

# Optimal Planning for Timed Partial Order Specifications

Kandai Watanabe<sup>1</sup>, Georgios Fainekos<sup>2</sup>, Bardh Hoxha<sup>2</sup>, Morteza Lahijanian<sup>1</sup>,  
Hideki Okamoto<sup>2</sup>, and Sriram Sankaranarayanan<sup>1</sup>

**Abstract**—This paper addresses the challenge of planning a sequence of tasks to be performed by multiple robots while minimizing the overall completion time subject to timing and precedence constraints. Our approach uses the Timed Partial Orders (TPO) model to specify these constraints. We translate this problem into a Traveling Salesman Problem (TSP) variant with timing and precedent constraints, and we solve it as a Mixed Integer Linear Programming (MILP) problem. Our contributions include a general planning framework for TPO specifications, a MILP formulation accommodating time windows and precedent constraints, its extension to multi-robot scenarios, and a method to quantify plan robustness. We demonstrate our framework on several case studies, including an aircraft turnaround task involving three Jackal robots, highlighting the approach’s potential applicability to important real-world problems. Our benchmark results show that our MILP method outperforms state-of-the-art open-source TSP solvers OR-Tools.

## I. INTRODUCTION

Workflow analysis and optimization techniques are of crucial importance in increasing efficiency across many domains, from manufacturing to administrative processes. These workflows are conventionally structured around tasks that have to be completed subject to precedence and timing constraints. Such constraints are often defined by the user or inferred from demonstrations [30], [27]. A recently-proposed model, *Timed Partial Order* (TPO) [30], provides a succinct, understandable, and easily analyzable representation of these timing constraints. It allows for precedence constraints using partial orders, and the timing constraints are specified using clocks that can be reset when events in the workflow happen. However, the problem of planning task sequences given environmental constraints and TPO specifications has not been solved. The main challenge lies in the combination of environmental constraints that specify how the events in the workflow can be achieved, timing costs for various events, and the TPO. All of these are to be taken into account while planning for such agents. In this work, we develop a planning framework with TPO specifications against environments modeled as deterministic transition systems.

Consider the agent operating in an aircraft turnaround task in Figure 1. Many tasks such as bulk loading/unloading and refueling can be completed in parallel. Additionally, there are precedence and timing constraints. For example, deplaning can only be done after the stair truck is placed, and deplaning

<sup>1</sup>Authors are with the Departments of Computer Science and Aerospace Engineering Sciences at University of Colorado Boulder, Boulder, Colorado. [first.lastname@colorado.edu](mailto:first.lastname@colorado.edu)

<sup>2</sup>Authors are with Toyota Motor North America, Research and Development. [first.lastname@toyota.com](mailto:first.lastname@toyota.com)



(a) Top-down view (b) Gridworld

Fig. 1: Aircraft Turnaround Example

takes at least 15 minutes. Catering can only begin after all passengers have deplaned. To this end, we need a plan synthesis algorithm that can find the minimum-makespan plan under the timing and partial-order constraints.

In this paper, we propose a new framework to solve the time-constrained plan synthesis problem for a single as well as multiple robots. We employ TPOs as task specifications and use Deterministic Transition Systems (DTS) as abstraction models of the robots in their environments. We show that the planning problem on DTS with TPO specifications reduces to a type of Traveling Salesman Problem (TSP), which asks, given a map of cities, to find the lowest cost path to visit every city once. TSP is well-studied and known to be NP-hard [9], but there exist tools that use highly-effective heuristics, allowing fast computations [5]. We formulate our DTS with TPO planning problem as an instance of TSP with the addition of timing and precedent constraints [24]. These constraints introduce the difficulty of applying off-the-shelf heuristics to our problem. Instead, we solve the problem as a Mixed Integer Linear Program (MILP), which finds an optimal solution rapidly for some of our benchmarks. We also show that for the multi-robot setting, a slight modification of the MILP can be employed. Furthermore, we provide an efficient approach to robustness analysis of the synthesized plans.

The contributions of this work are fourfold: we (i) introduce a general planning framework for TPO specifications that can be applied to one or more robots, (ii) formulate a MILP with time windows (global time with respect to the start event) and precedence constraints (local time between sub-tasks), and extend it to multiple robots, (iii) propose a method based on LP to quantify the robustness of the synthesized plans to capture the lower and upper bounds on the delays that the plan can tolerate w.r.t. the given TPO, and

(iv) provide a set of illustrative case studies and benchmarks that empirically show that TPO constraints actually narrow down the search space, speed up the computation time, and enable scaling up the algorithm to 160 nodes and 40 robots. We perform a physical experiment for an aircraft turnaround task with three Jackal robots that demonstrates the ability of our approach to plan for practically relevant problems.

## II. RELATED WORK

Simple Temporal Networks (STNs) [6] and Timed Automata (TA) [1] allow us to specify complicated timing specifications. A comparison of TPOs with related formalisms is provided by Watanabe et al [30].

*Model Checking Problem* [1]: The problem focuses solely on properties such as consistency of the specification model, neglecting the operating environment. The focus of this paper involves plan synthesis which combines the timing specification and the model of the operating environment.

*Plan Synthesis Problem*: The problem is to find a plan in the operating environment that satisfies the specification. This is a common problem in robotics, and many studies have been conducted for TAs [3] and Signal Temporal Logic (STL) [19] which can be translated into TAs [20]. In fact, the synthesis algorithm is known to be exponential in the number of clocks (Lemma 4.5 and Theorem 7.8 of [1]) and blows up very quickly in the number of tasks and environmental states. This makes the algorithm difficult to apply to larger instances and multi-agent systems [28]. In this paper, we focus on TPOs that can be turned into linear inequality constraints, which reduce the complexity of the problem.

*Task & Motion Planning Problem*: The problem is to assign (heterogeneous) tasks to robots, involving the problem of coalition formation [11]. The work in [29] tackles the task-allocation with timing and precedent constraints by forming coalitions of robots to accomplish tasks more efficiently. Similarly, [10] tackles the coalition formation problem to assign tasks to robots via a min-cost network flow approach. Our focus is not on allocating (heterogeneous) tasks to (heterogeneous) robots via forming coalitions, but rather it is on scheduling tasks that satisfy given formal specifications (e.g., precedence order, timing constraints, etc.). Others have also formulated the task allocation problem as a Traveling Salesman Problem (see [4] for a short survey), but without precedence and timing constraints. As highlighted in survey [23], some works have considered timing and precedent constraints along with others (multi-agent, hard/soft constraints, deterministic/stochastic, etc.). However, no work has looked into local timing dependencies, and more importantly, into tasks that can be performed at multiple different locations. The latter creates a choice of not only which robot should perform the task, but also in which location (for which we need to formulate a Generalized TSP).

## III. PROBLEM FORMULATION

In this section, we first introduce TPOs and present the single/multi-robot problem formulations.

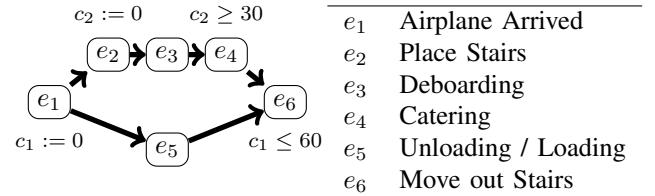


Fig. 2: Partial Order for an aircraft turnaround task. Events  $e_1, \dots, e_5$  represent events such as “airplane arrived” ( $e_1$ ) or the commencement of various tasks such as “Catering” ( $e_4$ ).

### A. Timed Partial Order (TPO) Specification

TPOs provide a simple yet useful model for specifying timing constraints. It is closely related to timed automata [2] and simple temporal networks (STNs) [6]. We provide a brief summary of TPOs, referring the reader to the paper by Watanabe et al for further details [30].

TPOs specify a timed sequence of *events*. These events may include the start and finish of a given task, or an environmental event (e.g., the temperature of the water has exceeded 100°C). The syntax of TPOs as defined in [30] does not allow an event to be repeated. Hence, we assume that repetitions of events are *disambiguated* by giving them unique labels.

Let  $\Pi = \{e_1, \dots, e_n\}$  be the set of events. A *timed trace*  $\tau : \Pi \rightarrow \mathbb{R}_{\geq 0}$  is a mapping  $\tau = \{e_1 \mapsto t_1, e_2 \mapsto t_2, \dots, e_n \mapsto t_n\}$ , wherein  $t_i \geq 0$  denotes the timestamp for event  $e_i \in \Pi$ . Timed partial orders specify which timed traces are feasible (allowed) and which ones are not.

Recall that a (*strict*) *Partial Order* (*PO*)  $P$  is a relation  $\prec$  on a set  $\Pi$  that is irreflexive, asymmetric, and transitive. We write  $e_i \preceq e_j$  if  $e_i \prec e_j$  or  $i = j$ . If  $e_i \preceq e_j$  holds, then for any timed trace  $\tau$  we require that  $\tau(e_i) \leq \tau(e_j)$ .

**Definition 1** (TPO). A timed partial order (TPO) is specified by a directed-acyclic graph (DAG)  $\varphi : (\Pi, \prec)$  describing a strict partial order over  $\Pi$  augmented with the following:

- 1) A finite set of clocks  $C = \{c_1, \dots, c_m\}$ ,
- 2) A guard map  $g$  that maps each event  $e_i$  to a guard condition, which is a conjunction of the form  $g(e_i) : \bigwedge_{j=1}^{n_i} c_j \bowtie a_j$ , wherein  $c_j \in C$  denotes a clock,  $\bowtie \in \{\leq, \geq\}$ , and  $a_j \in \mathbb{R}_{\geq 0}$  is a non-negative constant, and
- 3) A reset map  $R : \Pi \rightarrow 2^C$  that associates each event  $e_i$  with a subset of clocks  $R(e_i) \subseteq C$  that are to be reset to 0 whenever event  $e_i$  is encountered.

We initialize all the clocks to 0 and the clocks simply run to measure time. The edges in the TPO specify a precedence ordering between events. Thus, if the TPO has an edge  $e_i \rightarrow e_j$ , we require for any timed trace  $\tau$  satisfying the TPO that  $\tau(e_i) \leq \tau(e_j)$ . Similarly, TPOs associate constraints over the clocks on each node. These constraints must hold whenever an event  $e_i$  occurs and the reset actions corresponding to the events are executed to update the clock values.

**Example 1.** Figure 2 shows the TPO specification for the aircraft turnaround task depicted in Figure 1. The TPO expresses many partial order constraints. For example, stairs must be placed (event  $e_2$ ) before passengers deboard event

( $e_3$ ). After deboarding, the catering service can be started ( $e_4$ ). Meanwhile, bulk unloading and loading ( $e_5$ ) can be done in parallel. Only after everything is completed, the stairs can be removed ( $e_6$ ). The clocks enforce additional timing constraints. For instance, clock  $c_1$  is reset when we encounter event  $e_1$  and event  $e_6$  has the constraint  $c_1 \leq 60$ . This enforces the constraints that the stairs must be moved out within 60 time units of the aircraft arriving. Likewise, another clock  $c_2$  expresses the constraint that catering cannot be started until at least 30 minutes after the plane arrives (to allow sufficiently many passengers time to de-board).

Watanabe et al [30] showed that TPOs  $\varphi$  can be translated into a conjunction of the inequality forms,

$$\varphi : \bigwedge_{i,j \text{ s.t. } e_i \prec e_j} (t_j - t_i) \in [\ell_{j,i}, u_{j,i}] \wedge \bigwedge_{j=1}^n t_j \in [a_j, b_j], \quad (1)$$

wherein  $\ell_{j,i} \geq 0, a_j \geq 0$  form lower bounds and  $u_{j,i}, b_j \in \mathbb{R}_{\geq 0} \cup \{\infty\}$  are upper bounds that can be non-negative real numbers as well as  $+\infty$ . The calculation of these bounds can be easily automated but is not explained further here. Specifying constraints as inequalities is a simple way to specify the relationship between events.

### B. Single Robot Setting

In this work, we consider both single and multi-robot cases. For the single agent setting, we assume the agent operates deterministically in an environment. Abstractions can be made to represent it as a discrete Deterministic Transition System (DTS), similar to [15], [16], [17].

**Definition 2** (DTS). A single-robot deterministic transition system (DTS) is a tuple  $\mathcal{T} = (X, A, x_0, \delta_T, \Pi, L)$ , where

- $X$  is a finite set of states,
- $A$  is a finite set of controls or actions,
- $x_0 \in X$  is the initial state,
- $\delta_T : X \times A \rightarrow X$  is the (partial) transition function,
- $\Delta_T : X \times A \rightarrow \mathbb{R}_{\geq 0}$  is the transition duration function,
- $L : X \rightarrow \Pi \cup \{\emptyset\}$  is a labeling function that maps each state to an event or an empty set. Without loss of generality, we assume  $L(x_0) = \emptyset$ .

A plan  $\gamma = \gamma_0 \gamma_1 \dots \gamma_{n-1}$  is a sequence of actions, where  $\gamma_i \in A$  for all  $0 \leq i \leq n-1$ . A valid plan is plan  $\gamma$  that respects the transition function  $\delta_T$ , i.e.,  $\delta_T(s_i, \gamma_i)$  exists for all  $0 \leq i \leq n-1$ . We denote the set of all valid plans by  $\Gamma$ . By executing  $\gamma \in \Gamma$ , the robot generates a trajectory  $s^\gamma = s_0^\gamma s_1^\gamma \dots s_n^\gamma$ , where  $s_0^\gamma = x_0$  and  $s_{i+1}^\gamma = \delta_T(s_i^\gamma, \gamma_i)$ . The observation trace of a trajectory is the sequence of observed labels, i.e.,  $\rho^\gamma = L(s_0^\gamma) L(s_1^\gamma) \dots L(s_n^\gamma)$ . The duration of a trajectory is a sum of the transition durations  $D(s^\gamma) = \sum_{i=0}^{n-1} \Delta_T(s_i^\gamma, \gamma_i)$ . This induces a timed trace  $\tau^\gamma = (L(s_0^\gamma), t_0) \dots (L(s_n^\gamma), t_n)$  where  $t_i = D(s_0^\gamma, \dots, s_i^\gamma)$ , which is equivalent to the definition of the timed trace in TPO. We say plan  $\gamma \in \Gamma$  satisfies a TPO if its timed trace  $\tau^\gamma$  satisfies the inequalities  $\varphi$  in (1), denoted by  $\tau^\gamma \models \varphi$ .

**Example 2.** Again consider the aircraft turnout example in Figure 1b with the red agent in position coordinate  $(1, 17)$ .

It can be modeled as a transition system, where each cell of the grid is associated with a state  $x \in X$ . The agent can take actions  $A = \{\text{up, down, right, left, }\}$ . Then, plan  $\gamma = \text{down down}$  generates trajectory  $s^\gamma = (1, 17)(1, 16)(1, 15)$ , which induces timed trace  $\tau^\gamma = (\emptyset, 0)(\emptyset, 1)(e_{\text{red, floor}}, 2)$ .

Note that DTS can model a variety of different systems including robotic manipulators [13], [12].

**Problem 1.** (Single Robot Plan Synthesis) Given a robotic system as a DTS  $\mathcal{T}$  with a specification as a TPO  $\varphi$ , synthesize a valid plan  $\gamma^* \in \Gamma$  for the robot that satisfies the TPO in a minimum time duration, i.e.,

$$\gamma^* = \arg \min_{\gamma \in \Gamma} D(s^\gamma) \quad \text{s.t.} \quad \tau^\gamma \models \varphi$$

We further extend the problem to a multi-agent system.

### C. Multi-Robot Setting

We consider the same environment as above but now with  $p \in \mathbb{N}_{>1}$  robots, each with its own initial state. We define a multi-robot DTS (mDTS) by extending  $\mathcal{T}$  to have a set of initial states  $X_I = \{x_0^1, x_0^2, \dots, x_0^p\} \subseteq X$ , i.e.,  $\mathcal{T}^M = (X, A, X_I, \delta_T, \Pi, L)$ , where  $X, A, \delta_T, \Pi$ , and  $L$  are as in Def. 2. The notion of valid plan  $\gamma^i \in \Gamma^i$  for robot  $i$  is adopted from the single case, and the set of all valid plans for all robots is denoted by  $\Gamma = \bigcup_{i=1}^p \Gamma^i$ .

Similar to the single robot case, from initial state  $x_0^i \in X_I$ , plan  $\gamma^i$  induces a trajectory  $s^{\gamma^i}$ , and a timed trace  $\tau^{\gamma^i}$ , and timestamps  $t^{\gamma^i}$ . We assume that when a robot executes action  $a \in A$  at state  $x \in X$ , it remains in  $x$  for the entire duration  $\Delta_T(x, a)$  before transitioning to  $x' = \delta_T(x, a)$ . Then, the induced trajectory can be viewed as a piecewise function of time. With an abuse of notation, we use  $s^{\gamma^i} : \mathbb{R}_{\geq 0} \rightarrow X$  to denote this function, where  $s^{\gamma^i}(t)$  is the state visited by trajectory  $s^{\gamma^i}$  at time  $t$ . We say two trajectories  $s^{\gamma^1}$  and  $s^{\gamma^2}$  are *non-colliding* if, for all  $t \leq \max\{D(s^{\gamma^1}), D(s^{\gamma^2})\}$ ,  $s^{\gamma^1}(t) \neq s^{\gamma^2}(t)$ . We define a timed trace  $\tau^{\gamma^1 \dots \gamma^p}$  of the multi-robot system to be the union of the individual robot's timed traces. A timed trace is also viewed as a set with the order relation induced by the timestamps.

Then, the multi-robot problem is to find plans for the  $p$  robots that generate non-colliding trajectories with the timed trace  $\tau^{\gamma^1 \dots \gamma^p}$  that satisfies the TPO specification with a minimum time duration.

**Problem 2** (Multi-Robot Plan Synthesis). Given a system of  $p$  robots as a mDTS  $\mathcal{T}^M$  and a TPO specification  $\varphi$ , synthesize plans  $\gamma^{1*}, \dots, \gamma^{p*} \in \Gamma$  under which the multi-robot system satisfies the TPO in a minimum time duration:

$$\gamma^{1*}, \dots, \gamma^{p*} = \arg \min_{\gamma^1 \dots \gamma^p \in \Gamma} \max\{D(s^{\gamma^1}), \dots, D(s^{\gamma^p})\}$$

subject to

$$\tau^{\gamma^1 \dots \gamma^p} \models \varphi$$

$s^{\gamma^j}$  and  $s^{\gamma^j}$  are non-colliding  $\forall \gamma^i, \gamma^j \in \Gamma$ .

Note that Problems 1 and 2 ask for a satisfying plan that minimizes the maximum duration of the induced trajectories, which is also known as the *makespan* of the plan.

#### IV. APPROACH

In this section, we explain how a Problem 1 can be formulated as an instance of the Generalized Traveling Salesman Problem (GTSP), but with timing and precedent constraints. These additional constraints introduce difficulty in applying existing heuristics to our problem out of the box. In this paper, we first focus on the Mixed Integer Linear Program (MILP) formulation. We first introduce the problem of GTSP and later discuss how the problem can be translated into the graph representation of GTSP.

##### A. Problem 1 as Generalized Traveling Salesman Problem

The Generalized Traveling Salesman Problem [22] is the problem of finding the minimum cost path that visits exactly one city from given subsets of cities. GTSP is known to be an NP-hard problem, and it is a well-studied problem in combinatorial optimization research. There are many existing methods to obtain either exact or approximate solutions to this problem [25]. We want to translate Problem 1 into a GTSP, more specifically, GTSP with Time-Windows and Precedence Relations (GTSP-TWPR) [21] so that we can utilize the existing approaches and extend the problem formulation. Formally, GTSP-TWPR is defined as follows.

**Definition 3** (GTSP-TWPR). *Let  $G = (V, E)$  be a weighted directed graph with vertices  $V$  and edges  $E = V \times V$ . Node  $v_i \in V$  is associated with a vertex cost  $d_i$ , which represents the time delay at that vertex. Edge  $(v_i, v_j) \in E$  is assigned a time cost  $d_{ij}$ , which is the time required to move from  $v_i$  to  $v_j$ . Node  $v_0 \in V$  is designated as the depot with  $d_0 = 0$ .*

*The problem seeks a tour that visits some of the vertices  $v_{i_0}, v_{i_1}, \dots, v_{i_m}$  with starting and ending at the depot,  $v_{i_0} = v_{i_m} = v_0$ , while minimizing the total time of the tour:  $\sum_{j=0}^m d_{i_j} + d_{i_j, i_{j+1}}$ . Note that we can set  $d_{i,0} = 0$  for all  $i$  if return to the depot is not required for the problem. We refer to this total time as the makespan of the tour. The tour is subject to the following additional constraints: (a) We partition the set  $V \cup \{v_0\}$  into disjoint subsets  $V_1, V_2, \dots, V_k$ . The TSP tour is required to visit exactly one node from each subset  $V_i$ . Let  $t_i$  be the time at which the node in the set  $V_i$  is visited by our tour. (b) We require that  $l_i \leq t_i \leq u_i$  for a time interval  $[l_i, u_i]$  provided as input. (c) We specify qualitative precedence constraints of the form  $V_i \prec V_j$  that specifies that the tour must visit a node in  $V_i$  before it visits some node in  $V_j$ . (d) Whenever we have  $V_i \prec V_j$ , we require  $t_i \leq t_j$ . Additionally, we may also specify quantitative relative time window  $[l_{ij}, u_{ij}]$  requiring that  $l_{ij} \leq t_j - t_i \leq u_{ij}$ .*

We show how Problem 1 is mapped to a GTSP-TWPR.

**Definition 4** (Translation of Problem 1 to GTSP-TWPC). *We abstract DTS  $\mathcal{T}$  and the TPO  $\varphi$  to a GTSP-TWPR graph, by defining the set of node  $V = \{x_0\} \cup \{x \in X \mid L(x) \neq \emptyset\}$  to be the set of states with non-empty labels as well as  $x_0$ . Then, the depot node  $v_0 = x_0$ , and subset  $V_i$  is a set of states whose label is event  $e_i$ , i.e.,  $V_i = \{x \in X \mid L(x) = e_i\}$ . A directed edge  $(x_i, x_j)$  is added whenever  $L(x_i) \preceq L(x_j)$  or the events  $L(x_i), L(x_j)$  can happen in parallel.*

The edge cost  $d_{ij}$  between two states  $x_i$  and  $x_j$  is defined by the trajectory duration  $D(s, \gamma)$  where  $s$  is the shortest path between  $x_i$  and  $x_j$  on  $\mathcal{T}$ , i.e.,  $s = s_0 s_1 \dots s_n$  that induces a trace  $L(x_i) \emptyset \dots \emptyset L(x_j)$ . Likewise, if state  $x_i$  in  $T$  has a self-transition under action  $a$  with time cost  $\Delta_T(x_i, a)$ , then we set  $d_i = \Delta_T(x_i, a)$  as the vertex cost for node  $x_i$ ; otherwise, we set  $d_i = 0$ .

Edge costs are calculated by running an all-shortest path algorithms such as Floyd-Warshall algorithm [7]. We note that our method can be extended to a continuous space kinodynamic robots by running Stable Sparse RRT (SST) [18] to obtain the shortest paths between states. Generally, this is run once for environments where locations are fixed, e.g., manufacturing factories, hospitals, etc.

##### B. MILP Formulation

1) *Single Robot:* We solve Problem 1 on graph  $G$  exactly using a Mixed Integer Linear Programming (MILP) formulation. The formulation is shown in Figure 3. Recall that  $n = |\Pi|$  is the number of events, and per our construction of  $G$ , it is also the number of the subsets  $V_1, \dots, V_n$ . We define the square bracket  $[k]$  to represent the set  $\{1, \dots, k\}$ . Let  $N$  be the number of nodes  $N = |V|$ ,  $y_{ij}$  be an integer variable indicating the active edge  $i \rightarrow j$ , and  $\tau_i$  be the continuous time variable that represents the completion of the  $i$ th event, where  $\tau_{N+1}$  represents the time coming back to the depot. We additionally introduce the set  $V_0 = \{v_0\}$  with the depot node and the indices  $I = \{0\}$  to simplify the formulation.

Note that the continuous variables include  $N + 1^{\text{th}}$  variable  $\tau_{N+1}$  that represents the time at the end of the tour. Constraint (2a) expresses that the number of incoming and outgoing edges must be equal at every node. Constraints (2b) and (2c) represent that there is only one incoming and one outgoing edge for each subset. Together with the first constraint, we ensure that the incoming/outgoing edge to a particular subset  $V_i$  must involve the same node  $v_i \in V_i$ . Constraint (2d) represents all the TPO constraints, and (2e) delays the  $j^{\text{th}}$  event by  $d_{i,j} + d_j$  from  $i^{\text{th}}$  event only if the edge is activated ( $y_{i,j} = 1$ ). Constraint (2f) delays  $N + 1^{\text{th}}$  visit (makespan) by the edge cost between  $[N]$  nodes back to the initial nodes  $I$ .  $\implies$  represents “implies” and can be expressed by using the Big-M method [31].

A tour can be obtained by following the enabled edges  $y_{i,j} = 1$  from the depot node.

2) *Multiple Robots:* For the multi-robot setting, we construct graph  $G$  in a similar manner as the single-robot case, except that for each initial state  $x_0^i \in X_0$ , we add a depot node  $v_0^i$  and its associated subset  $V_0^i$  to  $G$ . For simplicity, we redefine the indices  $I = \{0_1, \dots, 0_p\}$  to denote the depot nodes and their subsets. The MILP formulation then becomes exactly the same as that of the single-agent case in Figure 3. In practice, we avoided introducing new depots, but instead, we changed the number of incoming and outgoing edges at the initial node  $v_0$  to prevent the increase in the number of integer variables. The right-hand sides of (2b) and (2c) equate to the number of robots starting at  $V_0$ .

$$\begin{aligned}
& \min \tau_{N+1} \quad \text{s.t.} \\
& \sum_i y_{i,j} - \sum_k y_{j,k} = 0, \quad j \in I \cup [N] \quad (2a) \\
& \sum_{v_j \in V_l} \sum_i y_{i,j} = 1, \quad l \in I \cup [n] \quad (2b) \\
& \sum_{v_j \in V_l} \sum_k y_{j,k} = 1, \quad l \in I \cup [n] \quad (2c) \\
& \tau_j - \tau_i \bowtie a_{i,j}, \quad t_j, t_i, a_{i,j} \in \varphi \quad (2d) \\
& y_{i,j} = 1 \implies \tau_j - \tau_i \geq d_{i,j} + d_j, \quad i \in I \cup [N], \quad j \in [N] \quad (2e) \\
& y_{i,j} = 1 \implies \tau_{N+1} - \tau_i \geq d_{i,j}, \quad i \in [N], j \in I \quad (2f) \\
& y_{i,j} \in \{0, 1\}, \quad i, j \in I \cup [N] \quad (2g) \\
& \tau_i \geq 0, \quad i \in I \cup [N+1] \quad (2h)
\end{aligned}$$

Fig. 3: MILP formulation of the GTSP-TWPR problem. Notation  $[N] = \{1, \dots, N\}$ .

The resulting tours minimize the makespan, and their induced timed trace satisfies the TPO specification. However, when the tours are mapped to plans on  $\mathcal{T}^M$ , they are not guaranteed to produce non-colliding trajectories. Hence, they need to be checked for collisions as a post process. If a collision is detected, then we repair the plans by introducing delays to the individual plans as in [14]. Acceptable limits on these delays can be calculated using a robustness analysis. Another approach is to resolve conflicts via the Conflict-Based Search (CBS) method for multi-agent as in [26].

### C. Robustness Analysis

Once a tour is returned by our MILP formalism, it is now possible to understand how robust the tour is to variations in the edge and node costs or in other words, variations in the delays associated with a given node or an edge between two locations. Such delays are common during plan execution. Consider a single agent tour with edge costs:  $v_0 \xrightarrow{d_{01}} v_1 \xrightarrow{d_{12}} v_2 \dots v_{m-1} \xrightarrow{d_{m-1,0}} v_0$ . Let  $d_i$  be the node cost of  $v_i$  with  $d_0 = 0$ . Our goal is to characterize all possible timing variations  $\delta(d_{ij})$  to the edge costs and  $\delta(d_j)$  to the node costs such that the TPO constraints will continue to hold. To this end, note that the nominal visit time for node  $v_i$  in the tour is given by  $t_i = \sum_{j=0}^{i-1} (d_j + d_{j,j+1})$  for  $i \in [1, m-1]$  while the visit time taking into account the unknown variations in  $d_j, d_{ij}$  will be  $t_i = \sum_{j=0}^{i-1} (d_j + d_{j,j+1} + \delta(d_j) + \delta(d_{j,j+1}))$ . Robustness analysis seeks a uniform bound  $\epsilon$  such that whenever each  $|\delta(d_j)| \leq \epsilon$  and  $|\delta(d_{i,j})| \leq \epsilon$ , the TPO constraints are guaranteed to hold. To compute such a limit  $\epsilon$ , we formulate the LP:

$$\begin{aligned}
& \max \epsilon \\
& \text{s.t. } t_i = \sum_{j=0}^{i-1} (d_j + d_{j,j+1} + \delta(d_j) + \delta(d_{j,j+1})) \\
& \quad \text{for } i = 1, \dots, m-1 \\
& t_i - t_j \bowtie a_{i,j}, \quad \text{TPO constraints } t_i, d_j, a_{i,j} \\
& \epsilon \leq \delta(d_j) \\
& \epsilon \leq \delta(d_{i,j})
\end{aligned}$$

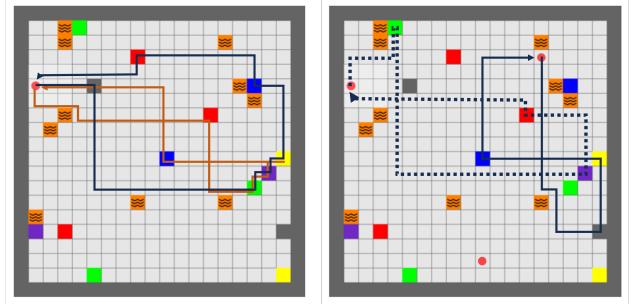
The LP above is always feasible and its optimal solution  $\epsilon$  denotes a uniform bound on the timing variations  $\delta(d_j), \delta(d_{i,j})$  such that as long as  $|\delta(d_j)| \leq \epsilon$  and  $|\delta(d_{i,j})| \leq \epsilon$  for each node and edge delay, the tour continues to be a feasible solution that satisfies the TPO constraints. The formulation for a multi-robot tour is almost identical except that the times  $t_i$  are computed differently for each robot. Unfortunately, trying to incorporate the robustness analysis as part of the solution to the TSP itself results in a robust optimization problem which often requires more expensive approaches to solve. Our future work will consider incrementally eliminating tours that fail a robustness criterion by adding a “blocking” constraint to force the MILP solver to return back with a different tour.

## V. EXPERIMENTS

In this section, we evaluate our algorithm for planning with TPO specification for single and multiple robots on various case studies. First, we demonstrate how different timing constraints of TPO cause different robot behaviors. Then, we illustrate scalability of the algorithm on a set of benchmarks. Lastly, we return to the aircraft turnaround example and show a physical experiment to demonstrate the applicability of the approach to a more realistic scenario.

### A. Illustrative Case Studies

We demonstrate our method on a gridworld environment in Figure 4 with multiple colored locations and two TPO tasks: with and without timing constraints. For the simple task, the robot must visit every color in any order. It has the option of visiting any location of the same color but must find a plan with a minimum makespan time. We ran the algorithm without any constraints and its trajectory is shown in orange in Figure 4a. The robot first visits red, green,



(a) Blue/Orange-With/out  $\varphi$       (b) Multi-robot trajectories

Fig. 4: Gridworlds with synthesized plans

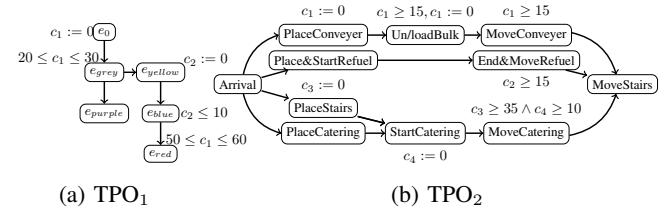


Fig. 5: TPO Specifications for the case studies.



Fig. 6: Experimental setup of the aircraft turnaround task (see attached video: <https://youtu.be/WUuWF1OoKW8>).

purple, yellow, blue, and grey in order. The second task is TPO<sub>1</sub> in Figure 5a. The robot visited grey, green, purple, yellow, blue and red in order. Observe that the robot now visits the grey first and the red last due to the constraints. In Figure 4b, we extend to two robots the same TPO<sub>1</sub> task. The result was two trajectories for the two robots that satisfy TPO<sub>1</sub>. The trajectories are long, since red has to be visited between 50 to 60 time units.

### B. Benchmarks

Here, we explore how our algorithm scales with the increasing number of states with non-empty labels, timing constraints, and number of robots. We generated a random set of events varying in number from 5 to 80 in a 30-by-30 gridworld environment. We incrementally add timing constraints of the form  $d_{i,j} + d_j - \Delta t \leq t_j - t_i \leq d_{i,j} + d_j + \Delta t$  where  $\Delta t > 0$  is a constant padding to see how the overall solution time depends on the number of constraints we add and the “tightness” of these constraints.

We ran benchmarks with varying the number of timing constraints (constraints involving  $p = 25, 50, 75, 100\%$  of all TPO edges),  $\Delta t = 10, 30, 50$ , and the number of robots of 1, 10, 20, 30, and 40. We compared our method to heuristic approaches implemented in OR-Tools Routing Library [8]. The time padding  $\Delta t$  and the percentage of the number of edges  $p$  did not have significant effects on the results. Figure 7 shows the result at when  $p = 25\%$  and  $\Delta t = 30$ .

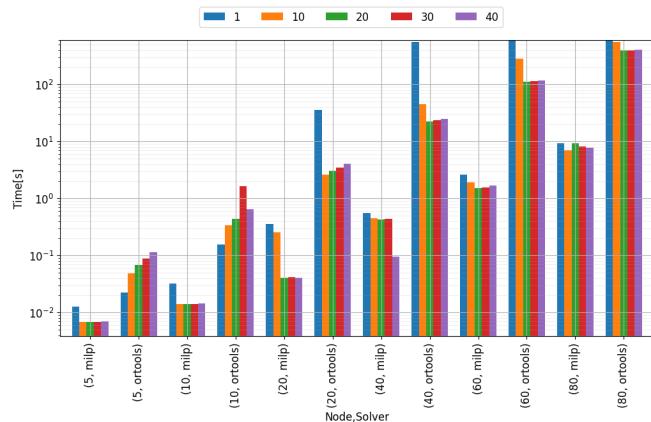


Fig. 7: Benchmark results. The x-axis is (#nodes, solver) and the y-axis is computation time in seconds in log scale. The timeout is set to 600 seconds. The colors of the bars indicate the number of robots (see legend).

As the number of nodes increases, the problem gets more difficult to solve, taking more time to find the optimal solution. Also, the computation time mostly stays the same as we increase the number of robots. Interestingly, the OR-Tools did not perform well compared to MILP. This is because the heuristics are disabled when unexpected additional constraints (e.g., local timing constraints) are added. Instead, they use constraint programming to solve the problem, which is slower than the Branch and Bound method employed in the MILP solver.

Also, the algorithm easily scaled up to 80 nonempty-label states as shown in the figure. We further ran the stretch test with a timeout of 30 minutes, and MILP was able to solve up to 160 nodes within  $182 \pm 102$  seconds excluding a case (out of 60 runs) when it hit the timeout. It took  $210 \pm 232$  seconds including the timeout.

### C. Aircraft Turnaround

We consider the aircraft turnaround example in Figure 1 with robots as ground staff. The task TPO<sub>2</sub> in Figure 5b, specifies vehicle movement, refuel, and bulk loading/unloading. The goal is to find the most efficient plans for completing the task according to TPO<sub>2</sub>. In Figure 6, we show the plans of all robots. Robot 1 places the stair truck, refueling vehicle, and moves out the stair truck. Robot 2 performs bulk unloading/loading and then moves out the refueling vehicle. Robot 3 performs the catering services. Almost all robots finish their assigned events at the same time. The overall makespan was 59 time units. We also repeated the same case study for a single robot from various initial states and obtain makespans 152, 155, 163 time units. A video of this experiment accompanies the submission<sup>1</sup>.

## VI. CONCLUSIONS

We introduce a general framework for planning under Timed Partial Order (TPO) specifications for multiple robots. Our solution maps the task allocation problem to the Generalized Traveling Salesman Problem (GTSP) with time windows and precedence constraints which we solve by a Mixed-Integer Linear Program (MILP). Our evaluations of the algorithm on various case studies demonstrate the time-effectiveness of our plans for up to 40 robots with 160 nodes.

For future work, we plan to investigate variants of the dynamic task assignment problem under TPO specifications and to consider robustness and contingencies in the MILP formulation.

<sup>1</sup>Video: <https://youtu.be/WUuWF1OoKW8>

## REFERENCES

- [1] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.
- [2] Rajeev Alur, Thomas A. Henzinger, Gerardo Lafferriere, George, and J. Pappas. Discrete abstractions of hybrid systems. In *Proceedings of the IEEE*, volume 88, pages 971–984, 2000.
- [3] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. *IFAC Proceedings Volumes*, 31(18):447–452, 1998.
- [4] Hamza Chakraa, Fran ois Gu erin, Edouard Leclercq, and Dimitri Lefebvre. Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art. *Robotics and Autonomous Systems*, page 104492, 2023.
- [5] William J. Cook. *In Pursuit of the Traveling Salesman*. Princeton University Press, Princeton, NJ, USA, September 2014.
- [6] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1-3):61–95, 1991.
- [7] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.
- [8] Vincent Furnon and Laurent Perron. Or-tools routing library.
- [9] Michael R. Garey and David S. Johnson. *Computers and Intractability: A guide to the theory of NP-Completeness*. W.H.Freeman, 1979.
- [10] Walker Gosrich, Siddharth Mayya, Saaketh Narayan, Matthew Mancina, Saurav Agarwal, and Vijay Kumar. Multi-robot coordination and cooperation with task precedence relationships. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5800–5806. IEEE, 2023.
- [11] Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. Recent trends in task and motion planning for robotics: A survey. *ACM Computing Surveys*, 2023.
- [12] Keliang He, Morteza Lahijanian, E Kavraki, Lydia, and Y Vardi, Moshe. Automated abstraction of manipulation domains for cost-based reactive synthesis. *IEEE Robotics and Automation Letters*, 4(2):285–292, Apr. 2019.
- [13] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Towards manipulation planning with temporal logic specifications. In *Int. Conf. Robotics and Automation*, pages 346–352. IEEE, May 2015.
- [14] Justin Kottinger, Shaull Almagor, Oren Salzman, and Morteza Lahijanian. Introducing delays in multi-agent path finding. *arXiv preprint arXiv:2307.11252*, 2023.
- [15] H. Kress-Gazit, G. Fainekos, and G. J. Pappas. Where's Waldo? sensor-based temporal logic motion planning. In *Int. Conf. on Robotics and Automation*, pages 3116–3121, Rome, Italy, 2007. IEEE.
- [16] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:211–236, May 2018.
- [17] Morteza Lahijanian, M. Kloetzer, S. Itani, C. Belta, and S.B. Andersson. Automatic deployment of autonomous cars in a robotic urban-like environment (RULE). In *Int. Conf. on Robotics and Automation*, pages 2055–2060, Kobe, Japan, 2009. IEEE.
- [18] Yanbo Li, Zakary Littlefield, and Kostas E Bekris. Asymptotically optimal sampling-based kinodynamic planning. *The International Journal of Robotics Research*, 35(5):528–564, 2016.
- [19] Oded Maler and Dejan Nickovic. Monitoring Temporal Properties of Continuous Signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems: Joint International Conferences on Formal Modeling and Analysis of Timed Systems*, pages 152–166. Springer, 2004.
- [20] Oded Maler, Dejan Nickovic, and Amir Pnueli. From MITL to Timed Automata. In *Proceedings of FORMATS*, volume 4202 of *LNCS*, pages 274–289. Springer, 2006.
- [21] Aristide Mingozzi, Lucio Bianco, and Salvatore Ricciardelli. Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints. *Operations research*, 45(3):365–377, 1997.
- [22] Charles Edward Noon. *The generalized traveling salesman problem*. University of Michigan, 1988.
- [23] Ernesto Nunes, Marie Manner, Hakim Mitiche, and Maria Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 90:55–70, 2017.
- [24] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems. *Part II: Transportation between pickup and delivery locations, to appear: Journal f r Betriebswirtschaft*, 2007.
- [25] Petric  C Pop, Ovidiu Cosma, Cosmin Sabo, and Corina Pop Sitar. A comprehensive survey on the generalized traveling salesman problem. *European Journal of Operational Research*, 2023.
- [26] Zhongqiang Ren, Sivakumar Rathinam, and Howie Choset. Conflict-based steiner search for multi-agent combinatorial path finding. In *Proceedings of Robotics: Science and Systems*, 2022.
- [27] Arik Senderovich, Kyle EC Booth, and J Christopher Beck. Learning scheduling models from event data. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 401–409, 2019.
- [28] Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuju Fan. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(2):3451–3458, 2022.
- [29] Elina Suslova and Pooyan Fazli. Multi-robot task allocation with time window and ordering constraints. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6909–6916. IEEE, 2020.
- [30] Kandai Watanabe, Georgios Fainekos, Bardh Hoxha, Morteza Lahijanian, Danil Prokhorov, Sriram Sankaranarayanan, and Tomoya Yamaguchi. Timed partial order inference algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pages 639–647, 2023.
- [31] H Paul Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.