A Network-Flow Reduction for the Multi-Robot Goal Allocation and Motion Planning Problem

João Salvado, Masoumeh Mansouri, Federico Pecora

***Abstract*— This paper deals with the problem of allocating goals to multiple robots with complex dynamics while computing obstacle-free motions to reach those goals. The spectrum of existing methods ranges from complete and optimal approaches with poor scalability, to highly scalable methods which make unrealistic assumptions on the robots and/or environment. We overcome these limitations by using an efficient graph-based method for decomposing the problem into sub-problems. In particular, we reduce the problem to a Minimum-Cost Max-Flow problem whose solution is used by a multi-robot motion planner that does not impose restrictive assumptions on robot kinodynamics or on the environment. We show empirically that our approach scales to tens of robots in environments composed of hundreds of polygons.**

I. Introduction

The *Multi-robot Motion Planning and Goal Allocation Problem* (MMP-GA) deals with finding feasible paths for multiple robots navigating in shared environments. Solving the MMP-GA involves allocating robots to goal configurations and computing trajectories for reaching them that are kinodynamically feasible, efficient, and avoid collisions with the environment and among robots. The different facets of the MMP-GA can be considered separately, as done in so-called *loosely-coupled* approaches for integrated task allocation, motion planning, coordination and control. While such methods have been shown to scale well (see related work in Section II), they usually lead to sub-optimal solutions. Conversely, considering goal allocation, motion planning and coordination jointly leads to highly optimized trajectories, but at the cost of high computational overhead. The spectrum of such *tightly coupled* approaches ranges from complete and optimal methods, which are adequate for small fleets, to highly-scalable methods, which scale to larger fleets by posing unrealistic assumptions on the robots, environment, or both. Tightly-coupled methods are beginning to be adopted in industrial settings, their limitations being overcome by means of costly re-engineering of the environment, robots and/or processes [1].

In this paper, we propose an efficient graph-based method for decomposing the MMP-GA problem into sub-problems involving subsets of robots. The method is based on the reduction of a discretized version of the MMP-GA to a minimum cost maximum flow problem [2]. The solution of this flow problem is then used to instantiate a series of independent multi-robot motion planning problems, which can be solved with existing continuous optimization methods. This allows to retain the advantages of tightly-coupled optimization for only those parts of the problem that require search in the joint configuration space of multiple robots. The method does not impose restrictive assumptions on robot kinodynamics or on the environment, and scales well to tens of robots, hence providing a viable solution for automating industrial applications without requiring costly re-engineering effort. The proposed method is based on prior work [3] in which the discrete MMP-GA is solved via heuristic search. We perform an empirical comparison of the two methods, which shows that the use of maximum flow allows better scaling with respect to the amount of robots and environment size without compromising solution quality. Moreover we demonstrate that the discretized MMP-GA can be solved for a tessellation of the environment composed of hundreds of polygons in a few seconds. Finally, we show how the sub-problems resulting from the decomposition of the overall MMP-GA can be solved concurrently with an existing optimization-based multi-robot motion planner [4], and evaluate the achieved improvement in scalability.

This paper is organized as follows. In Section II we outline the state of the art in multi-robot motion planning, path finding, coordination and control. Section III states the MMP-GA problem formally, while Section IV details the proposed reduction to maximum flow with capacity constraints and how its solutions are used to decompose the overall MMP-GA problem. We evaluate our approach empirically in Section V, and conclude with a brief outlook in Section VI.

II. Related Work

Existing methods often consider some facets of the MMP-GA problem while ignoring others. Some approaches rely on efficient single robot motion planners to compute a path for each individual robot, and then avoid collisions among robots by scheduling of their motions [5], [6] or via priority planning [7], [8]. These methods trade completeness and/or optimality for efficiency, and ignore the issue of goal allocation.

Approaches based on trajectory optimization exploit the existence of efficient convex optimization solvers, e.g., [9]. As the MMP-GA problem is non-convex, these approaches convexify the problem by assuming holonomic robots [10], robots navigating in obstacle-free space and linearizing collision constraints [11], decomposing the problem into simpler convex problems to be solved sequentially [12], allowing

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

collisions until a defined time horizon [13], or sequentially tightening collision constraints [14].

The Multi-Agent Path Finding (MAPF) problem is related to the MMP-GA problem, in that it addresses the issue of routing multiple agents through a graph. Efficient Integer Linear Programming (ILP) solvers have been used to solve a reduction of the MAPF problem to a dynamic network flow problem [15], and Conflict-Based Search (CBS) [16] has been used to resolve collisions in individually-obtained robot plans via conflict binary trees [17]. Some graph-based approaches plan in the combined configuration space of the fleet only when single robot paths are non-colliding [18]. These methods scale well with the number of robots [1], however they assume an unrealistic one-robot-one-cell paradigm. This assumption was relaxed in [19], allowing robots to occupy several cells while navigating in grid-like environments without considering robot kinematics. Honig et al. [20] use realistic dynamic models to post-process paths that are initially kinodynamically infeasible. The labeled MAPF problem, where robots are allocated to goals and/or are heterogeneous (known to be NP-hard [21]), can be solved by leveraging efficient solvers for the unlabeled case (flow reduction) while reasoning about conflicts between heterogeneous robots on a higher level via the Conflict-Based Min-Cost-Flow algorithm [22]. Others have exploited reductions to the problem of coordinating pebble motions on graphs [23], for which efficient algorithms exist [24]. The unlabeled variant of this problem, where homogeneous robots/pebbles are not allocated to specific goals, is known to be solvable efficiently under the assumption of one pebble per vertex and/or when start and goal configurations do not overlap [25]. These assumptions are relaxed in [26] by generating multiple unlabeled pebble graphs when start and goal co-location occurs. Most of the previous approaches ignore the problem of finding motions for robots with non-trivial kinodynamics; the few that do, do not address the issue of goal allocation.

Several methods presented in the literature adopt, as we do, the idea that different facets of the MMP-GA problem are tackled by solvers at several levels of abstraction. For instance, [27] partitions a single-robot roadmap into convex regions, solving the problem at multiple levels. Others leverage a discrete-space holonomic robot abstraction as a means to guide the search in multi-robot configuration space [28], or pre-compute single-robot roadmaps to be utilized during a multi-robot sampling phase [29]. Finally, some exploit solutions obtained from a fast CBS-based multi-robot path planner to warm-start an optimization-based solver [30]. None of these approaches considers the goal allocation problem in conjunction with multi-robot motion planning.

III. Problem Statement

A. Nomenclature

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

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

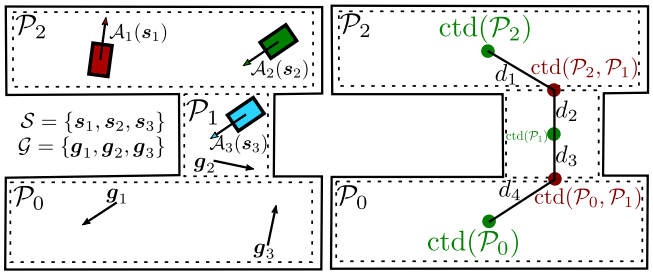


Fig. 1: Running example of a MMP-GA problem with three robots and three goals navigating on an $\mathcal{W}^{\text{free}}$ composed of three polygons. $\text{ctd}(\mathcal{P})$ and $\text{ctd}(\mathcal{P}_i, \mathcal{P}_i)$ are the centroids of polygons and of their intersections, and $d_i \in \mathbb{R}$ specify real-valued distances.

B. Preliminary Definitions

1) Robot and World: Robots navigate in a bounded 2D world $\mathcal{W} \subset \mathbb{R}^2$ with obstacles $\mathcal{O} \subset \mathcal{W}$. Each robot $\mathcal{R}_r$ is defined as a tuple $\mathcal{R}_r = \langle \mathcal{A}_r, \boldsymbol{x}_r, f_r \rangle$ of geometric shape $\mathcal{A}_r(\boldsymbol{q}_r)$, state $\boldsymbol{x}_r = [\boldsymbol{q}_r\, \dot{\boldsymbol{q}}_r]$ and dynamic model $\dot{\boldsymbol{x}}_r = f_r(\boldsymbol{x}_r(t))$, where $\boldsymbol{q}_r \in \mathcal{C}_r$ is a configuration in the robot's configuration space.

2) Trajectory: A robot's trajectory from a start configuration $\boldsymbol{s}_r \in \mathcal{C}_r$ to a goal configuration $\boldsymbol{g}_r \in \mathcal{C}_r$ is a time-continuous mapping $\tau_r : [0, 1] \mapsto \mathcal{C}_r$, where $\tau_r(0) = \boldsymbol{s}_r$ and $\tau_r(1) = \boldsymbol{g}_r$. A trajectory is *free* when $\tau_r : [0, 1] \mapsto \mathcal{C}_r^{\text{free}}$, where $\mathcal{C}_r^{\text{free}} = \{\boldsymbol{q}_r \in \mathcal{C}_r \mid \mathcal{A}_r(\boldsymbol{q}_r) \cap \mathcal{O} = \emptyset\}$, that is, the configurations where the robot does not collide with obstacles. A trajectory is *feasible* if it is free and it can be followed with the robot's dynamic model.

3) Multiple Robots: Previously defined variables are extended to the case of multiple robots by dropping the r subscript. That is, we have the multi-robot state space $\boldsymbol{x} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_R]$, configuration free space $\mathcal{C}^{\text{free}} = \prod_{r=1}^{R} \mathcal{C}_r^{\text{free}}$, geometric shape $\mathcal{A} = \cup_{r=1}^{R} A_r$ and dynamic model $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$.

4) Obstacle Free Space: We approximate the obstacle free space $\mathcal{W}^{\text{free}} = \mathcal{W} \backslash \mathcal{O}$ with a set of convex adjacent polygons $\{\mathcal{P}_1, \ldots, \mathcal{P}_P\}$ overlapping only on their boundary, e.g., as in a Voronoi tessellation [31] and as exemplified in Fig. 1. Let $\partial\mathcal{P}$ denote the boundary of polygon $\mathcal{P}$. We generate the obstacle-free space *connectivity graph* $G_{\mathcal{W}^{\text{free}}} = \langle V_{\mathcal{W}^{\text{free}}}, E_{\mathcal{W}^{\text{free}}} \rangle$, with vertices $V_{\mathcal{W}^{\text{free}}} = \{\mathcal{P}_1, \ldots, \mathcal{P}_P\}$ and edges $E_{\mathcal{W}^{\text{free}}} = \{(\mathcal{P}_i, \mathcal{P}_j) \in V^2_{\mathcal{W}^{\text{free}}} \mid \exists \boldsymbol{q}_r \in \mathcal{C}^{\text{free}}_r \text{ s.t. } \boldsymbol{q}_r \subset \partial\mathcal{P}_i \cap \partial\mathcal{P}_j\}$, meaning that an edge exists if a robot can be "placed" somewhere on the boundary intersection between polygons. We will refer to such polygons $\mathcal{P}_i$ and $\mathcal{P}_j$ as *connected*. The connectivity graph for our running example is shown in Fig. 2. Finally, we define the mappings c_s :

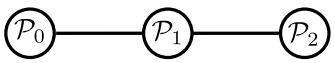


Fig. 2: The environment connectivity graph of our running example from Fig. 1.

$\{\mathcal{P}_1, \ldots, \mathcal{P}_P\} \mapsto \mathbb{R}$ for the cost of staying in a polygon and $u_p : \{\mathcal{P}_1, \ldots, \mathcal{P}_P\} \mapsto \mathbb{N}$ for the maximum number of robots allowed in each polygon.

5) Robot-Goal Allocation: Let $\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_R\}$ and $\mathcal{G} = \{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_G\}$ be the set of the multi-robot start and goal configurations. Let $\sigma : \mathcal{S} \mapsto \mathcal{G}$, be a mapping between robots and goals, e.g., $\sigma(\boldsymbol{s}_2) = \boldsymbol{g}_5$ means that the robot starting in $\boldsymbol{s}_2$ is allocated to goal $\boldsymbol{g}_5$. Thus, allocating robots to goals means computing σ.

6) Network Flow: A network $\mathcal{N} = \langle G^{\mathcal{N}}, u^{\mathcal{N}}, c^{\mathcal{N}}, \mathcal{S}^{\mathcal{N}} \rangle$ is composed of a directed graph $G^{\mathcal{N}} = \langle V^{\mathcal{N}}, E^{\mathcal{N}} \rangle$; a set of source $\mathcal{S}^{\mathcal{N}}_+$ and sink nodes $\mathcal{S}^{\mathcal{N}}_-$, where $\mathcal{S}^{\mathcal{N}} = (\mathcal{S}^{\mathcal{N}}_+ \cup \mathcal{S}^{\mathcal{N}}_-) \subset V^{\mathcal{N}}$; a mapping $u^{\mathcal{N}} : E^{\mathcal{N}} \mapsto \mathbb{N}$ specifying the edge capacities; and a mapping $c^{\mathcal{N}} : E^{\mathcal{N}} \mapsto \mathbb{R}$ defining the cost of a unit of flow traversing an edge. A Minimum-Cost Maximum-Flow problem [2] involves finding the maximum flow $f^{\mathcal{N}} : E^{\mathcal{N}} \mapsto \mathbb{N}$ that can be pushed through the network $\mathcal{N}$ from source to sink nodes while satisfying capacity and flow conservation constraints[1] and the objective

$$\text{minimize} \sum_{e \in E^{\mathcal{N}}} f^{\mathcal{N}}(e) \cdot c^{\mathcal{N}}(e). \tag{1}$$

C. Problem Formulation

We define the *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 objective is to find a multi-robot trajectory $\tau(t) = [\tau_1(t) \ \ldots \ \tau_R(t)]$ and a robot-goal allocation σ that minimizes some cost function $L(t, \boldsymbol{x})$ and such that

1) $\forall r \ . \ \tau_r(0) = s_r$ (initial condition),
2) $\forall r \ . \ \tau_r(1) = \sigma(s_r)$ (final condition),
3) $\forall r \ \forall t \in [0,1] \ . \ \tau_r(t) \in \mathcal{C}^{\text{free}}_r$ (feasibility),
4) $\forall r_i, r_{j \neq i} \ \forall t \in [0,1] \ . \ \mathcal{A}_{r_i}(\tau_{r_i}(t)) \cap \mathcal{A}_{r_j}(\tau_{r_j}(t)) = \emptyset$ (non-collision).

[1]Formulation is simplified, for more details consult [32].

D. Abstract Problem Formulation

We define a *discrete Multi-robot Motion Planning and Goal Allocation Problem* (dMMP-GA) as a discrete abstraction of the MMP-GA with the aim of finding a sequence of polygon transitions that bring the fleet from occupying the set of polygons it starts in to occupying the set of polygons in which the goal configurations are placed. Having $\mathcal{C}^{\mathcal{K}} = \{0, \ldots, R\}^P$ as an abstract state which represents the number of robots in each polygon, we define the mapping $\lambda : \ \mathcal{C} \mapsto \mathcal{C}^{\mathcal{K}}$ between a fleet configuration and its polygonal occupancy. Therefore, solving the dMMP-GA equates to finding a discrete trajectory $\tau^{\mathcal{K}}(k) \in \mathcal{C}^{\mathcal{K}}$ for $k \in \{1, \ldots, N\}$ such that:

1) the number of robots in each polygon at $k = 0$ reflects the start configuration $\boldsymbol{s}$, i.e., $\tau^{\mathcal{K}}(0) = \lambda(\boldsymbol{s})$
2) the number of robots in each polygon at $k = N$ reflects the goal configuration $\boldsymbol{g}$, i.e., $\tau^{\mathcal{K}}(N) = \lambda(\boldsymbol{g})$
3) each abstract state is feasible, i.e., $\forall k \ \exists \boldsymbol{q} \in \mathcal{C}^{\text{free}} : \lambda(\boldsymbol{q}) = \tau^{\mathcal{K}}(k)$
4) each abstract transition is feasible, i.e., if $\tau^{\mathcal{K}}(k) = \lambda(\boldsymbol{q}_k)$ and $\tau^{\mathcal{K}}(k+1) = \lambda(\boldsymbol{q}_{k+1})$ and $\mathcal{A}_r(\boldsymbol{q}_k[r]) \subset \mathcal{P}_i$ and $\mathcal{A}_r(\boldsymbol{q}_{k+1}[r]) \subset \mathcal{P}_j$, then either $\mathcal{P}_i = \mathcal{P}_j$ or $(\mathcal{P}_i, \mathcal{P}_j) \in E_{\mathcal{W}^{\text{free}}}$

Note that, the dMMP-GA problem is fully defined by the tuple $\langle G^{\mathcal{K}}, \lambda(\boldsymbol{s}), \lambda(\boldsymbol{g}) \rangle$, where graph $G^{\mathcal{K}} = \langle V^{\mathcal{K}}, E^{\mathcal{K}} \rangle$ is composed of a set of feasible vertexes $V^{\mathcal{K}} = \{\boldsymbol{c}^{\mathcal{K}} \in \mathcal{C}^{\mathcal{K}}\}$ and a set of feasible edges $E^{\mathcal{K}} = \{(\boldsymbol{c}^{\mathcal{K}}_i, \boldsymbol{c}^{\mathcal{K}}_j) \in \mathcal{C}^{\mathcal{K}} \times \mathcal{C}^{\mathcal{K}}\}$.

IV. Approach

The relationship between multi-robot path planning and network flow problems was explored in [15], where a reduction from collision-free unit-distance graphs (CUG's) to the dynamic network flow problem is shown. However, it is not possible to apply this reduction to the dMMP-GA problem, as the cost associated to transitioning between abstract states is a real number, and several robots can occupy the same polygon, thus breaking the unit-distance and collision-free assumptions. Recall that, a Minimum-Cost Maximum-Flow problem of a network $\mathcal{N}$ can be solved in polynomial time when $|S^{\mathcal{N}}_+| = |S^{\mathcal{N}}_-| = 1$ and $\forall e \in E^{\mathcal{N}} : c^{\mathcal{N}}(e) = -1$, which results in a maximum flow problem

$$\text{maximize} \sum_{e \in E^{\mathcal{N}}} f^{\mathcal{N}}(e). \tag{2}$$

This problem can be solved in polynomial time by the Ford-Fulkerson algorithm [33]. Interestingly, since we require $c^{\mathcal{N}} \in \mathbb{R}$ the problem becomes NP-Hard, although efficient algorithms exist, and we exploit a capacity-scaling algorithm [34] with time complexity $\mathcal{O}\left(|E^{\mathcal{N}}| \cdot \log\left(R|V^{\mathcal{N}}| + R|E^{\mathcal{N}}|\right) \cdot \log(V^{\mathcal{N}})\right)$.

In this paper we propose a polynomial-time reduction

$$\langle G^{\mathcal{K}}, \lambda(\boldsymbol{s}), \lambda(\boldsymbol{g}) \rangle \leq_P \langle G^{\mathcal{N}}, u^{\mathcal{N}}, c^{\mathcal{N}}, \mathcal{S}^{\mathcal{N}} \rangle, \tag{3}$$

meaning that the dMMP-GA problem is reduced to a network flow problem. In practice, we create a time-expanded network (also known as *dynamic network*) of the connectivity

graph of the obstacle free-space ($G_{\mathcal{W}^{\text{free}}}$) while (1) ensuring that the maximum amount of robots per polygon is not exceeded via $u^{\mathcal{N}}$; (2) considering traversal costs by defining $c^{\mathcal{N}}$; and (3) imposing abstract start and goal states via $\mathcal{S}^{\mathcal{N}}$. Traditionally, an time-expansion is considered in the strict sense of discrete time intervals, however our expansion is related to discrete concurrent transitions of robots between polygons. The number of expansions T is then a consequence of estimating the number of such concurrent transitions.

Finally, we exploit the fact that by reducing the original problem to a network flow problem, the dMMP-GA problem can be solved efficiently. By analyzing the abstract solution of the dMMP-GA, the full MMP-GA is de-composed into simpler problems, since it is known where (in which polygons) and when robot motions may affect each other.

A. Multi-Robot Dynamic Flow Network: a Discrete Abstraction

The procedure utilized to realize the reduction in (3) will be demonstrated via the running example shown in Fig. 1, where we will illustrate two consecutive expansions ($T = 2$) over a static network derived from the connectivity graph of the environment $G_{\mathcal{W}^{\text{free}}}$ shown in Fig. 2

A single dynamic network expansion, depicted in Fig. 3, aggregates three repeated sets of vertices, which for simplicity we refer to as *layers*. Each layer has a vertex for every polygon in $\mathcal{W}^{\text{free}}$. Edges between vertices from layer 0 to layer 1 account for the cost of transitioning between polygons (black arrows) or staying in them (blue arrows), and edges between layer 1 and 2 ensure that the maximum capacity of polygons is not exceeded (red arrows). Notice that multiple robots are allowed to stay and/or transition through polygons concurrently in a single expansion. In summary, a single expansion allows multiple robot transitions between *connected* polygons that do not violate maximum capacities of target polygons.

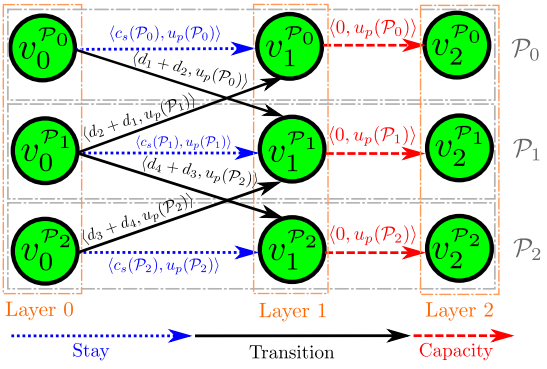


Fig. 3: Single dynamic network expansion for the environment of our running example. Distances d_1, d_2, d_3, d_4 can be seen in Fig. 1. The tuple over each directed edge shows its cost and capacity.

A complete dynamic network with two expansions ($T = 2$) is illustrated in Fig. 4 (top). We construct a network with a single source $\mathcal{S}^{\mathcal{N}}_{+} = \{v_+\}$ and single sink $\mathcal{S}^{\mathcal{N}}_{-} = \{v_-\}$ with a surplus and demand of R robots respectively. Therefore, the abstract start and goal configurations are modeled into the network by imposing maximal capacities as $\lambda(\boldsymbol{s})[i]$ on the outgoing edges of $\{v_+^{\mathcal{P}_i} : \mathcal{P}_i \in \mathcal{W}^{\text{free}}\}$ and $\lambda(\boldsymbol{g})[i]$ on the incoming edges of $\{v_-^{\mathcal{P}_i} : \mathcal{P}_i \in \mathcal{W}^{\text{free}}\}$. After the flow is computed (red edges), an abstract trajectory is extracted from the resulting flow on the incoming edges of every vertex in layer 3 at all expansions. For instance, assuming that $\mathcal{P}_1$ has a maximum capacity of one robot, we obtain the maximum flow shown in Fig. 4 (bottom), where $\tau^{\mathcal{K}}(0) = [0\,1\,2], \tau^{\mathcal{K}}(1) = [1\,1\,1], \tau^{\mathcal{K}}(2) = [2\,1\,0]$. In summary, querying the network means modifying such edge capacities according to $\lambda(\boldsymbol{s})$ and $\lambda(\boldsymbol{g})$, followed by a computation of the maximum flow that can be pushed through the network $\mathcal{N}$ with minimum cost, and finally transforming back to the original abstract configuration space.

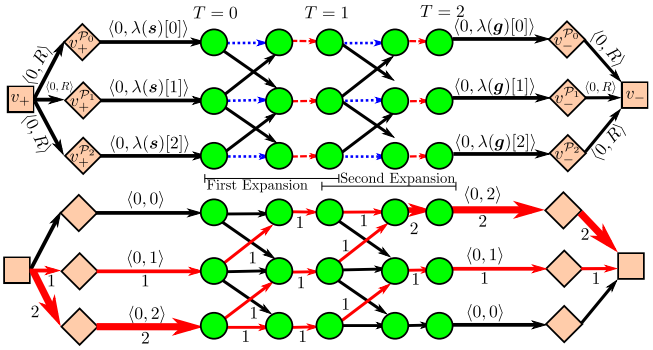


Fig. 4: Top: complete dynamic network $\mathcal{N}$ with $T = 2$ two expansions for our running example. Bottom: computed flow where $\mathcal{P}_1$ is traversed by one robot at a time due to the capacity constraint $u_p(\mathcal{P}_1) = 1$.

B. Multi-robot Dynamic Flow Network: Optimality considerations

Existing CUG-based approaches assume unitary edge cost, hence there is a direct correlation between the traversal time of the fleet and path length. This allows to exploit T (number of expansions) to minimize for different metrics. In our case, however, the number of expansions T in $\mathcal{N}$ is not correlated with path length, but to the number of polygon transitions. We explain how this affects different metrics below.

1) Minimizing sum of path lengths: The objective function defined in eq. 1 is to minimize the sum of edge costs multiplied by the amount of flow in those edges. Equivalently – retrieving the original abstract problem $\langle G^{\mathcal{K}}, \lambda(\boldsymbol{s}), \lambda(\boldsymbol{g})\rangle$ with robots as flow and path length as edge cost – we are thus directly minimizing for the sum of path lengths for some T. Moreover, a feasible solution is required to exist with at most T discrete polygon transitions. Therefore, we are required to use a *big enough* T. It is easy to see that the upper bound $T_{\text{max}} = P + R - 2$. Intuitively, a single robot r might need to traverse all polygons to navigate from start to goal configuration, thus $T \geq P - 1$. Additionally, if robot r traverses some polygon with unit capacity, and all other robots also need to traverse this polygon, then in the worst

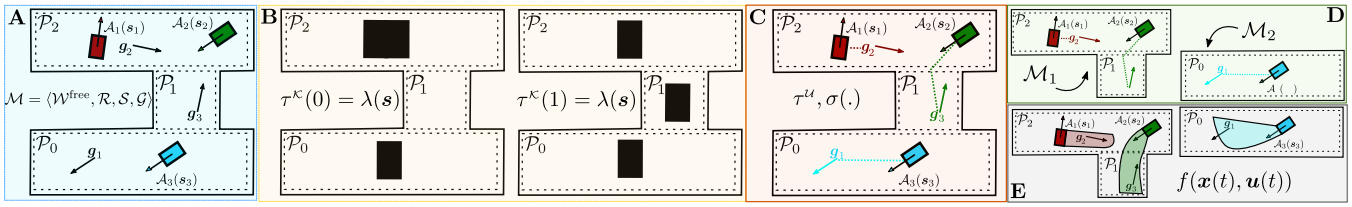


Fig. 5: Example with our method phases on solving a MMP-GA problem. (**A** blue) MMP-GA problem with three: robots, goals and polygons. (**B** yellow) Abstract trajectory $\tau^{\mathcal{K}}$ that is the solution of dMMP-GA problem, robots are in black to demonstrate they are unlabeled. (**C** red) Depiction of an optimal robot labeling with robot's traversed polygons and allocated goal. (**D** green) Initial problem de-composition in two MMP-GAs $\mathcal{M}_1$ and $\mathcal{M}_2$ to be solved concurrently. (**E** grey) Final phase where we account for robot's dynamics.

case robot r has to wait for all other robots, hence $T \geq R-1$. In our running example, we thus have $T_{\text{max}} = 4$.

2) Minimizing maximum path length: The value T_{max} often highly overestimates the length N of an abstract trajectory solution $\tau^{\mathcal{K}}$. Consequently, we can iteratively solve for $T \in [T_{\text{min}}, T_{\text{max}}]$ as a means to quickly get a feasible solution. A lower bound T_{min} can be estimated as the shortest path through the connectivity graph between $\lambda(\boldsymbol{s})$ and $\lambda(\boldsymbol{g})$. In our running example, $T_{\text{min}} = 2$.

Recall that expansions T are related to polygon transitions and not discrete time/path length, hence the first feasible solution found while iterating over T is not necessarily minimal with respect to maximum path length. One such case is shown in figure Fig. 6, where the solution found for $T = 4$ (multi-robot path 2) is sub-optimal with respect to both sum and maximum path length.

Observe that the polygons depicted in Fig. 6 do not have similar sizes. Conversely, if the tessellation $\mathcal{W}^{\text{free}}$ is composed of similarly- or equally-sized polygons, then the solution with lowest T would be optimal with respect to maximum path length.

Operationally, querying for $T = 1$ in the running example in Fig .1 implies deleting incoming edges on vertices $\{v_-^{\mathcal{P}_i} : \mathcal{P}_i \in \mathcal{W}^{\text{free}}\}$ (see Fig. 4) and creating edges between vertices in layer 2 for expansion $T = 1$ and vertices $\{v_-^{\mathcal{P}_i} : \mathcal{P}_i \in \mathcal{W}^{\text{free}}\}$ with the respective capacities and costs of the previously deleted edges.

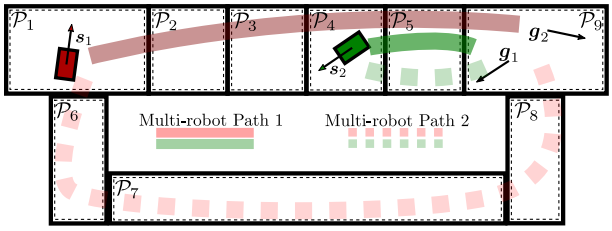


Fig. 6: Example where an optimal solution can not be found for $T < 5$, although a sub-optimal one exists for $T = 4$.

C. MMP-GA planner: exploiting dMMP-GA solution

The solution of a dMMP-GA problem abstraction is utilized to tackle the original MMP-GA problem in a divide-and-conquer manner. The key intuition is that the solution of the dMMP-GA tells us when and by how many robots each polygon is be traversed. Next, we describe the complete procedure of solving a MMP-GA problem by exploiting the solution of dMMP-GA. For that, we use a second running example shown in Fig. 5 (**A** blue). For a detailed explanation, please refer to [3].

Once an abstract trajectory $\tau^{\mathcal{K}}$ is computed, our method: (1) labels the fleet's robots along the abstract trajectory, consequently finding a robot-goal allocation; (2) de-composes the original problem in time and space, exploiting the fact that robots navigating different polygons do not interfere; (3) solves the previously de-composed problems concurrently while accounting for the robot's kinodynamics via a multi-robot motion planner [4]. These steps are briefly summarized below.

1) Labeling Robots: Solving the dMMPA-GA results in a discrete abstract trajectory $\tau^{\mathcal{K}}$ with unlabeled robots. In the example from Fig. 5, we the abstract trajectory is $\tau^{\mathcal{K}}(0) = [2\,0\,1]$ and $\tau^{\mathcal{K}}(1) = [1\,1\,1]$ (Fig. 5 **B** yellow). Here, although we known that the robot in $\mathcal{P}_1$ transitions to $\mathcal{P}_2$, we do not know which of the two robots in $\mathcal{P}_0$ transitions to $\mathcal{P}_1$. Therefore, to label the robots we construct a search search tree $\mathcal{U}$ along $\tau^{\mathcal{K}}$ considering the possible robot transitions. The tree is traversed in a *depth-first* manner for a labeling of robots that minimizes the sub-optimality resulting from an initial approximation of the transition costs for the start and goal configuration when creating the dMMP-GA. For example (see Fig. 5 **C** red), our labeling method selects robot r_2 to be the one transitioning to polygon $\mathcal{P}_1$, since it has a lower transitioning cost, whereas selecting robot r_1 would incur in an additional cost resulting from the robot's start configuration. The labeling leads to a multi-robot trajectory $\tau^{\mathcal{U}} = \prod_{r=1}^{R} \tau_r^{\mathcal{U}}$ prescribing the sequence of polygon-centroids traversed by all robots, and which implies a certain robot-goal allocation σ. For example, $\tau^{\mathcal{U}}(0) = [\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3]$ and $\tau^{\mathcal{U}}(1) = [\boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_1]$ (see Fig. 5 **C** red).

2) Decoupling the Problem: The fact that $\tau^{\mathcal{U}}$ defines when robot navigate through which polygons allows to decouple the problem in discrete time and space by computing a partition of independent robots. For the running example, we would have the partition $\{\{\mathcal{R}_1, \mathcal{R}_2\}, \{\mathcal{R}_3\}\}$ in a single discrete time step. Thus the original problem $\mathcal{M} = \langle \mathcal{W}^{\text{free}}, \mathcal{R}, \mathcal{S}, \mathcal{G} \rangle$ is now decomposed into the two sub-

problems

$$\mathcal{M}_1(\{\mathcal{P}_1,\mathcal{P}_2\},\{\mathcal{R}_1,\mathcal{R}_2\},\{\boldsymbol{s}_1,\boldsymbol{s}_2\},\{\boldsymbol{g}_2,\boldsymbol{g}_3\}) \otimes \mathcal{M}_2(\{\mathcal{P}_3\},\{\mathcal{R}_3\},\{\boldsymbol{s}_3\},\{\boldsymbol{g}_1\}), \quad (4)$$

that is, to solving in parallel the problem of finding a trajectory for robots 1 and 2 starting in $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ and ending in $\boldsymbol{g}_2$ and $\boldsymbol{g}_3$ respectively, while navigating inside polygons $\mathcal{P}_1$ and $\mathcal{P}_2$; similarly for robot 3 starting in $\boldsymbol{s}_3$ and ending in $\boldsymbol{g}_1$ while navigating in polygon $\mathcal{P}_3$. In summary, we have divided the original MMP-GA problem into two independent and simpler problems that can be solved concurrently.

3) Multi-robot Motion Planning: Finally, a dynamically feasible trajectory is computed for each of the previously generated independent sub-problems. The multi-robot motion planner used here must enforce the robots' non-holonomic dynamics from start to goal configuration while ensuring that robots remain in the polygons of the sub-problem while not colliding with each other.

V. Experiments and Results

A. Setup

Our approach was tested on a PC running Ubuntu 20.04 equipped with an 8-thread Intel Core i7-6820HQ CPU @ 2.70GHz. The obstacle-free space $\mathcal{W}^{\text{free}}$ is generated with OpenCV by iteratively expanding rectangles centered on randomly sampled points on a grayscale image map until colliding with obstacles or previously generated rectangles. The connectivity graph of the created rectangular polygon tessellation $G_{\mathcal{W}^{\text{free}}}$ is constructed using the Boost Polygon Library (BPL) [35] connectivity extraction algorithm. The graph $G^{\mathcal{N}}$ of the flow reduction was created with the LEMON Graph Library [36] and its capacity scaling algorithm was used to solve the Minimum-Cost Maximum-Flow problem. We exploit the Boost Graph Library (BGL) [37] connected components algorithm to decompose the original problem. For the joint multi-robot motion planning we utilize a trajectory based optimization planner introduced in [4]. Note that our approach allows other tessellations of the environment where polygons are convex and adjacent as defined in Section III.

B. Heuristic-based vs. Flow-based dMMP-GA solver

We start by comparing the flow-based method for solving the dMMP-GA problem introduced in this paper with an approach based on A* graph search [3]. Specifically, the latter approach constructs a graph $G^{\mathcal{K}}$ containing a vertex for each abstract multi-robot state, and an edge for each possible transition of one robot at a time. The comparison is conducted in terms of scalability and solution quality.

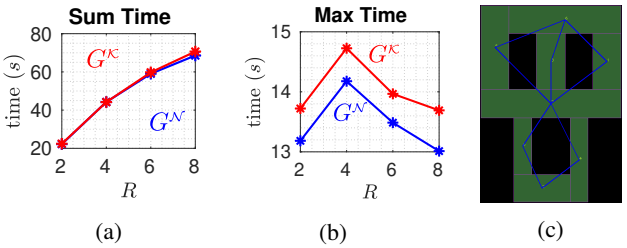
1) Scalability: We construct both graphs $G^{\mathcal{N}}$ and $G^{\mathcal{K}}$ for the scenario displayed in Fig. 7c for $R \in \{2,4,6,8,10\}$. As expected (see Table I), the number of vertices and edges grows exponentially for graph $G^{\mathcal{K}}$ and linearly for graph $G^{\mathcal{N}}$, which explains the much higher building time for the former. The amount of vertices can be estimated as: $|V^{\mathcal{K}}| \approx \binom{P+R}{P}$ and $|V^{\mathcal{N}}| \approx 2\times T\times(P+1)$. Consequently, we have $|V^{\mathcal{K}}| \gg |V^{\mathcal{N}}|$ when the number of polygons or robots increases. Note that $T \in \mathcal{O}(P+R)$.

R	2	4	6	8	10
$\lvert V^{\mathcal{N}}\rvert$	262	302	342	382	422
$\lvert V^{\mathcal{K}}\rvert$	55	715	5005	24290	91928
$\lvert E^{\mathcal{N}}\rvert$	534	622	710	798	886
$\lvert E^{\mathcal{K}}\rvert$	240	5280	48.048	274.440	1.162.860
$t^{\mathcal{N}}$(s) build	$\approx 10^{-3}$	$\approx 10^{-3}$	$\approx 10^{-3}$	$\approx 10^{-3}$	$\approx 10^{-3}$
$t^{\mathcal{K}}$(s) build	$\approx 10^{-4}$	$\approx 10^{-2}$	2	48	784
$t^{\mathcal{N}}$(s) query	$\approx 10^{-3}$	$\approx 10^{-3}$	$\approx 10^{-3}$	$\approx 10^{-3}$	$\approx 10^{-3}$
$t^{\mathcal{K}}$(s) query	$\approx 10^{-3}$	$\approx 10^{-2}$	$\approx 10^{-2}$	$\approx 10^{-1}$	$\approx 10^{-1}$

TABLE I: Creation and query of graphs $G^{\mathcal{N}}$ and $G^{\mathcal{K}}$ for the scenario in Fig. 7c. The table reports the amount of vertices and edges, and the time to build and query the respective graphs for different amounts of robots R.

To evaluate the performance of querying the graphs, we randomly create a set of 50 start and goal configurations for the multi-robot fleet for each value of R and compute an abstract path using each graph. For queries on $G^{\mathcal{N}}$, we compute the solution for T_{max} (see Section IV-B). Table I shows the average time to compute the abstract multi-robot trajectory. Again, the query time for graph $G^{\mathcal{K}}$ grows exponentially with R.

2) Solution Quality: Once an abstract multi-robot trajectory is generated using either graph $G^{\mathcal{K}}$ or $G^{\mathcal{N}}$ to obtain a solution to the dMMP-GA problem, robots are labeled (see Section IV-C.1 and example in Fig. 5**C**), and the original MMP-GA problem is decomposed into smaller problems (see Section IV-C.2 and example in Fig. 5**D**). An optimization-based planner (see Section IV-C.3 and example in Fig. 5**E**) is exploited to enforce realistic dynamics and the absence of robot-robot collisions. Even though the post-processing phase of the abstract solution is the same, the different abstract paths obtained by querying $G^{\mathcal{K}}$ or $G^{\mathcal{N}}$ lead to alternative trajectories. This comparison is shown in Fig. 7



(a) (b) (c)

Fig. 7: Sum of time and makespan for the scenario in (c) for the trajectories resulting from $G^{\mathcal{K}}$ (red) and $G^{\mathcal{N}}$(blue)

for 30 MMP-GA problems per value of R. The resulting trajectories were executed in simulation using the multi-robot coordination framework [6]. A video of one such execution is shown in the video available at `https://youtu.be/Wolhfgkt014`.

While the A*-based approach minimizes path length, the flow-based approach iterates over $T \in [T_{\text{min}}, T_{\text{max}}]$ and stops

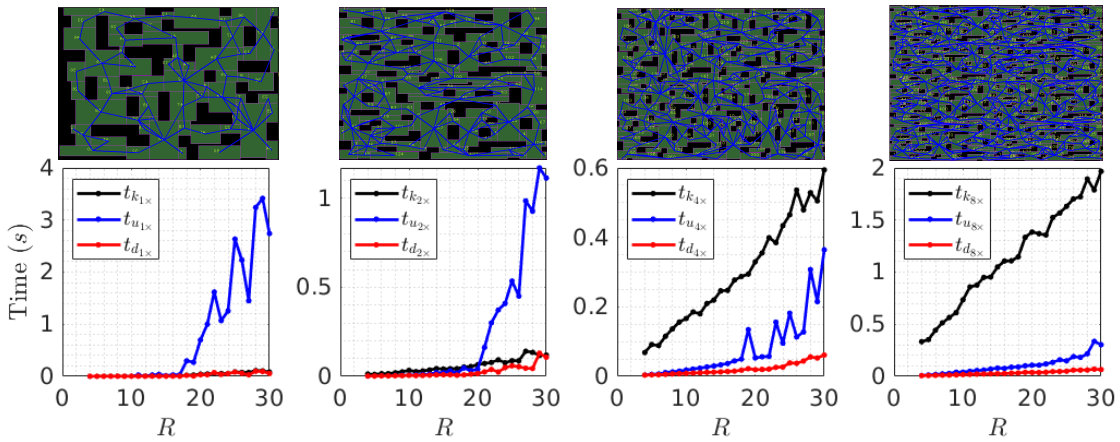


Fig. 8: Each plot summarizes the results obtained on the experiments conducted on the respective scenarios represented above. From left to right, a scenario is a concatenation of two duplicates of the previous scenario, resulting in scenarios with 60, 130, 260 and 510 polygons respectively. The curves show the average computation time for different amounts of robots, having: t_k as the computation time for solving dMMP-GA via $G^{\mathcal{N}}$ (described in Section IV-A, illustrated in Fig. 5**B**), t_u the computation time labeling phase (described in Section IV-C.1, illustrated in Fig. 5**C**), t_d the computation time for decomposition phase (described in Section IV-C.2, illustrated in Fig. 5**D**).

at the first feasible solution (i.e., with minimum polygon transitions). It is interesting to note that even though, as expected, the makespan (max time) is better in $G^{\mathcal{N}}$, the sum of traversal times is identical in the two approaches. We can attribute this fact to a suitable tesselation of the obstacle-free space.

C. Abstract Problem Scalability

In order to better assess scalability of the query phase, we conducted a set of 50 experiments for $4 \leq R \leq 30$. The dMMP-GA problem is solved via flow reduction while minimizing sum of path length (using $T_{\max}$), followed by robot labeling and problem decomposition for scenarios obtained by doubling the number of polygons, as shown in Fig. 8. These results reveal that the computational overhead of both decomposition (t_d) and dMMP-GA solving (t_k) grow linearly with the amount of robots and polygons, while the overhead of the labeling phase (t_u) decreases with the amount of polygons. This occurs because the labeling phase is rather related to the ratio of polygons to robots (similarly to the (n^2-1)-puzzle problem [38]). Also, the labeling phase used in these experiments does not stop at the first solution found, but continues searching the entire search space or times-out, and we retrieve the best solution found so far. Thus, if optimized labeling is not important, we can use the computed flow directly, e.g., in Fig. 5**C** we could arbitrarily allocate the red or green robot to $\boldsymbol{g}_2$ or $\boldsymbol{g}_3$.

D. MMP-GA Scalability and Example

In order to evaluate the computation time required to solve the entire MMP-GA, we conducted 50 experiments per value of R in the scenario shown in Fig. 10. The computation time is summarized in Fig. 9, where the upper and lower bounds are, respectively, the 90- and 10-percentile, while the middle line represents the average. The connectivity graph for this scenario is the same as the leftmost one shown in Fig. 8.

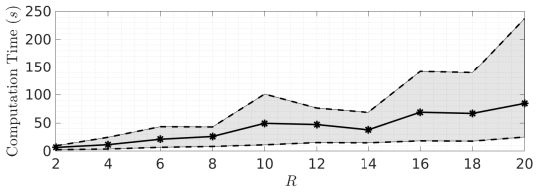


Fig. 9: Computation time per values of R for the first scenario shown in Fig. 8 with approximately 60 polygons.

Note that during the problem decomposition phase (see Section IV-C.2) the original MMP-GA problem is decomposed in space and time. Although we show the total computation time of the fleet trajectory from start to goal configuration, a receding horizon planner could be used to start trajectory execution during the multi-robot motion planning phase [3]. Furthermore, we allow three instances of the multi-robot motion planner to run concurrently (e.g., if a fleet of six robots is totally decoupled, we start computing trajectories for three robots and then the other three, if totally coupled a single planner is utilized).

VI. Future Work

We have presented a network flow reduction for solving the Multi-Robot Motion Planning and Goal Allocation problem. In the future, we intend to evaluate our approach on industrially-relevant environments, including underground mines and a bus depot. We also intend to extend our flow-based approach to account for heterogeneous fleets. Finally, we will compare our method with loosely-coupled approaches to goal allocation.

Acknowledgments. This work is supported by the EU H2020 programme under grant agreement No. 732737 (IL-

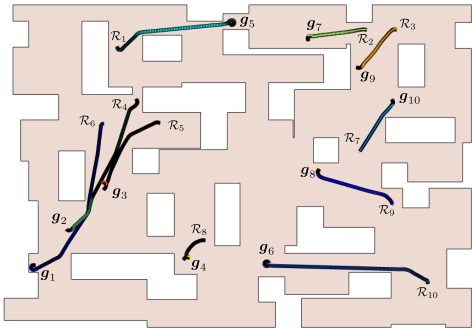


Fig. 10: MMP-GA problem example with start/goal configurations and solution trajectories.

IAD), Vinnova projects iQMobility and AutoHauler, and KKS research profile Semantic Robots.

REFERENCES

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

[2] A. Goldberg and R. Tarjan, "Solving minimum-cost flow problems by successive approximation," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, 1987, pp. 7–18.

[3] J. Salvado, M. Mansouri, and F. Pecora, "Combining multi-robot motion planning and goal allocation using roadmaps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, to appear.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] M. Grant and S. Boyd, "Cvx: Matlab software for disciplined convex programming, version 2.1," 2014.

[10] 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.

[11] 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.

[12] 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.

[13] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.

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

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

[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] J. Li, G. Gange, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "New techniques for pairwise symmetry breaking in multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 193–201.

[18] 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.

[19] J. Li, P. Surynek, A. Felner, H. Ma, T. S. Kumar, and S. Koenig, "Multi-agent path finding for large agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7627–7634.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

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

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

[32] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows," 1988.

[33] L. R. Ford and D. R. Fulkerson, "A simple algorithm for finding maximal network flows and an application to the hitchcock problem," *Canadian journal of Mathematics*, vol. 9, pp. 210–218, 1957.

[34] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM (JACM)*, vol. 19, no. 2, pp. 248–264, 1972.

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

[36] B. Dezső, A. Jüttner, and P. Kovács, "Lemon–an open source c++ graph template library," *Electronic Notes in Theoretical Computer Science*, vol. 264, no. 5, pp. 23–45, 2011.

[37] 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.

[38] D. Ratner and M. Warmuth, "The (n2- 1)-puzzle and related relocation problems," *Journal of Symbolic Computation*, vol. 10, no. 2, pp. 111–137, 1990.