POSTER TITLE: ALGORITHMS FOR TEMPORAL NETWORKS

Merrick Chang, Muhammad Furrukh Asif, Sudais Moorad
Luke Hunsberger

INTRODUCTION

The purpose of this URSI project was to help establish a (virtual) Temporal Reasoning Laboratory at Vassar College. The principal goal was to create a public repository that will include implementations of recently developed algorithms for manipulating Simple Temporal Networks (STNs) and Simple Temporal Networks with Uncertainty (STNUs). The algorithms were drawn from the recent literature on Temporal Networks. The public repository will enable researchers in the field to use the implemented algorithms to carry out reproducible empirical evaluations of the algorithms.

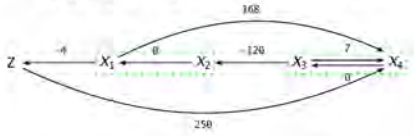
WHAT IS A TEMPORAL NETWORK

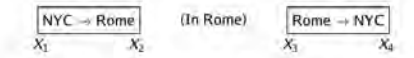
Temporal Networks are data structures for reasoning about temporal constraints on events in a variety of circumstances. For example, a computer agent responsible for planning a set of activities might use a Simple Temporal Network to represent the starting and ending times of those activities, as well as constraints on their durations, deadlines, and inter-activity constraints. A consistent STN is one that has a solution (i.e., a fixed schedule for doing those activities that is guaranteed to satisfy all of the constraints). Efficient algorithms are available for the computer agent to check the consistency of the STN, and for generating a variety of solutions. More complex problems involve scheduling activities in real time (i.e., as opposed to fixing a complete schedule in advance). Algorithms for converting a consistent STN into *dispatchable* form ensure that real-time execution can be guaranteed to succeed with minimal computational overhead.

An STNU augments an STN to include *contingent links* that can be used to represent actions with uncertain duration.

For example, our computer agent might control when it gets into a taxi, but not the duration of the taxi ride to the train station. Determining whether the activities in an STNU can be successfully executed in real-time is a more complex problem. However, assuming that the uncertainty is bounded by known values (e.g., the taxi ride might take between 10 and 25 minutes), efficient algorithms can check whether an STNU is *dynamically controllable*. Such algorithms allow an agent to use a *dynamic* strategy for deciding when to start actions, meaning that the agent's decisions can *react* to observations of the durations of contingent

$$\left\{\begin{array}{llll} Z - X_1 \le -4 & & X_4 - Z \le 250 \\ X_4 - X_1 \le 168 & & X_2 - X_3 \le -120 \\ X_4 - X_3 \le 7 & & X_1 - X_2 \le 0 \\ X_3 - X_4 \le 0 & & \end{array}\right\}$$





$T = \{Z, X_1, X_2, X_3, X_4\}$, where Z = Noon, June 8

$$C = \left\{\begin{array}{lll} Z - X_1 \le -4 & \text{(Leave NYC after 4 p.m., June 8)} \\ X_4 - Z \le 250 & \text{(Return NYC by 10 p.m., June 18)} \\ X_4 - X_1 \le 168 & \text{(Gone no more than 7 days)} \\ X_2 - X_3 \le -120 & \text{(In Rome at least 5 days)} \\ X_4 - X_3 \le 7 & \text{(Return flight less than 7 hrs)} \end{array}\right.$$

The above figures show a basic example of a scenario in which an STN could be applied and the associated diagram. A plane must make flights from NYC to Rome and back within set time constraints. Graphically, we can represent this situation using nodes and direct edges with weights.

LIBRARY FUNCTIONALITY

Our library features a wide variety of algorithms for STNs and STNUs, solving common problems such as verifying consistency, calculating shortest paths, and converting to dispatchable form. Users are also able to create visualizations for STNs and STNUs.

Some parts of our program have an accompanying GUI for ease of access.Our library is implemented in Python and available on Github [github.com/smoorad-vassar/temporal-networks]. Our code is fully documented and accompanied by a full suite of diagnostic tests and several algorithms to randomly generate networks for testing.

ALGORITHMS WE IMPLEMENTED

- Consistency Checking Algorithms for STNs
  - Bellman-Ford (Cormen et al. 2001)
  - Incremental Distance Matrix updaters (Rohnert 1985; Chleq 1995)
  - Consistency Checker (Morris 2014)
  - Potential function updaters (Ramalingam 1995; Hunsberger & Posentato, publication pending)
- All-Pairs Shortest-Paths (APSP) Algorithms
  - "Snowball" ASAP Algorithm (Planken 2013)
  - Johnson's Algorithm
  - Floyd-Warshall Algorithm
- Dijkstra's Single-Source, Shortest-Paths (SSSP) Algorithm (several versions)
- Dispatchability Algorithms
  - Greedy Dispatcher
  - Algorithms for converting into dispatchable form (Tsamardinos and Morris 1998)
  - Tarjan's Algorithm for finding Strongly Connected Components (1972)
- Path Consistency Algorithms
  - Directed Path Consistency (DPC) Converter (Dechter et al, 1991, Planken 2013)
  - Partial Path Consistency (PPC) Converters (Planken 2013)
  - DPC-Dispatch Algorithm (Planken 2013)
- Triangulation Algorithms (Planken 2013)
- Dynamic Controllability (DC) Checking Algorithms for STNUs
  - Morris 2014
  - Cairo, Hunsberger, Rizzi 2018