbol{x}_r^0[k], \boldsymbol{u}_r^0[k]) + \nabla f_r\left(\boldsymbol{x}_r^0[k], \boldsymbol{u}_r^0[k]\right) \begin{bmatrix} \boldsymbol{x}_r[k] - \boldsymbol{x}_r^0[k] \\ \boldsymbol{u}_r[k] - \boldsymbol{u}_r^0[k] \end{bmatrix}, \quad (3)$$

where $\boldsymbol{x}_r^0$ and $\boldsymbol{u}_r^0$ are the state and control of a robot's reference trajectory. We integrate this approximation for each discrete time interval and robot, resulting in the set of convex linear equalities,

$$\tilde{\boldsymbol{f}}_r[k] = \boldsymbol{x}_r[k] - \text{RK4}\left(\widetilde{\boldsymbol{x}}_r[k], \boldsymbol{x}_r[k-1], \boldsymbol{u}_r[k-1]\right) \quad (4)$$

$$\tilde{\mathcal{F}} = \{\tilde{\boldsymbol{f}}_r[k] = 0 \mid r \in \{1, \dots, R\}\, k \in \{1, \dots, N\}\} \quad (5)$$

where RK4 is a 4^{th}-order Runge-Kutta's integration method.

4) Obstacle Free Space: We approximate the free space in the environment with a set of convex polygons $\{\mathcal{P}_1, \dots, \mathcal{P}_P\}$. A polygon is defined as an intersection of multiple half-spaces. Let $\tau_{\mathcal{P}} : r \mapsto p$ be a mapping between each robot and a polygon, we assume such assignment is given. Furthermore, having L_r as the radius of the circle enclosing the geometry of robot r, we ensure that it remains in the obstacle free region by imposing the following set of convex linear inequalities,

$$\mathcal{H} = \{A_p q_r[k] - b_p + L_r \leq 0 \mid \forall_r p = \tau_{\mathcal{P}}[r]\} \quad (6)$$

where $A_p \in \mathbb{R}^{\text{H}_p \times 2}$ and $b_p \in \mathbb{R}^{\text{H}_p \times 1}$ and H_p represents the number of half-spaces of polygon $\mathcal{P}_p$.

5) Collisions: Robot-to-robot collisions are modeled via coupling constraints, where separation between robots is enforced with the following l_2-norm,

$$\boldsymbol{c}_{ij}[k] = ||\boldsymbol{q}_i[k] - \boldsymbol{q}_j[k]||_2^2 - (L_i + L_j)^2 \quad (7)$$

Similarly to the dynamics approximation in equation (3) we linearize the collision constraint of equation (7) with a first-order Taylor's approximation around a reference trajectory, which we will refer to as $\tilde{\boldsymbol{c}}_{ij}[k]$. Thus the set of linear inequality constraints that excludes robot-to-robot collisions is defined by the set of constraints,

$$\tilde{\mathcal{C}} = \{\tilde{\boldsymbol{c}}_{ij}[k] \geq 0 \mid i \neq j \in \{1, \dots, R\}\, k \in \{1, \dots, N\}\} \quad (8)$$

6) Trust Region: The approximations of collision and dynamic model constraints are reliable in a trust region around a reference trajectory $(\boldsymbol{x}^0, \boldsymbol{u}^0)$. Thus we impose that our decision variables have to be inside a trust region radius τ defined as follows,

$$\mathcal{T} = \left\{ (\boldsymbol{x}, \boldsymbol{u}) \mid \left|\left| \begin{vmatrix} \boldsymbol{x}^T - \boldsymbol{x}^{0^T} \\ \boldsymbol{u}^T - \boldsymbol{u}^{0^T} \end{vmatrix} \right|\right|_\infty \leq \tau \right\} \quad (9)$$

A bigger τ value entails the approximation correctly models the real problem in a bigger region around the reference trajectory while optimizing at faster convergence rate, with the *caveat* that a better approximation is more computationally demanding (e.g. convex second-order Taylor's expansion or particle fitting method).

7) Notes on Notation: Previously defined variables, functions and constraints are extended to the multi-robot case by dropping subscript r, as well as the opposite case, single-robot variables are extended to multi-robot case by introducing the subscript r. An upper-script tilde ("$\sim$") entails a first-order Taylor's approximation. E.g. $\tilde{\mathcal{F}}_r$ is the set of convex linear equalities that describe the approximated dynamics of the r-robot.

C. Optimization Problem

Based on the previous formulations, the problem of finding a trajectory for multiple robots navigating in an environment composed of static obstacles from start ($\boldsymbol{x}_{\text{start}}$) to a goal multi-robot state ($\boldsymbol{x}_{\text{goal}}$) while not colliding with each other can therefore be defined as follows,

Problem 1 (Multi-robot Trajectory Optimization):

$$\begin{aligned} \underset{\boldsymbol{x}, \boldsymbol{u}}{\text{minimize}} \;\; & f_0(\boldsymbol{u}) \\ \text{subject to} \;\; & \mathcal{F}, \mathcal{H}, \mathcal{C} \\ & \boldsymbol{x}[0] = \boldsymbol{x}_{\text{start}},\, \boldsymbol{x}[N] = \boldsymbol{x}_{\text{goal}} \\ & \underline{\boldsymbol{u}} \leq \boldsymbol{u} \leq \overline{\boldsymbol{u}}, \end{aligned}$$

with upper ($\overline{\boldsymbol{u}}$) and lower ($\underline{\boldsymbol{u}}$) bounds on the input controls. The *Problem 1* has realistic complex dynamics ($\mathcal{F}$) and coupling non-convex collision constraints ($\mathcal{C}$). In the next section, we exploit the approximations previously defined as a tool to efficiently solve this problem.

IV. Approach

The Multi-robot Trajectory Optimization *Problem 1* is hard to solve due to non-convexities introduced by complex robot dynamics and collision constraints that couple the robots' decision variables. Firstly, non-convexity can be handled by a process of successive convexification of non-convex optimal problems, an approach that is known as Sequential Convex Programming (SCP) [4]. For that, we will describe an l_1-penalty method for sequential convex programming tailored to solve the multi-robot trajectory optimization *Problem 1* (see Section IV-A). Secondly, the problem of coupling collision constraints is addressed by a distributed consensus optimization algorithm that can efficiently parallelize the problem in terms of robots while ensuring the solution convergence (see Section IV-B).

A. Multi-robot Sequential Convex Programming (mSCP) an l_1-penalty Method

We start by defining an l_1-penalty function approximation as follows

$$\tilde{\phi}(\boldsymbol{x},\boldsymbol{u}) = f_0(\boldsymbol{u}) + \sum_r \lambda_{f_r} \sum_k |\tilde{\boldsymbol{f}}_r[k]| + \lambda_c \sum_k \sum_{i\neq j} |\tilde{\boldsymbol{c}}_{ij}[k]|^+$$
$$\text{where } \tilde{\boldsymbol{f}}_r[k] = 0 \in \tilde{\mathcal{F}},\ -\tilde{\boldsymbol{c}}_{ij}[k] \leq 0 \in \tilde{\mathcal{C}} \tag{10}$$

, where we penalize dynamic infeasibility by manipulating λ_{f_r}, and robot collisions with λ_c. Note that $|c(.)|^+ = \max(c(.), 0)$. In other words, we move dynamic and collision equations from the set of constraints to the cost function, thus allowing and penalizing initial infeasibiliy.

Next, we define an approximation of *Problem 1* around a reference trajectory. This approximated Multi-robot Trajectory Optimization *Problem 2* is a combination of the previously defined l_1-penalty cost function (10) with the approximated constraints (i.e. dynamic , $\tilde{\mathcal{F}}$, and collision, $\tilde{\mathcal{C}}$, constraints), whereas the non-approximated constraints are straightforwardly imposed (e.g. obstacle free space constraints $\mathcal{H}$),

Problem 2 (Multi-robot Trajectory Optim. Approx.):

$$\underset{\boldsymbol{x},\boldsymbol{u}}{\text{minimize}}\ \ \tilde{\phi}(\boldsymbol{x},\boldsymbol{u})$$
$$\text{subject to}\ \ (\boldsymbol{x},\boldsymbol{u}) \in \mathcal{T} \tag{11}$$
$$\boldsymbol{x}[0] = \boldsymbol{x}_{\text{start}},\ \boldsymbol{x}[N] = \boldsymbol{x}_{\text{goal}} \tag{12}$$
$$\underline{\boldsymbol{u}} \leq \boldsymbol{u} \leq \overline{\boldsymbol{u}} \tag{13}$$
$$\mathcal{H} \tag{14}$$

where in summary, the decision variables $(\boldsymbol{x},\boldsymbol{u})$ of *Problem 2* have to lie on the trust region (11), robots navigate in the free space (14) while starting and finishing in a predefined state (12) with bounds on controls (13).

To solve the overall non-convex *Problem 1* we can now exploit sequentially computed solutions of the convex *Problem 2* as described in the multi-robot sequential convex programming (mSCP) **Algorithm 1**. This algorithm (mSCP) is composed of three main loops, where the outer loop **[line 1]** ensures there is no constraint violation while inner loops ensure convergence of the approximated problem solution trajectories **[line 3]** as well as trust region **[line 5]**.

In order to have better intuition on **Algorithm 1** we start our explanation on inner loops and proceed to the outter loops. The trust region loop **[line 5]** is responsible for expanding or shrinking this region depending on how accurate is the approximated $\tilde{\phi}$ versus actual cost ϕ reduction, as measured by the fraction $\tilde{\delta}/\delta$. On the approximation loop **[line 3]**, both dynamic and collision functions are approximated/convexified over the current trajectory **[line 4]**, using the first-order Taylor's expansion described in equation (3), and we test if either the solution trajectory or the overall cost has converged **[line 24]**. Moreover, on the outer violation loop **[line 1]** we check for constraint violations and increase by a factor of k the violation penalties λ if necessary. Additionally, *Problem 2* **[line 6]** is solved using an interior-point method implemented by an off-the-shelf solver.

Algorithm 1: Multi-robot SCP

Parameters:
$\boldsymbol{\tau^- \leq 1, \tau^+ \geq 1}$: expand and shrink factors
$\boldsymbol{\gamma_0 \approx 0, \gamma_1 \leq 1}$: convergence parameters
$\boldsymbol{x^{(0)}, u^{(0)}}$: initial guess trajectory
$\boldsymbol{k}$: violation penalty factor
$\boldsymbol{c_{tol} \approx 10^{-3}, f_{tol} = x u_{tol} \approx 10^{-2}}$: tolerances
$\boldsymbol{i \leftarrow 0}$: iteration number
Output: $\boldsymbol{x^{(i)}, u^{(i)}}$: multi-robot trajectory

1 **while** *(Constraint Violations)* **do**
2 $\tau \leftarrow \tau_0$ /* reset trust region */
3 **while** *(¬Approximation Converged)* **do**
4 $\tilde{f}, \tilde{c} \leftarrow$ convexity$(f, c, \boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)})$
5 **while** *(¬Trust Region Converged)* **do**
6 $\boldsymbol{x}^*, \boldsymbol{u}^* \leftarrow$ solve *Problem 2*
  /* Approx. and Actual Reduction */
7 $\tilde{\delta} = \phi(\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}) - \tilde{\phi}(\boldsymbol{x}^*, \boldsymbol{u}^*)$
8 $\delta = \phi(\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}) - \phi(\boldsymbol{x}^*, \boldsymbol{u}^*)$
9 **if** $\delta/\tilde{\delta} \geq \gamma_0$ **then** /* accept solution */
10 $(\boldsymbol{x}^{(i+1)}, \boldsymbol{u}^{(i+1)}) \leftarrow (\boldsymbol{x}^*, \boldsymbol{u}^*)$
11 $i \leftarrow i + 1$
12 **if** $\delta/\tilde{\delta} \geq \gamma_1$ **then**
13 $\tau \leftarrow \tau^+ \tau$ ▷**go to** 4 /* expand $\mathcal{T}$ */
14 **else**
15 $\tau \leftarrow \tau^- \tau$ /* shrink $\mathcal{T}$ */
16 **if** $\delta/\tilde{\delta} < 0$ **then** /* line-search */
17 $(\boldsymbol{x}^{(i+1)}, \boldsymbol{u}^{(i+1)}) \leftarrow$ recover $(\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}, \boldsymbol{x}^*, \boldsymbol{u}^*)$
18 $i \leftarrow i + 1$
19 ▷ **go to** 25
20 **if** $\tau < \tau_{min}$ **then** ▷**go to** 4 /* $\mathcal{T}$ small */
21
  /* cost change */
22 $\Delta f = |f_0(\boldsymbol{u}^{(i+1)}) - f_0(\boldsymbol{u}^{(i)})|$
  /* solution change */
23 $\Delta xu = \left\| \begin{vmatrix} \boldsymbol{u}^{(i+1)} - \boldsymbol{u}^{(i)} \\ \boldsymbol{x}^{(i+1)} - \boldsymbol{x}^{(i)} \end{vmatrix} \right\|_\infty$
24 **if** $(\Delta f \leq f_{tol} \cup \Delta xu \leq xu_{tol})$ **then** ▷**go to** 25
25 **for** $r = \{1, \ldots, R\}$ **do** /* dynamics violation */
26 **if** $\max(\mathcal{F}_r) \leq c_{tol}$ **then** $\lambda_{f_r} \leftarrow k\ \lambda_{f_r}$
27 **if** $\max(\mathcal{C}) \leq c_{tol}$ **then** $\lambda_c \leftarrow k\ \lambda_c$ /* colliding */
28

In conclusion, a standard l_1-penalty sequential convex programming method is applied in the context of the multi-robot trajectory optimization problem in hand, thus for additional details consult [7] and [31]. Nevertheless, there are some differences worth mentioning. Firstly, the violation penalty k is increased per individual robot once dynamic constraints are violated. Secondly, the exiting criteria of each loop is tailored to the problem in hand. And, finally we introduce a backtracking line-search mechanism that recovers a trajectory in case the predicted cost diverges.

Note that, the penalty function with no approximation $\phi(\boldsymbol{x},\boldsymbol{u})$ is similarly defined as $\tilde{\phi}(\boldsymbol{x},\boldsymbol{u})$ but with the actual dynamics (f_r) and collision constraints (c_{ij}).

Parameters:
$\boldsymbol{\alpha} < 0.5, \boldsymbol{\beta} > 1, \boldsymbol{t} = 1$

1 **Function** `recover` $(\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}, \boldsymbol{x}^*, \boldsymbol{u}^*)$ **:**

2 $\quad \Delta_x = \boldsymbol{x}^* - \boldsymbol{x}^i$, $\Delta_u = \boldsymbol{u}^* - \boldsymbol{u}^i$

3 $\quad$ **while** $\phi\left(\boldsymbol{x}^i - t\,\Delta_x, \boldsymbol{u}^i - t\,\Delta_u\right) \leq$
$\quad\quad \phi(\boldsymbol{x}^i, \boldsymbol{u}^i) - \alpha\, t\left(\phi(\boldsymbol{x}^i, \boldsymbol{u}^i) - \tilde{\phi}(\boldsymbol{x}^*, \boldsymbol{u}^*)\right)$ **do**

4 $\quad\quad t = \frac{t}{\beta}$

5 $\quad\quad \boldsymbol{x}^{(i)} \leftarrow \boldsymbol{x}^{(i)} + t\,\Delta_x$

6 $\quad\quad \boldsymbol{u}^{(i)} \leftarrow \boldsymbol{u}^{(i)} + t\,\Delta_u$

7 $\quad$ **return** $\left(\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)}\right)$

1) General Notes: Figure 1 illustrates the standard behavior of **Algorithm 1** (mSCP) per iteration until a solution is found. For instance, the actual (ϕ) and model ($\tilde{\phi}$) costs converge over iterations 3-5 whereas diverging on iteration 6. This means that 3-5 iterations occurred inside while loop on line 3, and once the costs have converged (i.e. on iteration 5), one exits the while loop. Next, the violation of the dynamic constraints (i.e. yellow curve iteration 5) is detected on line 25 and penalized, thus resulting in a reduction of the dynamic violation and an initial divergence on the costs on iteration 6. The same behavior occurs on iteration 8 but now with a collision constraint violation (i.e. purple curve). Finally and as expected, the free space constraints (i.e. blue curve) are always ensured and **Algorithm 1** (mSCP) finds a solution on iteration 11 with converged costs and no constraint violation.

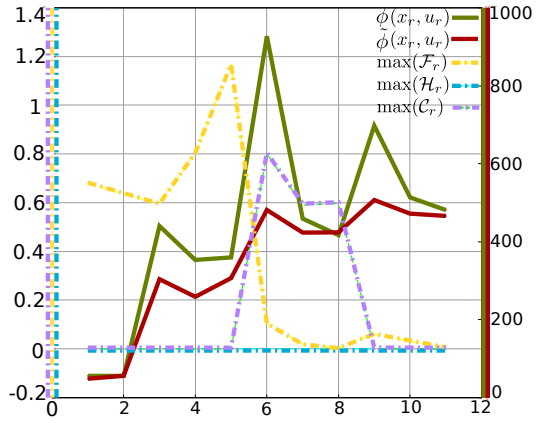

Fig. 1: Approximated and actual penalty costs (red and green curves with right y-axis) and maximum violation of dynamic, free space and collision constrains (yellow, blue and purple curves respectively with left y-axis) over each iteration (x-axis) of **Algorithm 1**. This represents the evolution of these quantities for a problem with 5 robots.

B. Distributed Multi-robot Optimization Method (DiMOpt)

The previously described method is able to handle non-convexity efficiently, nevertheless the amount of constraints increases combinatorially with the amount of robots, resulting in a poor problem scaling. Therefore, we have developed a Distributed Multi-robot Optimization (DiMOpt) method to overcome this issue.

Specifically, a key idea of this distributed method is to redefine collision constraints such that one can decouple *Problem 2* in terms of robots. Which in turn, we iteratively solve for each individual robot, in parallel, where all the other robot's trajectories are fixed. As a result, the collision constraint takes now the form,

$$\boldsymbol{c}_{i\hat{j}}[k] = ||\boldsymbol{q}_i[k] - \hat{\boldsymbol{q}}_j[k]||_2^2 - (L_i + L_j)^2 \tag{15}$$

where $\boldsymbol{q}_i[k]$ is a decision variable representing i-robot xy-coordinates, while $\hat{\boldsymbol{q}}_j[k]$ is constant representing j-robot xy-coordinates. As a consequence, it is now possible to decompose the l_1-penalty cost function shown in equation (10) per robot in the following manner,

$$\begin{aligned}
&\tilde{\phi}_r(\boldsymbol{x_r}, \boldsymbol{u_r}) = f_{0_r}(\boldsymbol{u_r}) + \\
&\quad \lambda_{f_r} \sum_k |\tilde{\boldsymbol{f}}_r[k]| + \lambda_{c_r} \sum_k \sum_{r \neq j} |\tilde{\boldsymbol{c}}_{r\hat{j}}[k]|^+ \\
&\text{where } \tilde{\boldsymbol{f}}_r[k] = 0 \in \tilde{\mathcal{F}},\ -\tilde{\boldsymbol{c}}_{r\hat{j}}[k] \leq 0 \in \tilde{\mathcal{C}}
\end{aligned} \tag{16}$$

and as expected $\tilde{\boldsymbol{c}}_{r\hat{j}}[k]$ is the first-order-taylor approximation of $\boldsymbol{c}_{r\hat{j}}[k]$.

Notice that the pair of collision constraints $\boldsymbol{c}_{i\hat{j}}[k]$ and $\boldsymbol{c}_{\hat{i}j}[k]$ refer to the same distance between robot i and j, where in $\boldsymbol{c}_{i\hat{j}}[k]$ robot j is fixed and vice-versa. In other words, each robot has a local representation of where are the others. Therefore, the missing piece of the puzzle is that these quantities need to converge, which is the same as stating that in the end each robot will know the other robot's trajectories. For that, a consensus optimization algorithm was developed, as described in the pseudo-code in **Algorithm 2**, where we introduce a consensus variable $\boldsymbol{z}_r[k] \in \mathbb{R}^2$ for each robot position $\boldsymbol{q}_r$, where we penalize for deviations of the consensus variable in augmented lagrangian form with scaled dual variables as,

$$\begin{aligned}
&\tilde{\psi}_r(\boldsymbol{x_r}, \boldsymbol{u_r}) = \tilde{\phi}_r(\boldsymbol{x_r}, \boldsymbol{u_r}) + \\
&\frac{1}{2}\rho_{\text{al}} \sum_k \left(||\boldsymbol{q}_r[k] - \boldsymbol{z}_r[k] + \boldsymbol{\lambda}_{z_r}[k]||_2^2\right)
\end{aligned} \tag{17}$$

where, $\boldsymbol{\lambda}_{z_r}[k] \in \mathbb{R}^2$ are multipliers that penalize for non-consensus and ρ_{al} is the augmented lagrangian constant. In summary and for clarity reasons, the consensus variable $\boldsymbol{z}_r$ can be understood as local copy of the r-robot trajectory that all the other robots have, and therefore we penalize r-robot for deviations of such trajectory. Finally, the full optimization problem each robot will iteratively solve is,

Problem 3 (Single-robot Consensus Optimization):

$$\begin{aligned}
\underset{\boldsymbol{x}_r, \boldsymbol{u}_r}{\text{minimize}}\ & \tilde{\psi}_r(\boldsymbol{x}_r, \boldsymbol{u}_r) \\
\text{subject to}\ & (\boldsymbol{x}_r, \boldsymbol{u}_r) \in \mathcal{T}_r \\
& \boldsymbol{x}_r[0] = \boldsymbol{x}_{\text{start}_r},\ \boldsymbol{x}_r[N] = \boldsymbol{x}_{\text{goal}_r} \\
& \underline{\boldsymbol{u}}_r \leq \boldsymbol{u}_r \leq \overline{\boldsymbol{u}}_r \text{ and } \mathcal{H}_r
\end{aligned}$$

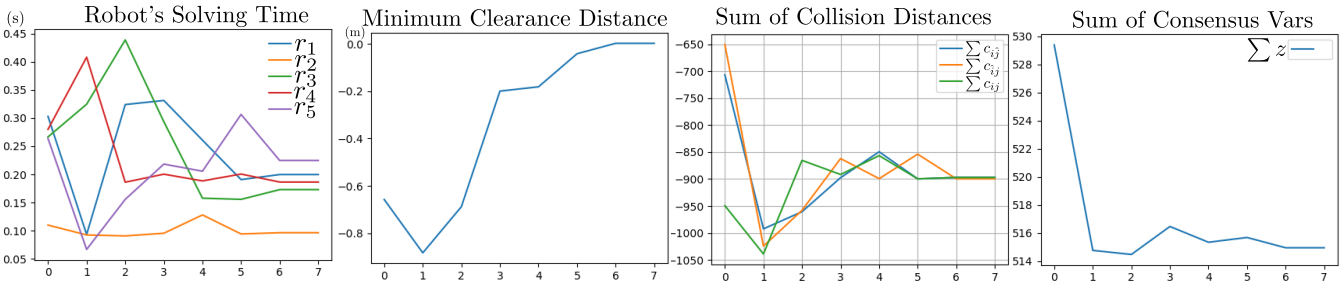

Fig. 2: Evolution of several quantities over each iteration until an optimal solution is found on Algorithm 2.

Algorithm 2: Distributed Multi-robot Optimization (DiMOpt)

Parameters:
$\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}$: initial guess trajectory
$\boldsymbol{\rho}_{\text{al}} \leftarrow \mathbf{0.1}$: augmented lagrangian constant
$\boldsymbol{z_r} \leftarrow \hat{\boldsymbol{q}}_{\boldsymbol{r}}$: consensus variables
$\boldsymbol{\lambda_r} \leftarrow \mathbf{0}$: consensus multipliers
Output: $\boldsymbol{x}^*, \boldsymbol{u}^*$: multi-robot trajectory

1 **while** ($\neg$ *Cost Converged* $\cup$ *Colliding*) **do**
  /* 1) Solve - In parallel */
2 **In Parallel**
3 **for** $r = \{1, \dots, R\}$ **do** /* robot scp */
4 update_*Problem_3* ($\hat{\boldsymbol{x}}_r, \boldsymbol{z}_r, \boldsymbol{\lambda}_r$)
5 $\boldsymbol{x}_r^*, \boldsymbol{u}_r^* \leftarrow$ Algorithm 1 [solve *Problem 3*]
  /* 2) Update / Share Knowledge */
6 **for** $r = \{1, \dots, R\}$ **do** /* trajectories */
7 $\hat{\boldsymbol{x}}_r \leftarrow \frac{1}{2}(\boldsymbol{x}_r^* - \hat{\boldsymbol{x}}_r)$
8 **for** $r = \{1, \dots, R\} \mid i \neq j$ **do** /* consensus */
9 **for** $k = \{1, \dots, N\}$ **do**
10 $\boldsymbol{z}_r[k] \leftarrow \frac{1}{2}(\boldsymbol{q}_r^*[k] + \boldsymbol{z}_r[k]) + b(\boldsymbol{q}_r^*[k] - \boldsymbol{z}_r[k])$
11 $\boldsymbol{\lambda}_r[k] \leftarrow \boldsymbol{\lambda}_r[k] + (\boldsymbol{q}_r^*[k] - \boldsymbol{z}_r[k])$
12 **if** $\max(\mathcal{C}) \leq 0 \cup \Delta_f \leq f_{tol}$ **then break**

in which, in contrast with *Problem 2* one has single robot decision variables and cost function as a combination of exact penalti and consensus terms.

The distributed multi-robot optimization described in **Algorithm 2** has two major steps. Firstly, each robot's problem (i.e. *Problem 3*) is updated with the last trajectory of the other robots $\hat{x}_r$, consensus variables z_r and penalti multipliers λ_r **[line 4]**, which is concurrently solved exploiting **Algorithm 1**. That is, in **line 6** we solve for *Problem 3* instead of *Problem 2* and exact penalti function $\phi(.)$ is substituted by $\psi(.)$; finally exit critireon for non-collision **[line 27]** has alternatively a set of constraints as defined in equation (15). Secondly, once all the robots have solved their individual problems then trajectories are updated as well as consensus multipliers and variables according to what is described in **lines 10-11**. That is, the new trajectory will be an average between the last trajectory and the newly computed one **[line 7]**, while the consensus variable and multiplers are computed according to consensus optimization via ADMM algorithm with scaled dual variable described in [5] such that $\boldsymbol{z}_r$ converges to $\boldsymbol{q}_r$. Notice that, we introduce a momentum term: $b(\boldsymbol{q}_r^*[k] - \boldsymbol{z}_r[k])$ **[line 10]** characteristic of heavy ball methods aimed to accelerate the convergence rate [32], where we decide $b = \frac{R-1}{R}$.

1) General Notes: The standard behaviour of Algorithm 2 is illustrated in Figure 2. This represents a particularly difficult problem with 5 robots since in almost all iterations the robots are colliding (Minimum Clearance distance is negative). Graph entitled Sum of Collision Distances represents three curves: the real sum of distances between robots c_{ij}, the sum as seen from i-robot $c_{i\hat{j}}$ and the other as seen form j-robot $c_{\hat{i}j}$ (i.e. collision constraints are pairwise) and we can see these converging as it happens for the consensus variables z. Notice that, the maximum robot's solving time as a downward trend with each iteration, this happens because we reuse the previously computed solution as a new initial guess. Moreover, this downward effect would be more accentuated when minimum clearance distance is not negative since intuitively the trajectories would not change as much.

V. Experiments and Results

A. Setup

The presented method was tested on a PC running Ubuntu 20.04.3 LTS (Focal Fossa) equipped with a AMD® Ryzen 9 5900x 12-core processor and 24 threads. Our software exploits pre-existing libraries on non-linear optimization and optimal control *CasADi* [33] and a message passage interface that facilitates parallel and distributed computing *Open MPI* [34].

The results are presented in the two following sections; where we firstly make a scaling comparison between Multi-robot SCP and DiMOpt approaches (section V-B), and secondy evaluate some important factors that increase the complexity of a problem solved by the DiMOpt method (section V-C).

B. Comparison between Multi-robot SCP and DiMOpt

The next sections V-B.1, V-B.3 and V-B.2 present different metrics on the same experimental setup. We have conducted a set of 200 experiments for each set of robots ranging from 2 to 18 robots, where robots with $0.1\,(\text{m})$ in diameter navigate in a square room of $25\,m^2$ (i.e. four free space constraints per robot, as described in equation 6).

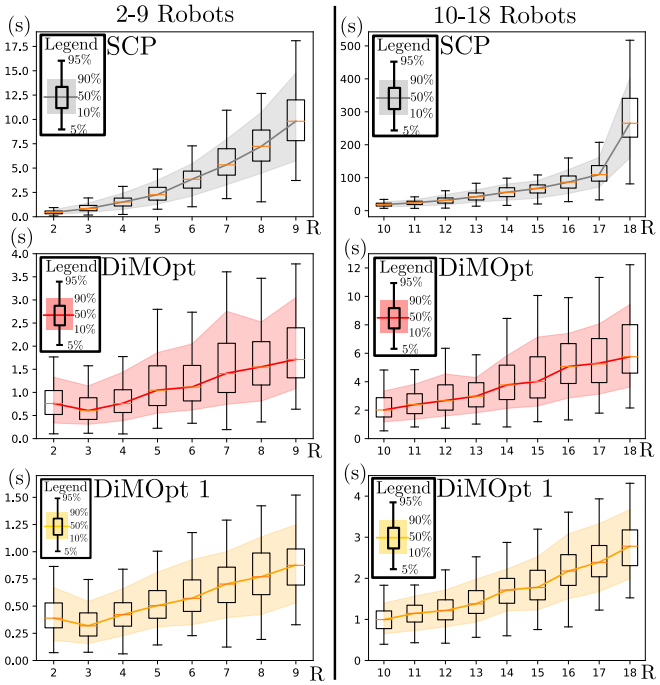

Fig. 3: Computation time (y-axis) of SCP, DiMOpt approaches and the first solution found DiMopt 1 with increasing amount of robots (x-axis), for a set of 2-9 robots on the left and 10-18 on the right.

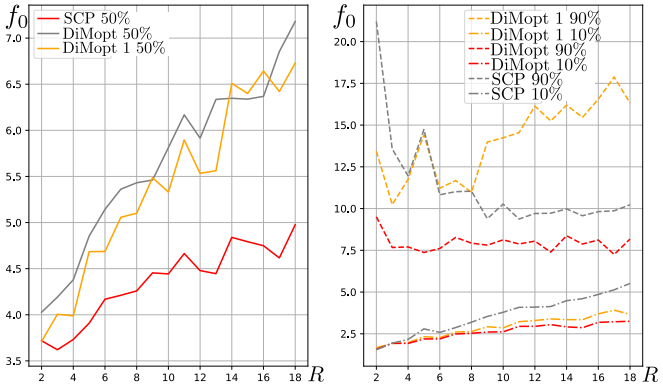

Fig. 4: Value of the cost function for the SCP and DiMOpt methods and for the first solution DiMOpt 1 for the 10%, 50% and 90% quantile.

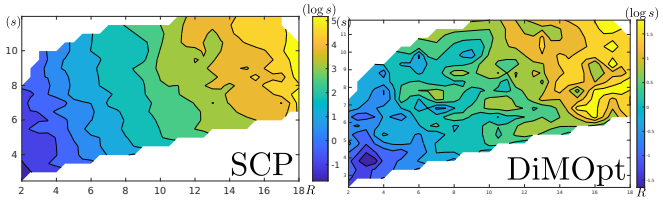

Fig. 5: Computation time in logarithmic scale (colobar) of SCP and DiMOpt methods with respect to the trajectory time (y-axis) and amount of robots (x-axis).

1) Scaling - Amount Robots: Figure 3 depicts the computation time with respect to the amount of robots for the SCP and DiMopt approaches, where one can detect a exponential growth for the SCP and a linear one for the DiMOpt method. Moreover, the first feasible and probably suboptimal solution of DiMOpt 1 can be found bellow $3(s)$ for 18 robots.

2) Solution Quality: In Figure 4, DiMOpt has an overall average and worst case lower cost than the SCP method. Interestingly, we can see that the cost of the final solution of SCP and DiMOpt 1 is relatively similar on average, however when the amount of robots is superior to 8 then DiMOpt 1 has an higher cost. This can be a result of the fact that, in the worst case, robots are required to execute more maneuvers thus the initial solution under DiMOpt being sub-optimal.

3) Scaling - Trajectory Time: Now, we will compare how the computation time is affected with an increasing amount of robots versus trajectory time (e.g. related to path length). Figure 5 shows that the method SCP is practically solely correlated with the amount of robots. On the other hand, although the DiMOpt method computation time is also highly correlated with the amount of robots, we can also confirm an higher correlation with the trajectory time.

C. Complexity of DiMOpt

Each of the sub-sections V-C.2 and V-C.1 has its own experimental setup. This section will evaluate the DiMOpt algorithm response when we increase the scenario occupancy and when the scenario occupancy is fixed with an increasing amount of robots.

1) Vary Scenario Occupancy: We have conducted a set of 30 experiments for each scenario occupancy percentage with a set of 5 robots and a maximum of 500 iterations. The problem of finding the maximum number of circles (robots) on a square (room) is known as the circle packing problem [35], since this computation is not trivial and out of the scope of this paper, then for simplicity reasons, we will assume a rectangular packing pattern for the scenario occupancy calculation. Therefore, the x-axis of Figure 6 represents the occupation percentage of the scenario by the fleet of robots and the radius of the robots is computed as follows:

$$L_r = \sqrt{\frac{Scenario\,Occupancy \times Room\,Size}{R}} \tag{18}$$

, where, $Scenario\,Occupancy \in [0, 1]$, $Room\,Size = 25$ and $R = 5$. Results reveal that the computation time of DiMOpt method is exponentially related to the scenario occupancy percentage. Furthermore, the success rate is around 70% when we have a 50% occupation of the scenario, such result is expected since individual robot trajectories will overlap over several iterations of DiMOpt, as we can see on the increase of the first iteration that DiMOpt algorithm found a feasible solution.

2) Fix Scenario Occupancy: In order to solely study the evolution of the computation time of DiMOpt approach by increasing the amount of robots, we have conducted 50 experiments per set of 2 to 20 robots while maintaining the scenario occupancy percentage at 5% as computed in equation (18). Figure 7 illustrates that the computation time

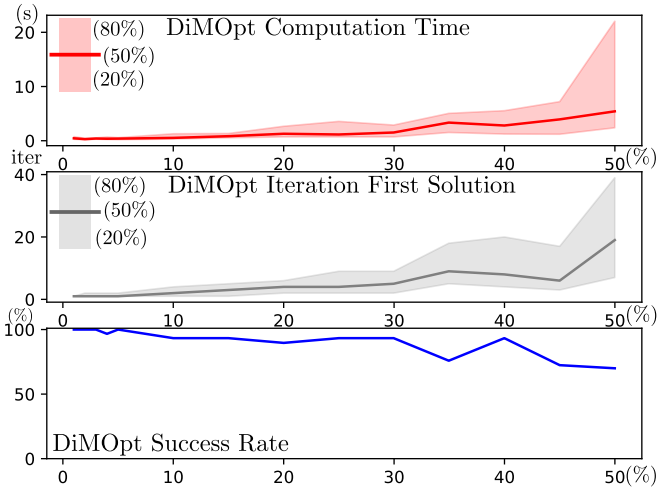

Fig. 6: Computation time versus scenario occupation percentage for the DiMOpt method (red), first solution DiMOpt 1 (grey) and the success percentage (blue).

and iteration of the first feasible solution increases with the amount of robots, however at a lower rate when comparing to what occurs when increasing the scenario occupancy, as depicted on the previous experiment in Figure 6. Finally, the computation time of DiMOpt approach is composed of a setup phase where the problem cost function and constraints are defined and a solving phase as described in Algorithm 2. Therefore, the last curve is the percentage of the setup time over the entire computation time, which as expected decreases with the amount of robots.

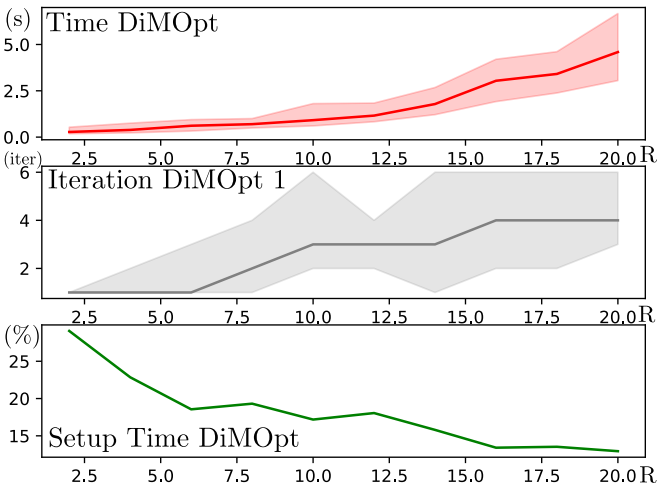

Fig. 7: Computation time versus amount of robots for the DiMOpt method (red), iteration of the first solution DiMOpt 1 (grey) and the problem setup time (green).

VI. Conclusions and Future Work

The multi-robot trajectory optimization problem can be efficiently tacked exploiting a mixture of sequential convex programming and distributed consensus optimization methods as described in **Algorithm 2** with pseudo-code of DiMOpt approach and corroborated on the conducted experiments. Moreover, a feasible sub-optimal solution can be found efficiently depending on the original problem complexity. We have also shown when the problem is particularly hard, e.g. robot scenario occupancy is high, then no efficient method exists; note that the same problem occurs with graph/tree search based methods. Finally, we foresee an increasing adoption of these distributed methods since the growth rate of the computation time over the number of robots is trailed by Moore's Law. Moreover, it would be interesting to investigate strategies for online deployment of this method, where a low hanging fruit can be savoured if one connects collision contraints and polygonal occupation, meaning for example, if robots do not occupy intersecting polygons then they can be considered separately.

Acknowledgments. This work is supported by the EU H2020 programme under grant agreement No. 732737 (ILIAD), Vinnova projects iQMobility and AutoHauler, and KKS research profile Semantic Robots.

References

[1] F. Pecora, H. Andreasson, M. Mansouri, and V. Petkov, "A loosely-coupled approach for multi-robot coordination, motion planning and control," in *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2018.

[2] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," *arXiv preprint arXiv:2005.07371*, 2020.

[3] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Siam, 2010, vol. 19.

[4] J. Duchi, S. Boyd, and J. Mattingley, "Sequential convex programming," in *Lecture Notes EE364b*. Stanford Univ., 2018.

[5] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[6] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2012, pp. 1917–1922.

[7] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Citeseer, 2013, pp. 1–10.

[8] A. Nedić and J. Liu, "Distributed optimization for control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 77–103, 2018.

[9] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[10] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proceedings 2001 ICRA. IEEE Int. Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 1. IEEE, 2001, pp. 271–276.

[11] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, p. 477, 1987.

[12] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *Int. J. Rob. Res. (IJRR)*, vol. 34, no. 12, pp. 1501–1514, 2015.

[13] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[14] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[15] K. Solovey and D. Halperin, "k-color multi-robot motion planning," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 82–97, 2014.

[16] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[17] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[18] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 3636–3641.

[19] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5954–5961.

[20] M. T. Shahab and M. Elshafei, "Distributed optimization of multi-robot motion with time-energy criterion," in *Path Planning for Autonomous Vehicles-Ensuring Reliable Driverless Navigation and Control Maneuver*. IntechOpen, 2019.

[21] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Dual decomposition for multi-agent distributed optimization with coupling constraints," *Automatica*, vol. 84, pp. 149–158, 2017.

[22] B. Houska, J. Frasch, and M. Diehl, "An augmented lagrangian based algorithm for distributed nonconvex optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1101–1127, 2016.

[23] A. Engelmann, Y. Jiang, B. Houska, and T. Faulwasser, "Decomposition of nonconvex optimization via bi-level distributed aladin," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1848–1858, 2020.

[24] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[25] M. Charitidou and D. V. Dimarogonas, "Signal temporal logic task decomposition via convex optimization," *IEEE Control Systems Letters*, 2021.

[26] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 434–440.

[27] S. Bhattacharya and V. Kumar, "Distributed optimization with pairwise constraints and its application to multi-robot path planning," *Robotics: Science and Systems VI*, vol. 177, 2011.

[28] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[29] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[30] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[31] Y. Mao, M. Szmuk, X. Xu, and B. Açikmese, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," *arXiv preprint arXiv:1804.06539*, 2018.

[32] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964.

[33] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[34] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[35] K. Stephenson, "Circle packing: a mathematical tale," *Notices of the AMS*, vol. 50, no. 11, pp. 1376–1388, 2003.