Combining Multi-Robot Motion Planning and Goal Allocation using Roadmaps

João Salvado, Masoumeh Mansouri, Federico Pecora

Abstract— **This paper addresses the problem of automating fleets of robots with non-holonomic dynamics. Previously studied methods either specialize in facets of this problem, that is, one or a combination of multi-robot goal allocation, motion planning, and coordination, and typically acrifice optimality and completeness for scalability. We propose an approach that constructs an abstract multi-robot roadmap in a reduced configuration space, where we account for environment connectivity and interference cost between robots occupying the same polygons. Querying the road-map results in a robot-goal assignment and abstract multi-robot trajectory. This is then exploited to de-compose the original problem into smaller problems, each of which is solved with a multi-robot motion planner that accounts for kinodynamic constraints. We validate the approach experimentally to demonstrate the advantage of considering task assignment and motion planning holistically, and explore some methods for balancing solution quality and computational efficiency.**

I. Introduction

Planning for multi-robot systems subsumes four problems: task allocation, which is the problem of determining how tasks should be assigned to robots in order to maximize fleet performance; coordination, which is to decide how robots should avoid each other while traversing the shared environment; motion planning, i.e., determining the trajectories robots take in completing tasks; and the problem of determining control inputs that bring about the realization of these trajectories under given constraints. These problems overlap in non-trivial ways. Loosely-coupled integrated approaches trade completeness for efficiency, and no such approach has thus far been developed that tackles all four aspects of the overall problem; trajectory optimization based methods employ constrained optimization formulations of the overall problem, which typically leads to near-optimal solutions, but at the cost of impractical computational overhead; graph-based methods leverage abstractions of the environment, robots and tasks to achieve scalability, but these abstractions are typically unrealistic.

In this paper, we propose a method which exploits key insights from all three types of approaches. A graph-based abstraction, which we call *multi-robot roadmap*, is used to determine how robots should traverse a partition of navigable space into convex polygons. Heuristic search is used to simultaneously determine task assignments and polygons to be traversed, while accounting for both individual robot performance and the cost of interference among robots (the balance of which can be regulated with a single parameter). Search results in an abstract multi-robot trajectory, which is then used to decompose the multi-robot motion planning problem into a partial order of smaller trajectory optimization problem instances. The solutions of these are then combined to obtain a kinodynamically feasible multi-robot trajectory.

We show empirically that our approach scales to tens of robots, and that considering task assignment and motion planning holistically leads to high quality solutions according to common metrics for measuring the performance of multi-robot systems. As for other roadmap-based approaches, our method allows some of the computation to be performed offline, i.e., without knowing the initial and final conditions of the robot fleet. We show empirically that the amount of computation that is done offline can be regulated by changing the granularity of the roadmap. We also show that a more fine-grained roadmap leads to better problem decomposition, which in turn improves both solution quality and query times. Finally, we demonstrate how increasing the weight of interference cost in the heuristic search leads to more efficient problem decompositions and improves the makespan of the fleet's trajectories.

II. Related Work

Considering task allocation, motion planning, coordination and control jointly can lead to highly-optimized solutions. However, this incurs significant computational overhead. In industrial practice, the problem is simplified by re-engineering the environment and/or the robots [1]. Others sacrifice completeness and optimality for scalability, e.g., by combining single-robot motion planning and coordination. Examples of such approaches include prioritized planning [2], [3], and coordination via velocity profile adjustment [4], [5]. None of these account for task allocation.

Task allocation has often been considered separately from the other problems [6]. For instance, Nam and Shell [7] formulate the optimal task assignment problem as a constrained optimization problem where interference among robots is considered as a cost. This and other similar approaches [8] rely on loosely-coupled integrations with other solvers, e.g., for coordination and motion planning [9].

Approaches based on trajectory optimization deal with the complexity of multi-robot planning by relaxing/convexifying certain constraints to exploit efficient sequential convex programming solutions [10]. These methods have been applied for holonomic robots [11], non-holonomic robots in obstacle-free environments [12], and for robots with non-convex

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

dynamics [13]. Roadmaps have also been utilized to warm-start non-linear solvers [14]. Again, though, none of these approaches spans the gamut of all four aspects of the problem without imposing restrictive assumptions.

Graph-based methods are known to scale to large fleets of robots, however, these make further unrealistc assumptions. Some exploit efficient network flow algorithms for solving multi-robot goal allocation and motion planning jointly. However, robots are required to move in "grid-like" environments represented as a graph, where edges/paths are non-colliding and vertices/states are occupied by at most one robot at a time [15]. Conflicts can be solved by imposing constraints over the paths of the robots (e.g., computed via a binary search tree [16]), or by constructing the resulting configuration free space [17], possibly also allocating tasks to heterogeneous robots [18]. Yet other approaches consider the joint configuration space of multiple robots only when conflicts cannot be solved locally [19]. Another class of graph-based methods considers disc-shaped robots as pebbles moving on a graph [20], [21] assuming robots do not overlap on start and goal configurations [22] (an assumption which can be relaxed by generating multiple pebble graphs [23]).

Some graph-based methods handle complexity by dividing the overall problem into multiple levels of abstraction. Honig et al. [24] use realistic dynamic models to post-process solutions derived from a high-level graph search. Overmars et al. [25] generate roadmap abstractions by partitioning a single-robot roadmap into regions, though this approach does not consider interference between robots in the same region. A probabilistically complete roadmap-based approach for multi-robot motion planning is proposed by Le and Plaku [26], in which sampling of the combined multi-robot configuration space is guided by a search in a discrete space with holonomic point robots. Similarly, Du et al. [27] exploit a pre-computed roadmap for single robots as a source of states to be found closer to the newly sampled state in the multi-robot configuration space. None of these approaches consider the task allocation problem.

III. Problem Statement

A. Nomenclature

In this article, we refer to tuples, sets and vectors as $\langle \dots \rangle$, $\{\dots\}$, and $[\dots]$, respectively. Non-bold uppercase letters are used for constants. Bold letters denote matrices (uppercase) and vectors (lowercase), and the notation $\boldsymbol{x}[i]$ indicates the i-th element of vector $\boldsymbol{x}$. To simplify exposition, we will consistently use $r \in \{1, \dots, R\}$ to index robots, $p \in \{1, \dots, P\}$ for polygons, $g \in \{1, \dots, G\}$ for goals, and $k \in \{1, \dots, N\}$ for discrete time. Continuous time is normalized w.r.t. the planning horizon, i.e., $t \in [0, 1]$, and discrete time k corresponds to continuous time $\frac{k}{N}$.

B. Preliminary Definitions

a) Robots and World: Consider a bounded 2D-world setting $\mathcal{W} \subset \mathbb{R}^2$, together with an obstacle region $\mathcal{O} \subset \mathcal{W}$. A robot r has a geometry $\mathcal{A}_r(\boldsymbol{q}_r)$ in configuration $\boldsymbol{q}_r \in \mathcal{C}_r$, where $\mathcal{C}_r$ is the configuration space of r. The robot's configuration-free space is $\mathcal{C}_r^{\text{free}} = \{\boldsymbol{q}_r \in \mathcal{C}_r \mid \mathcal{A}_r(\boldsymbol{q}_r) \cap \mathcal{O} = \emptyset\}$, that is, the region of the world where r's geometry intersects no obstacle. Each robot has state $\boldsymbol{x}_r = [\boldsymbol{q}_r \, \dot{\boldsymbol{q}}_r]$, a control input $\boldsymbol{u}_r$, and a dynamic model $\dot{\boldsymbol{x}}_r = f_r(\boldsymbol{x}_r(t), \boldsymbol{u}_r(t))$. A single robot is therefore fully defined by $\mathcal{R}_r = \langle \mathcal{A}_r, \boldsymbol{x}_r, \boldsymbol{u}_r, f_r \rangle$. The overall state and controls of a fleet of R robots are $\boldsymbol{x} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_R]$ and $\boldsymbol{u} = [\boldsymbol{u}_1 \dots \boldsymbol{u}_R]$; similarly, the overall configuration space and geometry of the fleet are, respectively, $\mathcal{C}^{\text{free}} = \prod_{r=1}^{R} \mathcal{C}_r^{\text{free}}$ and $\mathcal{A} = \cup_{r=1}^{R} A_r$.

b) Trajectory: A single robot trajectory between a start configuration $\boldsymbol{s}_r \in \mathcal{C}_r^{\text{free}}$ and goal configuration $\boldsymbol{g}_r \in \mathcal{C}_r^{\text{free}}$ is a continuous mapping $\tau_r : [0, 1] \mapsto \mathcal{C}_r^{\text{free}}$, where $\tau_r(0) = \boldsymbol{s}_r$ and $\tau_r(1) = \boldsymbol{g}_r$. We say that a trajectory is *free* when $\tau_r(t) \in \mathcal{C}_r^{\text{free}}$ for all t. Note that, for a robot to navigate along the trajectory $\tau_r(t)$ while adhering to kinodynamic constraints, a *kinematically feasible control policy* $\pi_r(f_r, \tau_r) = \boldsymbol{u}_r(\boldsymbol{x}_r(t), t)$ must be followed. We say that a *trajectory is feasible* if such a policy exists and the trajectory τ_r is free.

c) Obstacle Free Space: Let the obstacle free space be the set $\mathcal{W}^{\text{free}} = \mathcal{W} \backslash \mathcal{O}$. This can be approximated with a set $\{\mathcal{P}_1, \dots, \mathcal{P}_P\}$ of adjacent non-overlapping convex polygons. We say that polygons are *adjacent* if they intersect on their frontier as, e.g., in a Voronoi tessellation [28] with convex cells. Given such a set of adjacent convex polygons, one can construct the *connectivity graph* of the obstacle free space $G_{\mathcal{W}^{\text{free}}} = \langle V_{\mathcal{W}^{\text{free}}}, E_{\mathcal{W}^{\text{free}}} \rangle$, where the set of vertices $V_{\mathcal{W}^{\text{free}}} = \{\mathcal{P}_1, \dots, \mathcal{P}_P\}$ contains one vertex per polygon and $E_{\mathcal{W}^{\text{free}}} = \{(\mathcal{P}_i, \mathcal{P}_j) \in V_{\mathcal{W}^{\text{free}}}^2 \mid \exists \boldsymbol{q}_r \in \mathcal{C}_r^{\text{free}} \text{ s.t. } \boldsymbol{q}_r \subset \partial\mathcal{P}_i \cap \partial\mathcal{P}_j\}$, having $\partial\mathcal{P}$ as the boundary of a polygon, i.e., an edge exist if one can "place" a robot on the intersection of the two polygons. Figure 1(a) shows an example where the connectivity graph has set of four vertices $V_{\mathcal{W}^{\text{free}}} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4\}$ and edges $E_{\mathcal{W}^{\text{free}}} = \{(\mathcal{P}_1, \mathcal{P}_2), (\mathcal{P}_1, \mathcal{P}_4), (\mathcal{P}_2, \mathcal{P}_3), (\mathcal{P}_3, \mathcal{P}_4)\}$.

d) Robot Goal Allocation: Let $\mathcal{S} = \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_R\}$ and $\mathcal{G} = \{\boldsymbol{g}_1, \dots, \boldsymbol{g}_G\}$ be, respectively, a set of start and goal configurations for a fleet of R robots. Moreover, let $\sigma : \mathcal{S} \mapsto \mathcal{G}$, be a mapping from robot starting poses to goals. Hence, finding an allocation of robots to goals means computing σ.

C. Problem Formulation

We define a *Multi-robot Motion Planning and Goal Allocation Problem (MMP-GA)* as a tuple $\mathcal{M} = \langle \mathcal{W}^{\text{free}}, \mathcal{R}, \mathcal{S}, \mathcal{G} \rangle$. The task is to find function σ and a joint fleet trajectory $\tau(t) = [\tau_1(t) \; \dots \; \tau_R(t)]$ such that:

1) $\tau_r(0) = s_r \; \forall r \in \{1, \dots, R\}$ (start configuration)
2) $\tau_r(1) = \sigma(s_r) \; \forall r \in \{1, \dots, R\}$ (goal configuration)
3) $\tau_r(t) \in \mathcal{C}_r^{\text{free}}$ and $\exists \pi_r(f_r, \tau_r) = \boldsymbol{u}_r(\boldsymbol{x}_r, t) \; \forall r \in \{1, \dots, R\}, \forall t \in [0, 1]$ (all trajectories are feasible),
4) $\mathcal{A}_i(\tau_i(t)) \cap \mathcal{A}_j(\tau_j(t)) = \emptyset \; \forall t \in [0, 1], \forall i, j \neq i \in \{1, \dots, R\}$ (robots do not collide with each other),

while maximizing some performance criterion $L(t, \boldsymbol{x}, \boldsymbol{u})$ (we omit the subscript r when referring to the entire fleet).

IV. Approach

This paper solves a specific instance of the MMP-GA problem, where robots are homogeneous, meaning that they

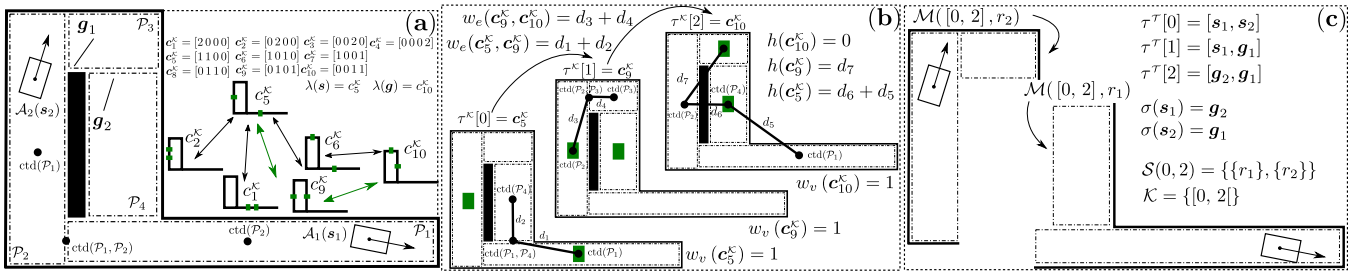


Fig. 1: Running example of a MMP-GA problem with two robots and two goals: (a) shows a partial representation of the graph $G^{\mathcal{K}}$ and trajectory $\tau^{\mathcal{K}}$ connecting abstract start and goal configurations; (b) extends the representation of the abstract trajectory $\tau^{\mathcal{K}}$, illustrating how step-cost and heuristic are computed; (c) demonstrates how this problem is de-composed.

have the same geometry, control and state space, and kinodynamics. The configuration space $\mathcal{C}_r$ of each robot is $SE(2) = \mathbb{R}^2\mathbb{S}^1$ (2D rigid bodies). Hence, the configuration of r can be defined by Euclidean coordinates and orientation, $\boldsymbol{q}_r = [x_r\, y_r\, \theta_r]$. Additionally, we assume that robots have car-like non-holonomic kinodynamics. For example,

$$\dot{x}_r = v_r \cos(\theta_r),\ \dot{y}_r = v_r \sin(\theta_r),\ \dot{\theta}_r = k_r v_r,\ \dot{v}_r = a_r \quad (1)$$

$$f_r(\boldsymbol{x}_r, \boldsymbol{u}_r) = \dot{\boldsymbol{x}}_r = [\dot{x}_r\, \dot{y}_r\, \dot{\theta}_r\, \dot{v}_r],\ \boldsymbol{u}_r = [k_r\, a_r], \quad (2)$$

where k_r is the curvature, v_r is the speed, and a_r the acceleration. In summary, the input of the MMP-GA is: (1) a partition of $\mathcal{W}^{\text{free}}$ into adjacent polygons $\{\mathcal{P}_1, \ldots, \mathcal{P}_P\}$; (2) a set of homogeneous robots $\{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$; (3) start configurations $[\boldsymbol{s}_1 \ \ldots \ \boldsymbol{s}_R] \in SE(2)^R$; (4) goal configurations $[\boldsymbol{g}_1 \ \ldots \ \boldsymbol{g}_G] \in SE(2)^R$; (5) a cost function $L(\cdot)$. A solution to this problem is an assignment $\sigma(\cdot)$ of robots to goals, and a fleet trajectory $\tau(t) = [\tau_1(t) \ \ldots \ \tau_R(t)]$.

Our method can be briefly described as follows. (A) First, we build a discrete abstraction of the MMP-GA problem, where the robots in the fleet are considered jointly in a reduced configuration space, $\mathcal{C}^{\mathcal{K}}$. Each configuration in this abstract space represents the number of robots per polygon in $\mathcal{W}^{\text{free}}$. Kinodynamics are not considered and collisions between robots are considered as a cost. (B) The previously constructed abstraction is queried in order to find an abstract trajectory $\tau^{\mathcal{K}}$. This trajectory is a sequence of abstract configurations in $\mathcal{C}^{\mathcal{K}}$, where the number of robots in each polygon changes in a discrete manner. At this level of abstraction, it is not possible to know which particular robot is entering or exiting a polygon, thus robots are *anonymous*. (C) An augmented trajectory $\tau^{\mathcal{T}}$ in continuous configuration space is obtained by associating each configuration in $\tau^{\mathcal{K}}$ to a representative continuous configuration in $\mathcal{C}$. This is further augmented by connecting the actual start configurations of all robots, and exploring the possible de-anonymizations of robots that reach the desired goal configurations along $\tau^{\mathcal{T}}$. (D) We exploit both $\tau^{\mathcal{K}}$ and $\tau^{\mathcal{T}}$ to de-compose the problem in time and space, based on the observation that robots navigating through different polygons do not require coordination. At this point, the overall MMP-GA problem is decomposed into a partial order of simpler MMP-GA instances, one for each subset of robots for which joint motion planning is necessary. (E) Finally, a multi-robot motion planner is employed to obtain a feasible trajectory for each simpler MMP-GA problem instance. The composite configuration space of subsets of robots and car-like kinodynamics are considered in each MMP-GA instance, and the combination of the resulting trajectories in $\mathcal{C}$ yields the overall fleet trajectory τ.

We call the abstract space constructed in phase (A) a *multi-robot roadmap*. As for other roadmap-based approaches to motion planning [29], [30], the idea is to allow some of the computation to be performed offline, i.e., without knowing the initial and final conditions (respectively, $\mathcal{S}$ and $(\sigma, \mathcal{G})$ in our approach). As a result, we can see phase (A) as the roadmap-building phase; phases (B–E) constitute the online query phase, in which the state of the actual fleet is connected to the roadmap and task assignment is performed (B–C), and kinematically feasible trajectories are computed for the robots in the fleet (D–E). A key intuition in our approach is that the roadmap also helps identify which subsets of robots need to be involved in multi-robot motion planning, and when. Next, we describe each stage in detail.

A. Multi-Robot Roadmap: a Discrete Abstraction

At the highest level of abstraction, the MMP-GA is seen as the problem of finding a sequence of polygon transitions from the fleet's starting set of polygons to the set of goal polygons. We define an abstract fleet configuration space $\mathcal{C}^{\mathcal{K}} = \{0, \ldots, R\}^P$ representing the number of robots present in each polygon. Let $\lambda : \mathcal{C} \mapsto \mathcal{C}^{\mathcal{K}}$ define the mapping of fleet configurations to the abstract space, that is, $\lambda(\boldsymbol{q}) = \boldsymbol{c}^{\mathcal{K}}$ represents the polygon occupancy by a fleet in configuration $\boldsymbol{q}$. An example of this can be found in figure 1(a), where $\lambda(\boldsymbol{s}) = \boldsymbol{c}^{\mathcal{K}}_5 = [1\,1\,0\,0]$, meaning the starting configuration has a robot in $\mathcal{P}_1$ and another robot in $\mathcal{P}_2$.

We construct offline a complete and weighted multi-robot road-map $G^{\mathcal{K}} = \langle V^{\mathcal{K}}, E^{\mathcal{K}} \rangle$ as follows:

- $V^{\mathcal{K}} = \{\boldsymbol{c}^{\mathcal{K}} \mid \exists \boldsymbol{q} \in \mathcal{C}^{\text{free}} \text{ s.t. } \lambda(\boldsymbol{q}) = \boldsymbol{c}^{\mathcal{K}}\}$, i.e., a vertex exists for each assignment of all robots to polygons such that their geometries are contained within the polygons;
- $(\boldsymbol{c}^{\mathcal{K}}_i, \boldsymbol{c}^{\mathcal{K}}_j) \in E^{\mathcal{K}}$ iff exactly one robot moves from one polygon to an adjacent polygon, that is,
  - $\left|\left|\boldsymbol{c}^{\mathcal{K}}_i - \boldsymbol{c}^{\mathcal{K}}_j\right|\right|_1 = 2$, and
  - the polygons $\underset{p \in \{1,\ldots,P\}}{\arg\max} \left(\boldsymbol{c}^{\mathcal{K}}_j - \boldsymbol{c}^{\mathcal{K}}_i\right)[p]$ and $\underset{p \in \{1,\ldots,P\}}{\arg\min} \left(\boldsymbol{c}^{\mathcal{K}}_j - \boldsymbol{c}^{\mathcal{K}}_i\right)[p]$ are adjacent;

- The weights of vertices and edges are given by functions $w_v : V^{\mathcal{K}} \mapsto \mathbb{R}$ and $w_e : E^{\mathcal{K}} \mapsto \mathbb{R}$.

In the following, given an edge $(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}}) \in E^{\mathcal{K}}$, we indicate with $\mathrm{out}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}}) = \underset{p \in \{1,\dots,P\}}{\arg\min} \left(\boldsymbol{c}_j^{\mathcal{K}} - \boldsymbol{c}_i^{\mathcal{K}}\right)[p]$ the polygon from which the robot transitions along the edge, and with $\mathrm{in}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}}) = \underset{p \in \{1,\dots,P\}}{\arg\max} \left(\boldsymbol{c}_j^{\mathcal{K}} - \boldsymbol{c}_i^{\mathcal{K}}\right)[p]$ the polygon to which the robot transitions. Hence, we can indicate the border (line or point) through which the robot transitions between the two polygons with $\mathcal{P}_{\mathrm{out}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})} \cap \mathcal{P}_{\mathrm{in}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})}$.

Given the original MMP-GA problem $\langle \mathcal{W}^{\mathrm{free}}, \mathcal{R}, \mathcal{S}, \mathcal{G} \rangle$, we can define a corresponding abstract problem where $\mathcal{W}^{\mathrm{free}} \approx G^{\mathcal{K}}$, $\mathcal{R}$ is represented as number of robots per polygon, $\mathcal{S} \approx \lambda(\boldsymbol{s})$, and $\mathcal{G} \approx \lambda(\boldsymbol{g})$ (see Figure 1(a)).

B. Multi-Robot Roadmap Query with Anonymous Robots

An A* search is conducted on graph $G^{\mathcal{K}}$ to compute a discrete path $\tau^{\mathcal{K}} : \{0, \dots, N\} \mapsto \mathcal{C}^{\mathcal{K}}$ connecting start configuration $\lambda(\boldsymbol{s}) \in \mathcal{C}^{\mathcal{K}}$ and goal configuration $\lambda(\boldsymbol{g}) \in \mathcal{C}^{\mathcal{K}}$. The step-cost function used in the search is

$$c\left(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}}\right) = \alpha\,(1-\gamma)\, w_v\left(\boldsymbol{c}_j^{\mathcal{K}}\right) + \gamma\, w_e\left(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}}\right), \quad (3)$$

where function α regularizes and factor $\gamma \in [0, 1]$ prioritizes edge cost w_e versus vertex cost w_v. To account for the geometry of the polygons in computing this cost (and the heuristic used by the A* search, as shown below), we compute the centroid $\mathrm{ctd}(\mathcal{P})$ of a polygon $\mathcal{P}$; similarly, let $\mathrm{ctd}(\mathcal{P}_i \cap \mathcal{P}_j)$ be the center point of the border of two connected polygons $\mathcal{P}_i$ and $\mathcal{P}_j$. We are interested in making the step cost function penalize solutions requiring complex fleet coordination or the traversal of highly constrained regions. This can be done by making w_v penalize configurations where many robots are in the same polygon, e.g., $w_v\left(\boldsymbol{c}^{\mathcal{K}}\right) = ||\boldsymbol{c}^{\mathcal{K}}||_\infty - 1$.

Euclidean distances between polygons are introduced via the edge cost. Since we consider holonomic point robots at this level of abstraction, the cost of an edge is the Euclidean distance between the centroids of the polygons between which the robot moves and the centroid of their border:

$$\begin{aligned} &w_e\left(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}}\right) = \quad (4)\\ &\left|\left|\mathrm{ctd}(\mathcal{P}_{\mathrm{out}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})}) - \mathrm{ctd}(\mathcal{P}_{\mathrm{in}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})} \cap \mathcal{P}_{\mathrm{out}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})})\right|\right|_2 + \\ &\left|\left|\mathrm{ctd}(\mathcal{P}_{\mathrm{in}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})}) - \mathrm{ctd}(\mathcal{P}_{\mathrm{in}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})} \cap \mathcal{P}_{\mathrm{out}(\boldsymbol{c}_i^{\mathcal{K}}, \boldsymbol{c}_j^{\mathcal{K}})})\right|\right|_2. \end{aligned}$$

We can also define the heuristic function in terms of Euclidean distances. However, as $G^{\mathcal{K}}$ is a multi-robot roadmap, we need to consider the transitions of multiple robots when estimating the distance to the goal configuration $\boldsymbol{c}_g^{\mathcal{K}} = \lambda(\boldsymbol{g})$.

Let $\zeta^-(\boldsymbol{c}^{\mathcal{K}}) = \{p \in \{1, \dots, P\} \mid \left(\boldsymbol{c}^{\mathcal{K}} - \boldsymbol{c}_g^{\mathcal{K}}\right)[p] < 0\}$ be the set of goal polygons that are missing robots in a given abstract configuration $\boldsymbol{c}^{\mathcal{K}}$. Dually, let $\zeta^+(\boldsymbol{c}^{\mathcal{K}}) = \{p \in \{1, \dots, P\} \mid \left(\boldsymbol{c}^{\mathcal{K}} - \boldsymbol{c}_g^{\mathcal{K}}\right)[p] > 0\}$ be the set of polygons with a robot surplus, i.e., more robots than goals. We can then define the heuristic estimator as

$$h(\boldsymbol{c}^{\mathcal{K}}) = \sum_{p^+ \in \zeta^+(\boldsymbol{c}^{\mathcal{K}})} \Big(\gamma\, \left(\boldsymbol{c}^{\mathcal{K}} - \boldsymbol{c}_g^{\mathcal{K}}\right)[p^+] \times \min_{p^- \in \zeta^-(\boldsymbol{c}^{\mathcal{K}})} \left(\left|\left|\mathrm{ctd}(\mathcal{P}_{p^-}) - \mathrm{ctd}(\mathcal{P}_{p^+})\right|\right|_2\right)\Big), \quad (5)$$

that is, the sum of distances between polygons with surplus robots and the closest goal polygon with missing robots, multiplied by that surplus.

The A* search yields[1] an abstract trajectory $\tau^{\mathcal{K}}$ where robots are *anonymous*: at each discrete transition $\tau^{\mathcal{K}}(k)$ to $\tau^{\mathcal{K}}(k+1)$ a single robot exits $\mathrm{out}(\cdot)$ and enters $\mathrm{in}(\cdot)$, but it is not decided which robot this is. Figure 1(b) details the trajectory $\tau^{\mathcal{K}}$ as a sequence of the abstract configurations $\boldsymbol{c}_5^{\mathcal{K}}, \boldsymbol{c}_9^{\mathcal{K}}, \boldsymbol{c}_{10}^{\mathcal{K}}$, that would be obtained in our running example.

C. De-Anonymization of Robots

In this phase, we de-anonymize robots along the abstract trajectory $\tau^{\mathcal{K}}$ obtained as a result of the query. To do this, we construct a weighted tree $\mathcal{T} = (V^{\mathcal{T}}, E^{\mathcal{T}})$ whose root is the fleet's starting configuration $\boldsymbol{s} \in \mathcal{C}$ (recall that $\lambda(\boldsymbol{s}) = \tau^{\mathcal{K}}(0)$). The children of a node $\boldsymbol{q}$ at depth k of the tree are all configurations $\boldsymbol{q'} \in \mathcal{C}$ such that $\lambda(\boldsymbol{q'}) = \tau^{\mathcal{K}}(k+1)$ and $\lambda(\boldsymbol{q}) = \tau^{\mathcal{K}}(k)$, and each robot r is either in a start configuration ($\boldsymbol{q'}[r] \in \mathcal{S}$), goal configuration ($\boldsymbol{q'}[r] \in \mathcal{G}$), or a polygon centroid ($\boldsymbol{q'}[r] = [\mathrm{ctd}(\mathcal{P}_p)\ 0]$ for some $\mathcal{P}_p \in \mathcal{W}^{\mathrm{free}}$). Expansion ceases at depth N, since $\lambda(\boldsymbol{g}) = \tau^{\mathcal{K}}(N)$.

Let edge $(\boldsymbol{q}, \boldsymbol{q'}) \in E^{\mathcal{T}}$, and let r_t be the robot that transitions from polygon $\mathcal{P}_{\mathrm{out}(\lambda(\boldsymbol{q}), \lambda(\boldsymbol{q'}))}$ to polygon $\mathcal{P}_{\mathrm{in}(\lambda(\boldsymbol{q}), \lambda(\boldsymbol{q'}))}$. Then, the edge weight is

$$\begin{aligned} &w_e^{\mathcal{T}}(\boldsymbol{q}, \boldsymbol{q'}) = \\ &\quad \left|\left|\boldsymbol{q}[r_t] - \mathrm{ctd}(\mathcal{P}_{\mathrm{in}(\lambda(\boldsymbol{q}), \lambda(\boldsymbol{q'}))} \cap \mathcal{P}_{\mathrm{out}(\lambda(\boldsymbol{q}), \lambda(\boldsymbol{q'}))})\right|\right|_2 + \\ &\quad \left|\left|\boldsymbol{q'}[r_t] - \mathrm{ctd}(\mathcal{P}_{\mathrm{in}(\lambda(\boldsymbol{q}), \lambda(\boldsymbol{q'}))} \cap \mathcal{P}_{\mathrm{out}(\lambda(\boldsymbol{q}), \lambda(\boldsymbol{q'}))})\right|\right|_2 + \\ &\quad \sum_{r \neq r_t} ||\boldsymbol{q'}[r] - \boldsymbol{q}[r]||_2, \end{aligned} \quad (6)$$

which is the distance as computed in w_e for robot r_t plus the distance of robots moving inside their initial polygon.

A path from root to leaf in $\mathcal{T}$ is a sequence of transitions for all robots in the fleet which leads all goals to be reached by a robot. This path is therefore a fleet trajectory $\tau^{\mathcal{T}} = \prod_{r=1}^{R} \tau_r^{\mathcal{T}}$. Each $\tau_r^{\mathcal{T}}$ is a trajectory for a single robot, starting in $\tau_r^{\mathcal{T}}(0) = \boldsymbol{s}_r$, followed by a sequence of polygon centroids for $k = \{1, \dots, N-1\}$, and ending in a goal configuration $\tau_r^{\mathcal{T}}(N)$. The overall fleet trajectory $\tau^{\mathcal{T}}$ implies a specific robot-goal assignment, that is, $\sigma(\boldsymbol{s}_r) = \tau_r^{\mathcal{T}}(N)$.

D. Decoupling the Problem in Time and Space

The fleet trajectory $\tau^{\mathcal{T}}$ tells us the sequence of polygons traversed by each robot. We exploit this information to decompose the original MMP-GA problem into independent sub-problems. We first partition the domain $\{0, \dots, N\}$ of $\tau^{\mathcal{T}}$ into intervals $[k_i, k_j[$ as follows:

[1] Since the heuristic is admissible (as it assumes that goals are reached by the robots that are closest, and that the lengths of the corresponding paths are straight line distances between polygon centroids), $\tau^{\mathcal{K}}$ is also optimal. However, this does not entail optimality of the final fleet trajectory τ.

- within an interval $[k_i, k_j[$, all robots must move unless they are already in a goal, i.e., $\tau_r^{\mathcal{T}}(k_i) = \tau_r^{\mathcal{T}}(k_j) \implies \tau_r^{\mathcal{T}}(k_j) = \sigma(\boldsymbol{s}_r)$.
- every pair of robots that are in the same polygon at k_i and k_j must not be co-located in the polygon centroid.

With the resulting intervals $\mathcal{K} = \{[k_1, k_2[, \ldots, [k_{m-1}, k_m[\}$ we can decompose the original MMP-GA problem into a sequence of MMP-GA problems. Each problem is defined as $\mathcal{M}(k_i, k_j, \mathcal{S}) = \langle \mathcal{W}^{\text{free}}_{k_i,k_j}, \prod_{r\in\mathcal{S}} \mathcal{R}_r, \prod_{r\in\mathcal{S}} \tau_r^{\mathcal{T}}(k_i), \prod_{r\in\mathcal{S}} \tau_r^{\mathcal{T}}(k_j)\rangle$, where $\mathcal{W}^{\text{free}}_{k_i,k_j} \subseteq \mathcal{W}^{\text{free}}$ is the union of polygons traversed in the interval $[k_i, k_j[$ by the subset of robot $\mathcal{S} \subseteq \{1, \ldots, R\}$.

We can further partition $\mathcal{M}(k_i, k_j, \mathcal{S})$ into sub-problems that involve subsets of robots traversing distinct polygons. Specifically, we construct a graph with a vertex for each robot and an edge for each pair of robots traversing the same polygon. The strongly connected components of this graph define a partition $\mathcal{S}(k_i, k_j)$ of set $\{1, \ldots, R\}$. Each element $\mathcal{S}$ of this partition defines the set of robots $\bigcup_{r\in\mathcal{S}} \mathcal{R}_r$ that need to be considered jointly in motion planning. Overall, we have therefore decomposed the original problem $\mathcal{M}$ into

$$\bigoplus_{[k_i,k_j[\in\mathcal{K}} \left(\bigotimes_{\mathcal{S}\in\mathcal{S}(k_i,k_j)} \mathcal{M}(k_i, k_j, \mathcal{S}) \right), \tag{7}$$

where the symbol $\oplus$ refers sequential problem resolution and $\otimes$ refers to parallel problem resolution.

Figure 1(c) shows an example where we decomposed the original MMP-GA problem into two MMP-GAs: $\mathcal{M}([0, 2], r_2)$ where robot r_2 traverses polygons $\{\mathcal{P}_2, \mathcal{P}_3\}$, and $\mathcal{M}([0, 2], r_1)$ where robot r_1 traverses polygons $\{\mathcal{P}_1, \mathcal{P}_4\}$, both with the same horizon $[0, 2[$.

E. Multi-robot Motion Planning

For a given (k_i, k_j), each $\mathcal{M}(k_i, k_j, \mathcal{S})$ can now be solved in parallel via some multi-robot motion planner which can compute a dynamically feasible trajectory τ for the multi-robot fleet. The planner will have as input a sequence of polygons, a start and goal configuration, kinodynamic constraints $\cup_{i\in\mathcal{S}} f_i$, continuous-time planning horizon $[\frac{k_i}{N}, \frac{k_j}{N}]$, and boundary conditions $\tau^{\mathcal{T}}(k_i)$ and $\tau^{\mathcal{T}}(k_j)$. As output, it will return a trajectory τ where non-holonomic robots traverse those polygons from start to goal configuration while not colliding with each other.

V. Experiments and Results

The following empirical evaluation aims to (1) assess the scalability of the approach in terms of number of robots and polygons, and (2) assess the quality of the obtained fleet trajectories in terms of overall fleet performance.

A. Setup

The approach is tested on a PC running Ubuntu 16.04 equipped with an 8-thread Intel Core i7-6820HQ CPU @ 2.70GHz. $\mathcal{W}^{\text{free}}$ was constructed manually via the Boost Polygon Library (BPL) [31] and $G_{\mathcal{W}^{\text{free}}}$ was computed via the connectivity extraction algorithm. Both $G^{\mathcal{K}}$ and $\mathcal{T}$ were generated via the Boost Graph Library (BGL) [32], and off-the-shelf A* and strongly connected component finding algorithms were utilized. The multi-robot motion planner used (Section IV-E) is the trajectory optimization based planner described in [13].

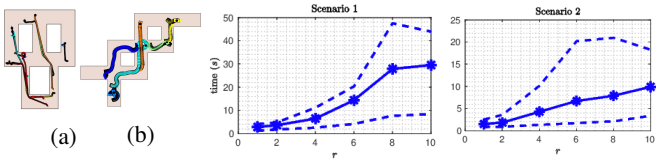


Fig. 2: An example solution is shown for scenario 1 (a) and scenario 2 (b). On the right, the computation time of the query phase (B–E) vs. number of robots for each scenario.

B. Scalability

R	1	2	4	6	8	10
$\lvert V_{\mathcal{K}}\rvert$	10	55	715	4985	23860	88135
$\lvert E_{\mathcal{K}}\rvert$	24	210	5280	47928	270540	1120136
time (s)	0.0002	0.0019	0.085	2.45	46	698

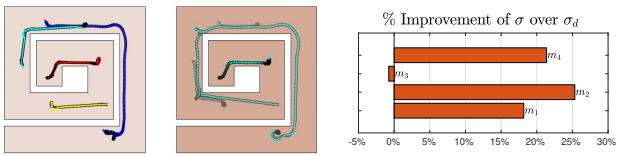
TABLE I: Roadmap building phase (Phase A) of scenario 1.

We test scalability using the scenarios depicted in Figure 2. Robots are non-holonomic with car-like kinodynamics, see eq. (1). Figure 2 shows the computation time of the query phase (B–E) vs. the number of robots in the scenario. Each asterisk represents the average solution computation time of a set of 100 runs. The top and bottom lines indicate, respectively, the 90- and 10-percentile. Table I summarizes how the size and time required to compute the roadmap scale with fleet size in scenario 1. Note that $|V_{\mathcal{K}}| \leq \binom{P+R}{P}$.

C. Solution Quality

To evaluate the quality of the computed trajectories, we simulate the execution of obtained fleet trajectories via the multi-robot coordination framework described in [5]. We measure the following metrics: sum of completion times (m_1), maximum completion time (aka makespan, m_2), sum of path lengths (m_3), and maximum path length (m_4).

1) Goal Allocation and Obstacle Free Space $\mathcal{W}^{\text{free}}$: As observed in Section IV-C, finding a robot-goal allocation σ is a consequence of computing $\tau^{\mathcal{T}}$. To assess the quality of the assignment σ, we compare it to the assignment σ_d that minimizes the sum of Euclidean distances between robots and goals, i.e., minimize $\sum_{r=1}^{R} \sum_{g=1}^{G} d_{rg} \cdot ||\boldsymbol{s}[r] - \boldsymbol{g}[g]||_2$, where $d_{rg} \in \{0, 1\}$, $\sigma_d(\boldsymbol{s}_r) = \boldsymbol{g}_g$ iff $d_{rg} = 1$, and each robot (goal) is constrained to be assigned to only one goal (robot). Since σ_d is not aware of the environment, for scenarios such as the one depicted in Figure 3 we expect to obtain better solution quality with our method. This is confirmed by our experimental results: as shown in Figure 3(b), we obtain a significant improvement in terms of m_1, m_2 and m_4. The naive assignment and our approach score similarly on metric m_3 (sum of path length). This is not surprising — for instance, if we assume that there are two robots $\mathcal{R}_1$



(a) (b)

Fig. 3: [(a) Left] trajectories obtained with our approach (σ); [(a) Right] trajectories obtained with σ_d. N.b. the yellow robot is assigned the dark blue robot goal in σ_d, which is closer to the yellow's start configuration if the $\mathcal{W}^{\text{free}}$ is not considered. (b) Average metrics of a set of 100 experiments with a fleet of 5 robots on the scenario in (a).

and $\mathcal{R}_2$ initially in poses $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, two goals $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, and that these are all located along the X axis in the order $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{g}_1, \boldsymbol{g}_2$; then, the assignment $\sigma(\boldsymbol{s}_1) = \boldsymbol{g}_2$ and $\sigma(\boldsymbol{s}_2) = \boldsymbol{g}_1$, which requires $\mathcal{R}_1$ to overtake $\mathcal{R}_2$, will measure similarly to other assignments on m_3, but better on m_1, m_2 and m_4.

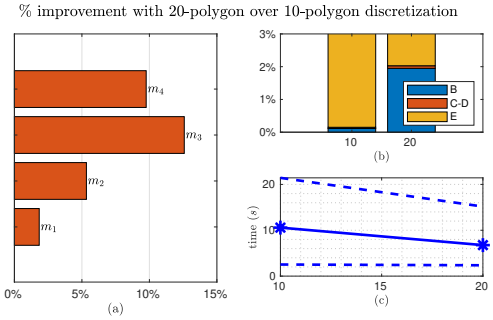


Fig. 4: (a) Average metrics of a set of 100 experiments with a fleet of 4 robots. (b) Run time divided by each phase of the algorithm as described in sections in IV, note that y-axis is broken at 3% since phase of section E will take the remaining time. (c) query phase B–E run time in both discretizations.

2) Polygon Discretization of $\mathcal{W}^{\text{free}}$: A further set of 100 experiments was conducted in Scenario 1 (see Figure 2) to assess the effect of the number of polygons into which $\mathcal{W}^{\text{free}}$ is discretized. We obtained a more granular discretization by subdividing each of the original polygons into two equally-sized polygons along the longer axis. The results for the same setting of start and goal configurations are shown in Figure 4(a). We observe that higher granularity of the discretization (more polygons) leads to an improvement in metric m_3 (sum of path lengths). Furthermore, it is interesting to notice that more polygons lead to faster queries, as shown in Figure 4(c). This is because, on one hand, the higher granularity leads to an increase in the computation time of phase B compared to other phases, as shown in Figure 4(b); but on the other, this leads to the abstract trajectory $\tau^{\mathcal{K}}$ having more steps, which in turn leads to more problem decomposition in phase E. This comes at the expense of more offline computation and bigger multi-robot road-maps.

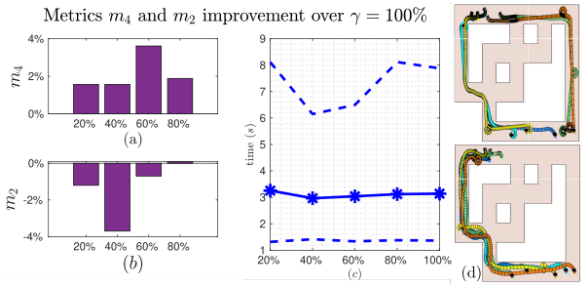


Fig. 5: Figures (a) and (b) report the average metrics improvement for $\gamma = \{20\%, 40\%, 60\%, 80\%\}$. Figure (c) reports on the average run time for the different gammas and also shows the 95-percentile and the 5-percentile lines. Figure (d) shows an example of the trajectories when $\gamma = 100\%$ on the top, and on the bottom when $\gamma = 40\%$.

3) Cost of Coordination γ: The factor γ in the step cost function in eq. (3) is used to balance Euclidean distance and the cost incurred by robots occupying the same polygon (interference cost). To evaluate the effect of γ, we conducted 50 experiments per γ value for a set of 5 robots in the scenario shown in Figure 5(d). Note that we are using Scenario 2 with two additional corridors, in order to allow multiple non-congested path options with longer detours. The results indicate that when interference cost is given more weight, longer paths might be chosen, see Figure 5(a). This decreases the overall time spent yielding, since robots traverse less congested regions, thus improving the makespan as shown in Figure 5(b). Note also that, for $\gamma = \{40\%, 60\%\}$, the computation time of the more complex problems is reduced, as shown by the lower 95-percentile in Figure 5(c). This occurs because robots traversing separate regions do not have to be considered jointly, thus making the computation of motion plans easier.

VI. Conclusions

We have presented a scalable roadmap based approach for solving the MMP-GA. In the future, we intend to exploit our spatio-temporal decomposition for achieving a receding-horizon fleet controller. Furthermore, given the fact that the suitability of a γ value depends on the shape of the environment robots operate in, we intend to investigate learning techniques that can appropriately estimate this value. Finally, we will investigate alternative graph search methods to A* for obtaining the abstract multi-robot trajectory $\tau^{\mathcal{K}}$, with the aim of further reducing the computational overhead of queries to the roadmap.

Acknowledgments. This work is supported by the EU H2020 project ILIAD, the Swedish Knowledge Foundation (KKS) research profile Semantic Robots, and Vinnova projects iQMobility and AutoHauler.

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, pp. 9–9, 2008.

[2] 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.

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

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

[5] 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.

[6] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Rob. Res. (IJRR)*, vol. 23, no. 9, pp. 939–954, 2004.

[7] C. Nam and D. A. Shell, "Assignment algorithms for modeling resource contention in multirobot task allocation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 889–900, 2015.

[8] J. Guerrero and G. Oliver, "Physical interference impact in multi-robot task allocation auction methods," in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*. IEEE, 2006, pp. 19–24.

[9] P. Forte, A. Mannucci, H. Andreasson, and F. Pecora, "Online task assignment and coordination in multi-robot fleets," *IEEE Robotics and Automation Letters*, 2021, to appear.

[10] 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.

[11] 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.

[12] 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.

[13] J. Salvado, R. Krug, M. Mansouri, and F. Pecora, "Motion planning and goal assignment for robot fleets using trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2018, pp. 7939–7946.

[14] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2986–2993.

[15] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 157–173.

[16] 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.

[17] 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.

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

[19] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[20] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2011, pp. 294–300.

[21] D. M. Kornhauser, G. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," Master's thesis, M. I. T., Dept. of Electrical Engineering and Computer Science, 1984.

[22] A. Adler, M. De Berg, D. Halperin, and K. Solovey, "Efficient multi-robot motion planning for unlabeled discs in simple polygons," in *Algorithmic foundations of robotics XI*. Springer, 2015, pp. 1–17.

[23] 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.

[24] 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.

[25] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robotics and autonomous systems*, vol. 23, no. 3, pp. 125–152, 1998.

[26] D. Le and E. Plaku, "Multi-robot motion planning with dynamics via coordinated sampling-based expansion guided by multi-agent search," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1868–1875, 2019.

[27] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning," *Int. J. Rob. Res. (IJRR)*, vol. 35, no. 5, pp. 501–513, 2016.

[28] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.

[29] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[30] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path," *IEEE Robotics & Automation Magazine*, vol. 15, no. 2, pp. 58–66, 2008.

[31] L. Simonson and G. Suto, "Geometry template library for stl-like 2d operations," *Colorado: GTL Boostcon*, vol. 4, 2009.

[32] K. Beevers and J. Peng, "A* graph search within the bgl framework," *Boost Graph Library 1.33. 0 (http://www. cs. rpi. edu/~ beevek/research/astar_bgl04. pdf October 2004)*, 2003.