2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Motion Planning and Goal Assignment for Robot Fleets Using Trajectory Optimization

João Salvado, Robert Krug, Masoumeh Mansouri, Federico Pecora

Abstract— **This paper is concerned with automating fleets of autonomous robots. This involves solving a multitude of problems, including goal assignment, motion planning, and coordination, while maximizing some performance criterion. While methods for solving these sub-problems have been studied, they address only a facet of the overall problem, and make strong assumptions on the use-case, on the environment, or on the robots in the fleet. In this paper, we formulate the overall fleet management problem in terms of Optimal Control. We describe a scheme for solving this problem in the particular case of fleets of non-holonomic robots navigating in an environment with obstacles. The method is based on a two-phase approach, whereby the first phase solves for fleet-wide boolean decision variables via Mixed Integer Quadratic Programming, and the second phase solves for real-valued variables to obtain an optimized set of trajectories for the fleet. Examples showcasing the features of the method are illustrated, and the method is validated experimentally.**

I. Introduction

Automating fleets of autonomous robots involves solving a multitude of problems. These include allocating goals to robots, computing feasible robot motions to reach these goals, and ensuring that these motions are coordinated, all the while maximizing some performance criterion. These can be seen as sub-problems of an overall *fleet management problem*. Fig. 1 shows a simple example scenario: two goals have to be reached by two of the three robots, and part of the problem is to decide which robot should reach which goal; the motions of the robots are constrained by the obstacles in the environment; and the way in which goals are assigned to robots will determine whether and how robot motions should be coordinated. Numerous methods for solving sub-problems underlying the fleet management problem have been studied in the fields of AI, Robotics and Operations Research. In addition to addressing only a facet of the overall problem, most existing methods make strong assumptions on the use-case, on the environment, or on the robots in the fleet. Deploying existing methods in industrial settings therefore requires catering to these assumptions, e.g., via additional infrastructure [1], providing hand-crafted paths [2], or specifying traffic rules for deadlock avoidance [3].

In this paper, we define the fleet management problem by formulating it as an Optimal Control Problem (OCP). From an optimal control point of view, the motion of a robot fleet is a consequence of control commands which transition the robots from a given initial state to a desired final state while obeying potentially time-varying constraints, and maximizing some performance criterion. Our formulation of the problem includes both real and boolean decision variables, the latter being used to model, $e.g.$, constraints on goal allocation. The formulation is intentionally general (and intractable), and is intended to highlight the component problems underlying the overall problem of controlling a fleet of autonomous robots. We also propose a concrete instantiation of the fleet management problem for fleets of non-holonomic robots in environments with obstacles. The instantiation consists of two problems, which are solved in sequence: solving the first one decides boolean variables subject to relaxed constraints on the real-valued variables; the second problem accounts for the constraints on real variables, subject to the assignments of boolean variables decided by solving the first problem. The method is detailed formally, and evaluated empirically on simulated multi-robot scenarios.

João Salvado, Federico Pecora and Masoumeh Mansouri are with the AASS Research Centre, Örebro University, <name>.<surname>@oru.se
Robert Krug is with the Robotics, Perception and Learning Lab, KTH Royal Institute of Technology, rkrug@kth.se

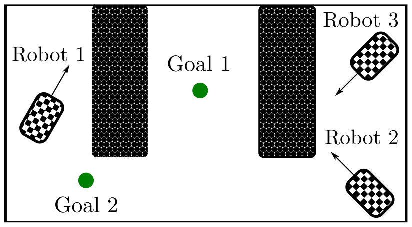

Fig. 1. Example of Fleet Management problem where two goals need to be reached by one robot each among the three available robots. The robots must reach the goals while not colliding with each other or with obstacles in the environment.

II. Problem Statement

A. Nomenclature

Robot index	$r = 1, \dots, R$
Discrete time index	$k = 1, \dots, N$
Goal index	$g = 1, \dots, G$
Polygon index	$p = 1, \dots, P$
Halfspace index	$h = 1, \dots, H$
Circular geometry index	$c = 1, \dots, C$

Bold letters are utilized to denote matrices (uppercase) and vectors (lowercase).

B. The Fleet Management Problem

We consider a multi-robot system in a two-dimensional world setting $\mathcal{W} \subseteq \mathbb{R}^2$. Each robot r is associated with a geometry $\mathcal{A}_r \subseteq \mathbb{R}^2$, a control space $\mathcal{U}_r = \{\boldsymbol{u}_r(t) \in \mathbb{R}^n\}$ and a state space $\mathcal{X}_r = \{\boldsymbol{x}_r(t) \in \mathbb{R}^m\}$. For notational convenience we assume that all robots have uniform control/state space dimension although this need not be the case in general. Let $\mathcal{U} = \left\{\boldsymbol{u}(t) \in \mathbb{R}^{Rn}\right\}$ be the compound fleet control space and $\mathcal{X} = \left\{\boldsymbol{x}(t) \in \mathbb{R}^{Rm}\right\}$ be the fleet state space formed by stacking the corresponding vectors of the individual robots. We describe regions covered by obstacles as a polyhedral space $\mathcal{O} \subseteq \mathbb{R}^2$ where robots cannot navigate. Furthermore, the obstacle state space $\mathcal{X}_o \subseteq \mathbb{R}^{Rm}$ is determined by robot-obstacle and robot-robot collisions and the free configuration space is defined as $\mathcal{X}_{\text{free}} = \mathcal{X} \backslash \mathcal{X}_o$. The fleet starts in an obstacle-free initial state $\boldsymbol{x}_0 \in \mathbb{R}^{Rm}$ and we are given a set of goal states to reach $\mathcal{X}_g = \{\boldsymbol{x}_g \in \mathbb{R}^m \mid g = 1, \dots, G\}$. Note that there can be less goals than available robots, *i.e.*, $G \leq R$. Additionally, we leverage boolean variables for decision making (*e.g.*, for robot goal assignment). These form the set $\mathcal{D} = \{d \in \{0,1\}\}$, for a given problem we stack all decision variables in the vector $\boldsymbol{d}(t) \in \mathbb{R}^{|\mathcal{D}|}$.

The objective is to allocate goals to individual robots in the fleet and to find a corresponding trajectory $\boldsymbol{\tau}(t) = [\boldsymbol{x}(t)^T, \boldsymbol{u}(t)^T]^T$ which transitions the fleet from its initial state to a valid goal configuration while traversing $\mathcal{X}_{\text{free}}$ in an optimal manner. This fleet management problem can be formalized as an Optimal Control Problem (OCP) where a cost functional in Bolza form is minimized

$$\min_{\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{d}(t)} L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) = \int_{t_0}^{t_f} l\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t, \boldsymbol{d}(t)\right) dt + \psi(\boldsymbol{x}(t_f), \boldsymbol{d}(t_f))$$

subject to

$$\boldsymbol{f}_r\left(\boldsymbol{x}_r(t), \boldsymbol{u}_r(t)\right) - \dot{\boldsymbol{x}}_r(t) = \boldsymbol{0}, \quad r = 1, \dots, R \tag{1}$$

$$\boldsymbol{B}(\boldsymbol{x}_0, \boldsymbol{x}_g, \boldsymbol{d}(t)) = \boldsymbol{0}, \quad g = 1, \dots, G \tag{2}$$

$$\boldsymbol{P}(\boldsymbol{x}(t), \boldsymbol{d}(t)) \leq \boldsymbol{0}, \tag{3}$$

$$\underline{\boldsymbol{u}} \leq \boldsymbol{u}(t) \leq \overline{\boldsymbol{u}}, \tag{4}$$

and where the constraints need to hold $\forall t \in [t_0, t_f]$. In the above formulation, the constraints in (1) ensure that the fleet's motion obeys the dynamics of the individual robots. Depending on the boolean decision variables, the boundary constraints in (2) enforce the initial and final states of the fleet while the path constraints in (3) ensure collision-free motion. Finally, the box constraints in (4) can be used to capture actuator limitations.

Finding an optimal solution to the fleet management OCP in form of a closed-loop policy $\boldsymbol{\pi}^* = \boldsymbol{u}(\boldsymbol{x}(t), \boldsymbol{d}(t), t)$ is daunting as the formulation comprises both real ($\boldsymbol{x}$, $\boldsymbol{u}$) and boolean ($\boldsymbol{d}$) decision variables. Furthermore, the dynamics of real-world robots are typically nonlinear. Therefore, in this work we limit ourselves to solving the trajectory optimization problem, *i.e.*, finding an optimal trajectory $\boldsymbol{\tau}^*(t)$ for a given initial state only. In practice, this will require a suitable tracking controller to stabilize the resulting trajectory. Here, we leave this aspect aside and focus on the theoretical foundation of trajectory optimization for the fleet-management problem.

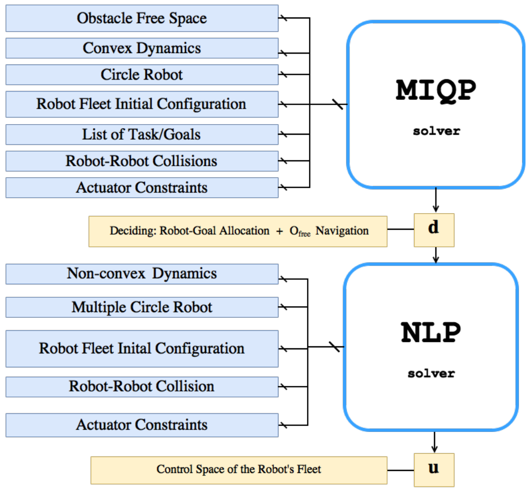

Fig. 2. Our two-phased approach to the fleet management problem

III. Approach

Our solution strategy is based on direct shooting to transform the continuous fleet-management OCP into a finite-dimensional optimization problem amenable to numerical solution [4]. To this end, we discretize in time according to $t_k = t_f k/N$, where t_f is a given final time. In shooting methods the optimal states $\boldsymbol{x}^*[k]$ are not directly solved for. Instead, they are obtained from the controls through numerical integration of the dynamics in (1), for which we use a 4^{th} order Runge-Kutta method. Discretizing the OCP in Section II-B yields a non-convex MINLP (Mixed Integer Nonlinear Problem) for which only a locally optimal solution can be found in general. The problem has the following input:

- Obstacle free polyhedral space $\mathcal{O}_{\text{free}}$
- Robot geometries $\mathcal{A}_1, \dots, \mathcal{A}_R$
- Fleet initial state $\boldsymbol{x}_0$
- Discrete robot dynamics
  $\boldsymbol{x}_r[k+1] = \boldsymbol{f}_r(\boldsymbol{x}_r[k], \boldsymbol{u}_r[k]),\ r = 1, \dots, R$
- Goals to reach $\boldsymbol{x}_1, \dots, \boldsymbol{x}_G$
- Fleet actuator constraints $\underline{\boldsymbol{u}}, \overline{\boldsymbol{u}}$

The solution of the fleet management problem is:

- A feasible sequence of control inputs $\boldsymbol{u}^*[0], \dots, \boldsymbol{u}^*[N]$ which minimizes a given cost function.

In order to leverage efficient existing solution strategies for integer programming and nonlinear programming respectively, we developed a two-phase solution procedure. The first phase solves for the boolean decision variables $\boldsymbol{d}^*[k]$ via Mixed Integer Quadratic Programming (MIQP). The solution of the MIQP is then used to parametrize a Nonlinear Programming Problem (NLP) which is solved subsequently. The latter yields the real-valued decision variables $\boldsymbol{u}^*[k]$ as illustrated in Fig. 2. The two solution stages will be discussed in detail in Sections III-B and III-C respectively.

A. Obstacle Avoidance

The obstacle-free space $\mathcal{O}_{\text{free}} = \mathcal{W} \backslash \mathcal{O}$ is approximated by the union of $\mathcal{P}_1, \dots, \mathcal{P}_P$ convex polygons.

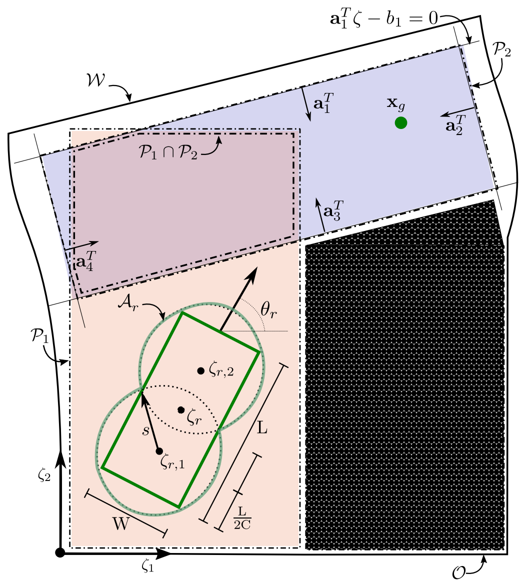

Fig. 3. A robot is navigating in world $\mathcal{W}$. The obstacle free region $\mathcal{O}_{\text{free}} = \mathcal{W}\backslash\mathcal{O}$ is decomposed into polygons $\mathcal{P}_1$ and $\mathcal{P}_2$, *i.e.*, $\mathcal{O}_{\text{free}} = \mathcal{P}_1 \cup \mathcal{P}_2$. In turn, Polygon $\mathcal{P}_2$ is defined by the intersection of $H = 4$ positive half-spaces, *i.e.*, $\mathcal{P}_2 = \{\boldsymbol{\zeta} \in \mathcal{W} \mid \boldsymbol{a}_h^T\boldsymbol{\zeta} - b_h \geq 0,\ h = 1,\ldots,4\}$. Robot r's rectangular geometry (parametrized by width W and length L) is approximated by the union $\mathcal{A}_r$ of two circular discs with radius s. The desired goal state is designated as $\boldsymbol{x}_g$.

Let us define a polygon as the intersection of positive half-spaces $\mathcal{P}_p = \{\boldsymbol{\zeta} \in \mathcal{W} \mid \boldsymbol{a}_h^T\boldsymbol{\zeta} - b_h \geq 0,\ h = 1,\ldots,H\}$, where $\boldsymbol{a}_h \in \mathbb{R}^2$ and $b_h \in \mathbb{R}$ denote the unit normal and offset of half-space h. We model the geometry of robot r by approximating its footprint in a conservative manner via the union of circular discs as shown in Fig. 3. For notational simplicity we assume that each robot geometry comprises the same number C of discs, that each disc has the same radius s and that each polygon $\mathcal{P}_p$ is formed by the same number H of half-spaces. However, these assumptions are not a general requirement in our approach. As in [5] we achieve obstacle avoidance by imposing linear constraints ensuring that each disc with center $\boldsymbol{\zeta}_{r,c}$ (which is a function of the configuration of robot r) is contained in at least one polygon $\mathcal{P}_p$. Corresponding constraints enforcing robot r to be contained in polygon $\mathcal{P}_p$ at time t_k can be written as

$$\boldsymbol{F}_p\boldsymbol{\zeta}_{r,c}[k] - \boldsymbol{b}_p - \mathbf{1}s \geq \mathbf{0}, \quad c = 1,\ldots,C \tag{5}$$

where the rows of $\boldsymbol{F}_p \in \mathbb{R}^{H\times 2}$ are formed by half-space normals $\boldsymbol{a}_h^T$ and the vector $\boldsymbol{b}_p \in \mathbb{R}^H$ contains the corresponding half-space offsets b_h. The vectors $\mathbf{1}$ and $\mathbf{0}$ in (5) are of appropriate dimension. The decision of which robot needs to be in what polygon at each time t_k is made during the MIQP stage. Robot-robot collision avoidance is solved differently in the MIQP stage and the NLP stage and will be detailed below.

B. MIQP Formulation

In the first phase we relax the overall problem by approximating the robots' dynamics with the uniform linear model

$$\boldsymbol{x}_r[k+1] = \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{x}_r[k] + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{bmatrix}}_{\boldsymbol{B}} \boldsymbol{u}_r[k]. \tag{6}$$

In (6) the robot's mass is denoted by m, the state is given by $\boldsymbol{x}_r[k] = [\boldsymbol{\zeta}_r^T, \dot{\boldsymbol{\zeta}}_r^T]^T$ and the control is defined as $\boldsymbol{u}_r[k] \in \mathbb{R}^2$. Here, we approximate each robot's footprint with a single circular shape, which is a conservative approximation and can lead to incompleteness. Nevertheless, note that robot orientation can be represented in the MIQP's state, allowing a better encapsulation of the robot's footprint. Furthermore, the cost function is chosen to be quadratic. As a result, for each possible assignment of discrete variables one has a convex problem with quadratic cost and linear constraints. Although solving a MIQP is still NP-hard, in practice solutions can be found efficiently via existing solvers, as we will show in the evaluation of our method in Section IV.

As obstacle avoidance requires assignment of robots to obstacle-free polygons at each time step we capture propositions of the form "Is robot r in polygon p at time t_k?" using RP boolean variables $e_{r,p}[k]$. Similarly, goal assignment "Is robot r assigned to goal g?" is formalized using RG boolean variables $y_{r,g}$. Finally, we formulate robot-robot avoidance in the MIQP by ensuring that all robot pairs are separated along at least one world coordinate direction (ζ_1 and/or ζ_2 respectively). To this end, we introduce boolean variables $l_{i,j}[k]$ whose value at time t_k is 1 if robots i and j are separated along ζ_2 and 0 if robots i and j are separated along ζ_1. For example, $(e_{3,2}[1] = 1)$ means that at t_1 robot 3 is in polygon 2, $(y_{4,2} = 1)$ means that robot 4 is assigned to goal 2 and $(l_{1,2}[3] = 1)$ means that robot 1 and 2 are separated along ζ_2 at t_3. All boolean decision variables are collected in vector $\boldsymbol{d}[k]$. The overall MIQP reads as

$$\min_{\substack{\boldsymbol{u}[k] \in \mathbb{R}^{2R},\ k=0,\ldots,N-1 \\ \boldsymbol{d}[k] \in \mathcal{D},\ k=0,\ldots,N}} L(\boldsymbol{u}(\cdot)) = \sum_{k=0}^{N-1} \boldsymbol{u}[k]^T\boldsymbol{R}\boldsymbol{u}[k] \tag{7}$$

subject to

$$\boldsymbol{x}[0] = \boldsymbol{x}_0, \tag{8}$$

$$\boldsymbol{x}_r[k+1] = \boldsymbol{A}\boldsymbol{x}_r[k] + \boldsymbol{B}\boldsymbol{u}_r[k], \quad r=1,\ldots,R;\ k=0,\ldots,N-1 \tag{9}$$

$$\mathbf{0} = (\boldsymbol{x}_g - \boldsymbol{x}_r[N])\, y_{g,r}, \quad r=1,\ldots,R;\ g=1,\ldots,G \tag{10}$$

$$1 = \sum_{g=1}^{G} y_{g,r}, \quad r=1,\ldots,R \tag{11}$$

$$1 = \sum_{r=1}^{R} y_{g,r}, \quad g=1,\ldots,G \tag{12}$$

$$1 \leq \sum_{p=1}^{P} e_{r,p}[k] \leq P, \quad r=1,\ldots,R;\ k=0,\ldots,N \tag{13}$$

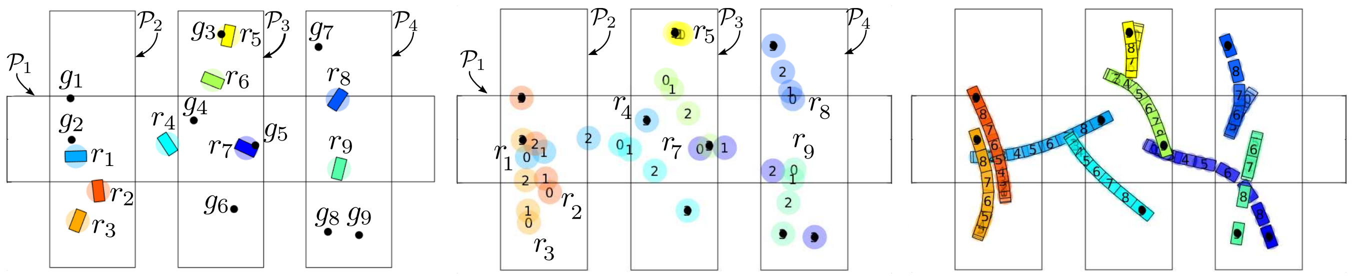

Fig. 4. (Left) Depicted is the initial configuration of a fleet consisting of 9 robots which navigate in obstacle-free space formed by the union of 4 polygons. In the illustration we use the shorthands r_i for $r = i$ and g_j for $g = j$. Also indicated are desired goal locations. (Center) Shown is the fleet trajectory obtained from the solution of a corresponding MIQP with $N = 3$ time-steps. Consider, *e. g.*, a subset of the solution for robot $r = 4$ which was allocated to goal $g = 6$ (g_6): $y_{4,6} = 1$, $e_{4,1}[1] = 1$, $e_{4,1}[2] = 1$, $e_{4,3}[2] = 1$, $e_{4,3}[3] = 1$. Therefore, the sequence of traversed polygons is $\mathcal{P}_1 \rightarrow \mathcal{P}_1 \cap \mathcal{P}_3 \rightarrow \mathcal{P}_3$. (Right) Represented is the final solution of the problem after the NLP phase with $N = 9$ time steps (solved by leveraging $\boldsymbol{d}^*[0], \ldots, \boldsymbol{d}^*[N]$ obtained from the MIQP solution shown in the center). Spatial and temporal coordination between, *e. g.*, robots $r = 7$ and $r = 9$ is evident. The robot paths are significantly different from the MIQP solution due to the non-holonomic robot dynamics considered in the NLP.

$$\mathbf{1}s \leq \boldsymbol{F}_p \boldsymbol{\zeta}_r[k] - \boldsymbol{b}_p + \mathbf{1}M(1 - e_{r,p}[k]), \tag{14}$$
$$\scriptstyle r=1,\ldots,R;\; p=1,\ldots,P;\; k=0,\ldots,N$$

$$\sum_{p=1}^{P} e_{p,r}[k] \geq \sum_{p=1}^{P} |e_{r,p}[k] - e_{r,p}[k+1]|, \tag{15}$$
$$\scriptstyle r=1,\ldots,R;\; k=0,\ldots,N-1$$

$$\mathbf{2}s \leq |\boldsymbol{\zeta}_{i,1}[k] - \boldsymbol{\zeta}_{j,1}[k]| + M \begin{bmatrix} l_{i,j}[k] \\ 1 - l_{i,j}[k] \end{bmatrix}, \tag{16}$$
$$\scriptstyle \forall (i,j) \in \{\{1,\ldots,R\} \times \{1,\ldots,R\} | j > i\};\; k=0,\ldots,N$$

$$\underline{\boldsymbol{u}} \leq \boldsymbol{u}[k] \leq \overline{\boldsymbol{u}}, \qquad \scriptstyle k=0,\ldots,N-1 \tag{17}$$

where $M \in \mathbb{R}_+$ in (14), (16) is a large positive number according to the Big M method. The quadratic cost function in (7) minimizes control energy and the constraint in (8) enforces the given initial state of the fleet. Each robot has to obey the chosen linear dynamics in (9). The task allocation in (10) ensures that the final states of those robots whose corresponding assignment variable $y_{g,r} = 1$ have to coincide with the corresponding goal states. Therefore, which robots are assigned to each goal is not decided a priori. Constraint (11) takes care that each robot is assigned no more than one goal whereas constraint (12) makes sure that each goal is assigned to a robot. Additionally, the constraint (13) ensures that each robot is in at least one obstacle-free polygon and (14) enforces that all circular geometries of robot r lie within polygon $\mathcal{P}_p$ at time t_k if the corresponding decision variable $e_{r,p}[k] = 1$ (*c. f.* (5)). Constraint (15) addresses the corner-cutting problem as shown in [5]. The right-hand side gives the accumulated number of polygons robot r enters or leaves between time-steps. As this needs to be smaller than the number of polygons robot r is currently in, the constraint enforces two consecutive states to lie within the same polygon. Consider the example in Fig. 3, where robot r has to travel from polygon $\mathcal{P}_1$ to $\mathcal{P}_2$ to reach goal x_g. Constraint (15) implies that it has to transverse the overlapping region $\mathcal{P}_1 \cap \mathcal{P}_2$. The partitioning of the obstacle free space is specified manually, although several off-the-shelf methods can be utilized, e.g. IRIS [6]. Finally, the constraint in (16) enforces separation of robot pairs along at least one coordinate direction and (17) takes care of actuator limitations. Note that the real absolute value function used in (14), (16) is convex but only piece-wise linear. However, it can be represented as a composition of linear constraints by introducing additional discrete slack variables [7]. Figure 4 depicts an exemplary MIQP solution. The illustrated configuration sequence of the fleet has been obtained by forward simulation of the dynamics using the obtained solution $\boldsymbol{u}^*[0], \ldots, \boldsymbol{u}^*[N]$ for the controls.

C. NLP Formulation

In the second solution phase we formulate a NLP with only real-valued decision variables as the boolean decision-making elements are taken from the solution $\boldsymbol{d}^*[k]$ of the MIQP. In particular, we exploit the pre-solved goal assignment ($y^*_{r,g}$) and fix the sequence of traversed obstacle-free polygons ($e^*_{r,p}[k]$). For notation purposes we assume the time-steps N in the MIQP and NLP to be the same, although this need not be the case. In fact, the amount of time steps in the MIQP only requires to match the amount of necessary polygon transitions, leaving a more fine grained time discretization for the NLP. In the case of adding additional NLP time-steps, the boolean $e^*_{r,p}[k]$ variables require post-processing to account for the dimension mismatch. We do not currently utilize the remaining solution elements from the MIQP ($\boldsymbol{u}^*[k]$ and $l^*_{i,j}[k]$) as the NLP builds upon more expressive robot dynamics and robot-robot avoidance formulation. Specifically, we approximate robot footprints with multiple circular discs and use a non-convex modified Reeds-Shepp model, where speed is not constant:

$$\boldsymbol{x}_r[k+1] = \boldsymbol{f}_r(\boldsymbol{x}_r[k], \boldsymbol{u}_r[k]) = \begin{bmatrix} v_r[k]\cos(\theta_r[k]) \\ v_r[k]\sin(\theta_r[k]) \\ \kappa_r[k] v_r[k] \\ \dot{v}_r[k] \end{bmatrix}. \tag{18}$$

In (18) the state vector is given by $\boldsymbol{x}_r[k] = [\boldsymbol{\zeta}_r^T,\, \theta_r,\, v_r]^T$ and we define a vector of controls as $\boldsymbol{u}_r[k] = [\dot{v}_r,\, \kappa_r]^T$. Here, $\theta_r[k]$ describes the heading of robot r, $v_r[k]$ is the speed and $\kappa_r[k]$ the curvature. Overall, the resulting NLP can be stated as

$$\min_{\substack{\boldsymbol{u}[k]\in\mathbb{R}^{2R},\\ k=0,\dots,N-1}} \quad L\left(\boldsymbol{u}(\cdot)\right) = \sum_{k=0}^{N-1} \boldsymbol{u}[k]^T \boldsymbol{R} \boldsymbol{u}[k] \tag{19}$$

subject to

$$\boldsymbol{x}[0] = \boldsymbol{x}_0, \tag{20}$$

$$\boldsymbol{x}_r[k+1] = \boldsymbol{f}_r(\boldsymbol{x}_r[k], \boldsymbol{u}_r[k]), \quad {\scriptstyle r=1,\dots,R;\ k=0,\dots,N-1} \tag{21}$$

$$\boldsymbol{0} = \left(\boldsymbol{x}_g - \boldsymbol{x}_r[N]\right), \tag{22}$$

$$\forall (g,r)\in\left\{\{1,\dots,G\}\times\{1,\dots,R\} \mid y^*_{g,r}=1\right\}$$

$$\boldsymbol{1}s \le \boldsymbol{F}_p \boldsymbol{\zeta}_{r,c}[k] - \boldsymbol{b}_p, \tag{23}$$

$$\forall (p,r,c,k)\in\left\{\{1,\dots,P\}\times\{1,\dots,R\}\times\{1,\dots,C\}\times\{1,\dots,N\} \mid e^*_{r,p}[k]=1\right\}$$

$$\boldsymbol{0} \le \|\boldsymbol{\zeta}_{i,c}[k] - \boldsymbol{\zeta}_{j,c}[k]\|_2 - \boldsymbol{2}s, \tag{24}$$

$$\forall (i,j,c,k)\in\{\{1,\dots,R\}\times\{1,\dots,R\}\times\{1,\dots,C\}\times\{1,\dots,N\} \mid j>i\}$$

$$\underline{\boldsymbol{u}} \le \boldsymbol{u}[k] \le \overline{\boldsymbol{u}}. \quad {\scriptstyle k=0,\dots,N-1} \tag{25}$$

The cost function in (19), the initial condition constraint in (20) and the actuator limitations in (25) have the same structure as in the MIQP. The discrete dynamics in (21) correspond to the non-holonomic model in (18). The goal allocation in (22) leverages the result of the previously solved MIQP and sets the final states of a subset of robots to their respective goals if the corresponding boolean variable $y^*_{r,g} = 1$. In a similar fashion, at each time-step t_k every robot r is assigned to at least one obstacle-free polygon according to $e^*_{r,p}[k]$. Robot-robot obstacle avoidance in the NLP is formulated in constraint (24). It ensures that the Euclidean distance between all circular discs approximating the geometries of robot pairs (i, j) does not fall below the common disc diameter. An example NLP solution is shown in Fig. 4. Again, the fleet configuration sequence has been computed by forward simulation using the obtained controls.

IV. Experiments and Results

The approach was implemented in Matlab and tested on a PC running Ubuntu 16.04 equipped with an 8-thread Intel Core i7-6820HQ CPU @ 2.70GHz and a Quadro M1000 GPU with 4GB RAM. The formulation described in Section III was implemented using the symbolic framework for algorithmic differentiation CasADi [8]. To solve the MIQP, we used IBM ILOG CPLEX solver, while the NLP solver used was IPOPT [9] with the linear solver MA27 [10]. Both solvers were given a 60-second timeout.

A. Scalability

We conducted test runs on variants of the scenario depicted in Fig. 4, with a varying number of horizontal and vertical polygons. Across all experiments we used constant actuator bounds for the absolute value of all elements of $\boldsymbol{u}[k]$ in the MIQP (1m/s^2) and the speed of robot r in the NLP ($|\dot{v}_r| \le 1$m/s^2). The curvature bound of $|\kappa_r| \le \tan(\phi_{\max})/L$, where ϕ is the steering angle, was varied across experiments.

We start by investigating which elements of the problem are computationally the most costly. In all boxplots (Fig. 5, 6 and 8), the vertical axis is computation time, each box represents a set of 50 experiments, the bottom line indicates

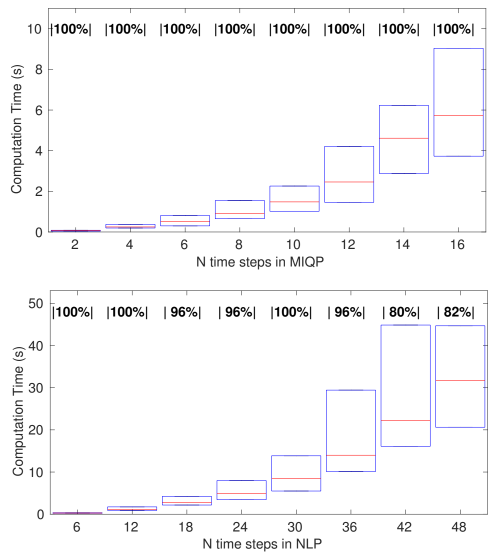

Fig. 5. Computation time vs. number of time steps in the MIQP and NLP. The solutions computed with a given number of time steps N_{MIQP} in the MIQP were provided as input to the NLP solver with $N_{\text{NLP}} = 3N_{\text{MIQP}}$ time steps. These experiments were conducted with the following settings: $P = 3$, $R = 3$, $C = 2$, $W = 0.5$m, $L = 1$m, $\phi_{\max} = 50°$. The obstacle free space is similar to the one depicted in figure 4, although with only two vertical (4×12m^2) polygons and one horizontal polygon (4×14m^2).

the 25^{th} percentile, the top line the 75^{th} percentile. The median is represented by the red line. The percentage of solutions found is also shown in the top of the box plots. We first investigate how computation time and solution success is affected by the number of time steps N_{MIQP} and N_{NLP} considered in the MIQP and NLP phases, respectively. The results are summarized in Fig. 5.

We also conduced experiments to assess how the approach scales with respect to the number P of polygons. We found no relation between P and computation time for $P = 2, \dots, 6$. Additionally, we conducted experiments to observe the effect of $\phi_{\max} \in \{10°, 20°, 30°, 40°, 50°, 60°, 70°, 80°\}$ on solution success rate. Also here, no relation was found, except for the case in which $\phi_{\max} \le 20°$, in which the amount of solutions found is lower because robots are essentially limited to moving straight ahead.

Next, we experimented with enabling/disregarding the robot-robot collisions in the MIQP via the constraints in (16). Note that their inclusion in the MIQP formulation is intended as a means to avoid allocations of boolean variables resulting in unavoidable robot-robot collisions in the subsequent NLP phase. Omitting these constraints in the MIQP does not affect the correctness of the solutions computed in phase two, as the absence of collisions between robots is also enforced in the NLP formulation. The computation time and percentage of solved problems after the second phase is shown in Fig. 6, with and without the robot-robot collision constraints in the MIQP formulation. Interestingly, the amount of solutions found is lower and the total computation time is higher when robot-robot collisions are considered in the MIQP. This can be explained by observing that the MIQP is a relaxation of

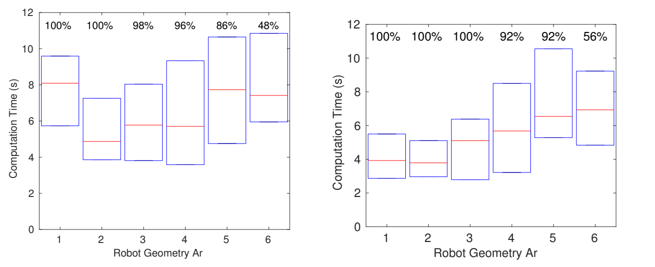

Fig. 6. Effect of considering robot-robot collision constraints in the MIQP (left graph) or not (right graph) on the computation time and the percentage of solutions found after phase two, while increasing the dimensions of the robot geometries. Robot geometry 1 is $W = 0.6(m)\, L = 1.2(m)$, robot geometry 2 is $W = 0.8(m)\, L = 1.6(m)$, robot geometry 3 is $W = 1(m)\, L = 2(m)$, robot geometry 4 is $W = 1.2(m)\, L = 2.4(m)$, robot geometry 5 is $W = 1.4(m)\, L = 2.8(m)$, robot geometry 6 is $W = 1.6(m)\, L = 3.2(m)$. These experiments were conducted with the following settings: $P = 2$, $R = 4$, $N_{\text{MIQP}} = 6$, $N_{\text{NLP}} = 24$, $\phi_{\max} = 60°$. The two polygons are rectangles crossing each other with dimensions $4 \times 10\text{m}^2$.

the fleet management problem, where the constraints in (16) effectively approximate the robot geometry to a square encapsulating $\mathcal{A}_r$. This overly-conservative approximation of the robot's geometry may result in a robot being less able to traverse narrow passages.

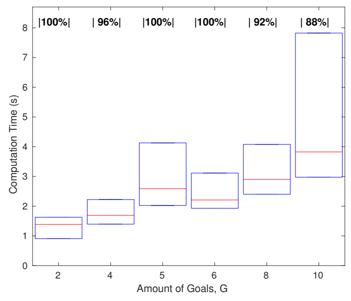

Fig. 7. Computation time vs. number of goals. These experiments were conducted with the following settings: $P = 4$, $R = 5$, $C = 2$, $W = 0.5$m, $L = 1$m, $\phi_{\max} = 50°$. The obstacle free space is the one depicted in figure 4

Furthermore, we conducted a set of experiments to evaluate the effect of the number of goals on the computation time, while the number of robots remains constant ($R = 5$). As shown in Fig. 7, the computation time grows as the number of goals is increased. However, this is not the case when the number of robots and goals are equal ($R = 5$ and $G = 5$). In general, having more goals than robots leads to a less constrained fleet management problem. Nevertheless, this "freedom" makes the robot-goal allocation problem harder as it requires more decisions to be made. The balance between these two factors can then explain the outlier (i.e., $R = 5$ and $G = 5$). When $G = 6$, the effect of having a harder robot-goal decision problem (deciding about which five of the six goals are going to be fulfilled) is lower than having a less constrained fleet management problem. On the other hand, when $G < R$ finding the control input variables for the non allocated robots is simpler since the best way to reduce the control energy is to remain in the initial configuration. Note that, for the case $R \neq G$ the constraints (11) and (12) have to be designed accordingly.

Finally, we assess how the approach scaled with respect to number of robots. Fig. 8 plots the total computation time of both phases versus the number of robots. As shown, computation time is highly affected by fleet size, and the majority of problems could be solved within the timeout for fleets of up to nine robots. Once again, the results were better both in terms of computation time and percentage of solutions found when robot-robot collisions were not considered in the MIQP.

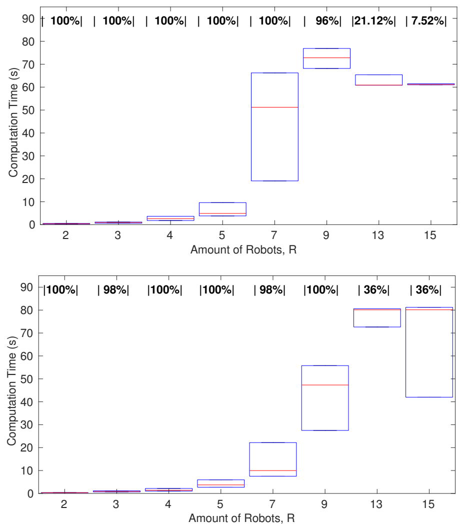

Fig. 8. Computation time vs. number of robots in the fleet. In the top plot, robot-robot collisions are considered in the MIQP, while they are not in the bottom plot. The experiments settings are the following: $P = 4$, $N_{\text{MIQP}} = 3$, $N_{\text{NLP}} = 9$, $W = 0.25$m, $L = 0.5$m, $\phi_{\max} = 50°$. The obstacle free space is the same as in 4, with vertical polygons with dimensions $4 \times 10\text{m}^2$ and horizontal polygon with dimension $4 \times 20\text{m}^2$.

B. Case Study

Next, we look at a case study which shows interesting features of the developed approach. In the problem depicted

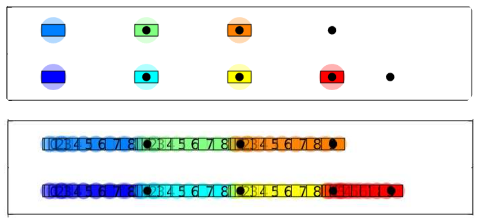

Fig. 9. Fleet management problem with initial configuration in the top and respective solution after the NLP in the bottom. The obstacle free space is composed of one polygon with dimensions $4 \times 10\text{m}^2$

in Fig. 9, the motions of the robots are a direct consequence of the OCP formulation of the fleet management problem. The problem requires all goals (black dots in the figure) to be allocated to a robot, while all but two of the robots are already at goal destinations in the initial condition. The computed solution requires all robots to move towards the right, thus achieving complete coverage of the goals. This solution shows the complex interplay between constraints on boolean and real-valued variables, and the fact that these are adhered to while an overall objective function is being minimized. The incentive for all robots to shift right derives from the need to minimize the overall fleet cost; this contrasts to greedily maximizing the private reward of individual robots.

V. RELATED WORK

In the multirobot motion planning problem [11], the objective is to find a path for each robot leading it from start to final configuration without colliding with other robots and obstacles. The configuration space within which search occurs is the Cartesian product of each robot's configuration space. Hence, each additional robot that is added to the fleet increases the size of the search space exponentially. Efficient graph-based search methods have been studied for multirobot motion planning, some relying on pre-computed motion primitives [12], others on the construction of obstacle-free configuration spaces [13]. None of these methods address the goal assignment problem, i.e., goal assignments are assumed given. The combined Target Assignment and Path Finding (TAPF) problem, where robots are not assigned to targets *a priori*, has received some attention [14]. Wagner et al. [15] propose a method in which the configuration spaces of robots are considered jointly only when a conflict cannot be solved. This effectively keeps the increase in search space size (due to the presence of many robots) to the minimum necessary; however, if there are no conflicts, the method will greedily minimize the path costs of single robots, thus failing to account for possible synergies between them, and potentially levitating the overall cost across the fleet. Also, these approaches have been shown to scale to large fleets only under the assumption of holonomic robot models. As a result, post-processing methods are required for enforcing adherence to realistic kinematic models [16].

Another possible strategy to solve the fleet management problem stated in Section II is to exploit the existence of efficient motion planning and coordination solvers, and solve the respective sub-problems separately while defining the necessary dependencies between them [17]. For example, the multi-robot motion planning problem has been decomposed into a motion planning problem (solved for each individual robot) and the subsequent problem of adaptating the temporal profiles of robot trajectories to avoid collisions. Conflict-based scheduling techniques have been used for this purpose [18], [19]. These approaches amount to decoupling the overall problem into spatial and temporal problems which are solved in sequence. As a result, these approaches may require re-planning of robot motions even in trivial cases, e.g., when robots need to exchange positions in a narrow corridor. Also, these approaches do not account for goal allocation, which is assumed given (albeit not necessarily known in advance [20]).

Advancements in numerical optimization have led to a surge in optimization-based motion planning approaches in recent years. Prominent examples include CHOMP [21] and derived approaches [22]. These methods optimize a performance criterion which incorporates penalties for obstacle avoidance and smoothness, but cannot handle explicit constraints such as robot dynamics. Schulman et al. [23] developed a sequential quadratic programming approach tailored to solving general motion planning problems formulated as NLPs. Nevertheless, as the previously discussed methods, their approach does not include boolean decision variables. The idea of using mixed integer programming to address discrete decision making in multi-robot path planning has been addressed by Schouwenaars et al. [24]. However, they rely on robot models being holonomic, and do not address goal assignment. A recent efficient approach [5] solves a mixed integer planning problem for a single non-holonomic UAV in a cluttered environment, where space is discretized into obstacle-free polygons. Furthermore, Augugliaro et al. [25] suggested a method considering hon-holonomic robot models for multiple UAVs in obstacle-free environments. Similar in spirit to our work, Kuindersma et al. [26] recently proposed to combine mixed integer programming for footstep planning with solving a subsequent NLP for generating trajectories for a single humanoid robot.

VI. CONCLUSIONS

We have proposed a general formulation of the fleet management problem, that is, the problem of assigning goals and computing mutually-consistent trajectories for a fleet of robots. Our formulation underscores two points: (1) automating fleets requires solving several sub-problems, which include goal assignment, motion planning, and coordination; (2) these sub-problems are tightly connected, hence requiring a holistic approach that goes beyond current state-of-the-art methods in motion planning and coordination.

We have also provided a concrete instantiation of the fleet management problem for fleets of non-holonomic robots in environments with obstacles. The proposed solution procedure relies on two phases: the first phase solves for the boolean decision variables under relaxed constraints via MIQP; solutions of phase one are then used by a NLP solver to compute jointly-feasible trajectories for the fleet. Through several examples, we have shown how complex fleet behavior occurs as a direct consequence of the interplay between constraints in the problem formulations of the two phases.

In our approach, the boolean variables considered in the first phase pertain to goal allocation and transitions through a polyhedral partitioning of obstacle-free space. It is worth noting that, as suggested by the general formulation of the fleet management problem, these may also include other aspects of fleet behavior we may wish to predicate upon, e.g., precedences among robot missions, or preferences on goal assignments. These aspects are often meaningful in real-world applications, and considering them jointly with the motion- and coordination-related sub-problems of fleet management can be very important.

In the future, we intend to transcribe the NLP with a collocation method which provides a better balance of time step discretization and solution quality for many problems. We also plan to study how this approach can be exploited for online fleet management. Finally, we intend to integrate the method into a fleet management framework with real robots, in which the computed trajectories are executed with a suitable tracking controller.

ACKNOWLEDGMENT

This work is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732737 (project ILIAD), by Vinnova project iQMobility, and by the Swedish Foundation for Strategic Research (SSF)

REFERENCES

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, p. 9, 2008.

[2] J. Larsson, "Unmanned operation of load-haul-dump vehicles in mining environments," Ph.D. dissertation, Örebro University, School of Science and Technology, 2011.

[3] H. Andreasson, A. Bouguerra, M. Cirillo, D. N. Dimitrov, D. Driankov, L. Karlsson, A. J. Lilienthal, F. Pecora, J. P. Saarinen, A. Sherikov, *et al.*, "Autonomous transport vehicles: where we are and what is missing," *IEEE Rob. Autom. Mag. (RAM)*, vol. 22, no. 1, pp. 64–75, 2015.

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

[5] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *Proc. IEEE Int. Conf. Robotics and Autom. (ICRA)*. IEEE, 2015, pp. 42–49.

[6] ——, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 109–124.

[7] FICO, "Mip formulations and linearizations," *Fair Isaac Corporation*, 2009.

[8] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, Arenberg Doctoral School, KU Leuven, 2013.

[9] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[10] "Hsl. a collection of fortran codes for large scale scientific computation." [Online]. Available: http://www.hsl.rl.ac.uk/

[11] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[12] M. Cirillo, T. Uras, and S. Koenig, "A lattice-based approach to multi-robot motion planning for non-holonomic vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2014, pp. 232–239.

[13] H. Ma, S. Koenig, N. Ayanian, L. Cohen, W. Hönig, T. Kumar, T. Uras, H. Xu, C. Tovey, and G. Sharon, "Overview: Generalizations of multi-agent path finding to real-world scenarios," *arXiv preprint arXiv:1702.05515*, 2017.

[14] H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," in *Proceedings of the International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1144–1152.

[15] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2011, pp. 3260–3267.

[16] W. Hönig, T. S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints." in *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2016, pp. 477–485.

[17] A. W. ter Mors, "Conflict-free route planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2011, pp. 2166–2171.

[18] M. Cirillo, F. Pecora, H. Andreasson, T. Uras, and S. Koenig, "Integrated motion planning and coordination for industrial vehicles." in *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2014.

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

[20] 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.

[21] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *Int. J. Rob. Res. (IJRR)*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[22] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2016, pp. 5332–5339.

[23] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Rob. Res. (IJRR)*, vol. 33, no. 9, pp. 1251–1270, 2014.

[24] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. European Control Conf. (ECC)*. IEEE, 2001, pp. 2603–2608.

[25] 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.

[26] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Rob. (AURO)*, vol. 40, no. 3, pp. 429–455, 2016.