Temporal Reasoning with Kinodynamic Networks

Han Zhang, Neelesh Tiruviluamala, Sven Koenig, T. K. Satish Kumar
{zhan645, tiruvilu, skoenig}@usc.edu, tkskwork@gmail.com

USC University of Southern California

Abstract

Temporal reasoning is central to Artificial Intelligence (AI) and many of its applications. However, the existing algorithmic frameworks for temporal reasoning are not expressive enough to be applicable to robots with complex kinodynamic constraints typically described using differential equations. For example, while minimum and maximum velocity constraints can be encoded in Simple Temporal Networks (STNs), higher-order kinodynamic constraints cannot be represented in existing frameworks. In this paper, we present a novel framework for temporal reasoning called Kinodynamic Networks (KDNs). KDNs combine elements of existing temporal reasoning frameworks with the idea of Bernstein polynomials. The velocity profiles of robots are represented using Bernstein polynomials; and dynamic constraints on these velocity profiles can be converted to linear constraints on the to-be-determined coefficients of their Bernstein polynomials. We study KDNs for their attractive theoretical properties and apply them to the Multi-Agent Path Finding (MAPF) problem with higher-order kinodynamic constraints. We show that our approach is not only scalable but also yields smooth velocity profiles for all robots that can be executed by their controllers.

STNs and KDNs

An STN is characterized by a directed graph. Each vertex represents an event with a to-be-assigned execution time. Each edge $\langle X_i, X_j \rangle$ is annotated with an interval $[lb, ub]$ and represents a simple temporal constraint, indicating that X_j must be scheduled between lb and ub time units after X_i, The task of solving an STN is to assign times to all events that satisfy all simple temporal constraints.

A KDN extends an STN by allowing *motion edges*. A motion edge corresponds to the traversal of a physical distance by a robot, which is subject to dynamic constraints, like minimum/maximum velocity and minimum/maximum acceleration constraints. The task of solving a KDN is to simultaneously find assignments of times to all event and a velocity profile for the robot for each motion edge that satisfies the dynamic constraints of the robot.

Bernstein Polynomials

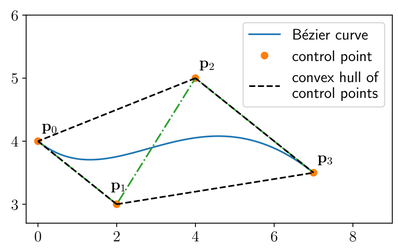
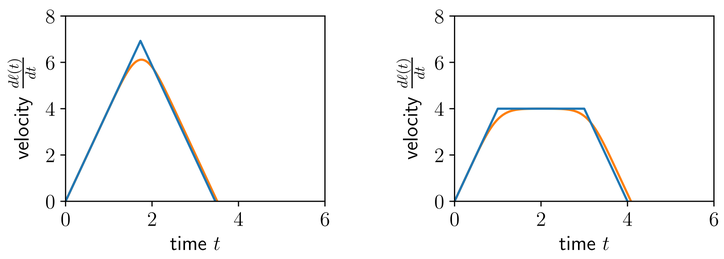


Figure 1: Shows an example of Bézier curve.

A Bézier curve is a linear combination of the Bernstein basis polynomials. The coefficients of the linear combination are called control points. A Bernstein polynomial $B(t)$ is a 1-dimensional Bézier curve with its control points being real numbers. These control points are also referred to as the Bernstein coefficients. Any continuous real-valued function defined on the real interval $[0, 1]$ can be uniformly approximated by Bernstein polynomials.

Dynamic Constraints on a Single Motion Edge

KDNs are significantly more complex than STNs, and even a single motion edge can introduce dynamic constraints that are hard to solve. In our paper, we formulate the problems of finding a velocity profile that minimizes the traversal time as Kinodynamic Differential Programs (KDDPs). We use a scaled Bernstein polynomials to approximate the velocity profile in a KDDP. Therefore, the constraints of a KDDP are converted to linear constraints on the Bernstein coefficients. We prove that the values of all feasible traversal times form a single interval, if any such value exists, and show that KDDPs can be solved using a binary search procedure.



(a) $L = 12\text{m}, v \leq 8\text{m/s}, |a| \leq 1\text{m/s}^2$ **(b)** $L = 12\text{m}, v \leq 4\text{m/s}, |a| \leq 1\text{m/s}^2$

Figure 2: Shows the result of applying our KDDP algorithm on two examples. The blue curves are the optimal velocity profiles derived from the physical interpretation. The orange curves are close approximations produced by our Bernstein polynomial approach. In both cases, 40 control points are used for the approximation.

Solving KDNs

At a high level, our approach for solving KDNs converts the dynamic constraints on a motion edge with fixed boundary conditions to an induced simple temporal constraint. For a feasible KDDP associated with the dynamic constraints of any motion edge, we prove that there exists only one single feasible time interval and our binary search procedure can be used to find the feasible time interval. A KDN can now be solved by searching in the space of all combinations of boundary conditions at the KDN vertices, which can be done using a Mixed Integer Linear Programming (MILP) solver.

KDNs and MAPF

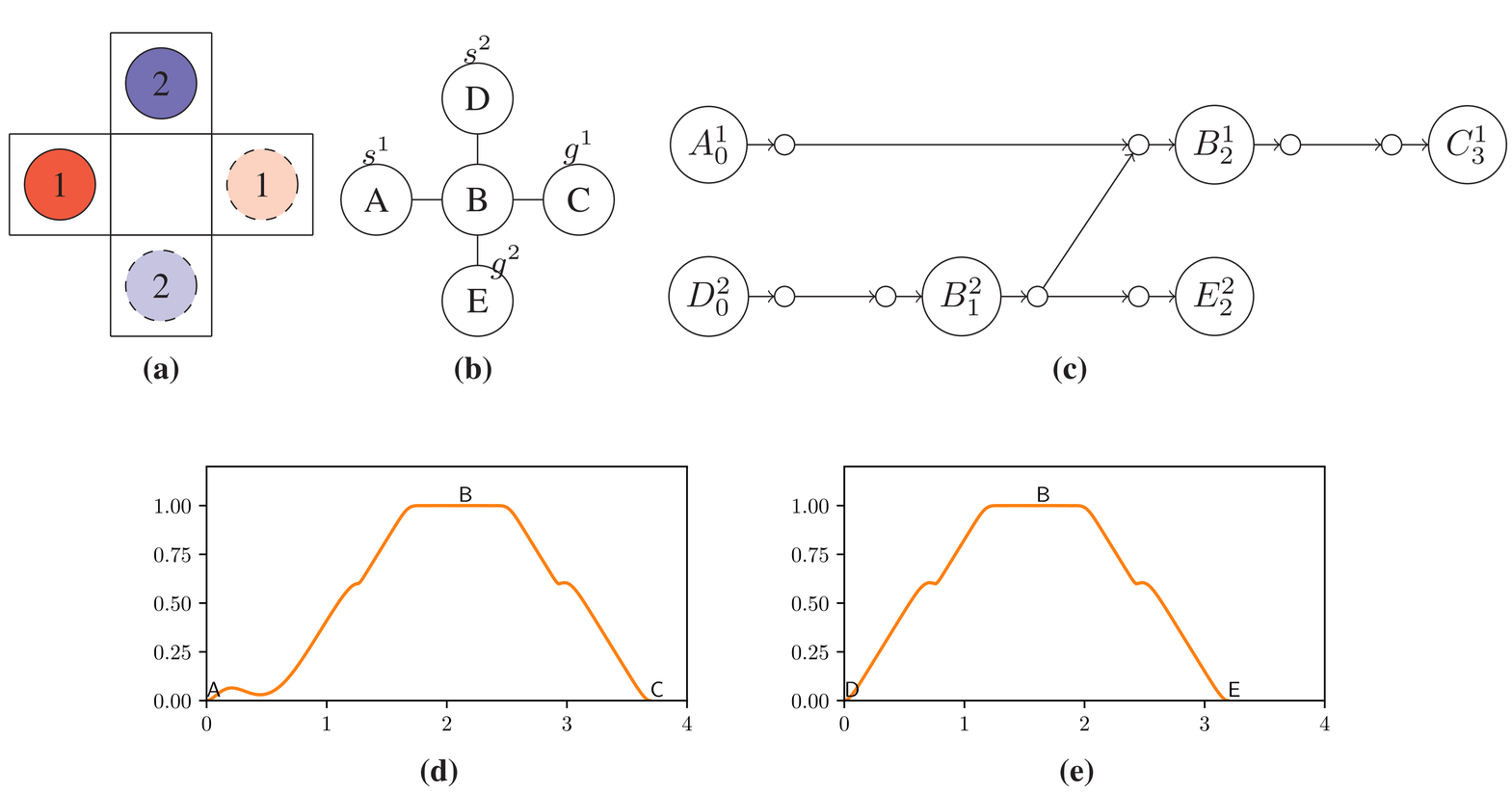


Figure 3: (a) shows a small MAPF instance with 2 robots on a four-neighbor grid map. (b) shows the same MAPF instance in a graph representation with vertices representing free cells and edges representing transitions between neighboring free cells. (c) shows the KDN built on a discrete plan generated by a MAPF solver. (d) and (e) show the velocity profiles of Robots 1 and 2, respectively, generated by our KDN algorithm.

#Robots	MAPF time (s)	KDN time (s)
04	00.053 (30/30)	10.064 (30/30)
06	00.214 (30/30)	18.551 (30/30)
08	11.409 (30/30)	31.882 (30/30)
10	25.819 (28/30)	43.191 (28/28)
12	68.112 (19/30)	72.172 (19/19)
14	36.938 (02/30)	92.952 (02/02)

(a)

%Obstacles	MAPF time (s)	KDN time (s)
10	00.296 (30/30)	5.170 (30/30)
15	00.507 (30/30)	5.468 (30/30)
20	01.142 (30/30)	6.694 (30/30)
25	19.076 (26/30)	6.757 (26/26)
30	53.692 (16/30)	9.707 (16/16)
35	08.621 (03/30)	6.144 (03/03)

(b)

Figure 4: (a) and (b) show experimental results on some MAPF instances with a varying numbers of robots and obstacle densities, respectively, on four-neighbor grid maps. In both cases, time is measured in seconds. MAPF time refers to the average time taken by CBSH [1], an optimal MAPF solver, to generate the discrete plans. KDN time refers to the average runtime of our KDN solver for generating the velocity profiles of the robots given the output of the MAPF solver. In both cases, the average is computed over the A successfully solved instances (within a runtime limit of 5 minutes) out of the given B instances, indicated by (A/B).

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779 and 1935712.

[1] Ariel Felner, Jiaoyang Li, Eli Boyarski, Hang Ma, Liron Cohen, T. K. Satish Kumar, and Sven Koenig. Adding heuristics to conflict-based search for multi-agent path finding. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 83–87, 2018.