Switched Linear Multi-Robot Navigation Using Hierarchical Model Predictive Control

Chao Huang[1], Xin Chen[1*], Yifan Zhang[1], Shengchao Qin[23*], Yifeng Zeng[2], Xuandong Li[1]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2]School of Computing, Teesside University, Tees Valley, TS1 3BX, UK
[3]Shenzhen University, Shenzhen, China
huangchao@seg.nju.edu.cn, chenxin@nju.edu.cn, s.qin@tees.ac.uk, y.zeng@tees.ac.uk, lxd@nju.edu.cn

Abstract

Multi-robot navigation control in the absence of reference trajectory is rather challenging as it is expected to ensure stability and feasibility while still offer fast computation on control decisions. The intrinsic high complexity of switched linear dynamical robots makes the problem even more challenging. In this paper, we propose a novel HMPC based method to address the navigation problem of multiple robots with switched linear dynamics. We develop a new technique to compute the reachable sets of switched linear systems and use them to enable the parallel computation of control parameters. We present theoretical results on stability, feasibility and complexity of the proposed approach, and demonstrate its empirical advance in performance against other approaches.

1 Introduction

A switched linear system is a special type of hybrid systems, consisting of several modes equipped with linear dynamics (differential/difference equations), and a switching rule specifying how to switch between them. It provides an expressive model for designing robots which embrace complex behaviors [Faust *et al.*, 2016; Bogomolov *et al.*, 2014; Lahijanian *et al.*, 2014]. For example, a robot with many gears which deliver different torques and speeds can be directly modeled by a switched linear system.

The multi-robot navigation problem attracts much attention for being not only academically challenging, but also of practical importance. It drives a group of robots from their initial positions to goal positions without any reference trajectory. A successful navigation should provide three guarantees: converging to the goal positions finally (*stability*), avoiding collision all the time (*feasibility*), and fast computing (*efficiency*).

There have been a number of works treating the navigation problems as path planning problems, where heuristic-search based methods are the most popular and well-developed solutions [Wagner and Choset, 2011; Karaman and Frazzoli, 2011; Janovský *et al.*, 2014]. For robots with discrete state space such as graph, A* based algorithms can quickly find the optimal path while ensuring the stability, and thus are well studied [Koenig and Likhachev, 2002; Wagner and Choset, 2011]. Before adopting it to continuous states, the continuous state space must be discretized in advance. Those methods that deal with continuous state space directly receive much attention in recent years. Optimal reciprocal collision avoidance (ORCA) approach [van den Berg *et al.*, 2011] is proposed to address the collision avoidance problem of robots. It is then combined with the RRT* algorithm [Karaman and Frazzoli, 2011], an extension of RRT (rapidly exploring random tree), for robot navigation in a continuous state space [Janovský *et al.*, 2014]. However neither of them ensures theoretical stability, therefore may easily lead robots to a situation where robots stick at one point as a result of infinite loops or deadlock [Janovský *et al.*, 2014]. To ease treatment, all the above methods assume that once the specified positions are given, all robots can easily compute the corresponding inputs and arrive the positions accurately, thus do not take the concrete dynamics of robots into consideration.

Unfortunately, as is pointed in [Pecora *et al.*, 2012], for robots with complex behaviors, treating the navigation problem as two separated steps: path panning and control inputs computation may cause severe collision issues. That is, due to the absence of robot behavior in path planning, even though the planned path is theoretically safe, in practice it might lead to collisions as the target positions in the given trajectory may be beyond the robots' ability. Therefore, for switched linear robots, due to their intrinsic complexity, the strategy of separation does not fit for handling the navigation problems.

Different from path planning approaches which only consider the trajectory, the switched linear robot navigation control requires to compute control inputs of all robots with consideration of robot dynamics. Model Predictive Control (MPC) is the most popular method whose basic idea is to first compute an input sequence of the next several steps using certain path finding strategy, and then choose the sequence head as the current input. MPCs fall into three categories: centralized MPC (CMPC), distributed MPC (DMPC) and hierarchical MPC (HMPC). CMPC provides theoretical guarantee on feasibility and stability, but suffers from a high computation complexity [Dunbar and Murray, 2002]. DMPCs can hardly ensure the stability under (non-convex) collision avoidance specification [Keviczky *et al.*, 2004; Kuwata and How, 2011; Mercangöz and Doyle, 2007]. Re-

*Corresponding authors: Xin Chen, Shengchao Qin

cently, HMPCs are proposed which balance the merits of both CMPC and DMPC schemes. In HMPC, a central controller determines the next state of every robot while for each robot, there is a distributed controller computing its control parameter driving it to the given state. Among them, the approach proposed in [Cirillo *et al.*, 2014] cannot ensure the dynamic feasibility generated by the central controller and the method proposed in [Huang *et al.*, 2016] replaces constraints of dynamics with reachable sets of robots so that the goal states can be efficiently produced by the central controller, ensuring both stability and feasibility for simple linear dynamical robots.

In this paper, starting from the reachable set based HMPC, we aim to address the navigation problem of robots whose behaviors are described by switched linear systems in continuous space. Here, the key challenge is how to compute the *exact* reachable set of robots in a very *short* time: the accuracy of the reachable sets assures the equivalence between the transformed optimization problem and the original one; if the overhead incurred by the computation of reachable sets is more than the time saving benefitted from parallel computation, it would render HMPC to be less useful. The reachable set computation of general switched systems is a well-known undecidable problem requiring heavy computation efforts [Alur *et al.*, 1995; Henzinger *et al.*, 1995]. We give a novel method to compute the reachable set function by slightly restricting the multi-robot system with a reasonable assumption. The reachable set function is computed in advance off-line, and the concrete reachable set is directly online derived by substituting the current state. Our main contributions are listed as follows:

- We propose a novel HMPC based method to address the navigation problem of switched linear multi-robot using our newly developed technique to compute the reachable set function of switched linear systems.
- We theoretically ensure the feasibility and stability of the proposed method and discuss its complexity;
- We demonstrate the performance of the new HMPC based method in several navigation scenarios and show its advantages empirically by varied comparison.

2 Problem Formulation

Consider a switched linear dynamical multi-robot system *S* containing *N* robots on a plane. Each robot has independent dynamics and inputs along the x-axis and y-axis. Assuming the sampling interval of the *i*-th robot is T_i, robots synchronizes with each other at the end of every *collaboration cycle*, which is the least common multiple of robots' sampling intervals.

Before formally defining the systems and problem, we first introduce the assumption throughout this paper.

Assumption 1. *Each robot stay stationary at a collaboration instant.*

Notably in current works of multi-robot navigation control, in particular heuristic-search methods, forcing robots to stop is a common choice when confronted with conflicting situations [Wei *et al.*, 2014].

Now, we model such a multi-robot system. For the *i*-th robot, the collaboration cycle is K_i times its local control cycle. A state of the *i*-th robot is denoted as $q_i = [x_i, \dot{x}_i, y_i, \dot{y}_i]^T$, where $x_i, \dot{x}_i, y_i, \dot{y}_i$ denote the position and velocity along the x-axis and the y-axis, respectively. Let $q_{i,x} = [x_i, \dot{x}_i]^T$ and $q_{i,y} = [y_i, \dot{y}_i]^T$ be the states along two axes. The dynamics of each robot along two axes are independent. We can model the dynamic along each axis separately as a switched linear system. Let $M_{i,x}$ and $M_{i,y}$ be the number of permissible *linear dynamical modes* along x-axis and y-axis. For the x-axis, the dynamic is modeled as the linear discrete-time time-varying state equality below:

$$q_{i,x}(k, k_i{+}1){=}A_{i,x}\, q_{i,x}(k, k_i){+}B_{i,x}(m_{i,x}(k, k_i))\, u_{i,x}(k, k_i), \tag{1a}$$

$$q_{i,x}(k{+}1, 0){=}q_{i,x}(k, K_i), \quad 0{\le}k_i{\le}K_i{-}1, k{\ge}0 \tag{1b}$$

where k denotes the *k*-th collaboration instant, $k_i{\in}\{0, 1, ..., K_i{-}1\}$ represents the k_i-th local sampling instant of the *i*-th robot, the pair $(m_{i,x}(k, k_i), u_{i,x}(k, k_i))$ is referred as the hybrid control along the x-axis, where:

Switching Control $m_{i,x}(k, k_i) \in \mathcal{M}_{i,x}{\triangleq}\{1, \cdots, M_{i,x}\}$ determines the dynamic mode;

Continuous Control $u_{i,x}(k, k_i) \in \mathcal{U}_{i,x}{\triangleq}[-U_{i,x}, U_{i,x}]$ determines the continuous evolution.

$A_{i,x}$ is the 2×2 coefficient matrix of state $q_{i,x}$ while $B_{i,x}$ is the 2×1 coefficient matrix of $u_{i,x}$ determined by $m_{i,x}(k, k_i)$. Equation (1b) defines the beginning state of one collaboration cycle as the ending state of the last collaboration cycle. Intuitively, Equation (1) indicates that the *i*-th robot can switch to one of the permissible dynamic modes along the x-axis at each sampling instant, then pick an appropriate input under the mode. Following the dynamic model along the x-axis, we model the one along the y-axis in the same way.

$$q_{i,y}(k, k_i{+}1){=}A_{i,y}\, q_{i,y}(k, k_i){+}B_{i,y}(m_{i,y}(k, k_i))\, u_{i,y}(k, k_i), \tag{2a}$$

$$q_{i,y}(k{+}1, 0){=}q_{i,y}(k, K_i), \quad 0{\le}k_i{\le}K_i{-}1, k{\ge}0 \tag{2b}$$

Let $u_i = [u_{i,x}, u_{i,y}]^T$ denote the continuous control of the *i*-th robot, $m_i = (m_{i,x}, m_{i,y})$ denote the switching control. The complete dynamic of the *i*-th robot is:

$$q_i(k, k_i{+}1) = A_i\, q_i(k, k_i){+}B_i(m_i(k, k_i))\, u_i(k, k_i), \tag{3a}$$

$$q_i(k+1, 0) = q_i(k, K_i), \quad 0{\le}k_i{\le}K_i{-}1, k{\ge}0 \tag{3b}$$

where $A_i{=}A_{i,x}{\oplus}A_{i,y}$, $B_i(m_i(k, k_i)){=}B_{i,x}(m_{i,x}(k, k_i)) \oplus B_{i,y}(m_{i,y}(k, k_i))$, and the direct sum operator $\oplus$ for two matrices A and B is defined as:

$$A \oplus B \triangleq \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}.$$

The constraint of switching control is:

$$m_i(k, k_i) \in \mathcal{M}_i \triangleq \mathcal{M}_{i,x} \times \mathcal{M}_{i,y}, \tag{4}$$

which means that the *i*-th robot has $M_{i,x}{\cdot}M_{i,y}$ dynamic modes. The switching control sequence of the *i*-th robot in the *k*-th collaboration cycle is denoted by $\tilde{m}_i(k) \triangleq m_i(k), m_i(k, 1), ..., m_i(k, K_i{-}1)$. The constraint of continuous control is:

$$u_i(k, k_i) \in \mathcal{U}_i \triangleq \mathcal{U}_{i,x} \times \mathcal{U}_{i,y} = [-U_i, U_i]. \tag{5}$$

where $U_i = [U_{i,x}, U_{i,y}]^T$. The continuous control sequence of the *i*-th robot in the *k*-th collaboration cycle is denoted by

$\tilde{u}_i(k) \triangleq u_i(k),\ u_i(k,1),...,u_i(k,K_i-1)$. We refer to the pair $(m_i(k,k_i), u_i(k,k_i))$ as the hybrid control of the *i*-th robot.

For a multi-robot system *S*, let $q(k) \triangleq [q_1(k),...,q_N(k)]$ denote the global state at the *k*-th collaboration instant, $\tilde{m}(k) \triangleq \tilde{m}_1(k),\cdots,\tilde{m}_N(k)$ and $\tilde{u}(k) \triangleq \tilde{u}_1(k),\cdots,\tilde{u}_N(k)$ denote the global switching and continuous control sequence in the *k*-th collaboration cycle respectively.

Assumption 1 is described as *collaboration specification*:

$$\dot{x}_i(k) = \dot{y}_i(k) = 0, \qquad k \geq 0, \tag{6}$$

We define the *collision avoidance specification* requiring that the (infinity norm) distance between any two robots must not be less than a given safety distance d_{safe}:

$$h(q) \leq 0, \qquad k \geq 0, \tag{7}$$

where $h(q) = [h_{i,j}(q_i,q_j)]_{i,j=1}^{N}$,

$$h_{i,j}(q_i,q_j) = \begin{cases} d_{safe} - \|q_{i,pos} - q_{j,pos}\|_\infty, & i \neq j, \\ 0, & i = j. \end{cases}$$

The infinite norm of a vector $x = [x_1,...,x_n]^T$ is defined as $\|x\|_\infty \triangleq \max(|x_1|,...,|x_n|)$.

Following [Huang *et al.*, 2016], we define two core concepts for MPC, namely *feasibility* and *stability*.

Definition 1. *(*Feasibility*) A MPC controller is* feasible *iff for any feasible initial state, the actual state $\{q(t), t\geq 0\}$ computed by the controller at every collaboration instant satisfies the collision avoidance specification $h(q(t))\leq 0$.*

Definition 2. *(*Stability*) A MPC scheme is* stable *iff starting from any feasible initial state, the state sequence $\{q(t)\}_{t=0}^{\infty}$ computed by the MPC controller converges to the goal state.*

Subsequently we formulate the navigation problem:

Problem 1. *Given a multi-robot system* S*, where*

- *the* i*-th robot has the kinematic model described as in (3a) and (3b) with switching control constraint (4) and continuous control constraint (5);*
- *a stationary initial state $q(0) = [q_1(0),...,q_N(0)]^T$ of* S*, where the initial velocity of each robot is zero;*
- *a stationary goal state $q' = [q'_1,...,q'_N]^T$ of* S*, where the goal velocity of each robot is zero;*
- *the collaboration specification and collision avoidance specification are described as in (6) and (7) respectively.*

Determine a control strategy to generate the hybrid control sequence $\{(\tilde{m}_i(k), \tilde{u}_i(k))\}_{k=0}^{\infty}$ for the i*-th robot, $i=1,...,N$, satisfying feasibility and stability.*

Remark 1. *In this work, we consider the obstacle-free navigation scenario. Note that for each robot, all other robots are treated as moving obstacles, and the collision avoidance is described by coordination constraints. With trivial extension, stationary obstacles can be treated similarly. Therefore, this setup does not lose the generality and practicality.*

3 HMPC Framework

In this section, we briefly introduce the basic idea of the HMPC control framework on switched linear multi-robot navigation. At a collaboration instant *t*, the HMPC scheme consists of two successive steps:

Compute the control goals: The central controller collects the current state $q(t)$ and minimizes the *state* cost in a finite horizon H:

$$\left.\begin{aligned} J_c^*(q(t)) \ &\triangleq \min_{q(1|t),\cdots,q(H|t)} \sum_{k=0}^{H-1} l_q(q(k|t)) + l_H(q(H|t)) \\ \texttt{s.t.}\quad & q_i(k+1|t) \in R_i(q_i(k|t)),\ k\geq 0,\ i=1,...,N, \\ & h(q(k|t)) \leq 0, \quad k=1,...,H-1, \\ & q(H|t) \in \tilde{\mathcal{Q}}_f, \quad q(0|t) = q(t), \\ & \dot{x}_i(k|t) = \dot{y}_i(k|t) = 0, \quad k=1,...,H-1. \end{aligned}\right\} \tag{8}$$

where the notation $(\cdot)(k|t)$ denotes the *predictive* value at the (*t*+*k*)-th collaboration instant computed at time t, $R_i(q_i(k|t))$ is the state set which the *i*-th robot can reach at the next collaboration instants from the state $q_i(k|t)$, l_H and $\tilde{\mathcal{Q}}_f$ are user-defined terminal cost and terminal constraint set to ensure stability. l_q and l_u denote the state cost and continuous control cost respectively, which are designed to be positive definite:

$$l_q(q) \begin{cases} = 0, & q = q', \\ > 0, & q \neq q'. \end{cases} \qquad l_u(u) \begin{cases} = 0, & u = 0, \\ > 0, & u \neq 0. \end{cases}$$

Let $q^*(1|t),...,q^*(H|t)$ denote the optimal solution. The first sample $q^*(1|t)$ will be used as the desired states $q(t+1)$ at the instant $t+1$, that is $q(t+1) = q^*(1|t)$. For the vector $q(t+1) = [q_1(t+1),...,q_N(t+1)]^T$, its *i*-th element $q_i(t+1)$ specifies the *control goal* of the *i*-th robot at the collaboration instant $t+1$.

Compute the hybrid controls: The *i*-th robot receives the *control goal* $q_i(t+1)$ and then derives the explicit control:

$$\left.\begin{aligned} J_i^*(q_i(t), q_i(t+1)) &\triangleq \min_{\tilde{m}_i(t),\tilde{u}_i(t)} l_u(\tilde{u}_i(t)) \\ \texttt{s.t.}\quad & q_i(t,k_i+1) = A_i\, q_i(t,k_i) + B_i(m_i(t,k_i))\, u_i(t,k_i), \\ & 0 \leq k_i \leq K_i - 1, \\ & q_i(t+1) = q_i(t,K_i), \\ & m_i(k,k_i) \in \mathcal{M}_i, \quad u_i(k,k_i) \in \mathcal{U}_i. \end{aligned}\right\} \tag{9}$$

Note that this optimization problem is non-convex. It can be convert to mixed integer programming and solved by optimization solvers according to [Richards and How, 2005]. The optimal solution $(\tilde{m}_i^*(t), \tilde{u}_i^*(t))$ is the actual hybrid control sequence for the *i*-th robot in the next collaboration cycle.

This procedure will be repeated at the next collaboration instant $t+1$ based on new measurements of the state $q(t+1)$ until $J_c^*(q(t))$ meets a certain convergence criteria.

Observing HMPC framework, the key lies on how to efficiently compute the reachable set for switched linear systems.

4 Computation of Reachable Set Function

In this section, We propose a novel method for computing the reachable set function $R_i(q_i(t))$ for a switched linear system.

Recall the formal definition of a reachable set function.

Definition 3. *The reachable set function $R_i(q_i(t))$ of the* i*-th robot is defined as the set comprising all states that the* i*-th robot can reach in a collaboration cycle from the state $q_i(t)$. In particular, $RP_i(q_i(t))$ denotes the reachable position set function, which consists all the positions in $R_i(q_i(t))$.*

Consider the switching control sequence $\tilde{m}_i(t)$ in the t-th collaboration cycle. Since $m_i(t,k_i)$ has $M_{i,x}\cdot M_{i,y}$ choices at

any k_i-th local sampling instant, $\tilde{m}_i(t)$ can have $(M_{i,x}{\cdot}M_{i,y})_i^K$ cases. And the reachable set under a certain switching control sequence is a polytope [Gritzmann and Sturmfels, 1993]. Hence a naive way to compute $R_i(q_i(t))$ is to directly compute the union of the $(M_{i,x}{\cdot}M_{i,y})^{K_i}$ convex polytopes. However, this approach faces a serious problem: if the representation of reachable set function includes the intermediate variables, its computation becomes time consuming and the optimization (9) solved in central controller degenerates to a CMPC. It indicates two key requirements of reachable set function.

- **Requirement 1**: The computation of reachable set should be efficient by reachable set function;
- **Requirement 2**: The representation of reachable set function should hide the intermediate variables between two collaboration instants.

In the following, we introduce our approach for computing the reachable set function $R_i(q_i(t))$ under Assumption 1. Since the collaboration specification (6) demands $\dot{x}_i(t{+}1) = \dot{y}_i(t{+}1){=}0$, if we obtain the reachable position set $RP_i(q_i(t))$, $R_i(q_i(t))$ can be easily computed as $R_i(q_i(t)){=}\{[x, \dot{x}, y, \dot{y}]^T{:}[x, y]^T \in RP_i(q_i(t)){\wedge}(\dot{x}_i{=}\dot{y}_i{=}0)\}$. Hence the key is to compute $RP_i(q_i(t))$.

Since the dynamic along the x-axis and y-axis is independent, we separately consider the dynamics along two axes. Let $RP_{i,x}(q_i(t))$ and $RP_{i,y}(q_i(t))$ denote the reachable position set along x-axis and y-axis, respectively. For the dynamic along the x-axis, we have the following conclusion.

Lemma 1. *The reachable position set $RP_{i,x}(q_i(t))$ along the x-axis is a closed interval.*

Proof. For a switching control sequence $\tilde{m}_i(t) = \{m_{i,x}(t, k_i)\}_{k_i=0}^{K_i-1}$ along with x-axis, let $RP_{i,x}(q_i(t), m_i(t))$ be the reachable position set along the x-axis with $\tilde{m}_i(t)$. Since $RP_{i,x}(q_i(t), \tilde{m}_i(t))$ is a convex polytope [Gritzmann and Sturmfels, 1993] and is 1-dimensional, $RP_{i,x}(q_i(t), \tilde{m}_i(t))$ is a closed interval. Furthermore, recall that

$$\begin{aligned} q_{i,x}(t+1) &= A_{i,x}\, q_{i,x}(t, K_i-1){+}\\ &\quad B_{i,x}(m_{i,x}(t, K_i-1))\, u_{i,x}(t, K_i-1)\\ &= A_{i,x}(A_{i,x}\, q_{i,x}(t, K_i{-}2){+}\\ &\quad B_{i,x}(m_{i,x}(t, K_i-2))\, u_{i,x}(t, K_i{-}2)){+}\\ &\quad B_{i,x}(m_{i,x}(t, K_i-1))\, u_{i,x}(t, K_i{-}1)\\ &= \cdots\\ &= (A_{i,x})^{K_i} q_{i,x}(t){+}\\ &\quad \sum_{p=0}^{K_i-1} (A_{i,x})^{K_i-p-1} B_{i,x}(m_{i,x}(t,p))\, u_{i,x}(t,p),\\ u_{i,x}(t,p) &\in \mathcal{U}_i, \quad m_{i,x}(t,p) \in \mathcal{M}_{i,x}, p = 0, ..., K_i{-}1 \end{aligned}$$

Obviously, $\{(m_{i,x}(t, k_i), 0)\}_{k_i=0}^{K_i-1}$ is a feasible hybrid input sequence to make $\dot{x}_i(t{+}1){=}\dot{y}_i(t{+}1){=}0$ for any switching control sequence $\tilde{m}_i(t)$. In this case, $q_{i,x}(t{+}1){=}\ (A_{i,x})^{K_i} q_{i,x}(t)$, i.e. $(A_{i,x})^{K_i} q_{i,x}(t) \in RP_{i,x}(q_i(t), \tilde{m}_i(t))$. So the reachable set under each switching control sequence shares a common element $(A_{i,x})^{K_i} q_{i,x}(t)$, i.e. $(A_{i,x})^{K_i} q_{i,x}(t) \in \bigcap_{\tilde{m}_i(t)} RP_{i,x}(q_i(t), \tilde{m}_i(t))$. Then Lemma 1 can be proved by the property of intervals. □

When $q_{i,x}(t)$ is the origin, following Lemma 1, let $RP_{i,x}(0) \triangleq [\min x_i, \max x_i]$ be the corresponding reachable position set along x-axis, where $\min x_i$ and $\max x_i$ are the nearest and farthest distance the *i*-th robot can reach along the x-axis. According to Assumption 1, we have

$$\begin{aligned} RP_{i,x}(0){=}\{x{:}\, \begin{bmatrix} x \\ 0 \end{bmatrix} &= \sum_{p=0}^{K_i-1} (A_{i,x})^{K_i-p-1} B_{i,x}(m_{i,x}(t,p))\, u_{i,x}(t,p),\\ u_{i,x}(t,p) &\in \mathcal{U}_i, m_{i,x}(t,p) \in \mathcal{M}_{i,x}, p{=}0, ..., K_i{-}1\} \end{aligned}$$

It immediately conducts the following corollary on the reachable position set from the above equation.

$$RP_{i,x}(q_i(t)){=}\{x : (\begin{bmatrix} x \\ 0 \end{bmatrix} - (A_{i,x})^{K_i} q_{i,x}(t)) \in RP_{i,x}(0){\times}\{0\}\}.$$

Similarly, we can obtain the same conclusion on the dynamic along the y-axis. Let $RP_{i,y}(0) = [\min y_i, \max y_i]$. These facts indicate that the reachable set $R_i(q_i(t))$ is a hyperrectangle:

Theorem 1. *The reachable set function $R_i(q_i(t))$ of the* i*-th robot is:*

$$\begin{aligned} R_i(q_i(t)){=}\{\begin{bmatrix} x \\ 0 \end{bmatrix} : (\begin{bmatrix} x \\ 0 \end{bmatrix} - (A_{i,x})^{K_i} q_{i,x}(t)) \in RP_{i,x}(0) \times \{0\}\}{\times}\\ \{\begin{bmatrix} y \\ 0 \end{bmatrix} : (\begin{bmatrix} y \\ 0 \end{bmatrix} - (A_{i,y})^{K_i} q_{i,y}(t)) \in RP_{i,y}(0) \times \{0\}\} \end{aligned}$$

$\min x_i, \max x_i, \min y_i, \max y_i$ can be computed by solving the corresponding optimization problems respectively. For example, we can obtain $\min x_i$ by solving a MIP problem:

$$\begin{aligned} &\min x_i \triangleq \min_{\tilde{m}_i(t), \tilde{u}_i(t)} x_i(t+1)\\ &\texttt{s.t.}\ \ q_i(t, k_i{+}1){=}A_i\, q_i(t, k_i){+}B_i(m_i(t, k_i))\, u_i(t, k_i),\ k_i{\geq}0\\ &\qquad q_i(t) = [0,0,0,0]^T, \quad q_i(t+1, 0) = q_i(t, K_i),\\ &\qquad \dot{x}_i(t+1) = \dot{y}_i(t+1) = 0,\\ &\qquad m_i(k, k_i) \in \mathcal{M}_i, \quad u_i(k, k_i) \in \mathcal{U}_i. \end{aligned}$$

Let us see if the reachable set function satisfies the two requirements. Since the reachable set function $R_i(q_i(t))$ can be obtained by simply substituting $q_i(t)$ value, Requirement 1 is achieved. Note that all the intermediate variables are hidden in the $R_i(q_i(t))$, thus Requirement 2 is also satisfied.

5 Algorithm Analysis

We discuss the two important properties of the algorithm: feasibility and stability. We also compare the complexity of the proposed method with CMPC on switched linear systems.

As the goal states given by the central controller at each collaboration cycle will be exactly reached by all robots, the feasibility of HMPC is determined by the optimal problem settled by the central controller. The optimal problem encodes the status of all robots as well as the constraints arising from the complete collision avoidance specification, so the feasibility is automatically assured by the solution.

Theorem 2 (Feasibility)**.** *Our* HMPC *scheme is feasible.*

The terminal constraint set and cost function methods are widely used to ensure the stability of traditional MPC approaches [Mayne *et al.*, 2000]. They differ from each other on the setting of *stable parameters* (namely, a terminal cost l_H,

a terminal constraint set $\tilde{\mathcal{Q}}_f$ and a controller function $\kappa(\cdot)$). Since CMPC deals with the multi-robot system as a whole, these various approaches can be easily applied on CMPC. So instead of giving the concrete parameter setting, we show the stability of our HMPC is equivalent to the stability of a specific linear CMPC. Thus users can choose their favorite methods to set parameters to ensure the stability.

Theorem 3 (Stability)**.** *Given a multi-robot system where each robot is modeled as (3a) and (3b), if a set of parameters $l_H, \tilde{\mathcal{Q}}_f, \kappa_f(\cdot)$ ensures the stability of a specific linear* CMPC*, the same setting also ensures the stability of our* HMPC*.*

Proof. We construct the following linear CMPC problem:

$$\left.\begin{aligned} J_c^*(q(t)) \;=\; & \min_{Q(H|t)} \sum_{k=0}^{H-1} l_q(q(k|t)) + l_H(q(H|t)) \\ \texttt{s.t.}\;\; & q(k+1|t) = A\,q(k|t) + B\,u'(k|t),\; k \geq 0, \\ & u'(k|t) \in \prod_{i=1}^{N} [\min x_i, \max x_i] \times [\min y_i, \max y_i] \\ & h(q_{k|t}) \leq 0,\; k = 1, ..., H-1, \\ & q(0|t) = q(t), \quad q(H|t) \in \tilde{\mathcal{Q}}_f, \end{aligned}\right\} \tag{10}$$

where

$$A = A_1^{K_1} \oplus \cdots \oplus A_N^{K_N}, \quad B = \overbrace{\begin{bmatrix}1&0&0&0\\0&0&1&0\end{bmatrix}^T \oplus \cdots \oplus \begin{bmatrix}1&0&0&0\\0&0&1&0\end{bmatrix}^T}^{N},$$

$I_n \in \mathbb{R}^n$ denotes the n-dimension identity matrix, u' is the $4N \times 2$ input variable in (10).

Provided that $l_H, \tilde{\mathcal{Q}}_f, \kappa_f(\cdot)$ ensures the stability for (10), i.e. the state sequence computed by (10) converges to the goal state. We have to prove the state sequence $\{q(t)\}_{t=0}^{\infty}$ computed by (8) is exactly the same with the one computed by (10).

Note that (8) has the same cost function with (10). The only difference between the constraints parts of (8) and (10) is the state function and input constraints, which actually provide the same feasible range of states. So given any $q(t)$, the optimal solutions of (8) and (10) are the same, which leads to the conclusion that the state sequence $\{q(t)\}_{t=0}^{\infty}$ computed by (8) will be identical to the one computed by (10).

Notice that each robot is ensured by (9) to reach its control goal computed by the central controller in each collaboration cycle. Stability provided by the central controller in turn guarantees that the HMPC scheme is stable. □

Due to the parallel computation of robots' concrete inputs, HMPC's efficiency is mainly determined by the optimization problem in the central controller. Thus a straight forward way to analyze theoretical efficiency is to quantitatively compare the complexity of programming problems in the central controllers of our HMPC and CMPC. Note that non-convex collision avoidance specification should be rewritten with Big-M method by introducing integer slack variables. Also as we mentioned above, the non-convex switched linear dynamic (3a) and (3b) should be treated by replacing the original switching control variable $m_i(k, k_i)$ with a $M_{i,x} \times M_{i,y}$ binary slack matrix variable, and the original continuous control variable $u_i(k, k_i)$ with a $M_{i,x} \times M_{i,y}$ cell matrix variable where each element is a 2×1 real vector [Richards and How, 2005]. It means that the optimization problems in HMPC and CMPC are mixed integer programming (MIP). There is no available theory of MIP complexity analysis, we analyze their complexity by comparing the number of (scalar) variables and (scalar) constraints instead. As integer variables have great impact on the MIP complexity, we consider continuous and integer variables separately.

Proposition 1 (Complexity)**.** *Comparing to the optimization problem in* CMPC*, the one in our* HMPC*'s central controller is reduced by $H \cdot \sum_{i=1}^{N}(2K_iM_{i,x}M_{i,y} + 4K_i - 4)$ continuous variables, $H \cdot \sum_{i=1}^{N} K_iM_{i,x}M_{i,y}$ integer variables and $H \cdot \sum_{i=1}^{N}(2K_iM_{i,x}M_{i,y} + 5K_i - 4)$ constraints.*

Proof. Collaboration specification and collision avoidance specification are treated the same in both control frameworks, thus we do not take them into account. For a robot i, at each sampling instant, there are $M_{i,x} \cdot M_{i,y}$ integer switching control variables, $2M_{i,x} \cdot M_{i,y}$ continuous control variables, 4 continuous state variables, $2M_{i,x} \cdot M_{i,y}+1$ control constraints and 4 state equation constraints, which should be all considered in CMPC. The intermediate variables and constraints in a collaboration cycle are reduced in our HMPC with the additional 4 reachable set constraints. Since the optimization problems in our HMPC and CMPC's central controller consider N robots in H collaboration cycles, the proposition is true. □

Proposition 1 demonstrates that our HMPC greatly reduces the number of variables and constraints comparing to CMPC, which is empirically supported by the following experiments.

6 Experimental Results

In order to explore the practicability of our HMPC based method on switched linear multi-robot systems, we have implemented other MPC schemes:centralized MPC (CMPC), basic DMPC (BDMPC), sequential DMPC (SDMPC), and iterative DMPC (IDMPC). CMPC [Dunbar and Murray, 2002] deploys a central controller to compute the concrete inputs of all the robots. On the contrary, DMPCs distribute the computation to many robots and differ from each other in terms of the policy of distribution: robots compute concurrently in BDMPC [Keviczky *et al.*, 2004], sequentially in SDMPC [Kuwata and How, 2011], and iteratively in IDMPC [Mercangöz and Doyle, 2007].

We derive the dynamic model of the *i*-th robot (Eq. 3) by discretizing a switched linear continuous system:

$$\dot{q}(t) = Aq(t) + B_{r,s}u(t), t \geq 0, r, s \in \{1, 2\}, \tag{11}$$

where

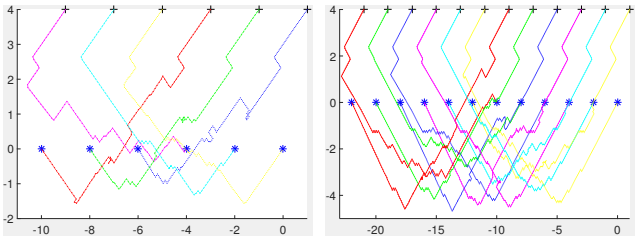
$$A = \begin{bmatrix}0&0&1&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{bmatrix}, B_{r,s} = \begin{bmatrix}0&0\\r&0\\0&0\\0&s\end{bmatrix},$$

at the sampling frequency:

$$\textit{sampling frequency of } \textit{i-th robot} = \begin{cases} 1/0.075, & \textit{i mod 3} \equiv 1, \\ 1/0.1, & \textit{i mod 3} \equiv 2, \\ 1/0.15, & \textit{i mod 3} \equiv 0. \end{cases}$$

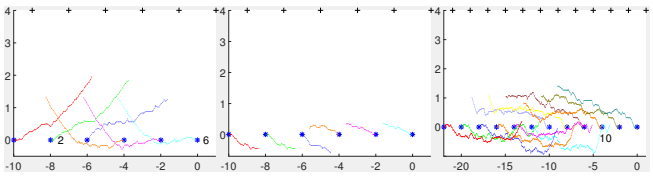
The continuous control constraint is $\mathcal{U}_i = [2, 2]^T$ and the safety distance is $d_{safe} = 0.6$.

Programming problems embedded in MPCs are solved by the CVX, a Matlab-based package [CVX Research, 2012; Grant and Boyd, 2008].



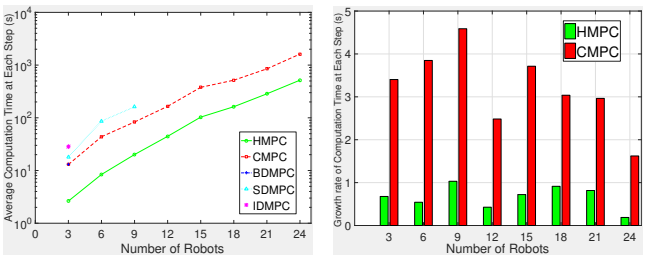
(a) HMPC in 6-robot case (b) HMPC in 12-robot case

Figure 1: Robot trajectories using HMPC



(a) BDMPC in 6-robot case (b) IDMPC in 6-robot case (c) SDMPC in 12-robot case

Figure 2: Typical failures of DMPCs



(a) Computation time vs. number of robots (b) Increasing rate of computation time vs. dynamic modes

Figure 3: Efficiency Comparison

6.1 Effectiveness

We compare the effectiveness by simulating MPCs with different numbers of robots $N{=}3, 6, 9, 12, 15, 18, 21, 24$. Our HMPC finishes all the simulations as CMPC. Due to the space limitation, we show the trajectories computed by our HMPC in 6/12-robot cases as Figure 1. In Figure 1 (and Figure 2), the x-axis and y-axis construct the plane which all the robots move in, and represent the positions of robots. Blue stars denote the initial positions, black crosses denote the goal positions, and dots in different colors denote the trajectories of different robots. In this configuration, robots have to cross the trajectories of others before they reach the goal positions. If all robots move directly towards to their corresponding goal positions, collision will inevitably happen.

Observing the trajectories computed by our HMPC, the robots in each scenario do not collide (satisfying feasibility) and finally converge to the goal states (satisfying stability), which conforms to the theoretical conclusion in Sec. 5.

However, DMPCs suffer from heavy failure issues, where several typical failed scenarios are shown in Figure 2. Note that the simulation is forced to stop, once the distance between any two robots is detected smaller than the safety distance. Reasons of failures fall into three main categories:

Incorrect Prediction: In Figure 2a, the *2*rd and *6*th robots collide, even though they are neighbors. Instead of knowing the actual behaviors of its neighbors, in BDMPC each robot computes its action by prediction. These neighbors often behave differently from what the *i*-th robot predicts. As a result, it is of high possibility that two adjacent robots make incorrect predictions on each other's behavior and thus collide.

Divergent Iteration: In Figure 2b, the robots get stuck in an infinite loop as the iteration process does not converge when using IDMPC. Before it was forcibly terminated, the iteration had tried over 100 times and spent more than four hours. It indicates that IDMPC may need additional treatment when encountering navigation problems with non-convex constraints.

Aggressive Action: In Figure 2c, the *10*th robot cannot get a feasible solution. In SDMPC, robots compute their actions sequentially in a fixed order. The aggressive actions taken by the robots with high priority may push the robots with low priority out of feasible regions.

6.2 Efficiency

Figure 3a shows the computation time of different MPCs in the scenarios in Sec. 6.1. For each scenario, the average computation time at each collaboration instant is plotted, if a MPC scheme succeeds. We study the efficiency improvement accompany with the increasement of robots. Notably, compared to CMPC, our HMPC improves the efficiency by at least 66.4% for all cases. This result complies with the theoretical conclusion in Sec. 5. The great reduction of constraints and variables makes our HMPC much more efficient in practice.

We then explore the efficiency improvement when the number of dynamic modes grows up. Figure 3b shows the result. The y-axis reflects the increasing rate of the average computation time at each collaboration instant, which is calculated by dividing the extra computation time in the case of four-mode robots by the computation time in the case of one-mode robots. The x-axis shows the number of robots involved. The average increasing rate of our HMPC is 61.8%, while that of CMPC is over 300%. It indicates that our HMPC is less sensitive to the growth of systems' complexity than CMPC. It is due to that in our HMPC the hyper-rectangle representation of reachable set hides the integer variables in the central controller introduced by multiple modes. Thus the complexity of the problem in the central controller increases smoothly with the robot dynamic modes.

7 Conclusion

In this paper, we propose a novel HMPC based approach for the navigation problem of switched linear dynamical robots with a new reachable set computation technique. We discuss feasibility, stability and complexity of the approach theoretically and shows the effectiveness and efficiency empirically.

Acknowledgement

Chao Huang was supported by the China Scholarship Council (No.201606190185). This research was supported by The National Key Research and Development Program of China (No.2016YFB1000802), National Natural Science Foundation of China (No.61632015, No.61561146394, No.61373033) and also by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[Alur *et al.*, 1995] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical computer science*, 138(1):3–34, 1995.

[Bogomolov *et al.*, 2014] Sergiy Bogomolov, Daniele Magazzeni, Andreas Podelski, and Martin Wehrle. Planning as model checking in hybrid domains. In *AAAI*, pages 2228–2234, 2014.

[Cirillo *et al.*, 2014] Marcello Cirillo, Federico Pecora, Henrik Andreasson, Tansel Uras, and Sven Koenig. Integrated motion planning and coordination for industrial vehicles. In *ICAPS*, 2014.

[CVX Research, 2012] Inc. CVX Research. CVX: Matlab software for disciplined convex programming, version 2.0. `http://cvxr.com/cvx`, aug 2012.

[Dunbar and Murray, 2002] William B. Dunbar and Richard M. Murray. Model predictive control of coordinated multi-vehicle formations. In *CDC*, volume 4, pages 4631–4636, 2002.

[Faust *et al.*, 2016] Aleksandra Faust, Hao-Tien Chiang, Nathanael Rackley, and Lydia Tapia. Avoiding moving obstacles with stochastic hybrid dynamics using PEARL: preference appraisal reinforcement learning. In *ICRA*, pages 484–490, 2016.

[Grant and Boyd, 2008] Michael C. Grant and Stephen P. Boyd. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. 2008.

[Gritzmann and Sturmfels, 1993] Peter Gritzmann and Bernd Sturmfels. Minkowski addition of polytopes: computational complexity and applications to gröbner bases. *SIAM Journal on Discrete Mathematics*, 6(2):246–269, 1993.

[Henzinger *et al.*, 1995] Thomas A Henzinger, Peter W Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *STOC*, pages 373–382. ACM, 1995.

[Huang *et al.*, 2016] Chao Huang, Xin Chen, Yifan Zhang, Shengchao Qin, Yifeng Zeng, and Xuandong Li. Hierarchical model predictive control for multi-robot navigation. In *IJCAI*, pages 3140–3146, 2016.

[Janovský *et al.*, 2014] Pavel Janovský, Michal Cáp, and Jirí Vokrínek. Finding coordinated paths for multiple holonomic agents in 2-d polygonal environment. In *AAMAS*, pages 1117–1124, 2014.

[Karaman and Frazzoli, 2011] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Rob. Res.*, 30(7):846–894, June 2011.

[Keviczky *et al.*, 2004] Tamás Keviczky, Francesco Borrelli, and Gary J. Balas. A study on decentralized receding horizon control for decoupled systems. In *ACC*, volume 6, pages 4921–4926. IEEE, 2004.

[Koenig and Likhachev, 2002] Sven Koenig and Maxim Likhachev. D* lite. In *AAAI/IAAI*, pages 476–483, 2002.

[Kuwata and How, 2011] Yoshiaki Kuwata and Jonathan P. How. Cooperative distributed robust trajectory optimization using receding horizon milp. *IEEE Transactions on Control Systems Technology*, 19(2):423–431, 2011.

[Lahijanian *et al.*, 2014] Morteza Lahijanian, Lydia E Kavraki, and Moshe Y Vardi. A sampling-based strategy planner for nondeterministic hybrid systems. In *ICRA*, pages 3005–3012. IEEE, 2014.

[Mayne *et al.*, 2000] David Q. Mayne, James B. Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[Mercangöz and Doyle, 2007] Mehmet Mercangöz and Francis J. Doyle. Distributed model predictive control of an experimental four-tank system. *Journal of Process Control*, 17(3):297–308, 2007.

[Pecora *et al.*, 2012] Federico Pecora, Marcello Cirillo, and Dimitar Dimitrov. On mission-dependent coordination of multiple vehicles under spatial and temporal constraints. In *IROS*, pages 5262–5269. IEEE, 2012.

[Richards and How, 2005] Arthur Richards and Jonathan How. Mixed-integer programming for control. In *ACC*, pages 2676–2683. IEEE, 2005.

[van den Berg *et al.*, 2011] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. *Reciprocal n-Body Collision Avoidance*, pages 3–19. 2011.

[Wagner and Choset, 2011] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *IROS*, pages 3260–3267, Sept 2011.

[Wei *et al.*, 2014] Changyun Wei, Koen V Hindriks, and Catholijn M Jonker. Multi-robot cooperative pathfinding: A decentralized approach. In *IEA/AIE*, pages 21–31. Springer, 2014.