CP-MILP: Mixed Integer Linear Programming for Multi-Agent Motion Planning with Linear Dynamics

Zhongqiang Ren[1], Allen George Philip[2], Shizhe Zhao[1], Sivakumar Rathinam[2] and Howie Choset[3]

***Abstract*—This paper considers a Multi-Agent Motion Planning (MAMP) problem that seeks collision-free paths for multiple agents from their respective start to goal locations among static obstacles, while minimizing the arrival times of the agents with linear dynamics. Among existing approaches such as graph search, sampling, and trajectory optimization, mixed integer programming (MIP) can often find high quality solutions with optimality guarantees. MIP approaches have been investigated extensively and many of them build upon a mixed-integer linear program (MILP) for single-agent, which depends on big-M constraints, a popular technique to formulate conditional constraints. We take the view that some big-M constraints there are unnecessary, and may potentially slow down the computation. This paper thus proposes a new MILP formulation using a perspective technique related to the control terms to bypass some of the big-M constraints, and hence the name Control Perspective MILP (CP-MILP). We analyze the property of our CP-MILP and experimental results show CP-MILP sometimes requires up to near an order of magnitude less runtime to solve to optimality.**

***Index Terms*—Motion and path planning, multi-robot systems, path planning for multiple mobile robots or agents.**

I. Introduction

MULTI-AGENT Motion Planning (MAMP) seeks collision-free paths for multiple agents from their respective start to goal locations among static obstacles. This paper considers agents with linear dynamics while minimizing the arrival times of the agents. This problem is fundamental in robotics and arises in logistics and surveillance. MAMP is challenging to solve to optimality especially when there are many agents [1], [2].

To address the challenge, a variety of approaches were proposed such as search, sampling, and optimization. Among them, mixed-integer programming (MIP) [3]–[7] can often find high quality paths with solution optimality guarantees by planning directly in the continuous spaces without relying on any discretized representation such as graphs or samples.

Manuscript received: June, 22, 2025; Revised September, 8, 2025; Accepted October, 8, 2025.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments. This work was partially supported by the Natural Science Foundation of Shanghai under Grant 24ZR1435900. This work was partially supported by National Science Foundation under Grant No. 2120219 and 2120529. *(Corresponding author: Zhongqiang Ren.)*

[1]Zhongqiang Ren and Shizhe Zhao are with Global College, Shanghai Jiao Tong University. zhongqiang.ren@sjtu.edu.cn; shizhe.zhao@sjtu.edu.cn

[2]Allen George Philip and Sivakumar Rathinam are with Texas A&M University, College Station, TX 77843-3123. y262u297@tamu.edu, srathinam@tamu.edu

[3]Howie Choset is with Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA. choset@andrew.cmu.edu

Digital Object Identifier (DOI):

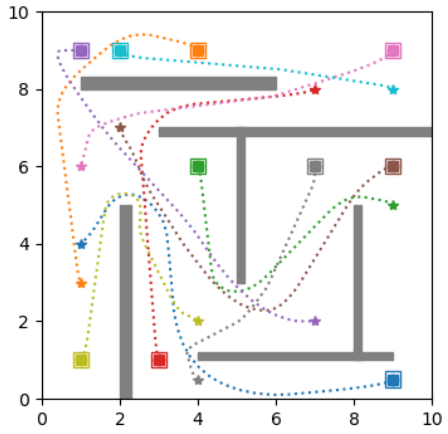

Fig. 1. An illustration of Multi-Agent Motion Planning with rectangle agents and obstacles. Each agent is subject to double integrator dynamics. The dashed curve shows the collision-free solution trajectories with global solution optimality guarantees.

MIP for motion planning has been investigated a lot over the past two decades, and many of them relies on a mixed integer linear programming (MILP) formulation of a single-agent path planning problem [3]. This MILP for single-agent relies on big-M constraints, a popular technique to formulate conditional constraints, to describe arrival times of the agents, which has been further extended to handle more sophisticated problem variants recently [5].

However, we take the view that some of the big-M constraints in this MILP [3], [5] are unnecessary, and may potentially slow down the computation in practice. This paper proposes a new formulation that avoids some of the big-M constraints related to the arrival times by redefining a set of binary variables and using a perspective technique over the control terms. We name our new formulation Control Perspective MILP (CP-MILP), and we show that CP-MILP also ensures solution optimality under a mild assumption that the goal states of the agents are at equilibrium, i.e., after reaching the goal states, the robots can stay at their goal states without any additional control.

We evaluate our CP-MILP against the existing MILP formulations for both single-agent and multi-agent in various maps with single integrator and double integrator dynamics. Our CP-MILP is usually faster to solve to optimality, and the runtime reduction is sometimes up to near an order of magnitude.

II. Related Work

MIP was used to solve various planning problems such as planning start-goal paths [3], traveling salesman problems [7], [8], connectivity maintenance [5]. Of close relevance to this work, the early work on MILP for MAMP uses big-M to encode agent-agent collision avoidance constraints [9], which is also adopted in this work. Subsequent research further uses big-M constraints to describe arrival times of the agents [3], which is avoided in this work via the proposed control perspective technique. Although big-M is convenient to formulate conditional constraints as linear constraints, it is known that [4], in practice, big-M constraints can slow down the computation if M is not properly chosen.

To address the MAMP problem, graph-based methods [10]–[15] usually discretize the workspace into a graph 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. Although these methods can often find high-quality solutions within the graph, how to obtain a graph representation to properly capture the obstacle-free space and the potential agent-agent interaction remains an open challenge.

Without relying on a graph, sampling-based methods [16]–[18] iteratively sample from the state space or the action space of the agents to build a tree that encodes collision-free trajectories. Sampling-based methods often run fast to return a first feasible solution and can then asymptotically refine the solution to optimality as the runtime approaches infinity. However, the solution quality returned by these approaches within a finite runtime can be poor without fine tuning the sampling process, especially when the environment is cluttered with bottlenecks. Recent literature seeks to use a finite number of samples while ensuring solution quality guarantees, but is often limited to small numbers of agents due to the heavy computational burden [16].

Trajectory optimization can solve similar multi-agent problems subject to nonlinear dynamics [19]–[23], but often rely heavily on initialization, and may get trapped into local minima or even converge to infeasible local minima in cluttered environments. Local collision avoidance strategies iteratively replan the motion locally around the agents so as to avoid collision in a reactive and decentralized fashion [24]–[26]. These approaches scale to a large number of robots, but provide no solution quality guarantees due to the myopic local planning. Finally, other work seeks to combine different techniques together, such as search and sampling [27], [28], search and optimization [29], [30] to bring together the benefits of different classes of methods.

Our recent work also seeks to leverage the notion of graphs of convex sets (GCS) [31] and mixed-integer conic programming (MICP) [32] to address the MAMP problem. These GCS and MICP methods can handle Euclidean distance objectives that cannot be directly represented by MILP, yet suffer from limited scalability and often run obviously slower than MILP-based formulations.

Finally, similarly to other MIP approaches [3], [5], [33], CP-MILP in this paper can plan in continuous space without discretization, and provide solution optimality guarantees, while suffering from limited scalability compared to recent search-based planners [14]. These approaches can possibly be fused together to combine their advantages in practice.

III. Problem Description

Let the index set $I_a = \{1, 2, \cdots, N\}$ denote a set of N agents. Each agent $i \in I_a$ is subject to linear dynamics $x^i_{t+1} = A^i_t x^i_t + B^i_t u^i_t$, where $t \in I_t = \{0, 1, \cdots, T\}$ is the index of time steps, $x^i_t \in X^i, u^i_t \in U^i$ are the state and control vectors of the agents at time step t, with $X^i \subseteq \mathbb{R}^{m_x}$ and $U^i \subseteq \mathbb{R}^{m_u}$ denoting the state space and control space, where m_x and m_u are positive integers indicating the dimension of the state space and control space respectively. All agents share a common workspace $W \subseteq \mathbb{R}^2$ with a set of obstacles $O = \{o_1, o_2, \cdots, o_K\} \subseteq W$, where each obstacle o_k is a convex polytope. Each agent $i \in I_a$ is also a convex polytope, and let $R^i(x^i) \subset W$ denote the subset of points that are occupied by the agent i in the workspace when the agent has state x^i. Let X^i_{free} denote the set of states that are collision-free with any obstacles, i.e., $X^i_{free} = \{x^i | R^i(x^i) \cap O = \emptyset\}$. The set of feasible control inputs U^i is a convex polytope in $\mathbb{R}^{m_u}$.

Let $x^i_s, x^i_g \in X^i_{free}$ denote the starting (or initial) state and goal state of agent $i \in I_a$. Let $x^i_{0:T} = \{x^i_s, x^i_1, x^i_2, \cdots, x^i_{T-1}, x^i_g\}$ denote a state trajectory of length $T+1$ of agent $i \in I_a$ connecting x^i_s and x^i_g, and let $u^i_{0:T-1} = \{u^i_0, u^i_1, \cdots, u^i_{T-1}\}$ denote the corresponding control trajectory of length T. A state trajectory is dynamically feasible if $x^i_{t+1} = A^i_t x^i_t + B^i_t u^i_t$ and $u^i_t \in U^i$ for all adjacent states and the corresponding control. Two agents are in collision if $R^i(x^i) \cap R^j(x^j) \neq \emptyset$. Let $|u^i_t|$ denote the L_1-norm of the control vector u^i_t.

Given a state trajectory x^i and the corresponding control trajectory u^i, let $t^i_{arr} \in I_t$ denote the *arrival time* of the agent, which is the earliest time step such that the agent arrives at its goal state x^i_g, and the agent stays at x^i_g for any time step thereafter, i.e., $x^i_t = x^i_g \ \forall \ t \geq t^i_{arr}, t \in I_t$. The maximum of all agents' arrival times is referred to as the makespan of the agents. Let $x = (x^1, x^2, \cdots, x^N), u = (u^1, u^2, \cdots, u^N)$ denote the joint state trajectory and the joint control trajectory of all agents. Let $J(x, u)$ denote the *cost function* of all agents' trajectories, which is the weighted average of the makespan and control efforts by factors $w_1, w_2 \geq 0, w_1 w_2 \neq 0$:

$$J(x,u) = w_1 \max_{i \in I} t^i_{arr} + w_2 \sum_{i \in I_a, t \in I_t \setminus \{T\}} |u^i_t| \tag{1}$$

The goal of the multi-agent motion planning (MAMP) problem considered in this paper is to find a set of collision-free and dynamically feasible trajectories x, u of all agents such that the cost function $J(x, u)$ is minimized.

Remark 1. *Both our CP-MILP and the existing MILP can be readily modified to handle other objectives such as the sum of arrival times, which will be discussed at the end of Sec. IV-D.*

IV. Methods

This section first presents MILP formulations for the single-agent problem. We then extend these formulations to the multi-

agent problem, by further considering the agent-agent collision avoidance constraints.

A. Existing MILP Formulation

In the literature, the single-agent problem can be formulated as an MILP [3], [5] as follows.[1]

$$\min_{x^i_{0:T}, u^i_{0:T-1}, b_{0:T}, h_{k,m,0:T}} \quad w_1 \sum_{t\in I_t} t b_t + w_2 \sum_{t\in I_t\setminus\{T\}} |u^i_t| \quad (2)$$

$$\text{s.t. } x^i_{t+1} = A^i_t x^i_t + B^i_t u^i_t,\ t \in I_t \quad (3)$$

$$x^i_t \in X^i, u^i_t \in U^i \quad (4)$$

$$x^i_0 = x^i_s \quad (5)$$

$$x^i_t \geq x^i_g - M(1-b_t),\ \ x^i_t \leq x^i_g + M(1-b_t) \quad (6)$$

$$a^T_{k,m} x^i_t \leq c_{k,m} + M(1-h_{k,m,t}) + M\sum_{j=0}^{t} b_j,$$
$$\forall k \in I_k, \forall m \in I_m, \forall t \in I_t \quad (7)$$

$$\sum_{m=1}^{M_k} h_{k,m,t} \geq 1, h_{k,m,t} \in \{0,1\},\ \forall k \in I_k, \forall t \in I_t \quad (8)$$

$$\sum_{t\in I_t} b_t = 1, b_t \in \{0,1\} \quad (9)$$

The binary variables $b_t, t \in I_t$ indicate when the agent arrives at its goal. Constraint (9) enforces that $b_t = 1$ for only one time step t, which is the arrival time of the agent. The term $\sum_{t\in I_t} t b_t$ is equal to t if the arrival time is t, and is zero otherwise. Constraints (6) uses the big-M technique such that, with a large enough real number M, the constraints are inactive (i.e., always holds) when $b_t = 0$, and is active when $b_t = 1$. When the constraints are active, they enforce the agent to arrive at the goal state as $b_t = 1$.[2]

Constraints (7) enforce agent-obstacle collision avoidance (Fig. 2 (a)). Let $I_k = \{1, 2, \cdots, K\}$ denote an index set representing the obstacles, and there are K convex polytopic obstacles in total. For each obstacle $o_k \in O, k \in I_k$ (the k-th obstacle), consider its Minkowski difference with the polytope representing the agent. The resulting convex polytope can be represented as the intersection of M_k half-spaces, $\{x | a^T_{k,m} x \leq c_{k,m}\}$, where m ranges from 1 to M_k.[3] Let $h_{k,m,t}$ denote a set of binary variables, indicating whether the agent at time t is on the outer side of the hyper-plane ($h_{k,m,t} = 1$), or on the inner side ($h_{k,m,t} = 0$) with respect to the k-th obstacle. Constraint (8) and the big-M term $M(1 - h_{k,m,t})$ in Constraint (7) require the agent to be on at least one side of the obstacle so that the agent is collision-free. The second big-M term $M\sum_{j=0}^{t} b_j$ in Constraint (7) relaxes (turns off) the constraint if the agent has reached the goal.

[1]Literature [5] summarizes the approach in [3] and is thus recommended as a reference. The paper [5] also considers the connectivity constraints among multi-agent, which is not the focus of this work.

[2]In this paper, all inequalities on vectors should be interpreted component-wise. When taking the sum of a vector and a scalar, the scalar is added to each component in that vector. E.g. on the right hand side of Eq. (6), x^i_g is a vector while $M(1-b_t)$ is a scalar. Their sum is a vector that adds $M(1-b_t)$ to each component of x^i_g.

[3]Here, $a^T_{k,m}$ is a row vector of the same length m_x as the state vector x^i_t, and in $a^T_{k,m}$, only the components, which correspond to the position components in x^i_t, are non-zero.

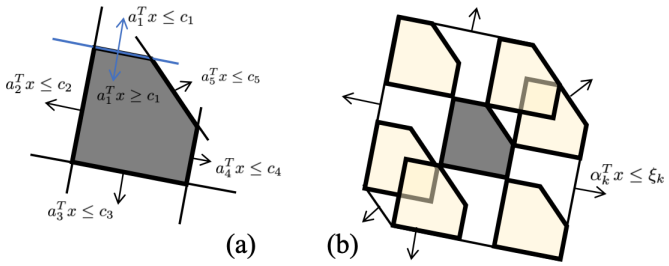

Fig. 2. (a) An illustration of the agent-obstacle collision avoidance constraints. (b) Minkowski difference with the polytope representing the agent.

B. CP-MILP for Single-Agent

Although big-M can help formulate conditional constraints easily, in practice, it can slow down the computation, and the selection of M often requires fine turning. Our CP-MILP seeks to avoid some of the big-M terms in Constraints (6) and (7) that are related to b_t.

We introduce a new set of binary variables $\beta_t, t \in I_t$ to represent the arrival time, instead of using b_t. Intuitively, $\beta_t = 1$ if the agent has not yet reached the goal (i.e., the robot is still active at time t), and $\beta_t = 0$ is the agent has reached the goal at an earlier time step (i.e., the robot is inactive at time t). Thus, as $t \in I_t$ increases, β_t can only drop from one to zero, and we have the following constraint.

$$\beta_t - \beta_{t+1} \geq 0,\ \forall t \in I_t\setminus\{T\},\ \ \beta_t \in \{0,1\},\ \forall t \in I_t \quad (10)$$

Next, let $v^i_t = u^i_t \beta_t$ denote the new control variable, which is the product of the control u^i_t and the binary variable β_t. The term $u^i_t \beta_t$ is bilinear and will be avoided in our formulation. We will remove u^i_t from the decision variables and use v^i_t as the decision variables instead. With the new control variable v^i_t, we can rewrite the dynamic constraint as follows.

$$\begin{aligned} x^i_{t+1} &= A^i_t x^i_t + B^i_t v^i_t, t \in I_t \\ x^i_t &\in X^i, (v^i_t, \beta_t) \in V^i \end{aligned} \quad (11)$$

Here, $V^i = \{(v^i_t, \lambda) | \lambda \geq 0, v^i_t \in \lambda U^i\}$ is the perspective of U^i. Intuitively, this constraint enforces that, when $\beta_t = 0$, the robot has reached the goal and the control must be zero, and when $\beta_t = 1$, the robot has not reached the goal yet, and v^i_t is same as the control u^i_t. We refer to such proposed reformulation based on v^i_t as the *control perspective* technique. The use of Constraints (11) relies on an assumption that the goal state of the robot must be an equilibrium, which is elaborated later in Sec. IV-C. Once the robot reaches its goal x^i_g, β_t and u^i_t are both zero. As a result, we only need to impose constraint $x^i_T = x^i_g$ on the last time step, and Constraint (6) becomes:

$$x^i_T = x^i_g \quad (12)$$

We observe that the relaxation of the obstacle avoidance constraint after the agent reaches the goal is unnecessary, since the robot's goal state must be collision-free. Otherwise (the goal state is not collision-free), there is no feasible solution. We thus rewrite Constraint (7) as follows.

$$a^T_{k,m} x^i_t \leq c_{k,m} + M(1 - h_{k,m,t}) \quad (13)$$

Finally, the objective function in CP-MILP is reformulated based on β_t, v_t^i. We summarize the entire program for CP-MILP as follows.

$$\min_{x_{0:T}^i, v_{0:T-1}^i, \beta_{0:T}, h_{k,m,0:T}} w_1 \sum_{t\in I_t} \beta_t + w_2 \sum_{t\in I_t\setminus\{T\}} |v_t^i| \quad (14)$$

$$\text{s.t. } (10), (11), (5), (12), (13), (8) \quad (15)$$

C. Analysis

This section shows the correctness of our CP-MILP under the following equilibrium goal assumption. We refer to the existing MILP in Sec. IV-A simply as MILP. Let J, J' denote the objective functions in Eq. (2) and Eq. (14) respectively.

Assumption 1. *The goal state x_g^i is an equilibrium for the dynamics, i.e., $x_g^i = A_t x_g^i, \forall t \in I_t$.*

Theorem 1. *When Assumption 1 holds, for any feasible solution to MILP, there is a corresponding feasible solution to CP-MILP and vice versa, and the two solutions have the same objective function values in MILP and CP-MILP.*

Proof. For a feasible solution $S = (x_t^i, u_t^i, b_t, h_{k,m,t})$ to MILP, a corresponding feasible solution $S' = (x_t^i, v_t^i, \beta_t, h_{k,m,t})$ can be constructed for CP-MILP by first copying $x_t^i, h_{k,m,t}$ as is. Then, let k denote the arrival time step, i.e., $b_k = 1$ is the only $b_t, t \in I_t$ variable that is equal to one, and build β_t by setting $\beta_t = 1, t \le k$ and $\beta_t = 0, t > k$. Additionally, due to Assumption 1 and that the arrival time is k, we know $u_t^i = 0, t > k$, and a corresponding set of v_t^i can be built by copying u_t^i. The resulting solution $S' = (x_t^i, v_t^i, \beta_t, h_{k,m,t})$ is a feasible solution to CP-MILP. Similarly, a feasible solution to MILP can be constructed for any given feasible solution to CP-MILP by copying $x_t^i, h_{k,m,t}$ as is, and constructing u_t^i, b_t based on v_t^i, β_t. Finally, by construction of the feasible solutions S, S' to MILP and CP-MILP and the form of the objective function J, J', we know $J(S) = J'(S')$. □

Theorem 2. *When Assumption 1 holds, MILP and CP-MILP have the same optimum, if one exists.*

Proof. We prove it by contradiction. Let $S1$ denote an optimal solution to MILP with objective function value $J(S1)$, and let $S1'$ denote the corresponding solution to CP-MILP with objective function value $J'(S1')$. By Theorem 1, we know $J(S1) = J'(S1')$. Assume that $S1'$ is not an optimal solution to CP-MILP. Then, there must be another feasible solution $S2'$ to CP-MILP whose objective function value $J'(S2')$ is strictly smaller than $J'(S1')$, i.e., $J'(S2') < J'(S1')$. With Theorem 1, a corresponding feasible solution $S2$ to MILP can be built out of $S2'$ with objective function value $J(S2) = J'(S2') < J'(S1') = J(S1)$, which contradicts that $S1$ is an optimal solution to MILP. □

D. CP-MILP for Multi-Agent

Both the existing MILP and our CP-MILP for the single-agent problem can be readily extended to multi-agent by introducing the additional agent-agent collision avoidance constraints [9]. We write down the entire program as follows.

$$\min_{x_t^i, v_t^i, \beta_t, h_{k,m,t}^i, i\in I_a} \sum_{t\in I_t\setminus\{T\}} (w_1\beta_t + \sum_{i\in I_a} w_2|v_t^i|) \quad (16)$$

$$\text{s.t. } x_{t+1}^i = A_t^i x_t^i + B_t^i v_t^i,$$
$$x_t^i \in X^i, (v_t^i, \beta_t) \in V^i, \forall t \in I_t, \forall i \in I_a \quad (17)$$
$$x_0^i = x_s^i, x_T^i = x_g^i, \forall i \in I_a \quad (18)$$
$$a_{k,m}^T x_t^i \le c_{k,m} + M(1 - h_{k,m,t}^i),$$
$$\forall k \in I_k, \forall m \in I_m, \forall t \in I_t, \forall i \in I_a \quad (19)$$
$$\sum_{m=1}^{M_k} h_{k,m,t}^i \ge 1, h_{k,m,t} \in \{0,1\},$$
$$\forall k \in I_k, \forall t \in I_t, \forall i \in I_a \quad (20)$$
$$\beta_t - \beta_{t+1} \ge 0, \ \forall t \in I_t\setminus\{T\},$$
$$\beta_t \in \{0,1\}, \ \forall t \in I_t, \forall i \in I_a \quad (21)$$
$$\alpha_{i,j,m}^T (x_t^i - x_t^j) \le \xi_{i,j,m} + M(1 - h_{t,m}^{i,j})$$
$$\forall t \in I_t, \forall i, j \in I_a, i \ne j \quad (22)$$
$$\sum_{m\in I_m} h_{t,m}^{i,j} \ge 1, \forall t \in I_t, \forall i, j \in I_a, i \ne j \quad (23)$$

Objective (16) and Constraints (17-21) are similar as aforementioned. Here, β_t describes the maximum arrival times of all agents, and the objective function seeks to minimize the sum of all agents' control efforts and the makespan. The binary variables $h_{k,m,t}^i$ related to collision avoidance among static obstacles are now defined for each agent.

In addition, Constraint (22) describes the inter-agent collision avoidance constraint, which is same as in [9]. As shown in Fig. 2 (b), for each pair of convex polytopic agents $i, j \in I_a$, consider their Minkowski difference, which can be described by a polytope enclosed by a set of M hyper-planes $\{x|\alpha_{i,j,m}^T x \le \xi_{i,j,m}\}$, where the index $m = 1, 2, \cdots, M_{i,j}$ indicates the index of the hyper-planes, with $M_{i,j}$ being the total number of hyper-planes. Let $h_{k,m}^{i,j}$ denote a set of binary variables, indicating whether the center of agent j is on the outer side of the hyper-plane $h_{t,m}^{i,j} = 1$, or on the inner side ($h_{t,m}^{i,j} = 0$). Constraint (23) ensures agent j must be on at least one side of the polytope $\{x|\alpha_{i,j,m}^T x \le \xi_{i,j,m}\}$, so that agents i, j are not in collision with each other.

Remark 2. *This formulation minimizes the makespan of the agents. It can be modified to minimize the sum of arrival times by defining $\beta_t^i, t \in I_t$ for each agent $i \in I_a$ as opposed to one common set of $\beta_t, t \in I_t$ for all agents. Specifically, the objective function becomes $\min_{x_t^i, u_t^i, \beta_t, h_{k,m,t}^i, i\in I_a} \sum_{i\in I_a}\sum_{t\in I_t\setminus\{T\}}(w_1\beta_t^i + w_2|v_t^i|)$. In addition, add the superscript i in β_t at all places in the formulation. Then, the constraint $(v_t^i, \beta_t^i) \in V^i, \forall t \in I_t$ in (17) ensures that only after an agent has reached its goal at some time step $t \in I_t$, the control of that agent becomes zero, and other agents $j \ne i$ are unaffected by β_t^i.*

V. Experimental Results

We evaluate our new CP-MILP against the existing MILP (i.e., the baseline) for both single-agent and multi-agent in various maps with single integrator and double integrator

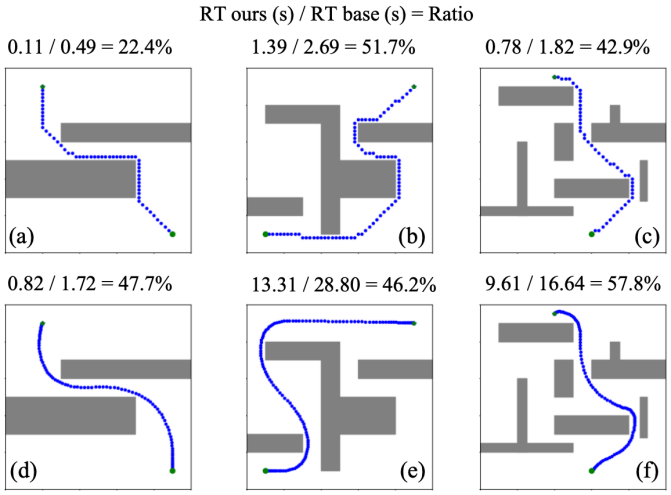

Fig. 3. Results for single-agent tests. The agent has single integrator dynamics in (a-c), and double integrator dynamics in (d-f). The numbers above each figure show: the runtime of CP-MILP/ the runtime of the baseline = the runtime ratio.

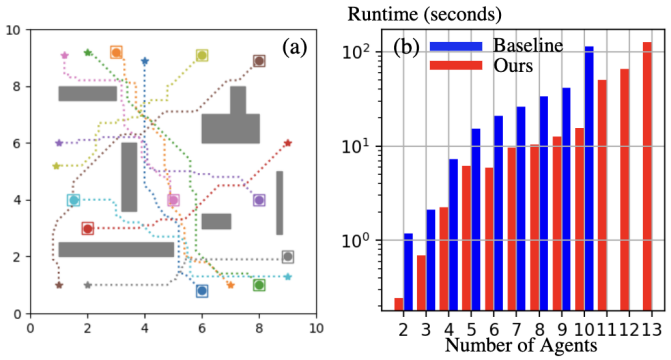

Fig. 4. Results for multi-agent tests with single integrator dynamics.

dynamics on a Macbook with M2 Pro CPU and 16GB RAM. All agents and obstacles are implemented as rectangles in our tests. The start and goal states of the agents are randomly sampled from the workspace with zero velocity terms for the double integrator, which satisfies the assumption that the goal states are equilibrium. We use Gurobi 11.0.3 as the underlying solver for all formulations.

A. Single-Agent Planning

We first plan for a single agent using the instances as shown in Fig. 3 with single integrator and double integrator dynamics in 2D environments with time horizon $T = 100$ (i.e., 100 time steps). Both formulations are solved to optimality and the solution costs always match each other. The numbers in Fig. 3 show the runtime ratio of our CP-MILP against the baseline, and our CP-MILP is always faster than the baseline in these tests. In Fig. 3 (a), our CP-MILP takes only around a quarter of the runtime needed by the baseline.

B. Multi-Agent Planning

We then test multi-agent planning and begin with single integrator dynamics with horizon $T = 100$.

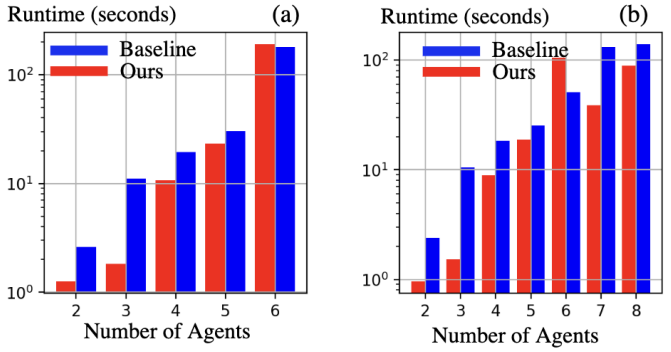

Fig. 5. Results for multi-agent tests with double integrator dynamics with 5% optimality gap (a) and 10% optimality gap (b).

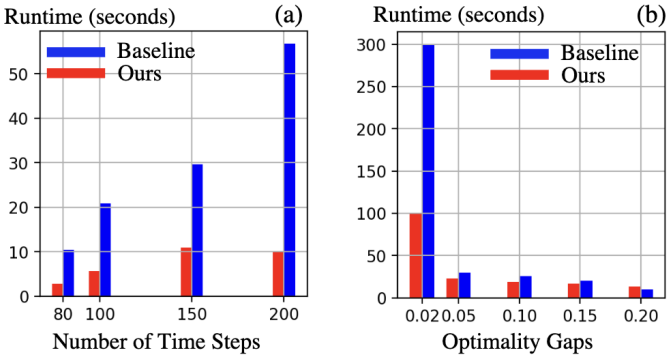

Fig. 6. (a) Results for multi-agent tests with varying planning time steps T. (b) Results for multi-agent tests with varying optimality gaps.

1) Single Integrator Dynamics: The test instance is shown in Fig. 4 (a). We vary the number of agents N from 2 to 13. As shown in Fig. 4 (b), our CP-MILP needs shorter runtime to solve to optimality than the baselines. Note that, the vertical axis is in log scale, and when $N = 10$, our CP-MILP takes near an order of magnitude less runtime than the baseline. We set a runtime limit of 5 minutes. The baseline times out after $N > 10$ and is thus omitted, while our CP-MILP can handle up to 13 agents.

2) Double Integrator Dynamics: We then switch to double integrator dynamics and the test instance is shown in Fig. 1. We vary the number of agents N from 2 to 8. Due to the difficulty of the problem, both formulations take long time to solve to optimality, we therefore set an optimality gap of 5% and 10% for the Gurobi solver in this test when solving both formulations. We will discuss the influence of varying optimality gaps in the following sections. As shown in Fig. 5, when the gap parameter is 5%, there is no obvious advantage of one formulation against the other, and sometimes (e.g. $N = 3$) our CP-MILP takes even more runtime than the baseline. When the gap parameter is 10%, our CP-MILP often runs faster, sometimes up to half an order of magnitude, than the baseline. However, there is still a case ($N = 6$ in Fig. 5(b)) where our approach is slower than the baseline.

The advantage of CP-MILP over the baseline is less obvious for double integrator than single integrator. A possible reason is that the double integrator has more state dimension than

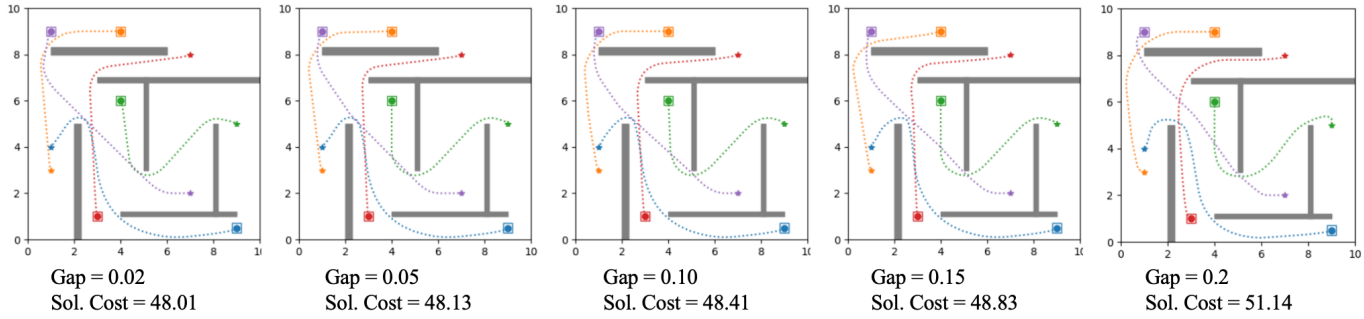

Fig. 7. Visualization of trajectories and their costs for different optimality gaps. Slight difference can be observed for the orange and the green agents.

the single integrator, which leads to more decision variables and more constraints, corresponding to a larger and more complicated feasible domain. As a result, the big M constraints bypassed by CP-MILP become less significant than those for a single integrator.

C. Influence of Parameters

This section evaluates the influence of the parameters on the formulations. We first test with different planning time horizons and then different optimality gaps.

1) Planning Horizon: Although both formulations are able to minimize the arrival times by optimizing over the binary variables b_t or β_t, the choice of planning horizon, i.e., the number of time steps T, may influence the runtime in practice. We fix the number of agents $N = 6$, use single integrator dynamics in this test, and vary the time steps T. As shown in Fig. 6(a), where the vertical axis is on a linear scale, as T increases, both formulations have more binary variables b_t, β_t and take longer runtime to solve to optimality.

2) Optimality Gaps: When using the Gurobi solver, one can set the gap parameter between the upper bound and the lower bound at termination, to let the solver terminate earlier, as long as a bounded sub-optimal solution is found. We tested different gap parameters with both formulations using a double integrator with $N = 5$ agents. As shown in Fig. 6(b), where the vertical axis is in linear scale, increasing the gap can obviously speed up the solution process of both formulations. As the gap varies, neither of the formulations consistently outperforms the other, which is aligned with the results in Fig. 5 for the double integrator dynamics.

Fig. 7 shows the trajectories of the agents and the solution costs at termination for different gaps. By tightening the optimality gap, the runtime increases a lot while the solution costs decrease slowly. Slight difference can be observed along the trajectories of the orange and green agents among those subfigures. It indicates that, in practice, solving the formulations with a moderate gap parameter should yield good enough solution trajectories within relatively short runtime.

VI. Conclusion and Future Work

This paper proposes a new MILP formulation called CP-MILP for single-agent and multi-agent motion planning problems that minimizes control efforts and arrival times. CP-MILP builds upon an existing MILP formulation, redefines a set of binary variables, and uses a perspective technique over the control terms to avoid some of the big-M terms. The paper shows that the proposed CP-MILP has the same optimum, if one exists, as the existing MILP as long as the goal states of all the agents are at equilibrium. As verified by the experimental results, CP-MILP often takes less runtime to solve to optimality, especially for systems with simple dynamics (such as single integrator). For systems with more complicated dynamics, the runtime benefits become less obvious. For future work, one can consider further extending CP-MILP with more sophisticated constraints such as connectivity maintenance [5], or combining the proposed CP-MILP with other MIP techniques to avoid inter-state collision [33], which is ignored in this work.

Acknowledgements

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.

References

[1] K. Solovey and D. Halperin, "On the hardness of unlabeled multi-robot motion planning," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1750–1759, 2016.

[2] J. K. Johnson, "On the relationship between dynamics and complexity in multi-agent collision avoidance," *Autonomous Robots*, vol. 42, no. 7, pp. 1389–1404, 2018.

[3] A. Richards and J. P. How, "Robust variable horizon model predictive control for vehicle maneuvering," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 16, no. 7, pp. 333–351, 2006.

[4] 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.

[5] R. J. Afonso, M. R. Maximo, and R. K. Galvão, "Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 14, pp. 5464–5491, 2020.

[6] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.

[7] 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)*. IEEE, 2024, pp. 8847–8853.

[8] ——, "A mixed-integer conic program for the multi-agent moving-target traveling salesman problem," in *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, 2025, pp. 492–498.

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

[10] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *SoCS*, vol. 10, no. 1, 2019, pp. 151–158.

[11] 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.

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

[13] Z. Ren, S. Rathinam, and H. Choset, "Loosely synchronized search for multi-agent path finding with asynchronous actions," in *2021 IROS*. IEEE, 2021, pp. 9714–9719.

[14] L. Pan, K. Hsu, and N. Ayanian, "Hierarchical large scale multirobot path (re)planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 5319–5326.

[15] 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.

[16] D. Dayan, K. Solovey, M. Pavone, and D. Halperin, "Near-optimal multi-robot motion planning with finite sampling," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3422–3436, 2023.

[17] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, no. 3, pp. 443–467, 2020.

[18] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "A general task and motion planning framework for multiple manipulators," in *IROS*, 2021, pp. 3168–3174.

[19] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[20] 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.

[21] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, 2022.

[22] 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*, vol. 7, no. 66, p. eabm5954, 2022.

[23] 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*, vol. 31, no. 2, pp. 809–824, 2022.

[24] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.

[25] 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.

[26] 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.

[27] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, 2022, pp. 13 494–13 499.

[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] 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.

[30] Y. Cai and Z. Ren, "Pwto: A heuristic approach for trajectory optimization in complex terrains," in *2024 ICAPS Workshop on Planning and Robotics (PlanRob)*. AAAI, 2024.

[31] S. Zhao, Y. Liu, H. Choset, and Z. Ren, "Mixed integer conic programming for multi-agent motion planning in continuous space," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Accepted*, 2025.

[32] 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 *2025 IEEE International Conference on Automation Science and Engineering (CASE)*, 2025, pp. 2208–2214.

[33] A. Richards and O. Turnbull, "Inter-sample avoidance in trajectory optimizers using mixed-integer linear programming," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 4, pp. 521–526, 2015.