Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

Privacy-Preserving Algorithm for Decoupling of Multi-Agent Plans with Uncertainty

Yuening Zhang, Brian Williams
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
zhangyn@mit.edu, williams@mit.edu

Abstract

The execution of multi-agent plans often requires communication between agents in order to synchronize their tasks. In cases where communication is unreliable or undesirable, temporal decoupling algorithms allow agents to find a distributed execution strategy beforehand without requiring perfect communication on the fly. The state-of-the-art Multi-Agent Simple Temporal Network with Uncertainty (MaSTNU) framework extends the decoupling problem for Multi-Agent Simple Temporal Network (MaSTN) to allow the modeling of uncertain durations and allow agents to communicate when certain events occur and communication is available. However, the existing approach assumes centralized knowledge of the MaSTNU, whereas in the multi-agent context, privacy is an important concern. In this paper, we propose a distributed, privacy-preserving algorithm for finding distributed execution strategies for MaSTNU. Experiments also showed significant speed-up of the proposed algorithm when the multi-agent plan is loosely coupled and mostly private.

Introduction

In many robotic tasks, such as the deployment of an AUV fleet to scout an interesting region of the ocean, robots may communicate with each other and with the operator, but such communication may be intermittent. When we consider human teams in collaboration, communication is important but they also do not update their progress too frequently. In these multi-agent plans, inter-dependencies between agents result in precedence or synchronization constraints. While communication between agents helps ensure the satisfaction of those constraints, in real life, communication may be unreliable during execution or simply undesirable.

The temporal decoupling problem, first introduced by Hunsberger (Hunsberger 2002), remedies the situation by finding distributed execution strategies for the multi-agent temporal plans, so that there is no need for communication between agents during execution. Temporal decoupling problem is first defined for Multi-Agent Simple Temporal Network (MaSTN) (Boerkoel Jr and Durfee 2013), which is a Simple Temporal Network (STN) (Dechter, Meiri, and Pearl 1991) partitioned among a set of agents. The decoupling solution is a set of local temporal constraints imposed on each agent's local network, such that when satisfied, the joint execution is guaranteed to succeed. Casanova et al. (Casanova et al. 2016) later extended the result to Multi-Agent Simple Temporal Network with Uncertainty (MaSTNU), which is a Simple Temporal Network with Uncertainty (STNU) (Vidal 1999) partitioned among a set of agents. In addition to allowing the modeling of uncertain durations, it also introduces communication links between agents, represented by inter-agent contingent temporal constraints, where an agent is allowed to report the occurrence of an event to another agent with some delay. Compared to the traditional decoupling problem where no communication is allowed, such a model is more realistic in reality, and it solves more decoupling problems where such communication is necessary for the execution to succeed.

On the other hand, distributed and privacy-preserving algorithms for the temporal decoupling problem have also generated interests (Boerkoel Jr and Durfee 2013; Mogali, Smith, and Rubinstein 2016). When agents coordinate on a shared task, they may not want to expose private or irrelevant information about their plan to others, such as their lunchtime, or other personal commitments they may have. Previous distributed algorithms have only considered MaSTNs. In this paper, we propose a distributed, privacy-preserving algorithm for the temporal decoupling problem for MaSTNUs. Instead of using a centralized approach (Casanova et al. 2016) that assumes centralized knowledge of the multi-agent plan, we adopt a generate-and-test approach where a master algorithm generates candidate decoupling solutions using only the limited shared information, and agents independently test the candidate on its local network, and report conflicts to guide the decoupling solution generation process. Our experiments showed significant performance gain of the distributed approach on loosely coupled MaSTNU problems, where the agents' networks are large but mostly private.

Background

Multi-Agent Simple Temporal Network with Uncertainty

Our multi-agent plan with uncertainty is represented by a MaSTNU (Casanova et al. 2016), which is a partition of an STNU (Vidal 1999) among a set of agents.

Definition 1 (STNU)**.** A *Simple Temporal Network with Uncertainty* (STNU) is a tuple $\langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$, where

- $\mathcal{V}$ is a set of events representing designated points in time.
- $\mathcal{E}$ is a set of *simple temporal constraints* scoped on $\mathcal{V}$, where $e \in \mathcal{E}$ is a tuple $\langle s, t, lb, ub \rangle$, in which
  - $s, t \in \mathcal{V}$ is the start and end event of the constraint.
  - $lb \in \mathbb{R} \cup \{-\infty\}$, $ub \in \mathbb{R} \cup \{+\infty\}$ is the lower bound and upper bound from s to t, i.e. $lb \leq t - s \leq ub$.
- $\mathcal{C}$ is a set of *simple contingent temporal constraints*, one for each uncontrollable event, where $e \in \mathcal{C}$ is a tuple $\langle s, t, lb, ub \rangle$, in which
  - $s, t \in \mathcal{V}$ is the start and end event of the constraint, where t is an *uncontrollable event*.
  - $lb, ub \in \mathbb{R}$ is the lower bound and upper bound from s to t and $0 \leq lb < ub < \infty$, i.e. $lb \leq t - s \leq ub$.

STNUs are an extension to STNs (Dechter, Meiri, and Pearl 1991) that, instead of assuming full control over the execution of all events, allows the modeling of uncontrollable events that can only be observed by the agent. A contingent constraint specifies the bound in which an uncontrollable event may occur. For example, it may take any time between 15 to 30 minutes to travel to a destination, depending on the traffic, and such a duration is not directly controlled by the agent. We also refer to $\mathcal{E}$ as *requirement constraints* to differentiate from the contingent constraints $\mathcal{C}$. An STNU is dynamically controllable (Vidal 1999) if there exists a *dynamic* and *valid* execution strategy. Intuitively, *dynamic* means that the strategy dynamically assigns values to the executable events based on the observed outcomes of uncontrollable events up to the present time, and *valid* means that all the temporal constraints are satisfied (Morris 2014; Casanova et al. 2016).

Definition 2 (MaSTNU)**.** A *Multi-Agent Simple Temporal Network with Uncertainty* (MaSTNU) is a tuple $\langle \mathcal{A}, f, \mathcal{V}, \mathcal{E}, \mathcal{C}, v_Z \rangle$, where

- $\langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$ is an instance of STNU.
- $\mathcal{A}$ is a set of agents.
- v_Z is a reference event, an absolute time point preceding all other events and shared by all agents, such as 12 pm.
- $f : \mathcal{V} \backslash \{v_Z\} \rightarrow \mathcal{A}$ is a partition function, which is a mapping of each event to a unique agent, except for v_Z.

The partition function f uniquely partitions the STNU into a set of local networks, one for each agent, and a set of external requirement and contingent constraints $\langle N^{\mathcal{A}}, \mathcal{E}_X, \mathcal{C}_X \rangle$, where:

- Each $N^a \in N^{\mathcal{A}}$ is the *local network* for agent $a \in \mathcal{A}$, which is an STNU $\langle \mathcal{V}^a, \mathcal{E}^a, \mathcal{C}^a \rangle$, where $\mathcal{V}^a = \{v \in \mathcal{V} | f(v) = a\} \cup \{v_Z\}$ is the set of *local events* owned by agent a, and $\mathcal{E}^a$ and $\mathcal{C}^a$ are the set of *local constraints* for agent a, which are constraints from $\mathcal{E}$ and $\mathcal{C}$ that are scoped on $\mathcal{V}^a$.
- $\mathcal{E}_X = \{e = \langle s, t, lb, ub \rangle \in \mathcal{E} | f(s) \neq f(t)\}$ is a set of inter-agent requirement constraints, or *external requirement constraints*.

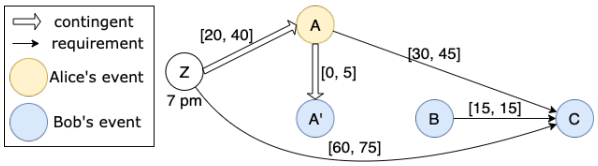

Figure 1: MaSTNU example modified from (Bhargava et al. 2018)

- $\mathcal{C}_X = \{e = \langle s, t, lb, ub \rangle \in \mathcal{C} | f(s) \neq f(t)\}$ is a set of inter-agent contingent constraints, or *external contingent constraints*.

Figure 1 shows an example MaSTNU, where two agents, Alice and Bob, are coordinating on a mission. Alice's local network is $\langle \{v_Z, v_A\}, \varnothing, \{e_{ZA}\} \rangle$, and Bob's local network is $\langle \{v_Z, v_{A'}, v_B, v_C\}, \{e_{BC}, e_{ZC}\}, \varnothing \rangle$, where v_Z is the reference event 7 pm. Additionally, e_{AC} is an external requirement constraint, and $e_{AA'}$ is an external contingent constraint. We also refer to external contingent constraints as *communication links*, since in this example, $e_{AA'}$ means that Bob can observe event $v_{A'}$ with a delay between 0 to 5 time units after the occurrence of event v_A that is unobservable to Bob.

Multi-Agent Temporal Decoupling Problem

While Casanova (Casanova et al. 2016) frames the problem as dynamic controllability for MaSTNU, its approach can be considered as one for the generalized temporal decoupling problem for MaSTNU, following from the definition of temporal decoupling problem for MaSTN (Boerkoel Jr and Durfee 2013):

Definition 3 (Temporal Decoupling)**.** Given a MaSTNU, the set of agents' local networks $N^{\mathcal{A}}$ forms a *temporal decoupling* of the MaSTNU if:

- All local networks $N^{\mathcal{A}} = \{N^{a_1}, N^{a_2}, \ldots, N^{a_n}\}$ are dynamically controllable, that is, there exists a dynamic and valid execution strategy for each local network.
- Merging *any* combination of dynamic and valid execution strategies for the local networks $N^{\mathcal{A}}$ yields a solution to the MaSTNU, that is, given that $\mathcal{C}_X$ are satisfied, all the external requirement constraints $\mathcal{E}_X$ are also satisfied.

Definition 4 (Temporal Decoupling Problem)**.** The objective of the temporal decoupling problem for MaSTNU is to find a set of *decoupling constraints* for each agent $\langle \mathcal{E}_d^a, \mathcal{C}_d^a \rangle$, such that the set of augmented local networks $N_{+\Delta}^a = \langle \mathcal{V}^a, \mathcal{E}^a \cup \mathcal{E}_d^a, \mathcal{C}^a \cup \mathcal{C}_d^a \rangle$ for each agent a forms a temporal decoupling of the MaSTNU.

We define $\langle \mathcal{E}_d, \mathcal{C}_d \rangle := \langle \cup_{a \in \mathcal{A}} \mathcal{E}_d^a, \cup_{a \in \mathcal{A}} \mathcal{C}_d^a \rangle$. Additionally, we say that $\langle \mathcal{E}_d, \mathcal{C}_d \rangle$ is a *feasible* decoupling solution, if the augmented local networks $N_{+\Delta}^a$ for all $a \in \mathcal{A}$ satisfy the first bullet point in Definition 3, that is, the addition of decoupling constraints does not make any local network become dynamically uncontrollable. We say that $\langle \mathcal{E}_d, \mathcal{C}_d \rangle$ is a *valid* decoupling solution, if the second bullet point is satisfied, that is, if the execution strategies satisfy the decoupling constraints, then the external constraints must be satisfied.

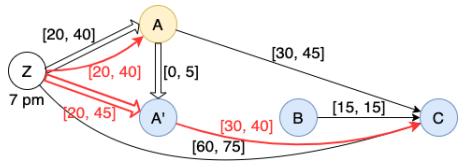

Figure 2: Example MaSTNU decoupling solution

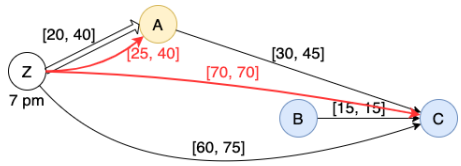

Figure 3: Infeasible decoupling without communication

Figure 4: Temporal decoupling without communication

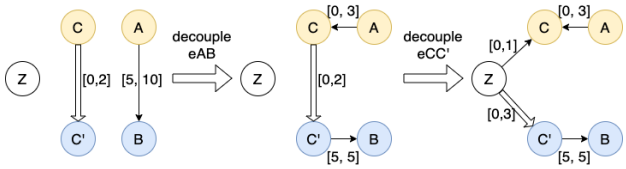

Figure 5: Temporal decoupling with communication

A decoupling solution for our example is shown in Figure 2. In this case, Alice's network is augmented with decoupling constraint $\langle\{e_{ZA}\}, \varnothing\rangle$, and Bob's network is augmented with $\langle\{e_{A'C}\}, \{e_{ZA'}\}\rangle$. The decoupling solution is feasible, since both local networks are dynamically controllable. Alice will communicate the occurrence of v_A to Bob, and Bob knows that he will receive the update at $v_{A'}$ sometime between 7:20 to 7:45 and continue his execution. The decoupling solution is also valid, because if $e_{AA'}$ is satisfied, then the merged local execution results will satisfy e_{AC}. Notice that while the local contingent constraint $e_{ZA'}$ specifies the possible occurrence of $v_{A'}$ from Bob's perspective, we still assume that $e_{AA'}$ is the actual functional contingent constraint guaranteed to be satisfied, since such communication is guaranteed by nature. This requires that $e_{ZA'}$ covers all possible time occurrences of $v_{A'}$, which is the case as $[20, 45]$ captures all possible realizations of e_{ZA} and $e_{AA'}$.

Without the communication link $e_{AA'}$, there is no decoupling solution in the above example. The traditional notion of decoupling would try to satisfy e_{AC} by fixing the time window for v_A and v_C with respect to the reference event v_Z, i.e. by enforcing e_{ZA} and e_{ZC}. As shown in Figure 3, even if we leave the full flexibility to Alice, her local network is not dynamically controllable due to the constraints involving v_Z and v_A. This shows that by allowing some degree of communication between agents during execution, the MaSTNU framework can solve more temporal decoupling problems than the traditional notion of decoupling.

Temporal Decoupling Algorithms

We describe the key ideas behind temporal decoupling algorithms (Hunsberger 2002; Casanova et al. 2016). The key intuition is to ensure that all the external constraints are made redundant by imposing a set of local decoupling constraints that are tighter or more restrictive, so that the external constraints can be satisfied without the need to explicitly propagate their values during execution.

Decoupling with Communication We will illustrate the idea using the following example in Figure 4 and 5 with two agents, the reference event $v_Z = 0$, an external requirement constraint e_{AB}, and optionally an external contingent constraint $e_{CC'}$. First, we consider the decoupling of the requirement constraint e_{AB}. As shown in Figure 4, if we do not take advantage of any communication, we can satisfy e_{AB} by imposing two local constraints e_{ZA} and e_{ZB}, since $v_B - v_A = [10, 10] - [0, 5] = [5, 10]$, which implies that $5 \leq v_B - v_A \leq 10$. In Figure 5, if we take the communication opportunity $e_{CC'}$, we can satisfy e_{AB} by imposing two local constraints e_{AC} and $e_{C'B}$. In this case, e_{AC}, $e_{CC'}$ and $e_{C'B}$ also implies the satisfaction of e_{AB}, since $v_B - v_A = (v_B - v_{C'}) + (v_{C'} - v_C) + (v_C - v_A) = [5, 5] + [0, 2] + [0, 3] = [5, 10]$ implies $5 \leq v_B - v_A \leq 10$. This decoupling procedure is called the internalization of external requirement constraints (Casanova et al. 2016).

Second, when communication is involved, as in Figure 5, the agent who receives $v_{C'}$ needs some expectation of when $v_{C'}$ will be received, since it cannot observe v_C. Therefore, we can impose local requirement constraint e_{ZC} and local contingent constraint $e_{ZC'}$, so that the agent is guaranteed to receive $v_{C'}$ some time between 0 and 3. Since $e_{ZC'}$ is a contingent constraint that describes when $v_{C'}$ might occur, $e_{ZC'}$ should cover all possible occurrences of time for $v_{C'}$, and in this case, $v_{C'} - v_Z = (v_{C'} - v_C) + (v_C - v_Z) = [0, 2] + [0, 1] = [0, 3]$, which covers all cases of e_{ZC} and $e_{CC'}$. This decoupling procedure is called the internalization of external contingent constraints (Casanova et al. 2016).

A temporal network can also be represented as a distance graph (Dechter, Meiri, and Pearl 1991), which is a directed graph where a temporal constraint $e_{ij} = \langle v_i, v_j, L_{ij}, U_{ij}\rangle$ induces two linear inequalities $v_j - v_i \leq U_{ij}$ and $v_i - v_j \leq -L_{ij}$, and is represented by two weighted directed edges $v_i \xrightarrow{U_{ij}} v_j$ and $v_j \xrightarrow{-L_{ij}} v_i$. To compute the above internalizations, we solve a mixed-integer linear program (MILP) by looking at the problem in its distance graph. In the following, we introduce continuous variables u_{ij} for each pair of events v_i and v_j that imposes constraint $v_j - v_i \leq u_{ij}$, which represents the weighted directed edge from v_i to v_j.

Internalization of External Requirement Constraints To internalize e_{AB}, we can look at the distance graph in Figure 6 (a). To make the constraints u_{AB} and u_{BA} redundant, the key is to enforce a shorter path that dominates u_{AB} and

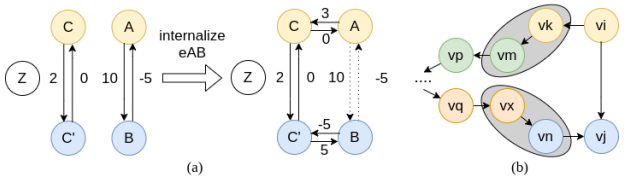

Figure 6: (a) Internalization of e_{AB} in the distance graph. (b) Shorter path though multiple communication links.

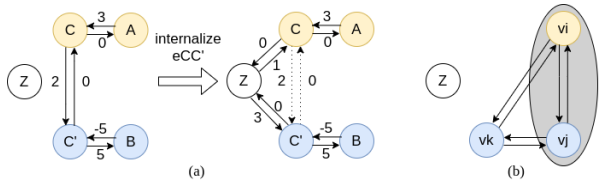

Figure 7: (a) Internalization of $e_{CC'}$ in the distance graph. (b) Choosing an arbitrary local event v_k for internalization.

u_{BA}, either through the reference point v_Z or through the communication link $e_{CC'}$. If we consider using the communication link first, we require that $u_{AC} + u_{CC'} + u_{C'B} \leq u_{AB}$, and $u_{BC'} + u_{C'C} + u_{CA} \leq u_{BA}$. The temporal constraints e_{AB} and $e_{CC'}$ also induce the constraints $u_{AB} \leq U_{AB} = 10$, $u_{BA} \leq -L_{AB} = -5$, $u_{CC'} \geq U_{CC'} = 2$ and $u_{C'C} \geq -L_{CC'} = 0$. Therefore, we can satisfy the linear inequalities by setting $u_{AC} = 3, u_{CC'} = 2, u_{C'B} = 5, u_{BC'} = -5, u_{C'C} = 0, u_{CA} = 0$. Similarly for the case of going through v_Z, we require $u_{AZ} + u_{ZB} \leq u_{AB} \leq U_{AB} = 10$, and $u_{BZ} + u_{ZA} \leq u_{BA} \leq -L_{AB} = -5$.

In general, as shown in Figure 6 (b), when there are multiple agents involved, the shorter path can go through multiple communication links across agents. Given such a path $v_i \to v_k \to v_m \to \cdots \to v_n \to v_j$ that dominates u_{ij}, the linear inequalities to be satisfied is $u_{ik} + u_{km} + \cdots + u_{nj} \leq u_{ij} \leq U_{ij}$, where each u_{pq} either belongs to an agent's local constraints, or is part of a communication link shaded in grey. The case of when no communication links are used becomes a special case, where we choose the path of $v_i \to v_Z \to v_j$, and $u_{iZ} + u_{Zj} \leq u_{ij}$.

Internalization of External Contingent Constraints To internalize the communication link $e_{CC'}$, we illustrate it with the distance graph in Figure 7 (a). We need to replace the communication link with a local contingent constraint that starts from a local event. In this case, we can use the reference event v_Z, and we require that $u_{ZC'} \geq u_{ZC} + u_{CC'}$ and $0 \geq u_{C'Z} \geq u_{CZ} + u_{C'C}$, so that $u_{ZC'}$ and $u_{C'Z}$ covers all possible occurrences of $v_{C'}$. Since $u_{CC'} \geq U_{CC'} = 2$ and $u_{C'C} \geq -L_{CC'} = 0$, we satisfy the linear inequalities by setting $u_{ZC'} = 3, u_{ZC} = 1, u_{CC'} = 2, u_{C'Z} = 0, u_{C'C} = 0, u_{CZ} = 0$.

In general, as shown in Figure 7 (b), to internalize an external contingent link e_{ij}, we can choose any local event $v_k \neq v_j$ to construct a local contingent constraint e_{kj}. Given that we have chosen the event v_k, we need to satisfy that $u_{kj} \geq u_{ki} + u_{ij}$, $0 \geq u_{jk} \geq u_{ik} + u_{ji}$, $u_{ij} \geq U_{ij}$ and $u_{ji} \geq -L_{ij}$. The result is a new internal contingent constraint $e_{kj} = \langle v_k, v_j, -u_{jk}, u_{kj} \rangle$ introduced to the agent that owns v_j. Notice also that this may introduce an additional external requirement constraint between event v_i and v_k, that is, $e_{ki} = \langle v_k, v_i, -u_{ik}, u_{ki} \rangle$, unless $v_k = v_Z$, which needs to be internalized as well.

The problem is combinatorial, since it involves deciding which communication links and which local events to select. We can encode the problem as a MILP, with the list of MILP constraints summarized below. Readers should refer to (Casanova et al. 2016) for more detail. When no communication links exist, the problem is effectively solving for a traditional decoupling solution. Notice that the constraints only guarantee finding a *valid* decoupling solution, rather than enforcing its feasibility.

MILP Formulation Given MaSTNU $\langle \mathcal{A}, \mathcal{V}, \mathcal{E}, \mathcal{C}, f, v_Z \rangle$, the MILP formulation includes the following variables:

(1) Real variables u_{ij} for $v_i, v_j \in \mathcal{V}$, with $u_{ii} = 0, v_i \in \mathcal{V}$.

(2) Boolean variables c_{kj} for $(v_i, v_j, v_k) \in T$, where $T = \{(v_i, v_j, v_k) | e_{ij} \in \mathcal{C}_X, v_k \in \mathcal{V}^{f(v_j)} \backslash \{v_j\}\}$.

(3) Boolean variables b_{ij} for $(v_i, v_j) \in \overline{E_X}$, where $\overline{E_X} = \{(v_i, v_j) | f(v_i) \neq f(v_j), e_{ij} \notin \mathcal{C}_X, e_{ji} \notin \mathcal{C}_X\}$.

(4) Boolean variables z_{ijkl} for $(v_i, v_j, v_k, v_l) \in Q$, where $Q = \{(v_i, v_j, v_k, v_l) | (v_i, v_j) \in \overline{E_X}, (v_k = v_l = v_Z) \vee (f(v_k) = f(v_i) \wedge (e_{kl} \in \mathcal{C}_X \vee e_{lk} \in \mathcal{C}_X))\}$.

(5) Integer variables $h_{ij} \in [0, H]$ for each tuple $(v_i, v_j) \in \overline{E_X}$, where $H = \max(|\mathcal{A}| - 2, |\mathcal{C}_X|)$.

The MILP constraints include:

(1) $\forall (v_i, v_j), u_{ij} + u_{ji} \geq 0$

(2) $\forall e_{ij} \in \mathcal{E}_X, (u_{ij} \leq U_{ij}) \wedge (u_{ji} \leq -L_{ij})$

(3) $\forall e_{ij} \in \mathcal{C}_X, (u_{ij} \geq U_{ij}) \wedge (0 \geq u_{ji} \geq -L_{ij})$

(4) $\forall e_{ij} \in \mathcal{E}_X, b_{ij} = 1 \wedge b_{ji} = 1$

(5) $\forall (v_i, v_j) \in \overline{E_X}, b_{ij} = \sum_{v_k, v_l | (v_i, v_j, v_k, v_l) \in Q} z_{ijkl}$

(6) $\forall (v_i, v_j, v_k, v_l) \in Q, u_{ij} \geq u_{ik} + u_{kl} + u_{lj} + (z_{ijkl} - 1)M$, where M is a large constant

(7) $\forall (v_i, v_j, v_k, v_l) \in Q$ s.t. $(v_l, v_j) \in \overline{E_X}, z_{ijkl} \leq b_{lj}$

(8) $\forall (v_i, v_j, v_k, v_l) \in Q$ s.t. $(v_l, v_j) \in \overline{E_X}, h_{ij} + (1 - z_{ijkl})(H + 1) \geq h_{lj} + 1$

(9) $\forall e_{ij} \in \mathcal{C}_X, \sum_{v_k | (v_i, v_j, v_k) \in T} c_{kj} = 1$

(10) $\forall (v_i, v_j, v_k) \in T, (u_{kj} \geq u_{ki} + u_{ij} + (c_{kj} - 1)M) \wedge (0 \geq u_{jk} \geq u_{ik} + u_{ji} + (c_{kj} - 1)M)$

(11) $\forall (v_i, v_j, v_k) \in T$ s.t. $(v_i, v_k) \in \overline{E_X}, c_{kj} \leq b_{ik}$ and $\forall (v_i, v_j, v_k) \in T$ s.t. $(v_k, v_i) \in \overline{E_X}, c_{kj} \leq b_{ki}$

Distributed Decoupling Algorithm

Previously, a centralized algorithm is proposed (Casanova et al. 2016) that computes an optimal decoupling solution for MaSTNU by encoding both the constraints for finding a valid decoupling and the constraints for ensuring dynamic controllability (Cui and Haslum 2017) of the local networks

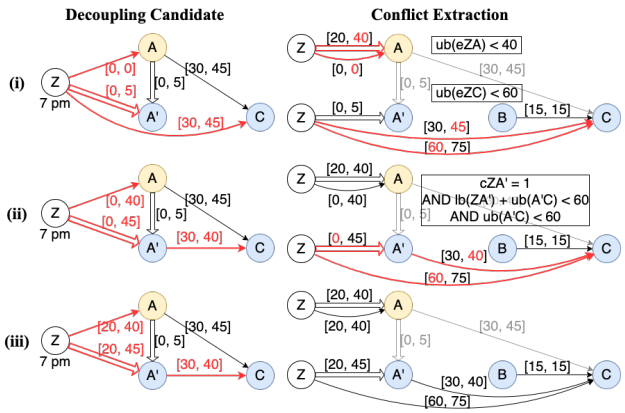

Figure 8: Generate-and-test approach for finding a decoupling solution. Each row represents an iteration of the algorithm. Left column shows the decoupling candidate with decoupling constraints highlighted in red, and right column shows the feasibility checking, with negative cycles highlighted in red. Extracted conflicts are shown in boxes.

Algorithm 1: DistributedDecoupling

Input : external constraints $\mathcal{E}_X, \mathcal{C}_X$, shared events $\mathcal{V}_S$
Output: decoupling solution $\langle \mathcal{E}_d, \mathcal{C}_d \rangle$

1 $conflicts \leftarrow \{\}$
2 $\langle \mathcal{E}_d, \mathcal{C}_d \rangle \leftarrow$ DECOUPLE$(\mathcal{E}_X, \mathcal{C}_X, \mathcal{V}_S, conflicts)$
3 **while** $\langle \mathcal{E}_d, \mathcal{C}_d \rangle \neq None$ **do**
4 $suc \leftarrow True$
5 **foreach** $a \in \mathcal{A}$**, in parallel do**
6 $controllable, conflict \leftarrow$ CHECKDECOUPLING$(a, \langle \mathcal{E}_d^a, \mathcal{C}_d^a \rangle)$
7 **if** $\neg controllable$ **then**
8 $suc \leftarrow False$
9 $conflicts \leftarrow conflicts \cup \{conflict\}$
10 **if** suc **then**
11 **return** $\langle \mathcal{E}_d, \mathcal{C}_d \rangle$
12 $\langle \mathcal{E}_d, \mathcal{C}_d \rangle \leftarrow$ DECOUPLE$(\mathcal{E}_X, \mathcal{C}_X, \mathcal{V}_S, conflicts)$
13 **return** $None$

in a single mixed-integer linear program (MILP). However, when privacy is of concern, and the agents do not want to expose the private parts of their local networks, a privacy-preserving algorithm is desirable. We define what privacy means as follows.

Definition 5 (Privacy)**.** We say that *privacy* is satisfied if each agent only communicates information about its shared events $\mathcal{V}_S^a$, and any temporal constraints scoped on the shared events $\mathcal{V}_S^a$. The *shared events* $\mathcal{V}_S^a \subseteq \mathcal{V}^a$ for an agent a are the set of local events that the agent is willing to share, and must include the reference event v_Z and the set of local events connected to external constraints, that is, $\{v_Z\} \cup (\mathcal{V}^a \cap \mathcal{V}_X) \subseteq \mathcal{V}_S^a$, where $\mathcal{V}_X = \{s | e = \langle s, \cdot, \cdot, \cdot \rangle \in \mathcal{E}_X \cup \mathcal{C}_X\} \cup \{t | e = \langle \cdot, t, \cdot, \cdot \rangle \in \mathcal{E}_X \cup \mathcal{C}_X\}$ is the set of all events connected to external constraints. The rest of the local events are *private events* $\mathcal{V}_P^a = \mathcal{V}^a \backslash \mathcal{V}_S^a$. $\mathcal{V}_S = \cup_{a \in \mathcal{A}} \mathcal{V}_S^a$ is the set of all shared events.

The key to a privacy-preserving algorithm is two-fold. First, we adopt a generate-and-test approach, where a master algorithm iteratively generates a *valid* candidate decoupling solution based on the information on the external constraints $\mathcal{E}_X \cup \mathcal{C}_X$ and the shared events $\mathcal{V}_S$, and agents test the *feasibility* of the candidate independently on their local network, and extract any feedback in the form of conflicts to guide the master algorithm. Second, since the decoupling constraints sent to each agent and the extracted conflicts returned to the master are only scoped on the shared events $\mathcal{V}_S^a$, it also satisfies privacy requirement. Figure 8 illustrates the solving process for our example MaSTNU, where a valid and feasible decoupling solution is found in three iterations.

Note that for a fully distributed algorithm, any agent could take the role of the master to generate the candidate decoupling solutions, as long as the agent has the knowledge of $\mathcal{E}_X \cup \mathcal{C}_X$ and $\mathcal{V}_S$. Additionally, our definition of privacy can be considered as a notion of weak privacy instead of strong privacy. That is, the agents do not exchange private information about the network, but information about the network may potentially be deduced under adversarial context.

Algorithm The distributed algorithm is shown in Algorithm 1. At each iteration, the master algorithm finds a valid candidate decoupling solution $\langle \mathcal{E}_d, \mathcal{C}_d \rangle$ (line 2 and line 12), and sends the corresponding decoupling constraints $\langle \mathcal{E}_d^a, \mathcal{C}_d^a \rangle$ to each agent to check its feasibility, that is, if the decoupling constraints over-constrain its local network (line 6). This step can be done in a distributed and parallel fashion. If the local network is not dynamically controllable, the agent returns a conflict to the master algorithm, which is incorporated in order to generate a new candidate that avoids those conflicts (line 9). If all the local networks are controllable, then the decoupling solution is also feasible, and thus is returned as a solution (line 10 - 11).

For the sub-routine DECOUPLE, a candidate decoupling solution is generated based on the external constraints $\mathcal{E}_X$ and $\mathcal{C}_X$, the set of shared events $\mathcal{V}_S$, and the set of conflicts returned from the agents. This sub-routine solves a MILP problem that includes three sets of constraints: first, the constraints for a valid decoupling solution summarized above, second, encoding for the resolution of conflicts, and third and optionally, any other pre-compiled constraints that can be added in the beginning, instead of being discovered in the iterative process. The MILP formulation is described in more detail later.

The sub-routine CHECKDECOUPLING is shown in Algorithm 2. The agent receives the candidate decoupling constraints $\langle \mathcal{E}_d^a, \mathcal{C}_d^a \rangle$ from the master and checks the controllability of the local network after it is augmented with the decoupling constraints $N_{+\Delta}^a$ (line 1 - 2). For this paper, we require that the dynamic controllability checker supports conflict extraction, such as Bhargava's fast DC checker (Bhargava, Vaquero, and Williams 2017). If the network is not controllable, the checker returns a temporal conflict. The algorithm then compiles the conflict so that it is privacy-

Algorithm 2: CheckDecoupling

Input : local network $N^a = \langle \mathcal{V}^a, \mathcal{E}^a, \mathcal{C}^a \rangle$, decoupling constraints $\langle \mathcal{E}_d^a, \mathcal{C}_d^a \rangle$
Output: boolean $controllable$, conflict $conflict$

1 $N_{+\Delta}^a \leftarrow \langle \mathcal{V}^a, \mathcal{E}^a \cup \mathcal{E}_d^a, \mathcal{C}^a \cup \mathcal{C}_d^a \rangle$
2 $controllable, conflict \leftarrow$ DYNAMICALLYCONTROLLABLE$(N_{+\Delta}^a)$
3 **if** $\neg controllable$ **then**
4 $\quad conflict \leftarrow$ COMPILECONFLICT$(conflict)$
5 **return** $controllable, conflict$

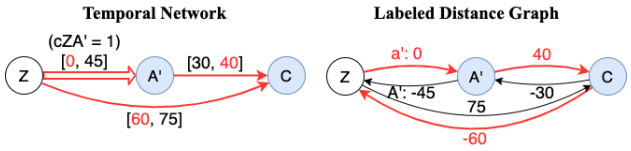

Figure 9: Conflict in STNU and labeled distance graph

preserving and only conveys information about the decoupling constraints, and sends it to master (line 3 - 5). Next, we describe what conflicts are, and how they are made privacy-preserving.

Privacy-Preserving Conflicts In constraint satisfaction problems (CSP), a *minimal conflict* as a minimal set of constraints that make the problem inconsistent. For dynamic controllability of STNUs, a conflict is a subset of the temporal constraints that makes the network uncontrollable. Figure 9 (left) shows an example conflict from iteration (ii) of Figure 8, which is the set of constraints $\{e_{ZA'}, e_{A'C}, e_{ZC}\}$. An STNU is determined as dynamically uncontrollable if a semi-reducible negative cycle is found in its equivalent labeled distance graph (Morris 2006, 2014), as shown in Figure 9 (right). Yu (Yu and Williams 2013) extended the idea of a conflict from a discrete set of constraints to a *continuous conflict* in the temporal domain, expressed as a linear inequality, so that we can express the degree of violation in the negative cycle such as $lb(e_{ZA'}) + ub(e_{A'C}) - lb(e_{ZC}) = -20 < 0$, where $lb(e)$ and $ub(e)$ represents the lower and upper bound of a temporal constraint.

More generally, a *hybrid conflict* (Yu, Fang, and Williams 2014) is a conflict that allows both discrete and continuous elements, that is, a conjunction of linear inequalities and discrete constraints. In general, we need a hybrid conflict to capture the semi-reducible negative cycle from an uncontrollable STNU, as it may have additional supporting conditions. In the example in Figure 9, $ub(e_{A'C}) - lb(e_{ZC}) < 0$ is a necessary condition for the cycle to be semi-reducible, and hence the hybrid conflict is $(lb(e_{ZA'}) + ub(e_{A'C}) - lb(e_{ZC}) < 0) \wedge (ub(e_{A'C}) - lb(e_{ZC}) < 0)$. Additionally, if any temporal constraint is conditional, for example, if $e_{ZA'}$ has a guard condition $c_{ZA'} = 1$, meaning that $e_{ZA'}$ only needs to hold if the decoupling candidate imposes such a contingent constraint by assigning $c_{ZA'} = 1$, then such a guard $c_{ZA'} = 1$ is also included in the conjunction as a discrete constraint. Hybrid conflicts in the temporal domain have proven useful in solving the temporal relaxation problems (Yu, Fang, and Williams 2014), but this is the first time it is applied to the temporal decoupling problem.

In our algorithm, the dynamic controllability checker extracts a hybrid conflict, which may involve local constraints that are private information. Therefore, the conflict is processed by COMPILECONFLICT (line 4) before sending to the master. For example, the extracted conflict from the checker $(lb(e_{ZA'}) + ub(e_{A'C}) - lb(e_{ZC})) < 0) \wedge (ub(e_{A'C}) - lb(e_{ZC}) < 0)$ in Figure 9 is compiled to $(c_{ZA'} = 1) \wedge (lb(e_{ZA'}) + ub(e_{A'C}) < 60) \wedge (ub(e_{A'C}) < 60)$ as shown in Figure 8 (ii).

In order to compile an extracted conflict into a privacy-preserving one as desired, we need to add any necessary discrete guards and compile out the private constraints. First, any contingent constraint e_{kj} in the conflict that is also a decoupling constraint received from the master needs to include its guard condition $c_{kj} = 1$ in the hybrid conflict. This is because e_{kj} only needs to hold when the master assigns $c_{kj} = 1$ when internalizing some external contingent constraint e_{ij}. In our example, since $e_{ZA'}$ is a contingent constraint and also a decoupling constraint, $c_{ZA'} = 1$ is included. Second, we project out any constraints in the conflict that are not part of the decoupling constraints by replacing the corresponding terms by constant values. For example, by replacing $lb(e_{ZC})$ by its value 60, we obtain the projected linear inequalities $(lb(e_{ZA'}) + ub(e_{A'C}) < 60) \wedge (ub(e_{A'C}) < 60)$, which includes only the decoupling constraints $e_{ZA'}$ and $e_{A'C}$.

Finally, the compiled conflict is a tuple $\langle \mathcal{O}, \mathcal{I} \rangle$, where $\mathcal{O}$ and $\mathcal{I}$ in conjunction form the hybrid conflict. Each $c \in \mathcal{O}$ is a constraint of the form $c_{kj} = 1$, where c_{kj} is a reference to the corresponding variable defined in MILP Formulation. Each $ineq \in \mathcal{I}$ is a linear inequality of the form $\sum_{e_{ij} \in \mathcal{E}_d^a \cup \mathcal{C}_d^a} \pm bound(e_{ij}) < N$, where $bound$ can be either the lb or the ub function, and N is a constant.

Solving Decoupling as a MILP Problem We encode the temporal decoupling problem as a MILP, which includes constraints (1) - (11) summarized before, encoding for the resolution of conflicts, and any other pre-compiled constraints. For constraints (1) - (11), it suffices to apply those linear inequality constraints to the shared network made up of the shared events and the external constraints, that is, $\langle \mathcal{V}_S, \mathcal{E}_X, \mathcal{C}_X \rangle$. We now describe how to encode the other two sets of constraints.

(1) Encoding Conflicts To ensure the decoupling solution avoids any conflicts, we encode the negation of conflicts in our MILP formulation. A hybrid conflict returned by an agent is a tuple $\langle \mathcal{O}, \mathcal{I} \rangle$, which can be resolved if the conjunction is negated, that is, if any of the conjuncts in $\mathcal{O}$ or $\mathcal{I}$ is negated. More specifically, a constraint $c_{kj} = 1$ in $\mathcal{O}$ can be negated by setting $c_{kj} = 0$. For a linear inequality in $\mathcal{I}$, we can rewrite it so that it is expressed in variables u_{ij} as defined in the MILP formulation, by substituting $lb(e_{ij}) = -u_{ji}$ and $ub(e_{ij}) = u_{ij}$. The linear inequality then becomes $\sum_{u_{ij}} \pm u_{ij} < N$. The negation of such an inequality is $\sum_{u_{ij}} \pm u_{ij} \geq N$. To put it in logical statement,

we need to enforce:

$$\Big(\bigvee_{(c_{kj}=1)\in\mathcal{O}} c_{kj}=0\Big)\vee\Big(\bigvee_{(\sum_{u_{ij}}\pm u_{ij}<N)\in\mathcal{I}}\sum_{u_{ij}}\pm u_{ij}\geq N\Big)$$

To encode it in a mixed-integer linear program, we introduce an indicator variable ind for each linear inequality constraint $\sum_{u_{ij}}\pm u_{ij}\geq N$ to indicate when the constraint should hold. Additionally, since c_{kj} is a boolean variable, at least one of the indicator variables or the negation of c_{kj} should hold. The above logical statement can be encoded as:

- $\forall(\sum_{u_{ij}}\pm u_{ij}<N)_k\in\mathcal{I},\sum_{u_{ij}}\pm u_{ij}\geq N+(ind_k-1)M$, where $(\sum_{u_{ij}}\pm u_{ij}<N)_k$ is the kth inequality constraint in $\mathcal{I}$, ind_k is the auxiliary boolean indicator variable, and M is a large constant.
- $\sum_{c_{kj}|(c_{kj}=1)\in\mathcal{O}}(1-c_{kj})+\sum_k ind_k\geq 1$.

(2) Pre-compiled Constraints Since each iteration of the algorithm aims to discover conflicts, we can potentially improve the efficiency by adding some universal constraints, or any pre-compiled constraints from the agents to avoid discovering less informative conflicts. For example, one universal constraint that should hold is the shortest path constraint $\forall v_i,v_j,v_k\in\mathcal{V}_S^a,u_{ij}\leq u_{ik}+u_{kj}$ for each agent. Additionally, the agents may choose to pre-compile a set of abstracted constraints scoped on V_S^a that must hold. For example, our agent Bob can inform the master generator that $60\leq v_C-v_Z\leq 75$. There may be a variety of compilation strategies, such as using bucket elimination to project the local network onto a subset of the shared network. We will not go into details for the scope of this paper.

Objective Function We can define the objective function for the MILP problem according the desired condition of optimality, or simply find a feasible solution. Examples of objective functions include maximizing temporal flexibility $\sum_{i<j}u_{ij}+u_{ji}$ (Hunsberger 2002), maximizing the minimum normalized flexibility for any individual, i.e. $\min_{a\in\mathcal{A}}\frac{1}{|\mathcal{V}^a|}\sum_{i,j\in\mathcal{V}^a,i<j}(u_{ij}+u_{ji})$ (Casanova et al. 2016). It is important to note that in the distributed setting, the optimal decoupling is considered at the abstracted level of the shared network, instead of considering the entirety of all of agents' local networks.

Post-Processing MILP Solution When the solver returns a solution, it provides an assignment to each variable in the MILP Formulation, including $u_{ij},c_{kj},b_{ij},z_{ijkl},h_{ij}$. To extract the decoupling solution between iterations, we take the simplest approach of:

- For each (v_i,v_j) and $f(v_i)=f(v_j)=a$, such that there exists no c_{ij} assigned to 1, create a simple temporal constraint $e=\langle v_i,v_j,-u_{ji},u_{ij}\rangle$ for agent a. If $c_{ij}=1$, create a contingent constraint instead.

Notice that the compiled decoupling constraints may not all be necessary. Since decoupling constraints in general reduce the overall flexibility of the network, we extract a minimal and necessary set of decoupling constraints at the last iteration when a valid and feasible decoupling solution is found. We can do so by going through each external requirement or contingent constraint, checking the variables z_{ijkl}, c_{kj} to identify the path that internalizes the constraint, and only record those decoupling constraints on the path.

Discussion & Experiments

While MaSTNU is a powerful framework for modeling distributed multi-agent execution, two current limitations of the centralized approach (Casanova et al. 2016) may hinder its practical adoption: (1) the centralized approach implicitly requires centralized knowledge of the local networks, (2) because of the combinatorial nature of the problem, solving the MILP problem tends to be slow. Our proposed algorithm resolves the first limitation since it is privacy-preserving. We also show that it mitigates the second limitation in the following experiments, by showing its runtime improvement over the centralized approach for loosely coupled MaSTNU problems with large local networks that are mostly private.

Experiments

We compare the runtime of the distributed algorithm (dis) and the centralized MILP approach ($milp$). We use Gurobi as the solver, and the experiments are run on a machine with 8 GB RAM and 2.4 GHz CPU. The experiments are run on a set of randomly generated MaSTNUs, parameterized by the number of agents N_a, the number of local activities N_{local}, the number of external requirement constraints N_{req}, and the number of external contingent constraints N_{cont}. Each local network is dynamically controllable, with N_{local} contingent constraints of bound [0, 1 - 5], representing local activities, and N_{local} requirement constraints of bound [0, 5 - 20] randomly connecting local events. Each external requirement constraint randomly connects two events from two agents, with lower bound 0 and an upper bound sampled from {10, 20, 30, 40, +inf}. Each external contingent constraint randomly samples an event v from an agent, and creates a corresponding received event v' at another sampled agent with a bound of [0, 1]. The shared events include v_Z and $\mathcal{V}_X$. The objective function maximizes $\sum_{i<j}u_{ij}+u_{ji}$. Notice that we simulate the distributed algorithm by running it in a sequential fashion, and no pre-compilation techniques are used. Each case is run over 30 samples with a 3-minute timeout, where we show the average runtime.

Varying Local Network Size With $N_a=2$, $N_{req}=3$, we vary the size of the local networks by changing N_{local} under $N_{cont}\in[0,5]$. The result in Figure 10 shows that as N_{local} increases, the distributed algorithm has an obvious advantage over the centralized MILP algorithm, which mostly timed out when N_{local} is large with the curves leveling off at the 180-seconds timeout. Overall, we observe that the runtime of the distributed algorithm is minimally affected by the size of the local networks N_{local} compared to the centralized approach, when the shared network is fixed. The performance gain is attributed to the separation of the dynamic controllability checking process from the core decoupling solving process, and the use of state-of-the-art $O(n^3)$ dynamic controllability checkers, whereas the

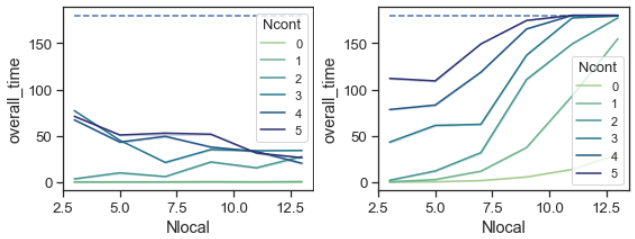

Figure 10: Overall runtime in relationship to N_{local}, under varying N_{cont}. Left: distributed. Right: centralized

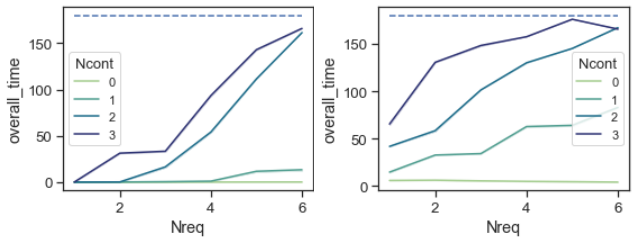

Figure 11: Overall runtime in relationship to N_{req}, under varying N_{cont}. Left: distributed. Right: centralized

centralized approach encodes a single MILP problem using Cui's MILP dynamic controllability formulation (Cui and Haslum 2017).

Varying Number of Agents With $N_{local} = 6$, $N_{req} = 3$, we vary the number of agents $N_a \in [2, 7]$ under $N_{cont} \in [0, 5]$. The results (figure not shown) show that when $N_{cont} = 0$, the overall runtime for centralized MILP grows at a linear rate, whereas the distributed algorithm is barely affected. However, with a larger N_{cont}, we observe that the overall runtime decreases as N_a increases for both algorithms. We conjecture that this is because with fixed N_{cont}, the fewer the agents, the denser the external constraints are, which increases the complexity of the core decoupling problem that ended up dominating the runtime. The relatively small size of the local network $N_{local} = 6$ also means that the feasibility condition of the decoupling solution contributes less to the runtime. While the empirical results are inconclusive, we think that theoretically, by distributing the feasibility checking process, the distributed algorithm should provide benefits as the number of agents grows in problems where ensuring the feasibility condition is dominating the runtime.

Varying Number of External Constraints With $N_a = 2$, $N_{local} = 9$, we vary the number of external requirement constraints N_{req} under $N_{cont} \in [0, 3]$. As seen in Figure 11, while the distributed algorithm shows smaller runtime overall, its runtime starts to grow at a faster rate when $N_{cont} = 2$. The distributed algorithm may be less effective with a larger N_{cont}, as there tend to be more hybrid conflicts discovered and more iterations to go through.

Relationship with Number of Conflicts We also characterize how the runtime of the distributed algorithm depends on the number of conflicts discovered, which roughly corresponds to how many iterations the algorithm has to go through. We use the data from the first experiment. Figure 12 (left) shows that there is a strong positive correlation between runtime and the number of conflicts. When N_{cont} is larger, the rate of increase is larger, which may be due to the increase in the complexity of the core decoupling MILP problem resulting in the increase in the solving time for each iteration, as evidenced by Figure 12 (right). Additionally, Figure 12 (right) shows that a single iteration solving time takes much less time than the centralized MILP solving time, with 93% of the cases having a last-iteration solving time under 0.5 seconds.

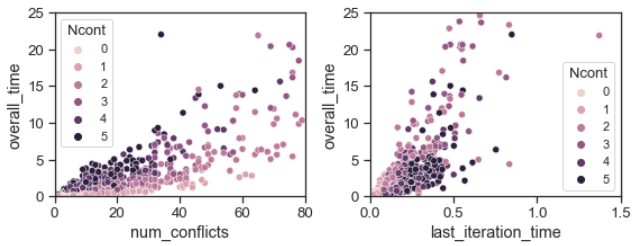

Figure 12: (Left) Overall runtime VS number of conflicts, showing 95% of the data points. (Right) Overall runtime VS last-iteration runtime, showing 96% of the data points.

Discussion

When no communication is allowed, the distributed algorithm almost always performs better than the centralized approach. In this case, it can be considered as an extension to the previous distributed decoupling algorithms by allowing local networks to be STNUs rather than STNs (Boerkoel Jr and Durfee 2013; Mogali, Smith, and Rubinstein 2016). However, the distributed algorithm still does not solve the scalability problem of MaSTNUs with larger N_{cont} and N_{req}. The distributed algorithm is thus more suitable for loosely coupled MaSTNUs where the agents' local networks are large but mostly private.

Conclusion

We proposed a privacy-preserving distributed temporal decoupling algorithm for the MaSTNU problems with two contributions. First, it does not assume centralized knowledge of the MaSTNU, and preserves the agents' privacy on their local networks. Second, we showed empirically that it has significant performance gain in runtime over the centralized approach (Casanova et al. 2016) on MaSTNU problems where agents' local networks are loosely coupled and mostly private. The MaSTNU framework is powerful, and we take one step further to its practical application in modeling multi-agent execution.

Future work can look into approaches providing stronger privacy guarantees, and extending the generate-and-test decoupling framework to allow richer types of local plan representations such as temporal plan networks (TPN) (Kim, Williams, and Abramson 2001), as long as solvers exist that support similar notions of conflict extraction.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0035. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

References

Bhargava, N.; Muise, C.; Vaquero, T.; and Williams, B. 2018. Delay Controllability: Multi-Agent Coordination under Communication Delay. Technical report, MIT Computer Science and Artificial Intelligence Laboratory.

Bhargava, N.; Vaquero, T.; and Williams, B. 2017. Faster Conflict Generation for Dynamic Controllability. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*.

Boerkoel Jr, J. C.; and Durfee, E. H. 2013. Distributed reasoning for multiagent simple temporal problems. *Journal of Artificial Intelligence Research* 47: 95–156.

Casanova, G.; Pralet, C.; Lesire, C.; and Vidal, T. 2016. Solving dynamic controllability problem of multi-agent plans with uncertainty using mixed integer linear programming. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, 930–938. IOS Press.

Cui, J.; and Haslum, P. 2017. Dynamic controllability of controllable conditional temporal problems with uncertainty. In *27th International Conference on Automated Planning and Scheduling (ICAPS 2017)*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial intelligence* 49(1-3): 61–95.

Hunsberger, L. 2002. Algorithms for a temporal decoupling problem in multi-agent planning. In *AAAI/IAAI*.

Kim, P.; Williams, B. C.; and Abramson, M. 2001. Executing Reactive, Model-Based Programs through Graph-Based Temporal Planning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 487–493.

Mogali, J. K.; Smith, S. F.; and Rubinstein, Z. B. 2016. Distributed decoupling of multiagent simple temporal problems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 408–415.

Morris, P. 2006. A structural characterization of temporal dynamic controllability. In *International Conference on Principles and Practice of Constraint Programming*, 375–389. Springer.

Morris, P. 2014. Dynamic controllability and dispatchability relationships. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, 464–479. Springer.

Vidal, T. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence* 11(1): 23–45.

Yu, P.; Fang, C.; and Williams, B. C. 2014. Resolving Uncontrollable Conditional Temporal Problems Using Continuous Relaxations. In *Proceedings of the Twenty-Fourth International Conferenc on International Conference on Automated Planning and Scheduling*, 341–349.

Yu, P.; and Williams, B. C. 2013. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In *Twenty-Third International Joint Conference on Artificial Intelligence*.