Mixed Integer Conic Programming for Multi-Agent Motion Planning in Continuous Space

Shizhe Zhao[1], Yongce Liu[1], Howie Choset[2] and Zhongqiang Ren[1†]

***Abstract*— Multi-Agent Motion Planning (MAMP) seeks collision-free trajectories for multiple agents from their respective start to goal locations among static obstacles, while minimizing a cost function over the trajectories. Existing approaches for this problem include graph-based, Mix-Integer Programming (MIP) based and trajectory optimization-based, each with its own limitations. This paper introduces a new approach for MAMP based on Mixed Integer Conic Programming (MICP) formulation that complements these existing approaches. We show that our formulation is valid and test our approach against various baselines, including a graph-based method that combines search and sampling, as well as different MIP formulations. The numerical results show that the solutions found by our approach are sometimes eight times closer to the true optimum than the ones found by the baseline when given the same amount of runtime limit. We also verify our approach with multiple drones in a lab setting.**

I. Introduction

This paper investigates a Multi-Agent Motion Planning (MAMP) problem, which seeks collision-free trajectories for multiple agents from their respective start to goal locations among static obstacles, while minimizing the sum of trajectory lengths of the agents subject to a time bound, within which all agents must arrive at their goals. The agents have speed limits and can move in any direction. This problem is fundamental in robotics and arises in applications such as logistics and surveillance. Consider autonomous guided vehicles or forklifts transporting materials in factories, or drones flying in a city to delivery packages in a coordinated fashion among buildings. MAMP naturally arises in these scenarios to optimize the operation of the robots. MAMP is challenging when producing high quality trajectories among obstacles while avoiding agent-agent collision [1], [2].

A. Related Work

To address MAMP, Mixed Integer Programming (MIP) was used to solve MAMP. MILP [3] uses a set of linear constraints to describe the obstacles, and then finds a solution by calling an off-the-shelf solver. However, MAMP often involves nonlinear constraints (e.g., speed limit) and objectives (e.g., Euclidean distance), which cannot be directly represented by MILP. Directly encoding nonlinearity yields a mixed integer nonlinear programming (MINLP) model, which is often much slower. Alternatively, these nonlinearity can be addressed by approximation [4], [5], which may result in highly suboptimal solutions (Fig. 1b).

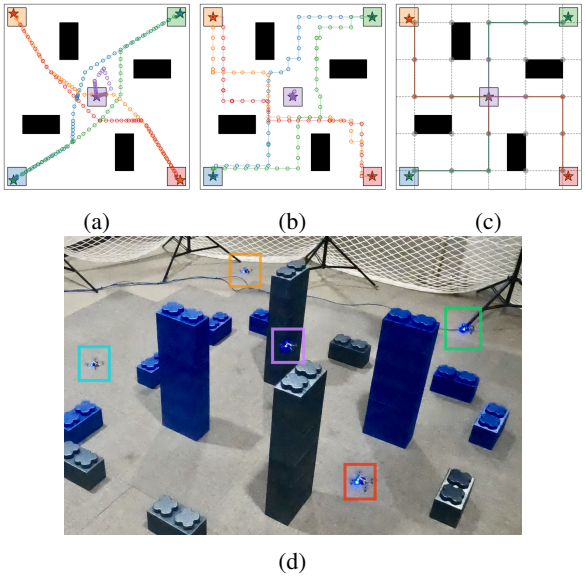


Fig. 1: **(a)** shows an example of MAMP in continuous space with drones, where the trajectories are obtained from our MICP. **(b)** shows the trajectories obtained from the existing MILP model [3]. **(c)** illustrates a possible discretization of the workspace, which can be represented as an undirected graph G. The gray dots and solid lines represent the vertices and edges of G, respectively. **(d)** shows the setup of drones in a motion capture system to execute the planned paths. The color of the rectangular frame on the drone indicates the corresponding agents in (a), (b) and (c). Our MICP often finds shorter paths.

Besides, graph-based methods [6]–[12] usually discretize the workspace into a graph (such as a state lattice or roadmap) and the action space of the agent into a set of motion primitives (i.e., short trajectories connecting two states) to iteratively plan trajectories for the agents from their starts towards their goals. These methods often find high-quality solutions within the graph. However, generating a graph representation to properly capture the obstacle-free space and the potential agent-agent interaction is challenging, since too coarse a discretization may lead to no solution (Fig. 2a) while too fine a discretization may lead to high computational burden (Fig. 2b).

Sampling-based methods [13]–[15] address the aforementioned challenge by iteratively sampling from the state space

[1] Shizhe Zhao, Yongce Liu and Zhongqiang Ren are at Shanghai Jiao Tong University, China. Emails: {shizhe.zhao, yongce.liu, zhongqiang.ren}@sjtu.edu.cn

[2] Howie Choset is at Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA. Emails: {choset@andrew.cmu.edu}

[†] Corresponding author.

or the action space of the agents to find collision-free trajectories. Sampling-based methods run fast to find a first feasible solution and are able to asymptotically converge to an optimal solution when the runtime goes to infinity. However, the solution quality returned by these approaches within a finite runtime can be poor without fine tuning the sampling process (Fig. 2c), especially when the environment is cluttered with bottlenecks. Recent effort seeks to provide solution quality guarantees with a finite number of samples, but are often limited to a small number of agents in practice due to the heavy computational burden [13].

Another method that avoids discretization is using a graph of convex sets (GCS) to represent the workspace and planning motions by optimization. These approaches are fast, produce high quality solution, and can operate in large and cluttered environments [16]–[19]. However, GCS for multi-agent is under-explored, and how to handle time-dependent constraints for agent-agent collision avoidance in MAMP remains a challenge. Recent work makes attempts to adapt GCS to a space-time domain to solve MAMP, but they either result in unbounded suboptimal solutions [20], or has limited scalability [21].

Finally, trajectory optimization solves similar multi-agent planning problems by planning the motion of multiple agents with dynamics [22]–[26]. However, trajectory optimization often relies heavily on initialization, and can get trapped by local minimum or even fail to find a feasible solution in cluttered environments. Local collision avoidance strategies iteratively plan and replan the motion locally around the robots, so that the robots avoid collision with each other in a reactive and decentralized fashion [27]–[29]. These approaches can readily scale to a large number of robots, but provide no solution quality guarantee due to their myopic local coordination strategy. Some other work seeks to combine different techniques together, such as search and sampling [30], [31], search and optimization [32] to bring together the benefits of different classes of methods.

B. Contributions

This paper introduces a Mixed Integer Conic Programming (MICP) approach for MAMP under discretized time steps. This method does not require discretization of the workspace and can provide near-optimal solutions within a finite runtime limit. Compare to aforementioned methods, our method focuses on providing posterior bounds by handling speed limit constraints and minimizing Euclidean distance travelled by the agents, which were not investigated before.

We compare MICP with a recent sampling-based method, KCBS [30], as well as different formulations, MILP [3] and MINLP. The experiments include several non-trivial settings with up to 10 agents. The results show that our approach can find solutions in challenging cluttered space where the KCBS fails. In other settings, the solutions found by our approach are around 8 times closer to the true optimum than the ones found by the KCBS when given the same amount of runtime. Our MICP demonstrates balanced performance compared to other formulations; it is much more scalable than MINLP and offers better solution quality than MILP. We also showcase the use of our approach on multiple drones under a motion capture system in non-trivial cluttered environments (Fig. 1d).

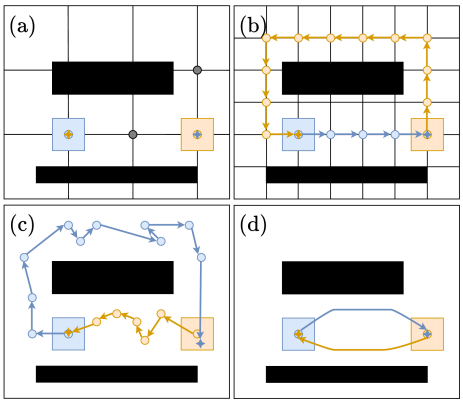


Fig. 2: An illustration of Multi-Agent Motion Planning. The two square agents need to move in a collision-free manner among obstacles. The goal of each agent coincides with the start position of the other agent. (a) Search methods with too coarse a discretization can lead to no feasible solution. (b) An optimal solution in the discretized representation can be highly sub-optimal in the continuous space. (c) Sampling-based method may return sub-optimal solutions. (d) Our MICP based method seeks to directly plan in the continuous space to find high quality solutions.

II. Problem Statement

Let $\mathcal{W} \subset \mathbb{R}^2$ denote a bounded 2D workspace. For any point $w \in \mathcal{W}$, let $w(x)$ and $w(y)$ denote the x and y coordinate of w. We assume the agents and static obstacles are convex polygons. A polygon P is defined by its vertices $V(P) = [w_0^P, w_1^P, \cdots, w_k^P]$,[1] where w_i^P is called the ith (corner) vertex of P and (w_i^P, w_{i+1}^P) is the ith edge. All vertices in $V(P)$ are in *counter-clockwise* order. Let ∂P denote the boundary of the polygon P. Without causing confusion, from now on, we always use P to represent the interior of the polygon, i.e., an open set that excludes its boundary. The reference point of a polygon P is its centroid, denoted as $r(P)$:

$$r(P) = \frac{\sum_{w \in V(P)} w}{|V(P)|} \tag{1}$$

Let $\mathcal{O} = \{o_1, o_2, \ldots, o_s\}$ denote the set of static obstacles. Let the index set $I = \{1, 2, \ldots, n\}$ denote a set of n agents. At any time t, let r_t^i denote the reference point of agent i at t, and let $A_t = \{A^1(r_t^1), \ldots, A^n(r_t^n)\}$ denote the set agent polygons at t, where $A^i(r_t^i)$ denotes the polygon of the ith agent when the reference point is at $r_t^i \in \mathcal{W}$. Additionally, $A^i(r_t^i)$ is simply referred to as A_t^i when there is no confusion.

Agents can move in any direction. Let $s^i, g^i \in \mathcal{W}$ denote the start and goal position of the agent i. All the agents share

[1]We use bracket to indicate that $V(P)$ is an ordered list of vertices.

the same speed limit, denoted as $V_{max} \in \mathbb{R}^+$, and share the same global clock. Each agent starts its motion at the time $t = 0$ from s^i and ends its motion at g^i at a time that is no later than $T_{max} \in \mathbb{R}^+$, where T_{max} is called the *Time Constraint*.

Let $\tau^i : [0, T_{max}] \to \mathcal{W}$ denote the trajectory of agent $i \in I$, where $\tau(0) = s^i$ and $\tau(T_{max}) = g^i$. The cost of a trajectory $c(\tau^i) = |\tau^i|$ is defined as its length.

Problem 1 (TB-MAMP): The goal of the Time Bounded Multi-Agent Motion Planning (TB-MAMP) problem is to find a trajectory τ^i for each agent $i \in I$ such that for any $t \in [0, T_{max}]$, (i) each trajectory τ^i is collision free with respect to the static obstacles

$$\forall o \in \mathcal{O},\ A^i(\tau^i(t)) \cap o = \emptyset \tag{2}$$

(ii) there is no agent-agent collision along the trajectories for each pair of agents

$$\forall i \in I, \forall j \in I \backslash \{i\},\ A^i(\tau^i(t)) \cap A^j(\tau^j(t)) = \emptyset \tag{3}$$

and (iii) the sum of individual trajectory cost $\sum_{i \in I} c(\tau^i)$ reaches the minimum.

Remark 1: Arrival time and trajectory length are two common optimization objectives in the MAMP literature, used by different methods. Search-based methods usually minimize the sum of arrival times at the goals [6]–[8], while sampling-based methods usually minimize the sum of trajectory lengths [13], [14]. Some recent papers also seek to combine both objectives as a multi-objective planning problem [33]. This work seeks to minimize the trajectory lengths subject to a time constraint.

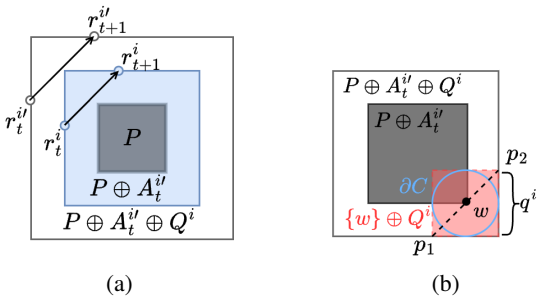
III. Mixed Integer Conic Program (MICP)

Our formulation discretizes the time dimension into a finite number of time steps. Let $\delta t \in \mathbb{R}^+$ denote the time unit, and let $T = \{0, 1, 2, \ldots, m\}$ denote the index set of a sequence of time steps, where $m = T_{max}/\delta t$. We first describe how to represent obstacles to ensure collision avoidance at any time step and during the transition between time steps. Then we introduce the speed limit constraint of the agents and the objective function to be minimized. Finally, we summarize the entire MICP formulation.

A. C-Space Obstacles

Definition 1: Given an agent $i \in I$ and polygon P that may potentially collide with i. Let $\mathcal{C}^i(P)$ denote the C-space (configuration space) obstacle of P to the agent i, i.e., $\forall p \in \mathcal{C}^i(P), A^i(p) \cap P \neq \emptyset$.

In this paper, P is a polygon corresponding to either a static obstacle or another agent as explained later. When we say agent i does not enter $\mathcal{C}^i(P)$, we mean agent i's reference point r^i_t does not enter $\mathcal{C}^i(P)$. To compute $\mathcal{C}^i(P)$, let $P_1 \oplus P_2$ denote the Minkowski sum of polygons P_1 and P_2, i.e.,$P_1 \oplus P_2 = \{w_1 + w_2 \mid w_1 \in P_1, w_2 \in P_2\}$. Let P' denote a polygon that moves P to the origin $(0, 0)$ (i.e., the reference point of P is moved to the origin), and then reflects about the origin, i.e., $P' = \{-w + r(P) \mid w \in P\}$. For example, $A^{i\prime}_t$ means the polygon of agent i is first moved to the origin and then reflected about the origin.



(a) (b)

Fig. 3: **(a)** illustrates the region of $P \oplus A^{i\prime}_t$ (blue) and $P \oplus A^{i\prime}_t \oplus Q^i$ (white) of an obstacle P (grey) in the configuration space of an agent i. Here, $r^{i\prime}_t$, $r^{i\prime}_{t+1}$ and r^i_t, r^i_{t+1} indicate the reference points of agent i with and without the buffer region respectively. It shows that, without buffer region Q^i, the agent may enter $P \oplus A^{i\prime}_t$ during the transition, resulting in a inter-sample collision. **(b)** The larger rectangle represents the C-space obstacle $\mathcal{C}^i(P)$, and the smaller gray rectangle represents the region $P \oplus A^{i\prime}_t$. Given any $w \in P \oplus A^{i\prime}_t$, the red square is the point set $\{w\} \oplus Q^i$, and it contains a circle C centered at w with diameter q^i. Any segment $\overline{p_1 p_2}$ that ending at the border and passes w must be no less than q^i, which exceed speed limit.

To avoid collision at each discretized time step, let $P \oplus A^{i\prime}_t$ represent the C-space obstacle, which is a region generated by inflating obstacle polygon P with respect to the agent polygon A^i_t. It ensures collision avoidance between the agent i and the obstacle P at any discretized time step $t \in T$ by restricting that agent i does not enter $P \oplus A^{i\prime}_t$ [34].

To avoid collision during the transition between two subsequent time steps t and $t+1$, an additional buffer region is needed when inflating the obstacles. Let v^i_P denote the maximum relative speed of agent i to P:

$$v^i_P := \begin{cases} V_{max}, & \text{if } P \in \mathcal{O} \text{ (P is a static obstacle)} \\ 2V_{max}, & \text{otherwise (P is another agent)} . \end{cases} \tag{4}$$

Let $Q^i = q^i \times q^i$ denote a square with side length q^i, centered at the origin $(0, 0)$, using the origin as its reference point, where $q^i = v^i_P \delta t$. It ensures that the agent i does not collide with P during the transition between t and $t+1$, as long as at t, the agent i does not enter:

$$\mathcal{C}^i(P) = P \oplus A^{i\prime}_t \oplus Q^i, \tag{5}$$

which is the corresponding C-space obstacle after considering the buffer region Q^i. Fig. 3a provides an illustration. Fig. 3b illustrates that $q^i = v^i_P \delta t$ is sufficient enough to ensure collision-free motion, theoretically proof is skipped due to the space limit.

B. Formulation

The trajectory of each agent can be represented by a path $\pi^i = [r^i_0, r^i_1, \cdots, r^i_m] \subset \mathcal{W}$, and the cost of a path is defined

as the length of the path: $c(\pi^i) = \sum_{t \in T \setminus \{m\}} \|r_t^i, r_{t+1}^i\|$.

1) Start and Goal Constraints: The first and last point of each path must coincide with the start and goal of the agent:

$$\begin{aligned} r_0^i &= s^i, i \in I \\ r_m^i &= g^i, i \in I \end{aligned} \tag{6}$$

2) Speed Constraint: Let auxiliary variables $l(x_t^i), l(y_t^i) \in \mathbb{R}$, which are real numbers, either positive or negative, denote the amount of translation along the x-axis and y-axis respectively from time step t to $t+1$. Let $l_t^i \in \mathbb{R}^+$ denote the traversed distance in the workspace. For all $i \in I, t \in T \setminus \{m\}$:

$$\begin{aligned} l(x_t^i) &= r_{t+1}^i(x) - r_t^i(x) \\ l(y_t^i) &= r_{t+1}^i(y) - r_t^i(y) \\ l(x_t^i)^2 + l(y_t^i)^2 &\leq l_t^{i2} \\ l_t^i &\leq V_{max} \delta t \end{aligned} \tag{7}$$

Constraint $l(x_t^i)^2 + l(y_t^i)^2 \leq l_t^{i2}$ is a quadratic cone defined over $l_t^i, l(x_t^i), l(y_t^i)$.

3) Agent-Obstacle Collision Avoidance: Given an agent i, an obstacle $o \in \mathcal{O}$ and a time step t. Recall that $\mathcal{C}^i(o)$ is the C-space obstacle of o to agent i. To simplify the notation, let $P = \mathcal{C}^i(o)$ for this sub-section. Let a_k^P denote the normal vector of the kth edge of polygon P pointing outward of the P, and let $B = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ denote a matrix helping represent the normal vector of an edge. Then, for each vertex of P ($\forall w_k^P \in V(P)$):

$$a_k^P = B(w_{k+1}^P - w_k^P) \tag{8}$$

Here, the index of a polygon's vertices and edges follows modular arithmetic, i.e., $w_{i+1}^P = w_{(i+1) \bmod |V(P)|}^P$, so that Eq. (8) is well defined for the last vertex $k = |V(P)| - 1$. Then, any point $r \in \mathcal{W}$ not in P must be contained in at least one of the half plane related to an edge of P Fig. 4 provides an illustration when P is a rectangle. In other words, point r must satisfy at least one of the following constraints:

$$\begin{cases} (a_0^P)^T \cdot (r - w_0^P) \geq 0, \text{or} \\ (a_1^P)^T \cdot (r - w_1^P) \geq 0, \text{or} \\ \vdots \\ (a_{|V(P)|-1}^P)^T \cdot (r - w_{|V(P)|-1}^P) \geq 0 \end{cases} \tag{9}$$

Here, each constraint in Eq. (9) encodes a half plane corresponding to an edge of P. The "or" relationship in Eq. (9) can be expressed by *big-M* constraints. Let $h_{k,t}^{i,o} \in \{0,1\}$ denote auxiliary variables, indicating whether a constraint is active or inactive. We now replace r in Eq.9 with r_t^i the position of agent i at a time step t. Then, for all $i \in I, o \in O, t \in T$:

$$\begin{aligned} \forall w_k^P \in V(P) : (a_k^P)^T \cdot (r_t^i - w_k^P) &\geq -M(1 - h_{k,t}^{i,o}), \\ \sum_k h_{k,t}^{i,o} &= 1, \end{aligned} \tag{10}$$

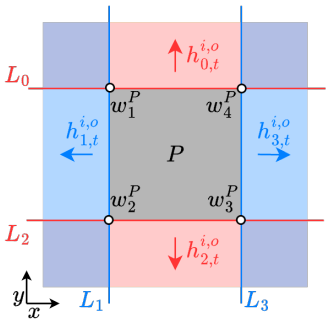


Fig. 4: Rectangle $P = \mathcal{C}^i(o)$ is a C-space obstacle of $o \in \mathcal{O}$ to the agent i. For each $k \in \{0,1,2,3\}$, L_k is a line that defines a subspace, and $h_{k,t}^{i,o}$ is a boolean value that indicates whether r_t^i is in the subspace of L_k. For example, $h_{3,t}^{i,o} = 1$ means the agent is to the right of P, i.e., r_t^i is on the right side of L_3.

where $M \in \mathbb{R}^+$ is a sufficiently large constant number. The constraint $\sum_k h_{k,t}^{i,o} = 1$ ensures only one of the constraints in Eq. (9) is active.

4) Agent-Agent Collision Avoidance: Agent-Agent collision avoidance constraints can be expressed in a similar way, the difference is that, the reference point of the obstacle (i.e., the other agent) is a decision variable, which can be different at each time step. Given agent i and j ($i \neq j$), and a time step t, let $P_t^{i,j}$ denote the C-space obstacle of A_t^j to agent i at t, when agent j is considered the obstacle. $P_t^{i,j}$ is a polygon of fixed shape that translates in the workspace:

$$P_t^{i,j} = \mathcal{C}^i(A_t^j) = \mathcal{C}^i(A_0^j) \oplus \{-s^j\} \oplus \{r_t^j\} \tag{11}$$

Let $a_k^{i,j}$ denote the normal vector of the kth edge of $P_t^{i,j}$ pointing outward of polygon $P_t^{i,j}$. Then, for each vertex $w_{k,t}^{i,j}$ of polygon $P_t^{i,j}$, $k = 1, 2, \cdots, |V(P_t^{i,j})|$:

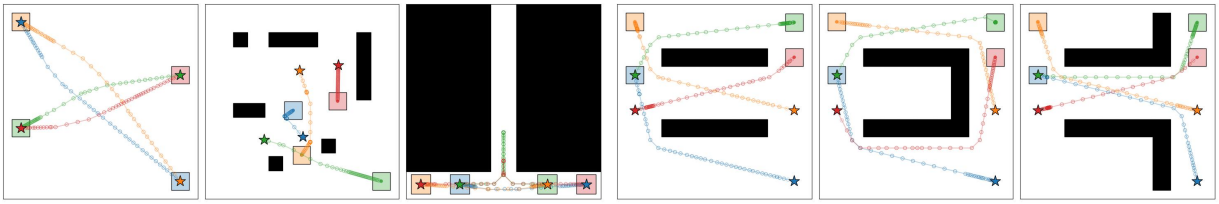
$$\begin{aligned} a_k^{i,j} &= B(w_{k+1,t}^{i,j} - w_{k,t}^{i,j}) \\ &= B((w_{k+1,0}^{i,j} - s^j + r_t^j) - (w_{k,0}^{i,j} - s^j + r_t^j)) \\ &= B(w_{k+1,0}^{i,j} - w_{k,0}^{i,j}) \end{aligned} \tag{12}$$

This equation shows that the normal vector of each edge of polygon $P_t^{i,j}$ remains constant at any time steps, which is not a surprise since all agents can only translate at any time, and translation does not change the direction of the normal vectors of each agent polygon.

Similar to the Eq. (10), let $h_{k,t}^{i,j} \in \{0,1\}$ denote auxiliary variables for the *big-M* constraints, then $\forall i, j \in I, i < j, t \in T, \forall w_{k,t}^{i,j} \in V(P_t^{i,j})$:

$$\begin{aligned} (a_k^{i,j})^T \cdot (r_t^i - (w_{k,0}^{i,j} - s^j + r_t^j)) &\geq -M(1 - h_{k,t}^{i,j}), \\ \sum_k h_{k,t}^{i,j} &= 1, \end{aligned} \tag{13}$$

Since $a_k^{i,j}$ is a constant that is independent from any decision variable, Eq. (10) is a linear combination of r_t^i and r_t^j.



(a) Empty (b) Random (c) Narrow (d) Parallel (e) Wall (f) Corridor

Fig. 5: Test environments used in our experiments. Black rectangles are obstacles, squares are agents, solid stars are their goals, and hollow circles indicate their positions at each time step.

Notation	Domain	Meaning
r_t^i	$\mathcal{W}$	Reference point of i at time step t
l_t^i	R^+	Travelled distance of i from time step t to $t+1$
$h_{k,t}^{i,o}, h_{k,t}^{i,j}$	$\{0,1\}$	Auxiliary variables for *big-M* constraints

TABLE I: Decision variables in the MICP formulation. The subscripts and superscripts take value: $\forall i \in I, t \in T, o \in \mathcal{O}$.

5) MICP Formulation: We summarize all the decision variables $\mathcal{D}$ in Table I, and the full MICP formulation is:

$$\min_{\mathcal{D}} \quad \sum_{i \in I, k \in T} l_k^i$$
$$\text{subject to} \quad (6), (7), (10) \text{ and } (13)$$

Eq. (6) are linear constraints. Eq. (7) are linear constraints and second order conic constraints for any $i \in I, k \in T$. Eq. (10) and (13) are linear combinations of decision variables. As a result, the entire formulation is a Mixed Integer Conic Program (MICP).

Remark 2: The lower bound from the aforementioned MICP may not be the lower bound of our TB-MAMP problem, due to the buffer region Q^i in Eq (5). To certify the solution quality, we can relax the MICP formulation by setting $Q^i = \emptyset$, which ignores collision avoidance during the transition. This allows us to obtain a posterior lower bound from the lower bound of the relaxed formulation.

IV. Experimental Results

This section begins with a description of the experimental settings and then reports the results of three experiments. In the first experiment, we provide an overview of all baseline methods, highlighting the advantage of MICP on solution quality. The second experiment demonstrates the effectiveness of MICP in various environment types and inspects the solution process. The third experiment demonstrates the use of our approach on multiple drones under a motion capture system. Our code is publicly available[2].

A. Test Settings

1) Test Instance Generation: We create 6 workspaces (also called environments) of size 10×10. Figure 5 shows an example of four agents in each workspace. In the *Empty* and *Narrow*, the agents need to swap their locations, i.e., each agent's goal is the starting location of another agent. These two environments describe the scenarios where all single-agent shortest paths cross at the same position within an open and narrow area, respectively. The shortest path of each agent is simply the straight line between its start and goal location. We intend to use these environments to evaluate our method's performance for coordination in cluttered spaces. In the *Random* environment, static obstacles and agents' start and goal locations are randomly sampled from the workspace.

In the *Parallel, Wall* and *Corridor* environments, the agents are divided into two groups, where each group starts at one side of the workspace and moves towards the opposite side. We intent to use these environments to evaluate our method under various workspace topologies. Specifically, in *Parallel*, the obstacles in the middle form three corridors. The *Wall* involves an additional blockage at the central corridor, forcing agents to avoid the obstacle, and the agents thus have to have more interaction. The *Corridor* involves additional blockages in both the top and bottom areas, forcing all agents to navigate through the middle corridor, which further increases the chance of interaction among the agents.

2) Implementation and Baseline: We implement our MICP formulation in Python and use Gurobi 11 as the solver. In our tests, we let Gurobi terminate if the optimality gap (i.e., the gap between the lower bound and the upper bound computed by the solver) is within 5%. All experiments were run on a desktop with a 16-core i7-13700 CPU and 32GB RAM on Ubuntu 22.04. Baseline methods are as follows:

a) KCBS [30]: It follows the workflow of Conflict-Based Search (CBS) [35]. *KCBS* employs a sampling-based method (e.g. RRT [36]) at the low-level to find path for each agent in the continuous space. The original *KCBS* incorporates kinodynamic constraints into the low-level planner, which is not required in this work. In addition, *KCBS* does not consider the T_{max} constraint (Sec. II) while our MICP does. Finally, the original *KCBS* prioritizes the search by the number of collisions rather than the traversed distance and terminates after the first solution is found. We adapted *KCBS* to align with our problem setting by allowing it to keep running CBS-like search for further improvement after finding the first solution. This will persist until either the time limit is reached or all CBS search nodes are explored.

b) MIP and MINLP [3]: We replace the expression of objective function for MIP and MINLP with the sum of L_1-

[2] https://github.com/rap-lab-org/public_MICP_MAMP

norm and L_2-norm respectively

$$L_1 = \sum |r_t^i, r_{t+1}^i| \tag{14a}$$

$$L_2 = \sum \|r_t^i, r_{t+1}^i\|^2, \tag{14b}$$

and the replace the speed limit constraints (i.e., Eq. (7)) with

$$\text{MILP}: \ |r_t^i, r_{t+1}^i| \leq \sqrt{2}\delta t V_{max} \tag{15a}$$

$$\text{MINLP}: \ \|r_t^i, r_{t+1}^i\|^2 \leq (\delta t V_{max})^2, \tag{15b}$$

where Eq. (15a) approximates a circular region with a radius of $\delta t V_{max}$ using a inner square.

We refer to a (problem) instance as a specific number of agents in a specific environment. We allocate a 500 seconds runtime limit (i.e., *TLimit*=500)[3] for each instance, and fix the parameters $\delta t = 0.2, V_{max} = 2, T_{max} = 10$. Each agent is a 1×1 square.

3) Metrics for Comparison: For each instance, we examine two metrics: the shortest distance D^* and the optimality gap Δ. Here, D^* is the total traversed distance of all the agents in the best solution obtained within the runtime limit. It represents an upper bound of an instance. An instance is considered failed if a method cannot find a feasible solution within the runtime limit, and the corresponding D^* is infinite, denoted as `inf`. The gap Δ is evaluated by $\frac{D^* - D_{lb}}{D^*}$, where D_{lb} can be computed as described in Remark 2. Note that the gap Δ is not equivalent to the optimality gap of *MICP*, as the obstacle inflation influences the evaluation on both lower and upper bound in MICP. Both D^* and Δ are better when they are smaller.

B. Experiment 1: Overview of All Methods

This experiment aims to provide an overview of the behaviour of all methods. To do this, we run both methods in Empty and Random environments, and for each environment, we vary the number of agents $n \in \{4, 6, 8, 10\}$. For each given n, we test 10 randomly generated instances.

Table II shows the average solution quality (D^*) of 10 instances for each setting, along with the corresponding number of failures. We can see that, KCBS has significantly worse D^* than other methods, which is expected for a sampling-based method. MILP also has noticeable worse D^*, compared to MICP and MINLP. This is because the L_1-norm objective (i.e., Eq (14a)) does not perfectly align with the Euclidean distance, Fig. 1b shows an example. MINLP and MICP have similar D^*, which are better than other baselines. However, MINLP can only solve up to 8 agents in the Random environment and up to 6 agents in the Empty environment.

Fig. 6 shows the runtime distribution of 10 instances in each setting. It shows that both MICP and MINLP can terminate early within the runtime limit when n is small, as the optimality gap calculated by the Gurobi solver is within 5% which satisfies the termination condition we set for the solver. When n increases, the runtime distribution of MICP and MINLP fall into narrow range towards the runtime limit. MILP exhibits a similar trend but generally has a smaller runtime than all others. This is because a linear model for MAMP is easier to solve. Moreover, MILP uses L_1-norm as the objective, so it cannot properly evaluate the optimality bound in Euclidean distance, leading to a early termination even if it has enough time to further improve the solution. In contrast, since KCBS lacks knowledge of optimality bound knowledge during the search, it continuously improves the solution quality throughout the entire runtime limit unless it has exhausted all the search nodes.

[3]We use "time budget T_{max}" to denote the time constraint in the TB-MAMP problem formulation as described in Sec. II, and use "runtime limit $TLimit$" to denote the runtime limit for each instance.

	n\avg.D^*	KCBS	MICP	MILP	MINLP
Empty	4	55.3	42.4	53.5	**42.1**
	6	94.5	64.0	80.8	**63.2** (#F 2)
	8	134.0	**85.4**	108.3	inf (#F 10)
	10	171.4 (#F 1)	**115.4**	138.3	inf (#F 10)
Random	4	26.7	**19.0**	23.9	19.8 (#F 1)
	6	42.73	**28.0**	34.8	28.8
	8	77.35	**42.1**	52.6	42.7 (#F 2)
	10	100.0	**51.5**	62.1	inf (#F 10)

TABLE II: Average total distance of all agents' trajectories (avg. D^*). The `#F` in bracket shows the number of failures of each method, with all others being successful by default.

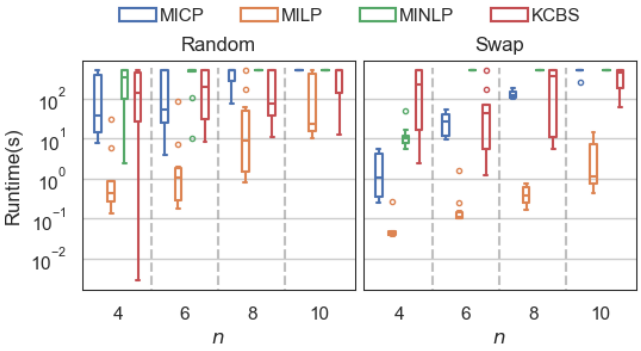


Fig. 6: Runtime distribution of all instances in Random and Swap environments.

C. Experiment 2: Solution Process of MICP and KCBS

This experiment closely examines the solution process of MICP and KCBS. To achieve this, we run MICP and KCBS on a single instance in all environments shown in Fig. 5.

As shown in Table III, our MICP method consistently returns solutions of better quality (i.e., smaller D^*) than KCBS within the same amount of runtime limit. The data on the gap Δ also verifies the solution quality. For example, in the *Empty* environment with 10 agents, our MICP method returns a solution that is at most 5.48% away from the true optimum while the solution returned by KCBS is at most 43.25% away from the true optimum, which is about 8 times larger than 5.48%.

To better understand the method, we examine the solution process of both methods for some representative instances. For each instance, we record the best solution over time.

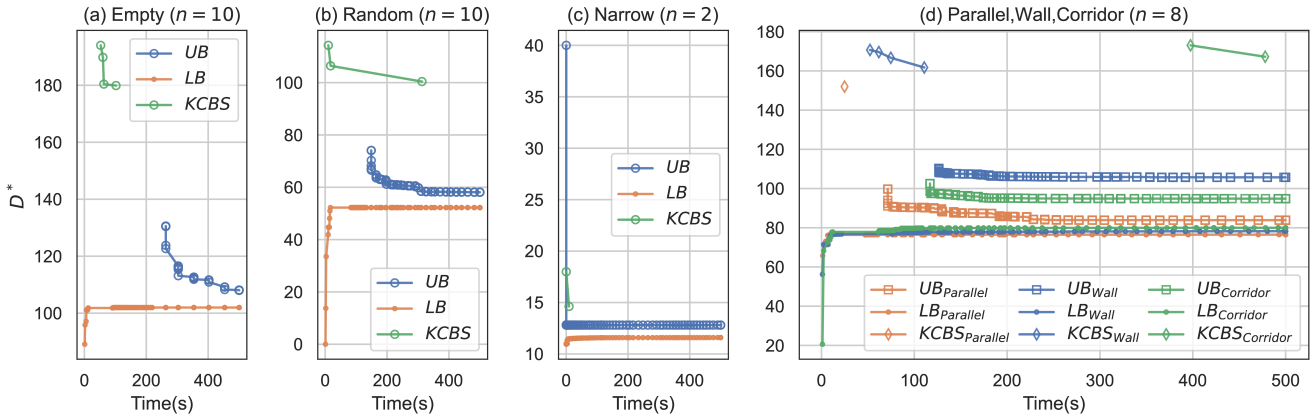


Fig. 7: Each figure shows the solution process on the instance with the largest n that is solved by both methods in each of the six environments.

		D^*		$\Delta(\%)$	
	n	MICP	KCBS	MICP	KCBS
Empty	10	108.01	179.89	5.48	43.25
Random	10	58.11	99.11	5.91	50.54
Narrow	2	12.81	14.62	10.69	21.75
	4	32.41	inf	19.19	NaN
Parallel	4	44.23	62.43	5.20	32.84
	6	61.67	102.84	4.02	42.44
	8	83.86	151.99	8.23	49.37
	10	102.97	168.83	8.09	43.94
Wall	4	50.72	95.17	11.63	52.91
	6	78.11	108.88	21.02	43.34
	8	105.73	161.74	27.13	52.36
	10	135.78	188.40	30.29	49.76
Corridor	4	45.12	71.18	3.77	39.00
	6	66.07	115.56	7.57	47.15
	8	94.85	167.23	17.40	53.15

TABLE III: Result of all instances. Δ is the solution optimality bound.

As shown in Fig. 7, for some instances (Fig. 7(a,b)), the solution quality (D^*) of our MICP improves over time, while for the other instances (e.g. Fig. 7(c,d)), the solution quality has little change as time evolves. In Fig. 7(c), the reason is that D^* is already close to the lower bound, making it hard to further improve. In Fig. 7(d), a possible reason is the cluttered environment and the complicated interaction among the agents, which slows down the solution process of MICP and makes it hard to improve the upper bound.

On the other hand, the lower bounds calculated by MICP plateau after reaching a certain value in all instances, which indicates the difficulty of improving the lower bound. A better lower bound estimation results in smaller optimality gap during the computation, allowing the early termination of some instances within the time budget (e.g. Empty with 4 agents in Table III).

From Fig. 7(d), we can observe that in *Parallel, Wall* and *Corridor*, the upper bound of MICP increases with the growing number of blockages, whereas the lower bound remains similar. This suggests that in environments with narrow corridors where the agents have to interact with each other for collision avoidance, the MICP method may end up with a large optimality gap due to the difficulty on improving lower bound. We also notice that, for both KCBS and MICP, the time to find first solution increases with the increasing number of blockage as well, and the difference in the runtime to the first solution between the two methods becomes smaller. The baseline KCBS returns the first feasible solution earlier than our MICP method in most environments, except for *Corridor*. In both environments, KCBS rarely shows improvement in the remaining time.

D. Experiment 3: Drone Experiment

This experiment demonstrates the use of the proposed method in a multi-drone system. We test two instances in this experiment. For each instance, multiple drones (*CrazyFlie 2.0*) execute their paths planned by our MICP. Readers can find more details from our video.

In the first instance, as shown in Fig. 1, four agents are positioned at the four corners of the workspace, and their goal location is the opposite corner. Meanwhile, the agent in the middle needs to avoid the other passing agents. This instance highlights the advantage of planning in a continuous space. Given the same V_{max} and T_{max}, the D^* in continuous space (Fig. 1a) is 55.52. In contrast, when discretizing the workspace into a grid as shown in Fig. 1c, the solution paths have longer length and D^* is 83.33, which is larger than 55.52.

In the second instance, as shown in Fig. 5c, four agents swap their locations in a narrow corridor. This instance highlights the advantage of MICP in extremely crowded environment, where *KCBS* failed (Table III).

V. Conclusion and Future Work

This paper introduces a novel MICP formulation of MAMP. We show that MICP is valid and verify our approach in various test settings. Although the MICP is based on squared-agents are in the same velocity bound, the formulation can be easily extended to the scenario that agents are in

different shape of convex polygons, and each with its own velocity constraints.

A limitation of MICP formulation is that it does not support non-holonomic agents, which restricts its application scenarios. One possibility to remedy this is to combine MICP with a search-based method that handles non-holonomic constraints. Another future direction for future work is to integrate MICP into a multi-level framework to improve scalability.

VI. ACKNOWLEDGMENT

The main ideas in this paper originated during Dr. Ren and Allen's work at CMU and TAMU respectively. This material is partially based on the work supported by the National Science Foundation (NSF) under Grant No. 2120219 and 2120529. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect views of the NSF. Dr. Ren and Dr. Zhao were also partially supported by the Natural Science Foundation of Shanghai under Grant 24ZR1435900, and the Natural Science Foundation of China under Grant 62403313.

REFERENCES

[1] K. Solovey and D. Halperin, "On the hardness of unlabeled multi-robot motion planning," *The International Journal of Robotics Research*, 2016.

[2] J. K. Johnson, "On the relationship between dynamics and complexity in multi-agent collision avoidance," *Autonomous Robots*, 2018.

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

[4] M. G. Earl and R. D'andrea, "Iterative milp methods for vehicle-control problems," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1158–1167, 2005.

[5] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, "Mixed-integer programming in motion planning," *Annual Reviews in Control*, vol. 51, pp. 65–87, 2021.

[6] L. Cohen, T. Uras, T. S. Kumar, and S. Koenig, "Optimal and bounded-suboptimal multi-agent motion planning," in *SOCS*, 2019.

[7] A. Andreychuk, K. S. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artif. Intell.*, 2022.

[8] Z. Ren, S. Rathinam, and H. Choset, "Loosely synchronized search for multi-agent path finding with asynchronous actions," in *2021 IROS*. IEEE, 2021.

[9] S. Zhou, S. Zhao, and Z. Ren, "Loosely synchronized rule-based planning for multi-agent path finding with asynchronous actions," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 39, Apr. 2025, pp. 14 763–14 770.

[10] L. Wen, Y. Liu, and H. Li, "Cl-mapf: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints," *Robotics and Autonomous Systems*, 2022.

[11] I. Solis, J. Motes, R. Sandström, and N. M. Amato, "Representation-optimal multi-robot motion planning using conflict-based search," *IEEE Robotics Autom. Lett.*, 2021.

[12] J. Chen, J. Li, C. Fan, and B. C. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," in *Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, 2021, pp. 11 237–11 245.

[13] D. Dayan, K. Solovey, M. Pavone, and D. Halperin, "Near-optimal multi-robot motion planning with finite sampling," *IEEE Transactions on Robotics*, 2023.

[14] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, 2020.

[15] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "A general task and motion planning framework for multiple manipulators," in *IROS*, 2021.

[16] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, 2023.

[17] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," *IEEE Transactions on Robotics*, 2024.

[18] K. Sundar and S. Rathinam, "A* for graphs of convex sets," *arXiv preprint arXiv:2407.17413*, 2024.

[19] A. G. Philip, Z. Ren, S. Rathinam, and H. Choset, "A mixed-integer conic program for the moving-target traveling salesman problem based on a graph of convex sets," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 8847–8853.

[20] J. Tang, Z. Mao, L. Yang, and H. Ma, "Space-time graphs of convex sets for multi-robot motion planning," *arXiv preprint arXiv:2503.00583*, 2025.

[21] S. Zhao, A. G. Philip, S. Rathinam, H. Choset, and Z. Ren, "Cb-gcs: Conflict-based search on the graph of convex sets for multi-agent motion planning," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2025.

[22] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, 2014.

[23] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics Autom. Lett.*, vol. 5, no. 2, pp. 604–611, 2020.

[24] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, 2022.

[25] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, *et al.*, "Swarm of micro flying robots in the wild," *Science Robotics*, 2022.

[26] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Distributed nonlinear trajectory optimization for multi-robot motion planning," *IEEE Transactions on Control Systems Technology*, 2022.

[27] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, 1998.

[28] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE ICRA*, 2008, pp. 1928–1935.

[29] S. Ruan, Q. Ma, K. L. Poblete, Y. Yan, and G. S. Chirikjian, "Path planning for ellipsoidal robots and general obstacles via closed-form characterization of minkowski operations," in *Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020.

[30] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *IROS*. IEEE.

[31] 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*, 2019.

[32] B. Senbaslar, W. Hönig, and N. Ayanian, "RLSS: real-time, decentralized, cooperative, networkless multi-robot trajectory planning using linear spatial separations," *Auton. Robots*, vol. 47, no. 7, pp. 921–946, 2023.

[33] Z. Ren, C. Zhang, S. Rathinam, and H. Choset, "Search algorithms for multi-agent teamwise cooperative path finding," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[34] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars, *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[35] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, 2015.

[36] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.