

Multi-Agent Path Finding with Tasks Under Time Uncertainty

Alexandre Albore ✉ 

ONERA, University of Toulouse, France

Christophe Grand ✉ 

ONERA, University of Toulouse, France

Noëlie Ramuzat ✉ 

ONERA, University of Toulouse, France

Ajdin Sumic ✉ 

ONERA, University of Toulouse, France

Abstract

Multi-Agent Path Finding under Time Uncertainty (MAPF-TU) extends classical MAPF by considering uncertain traversal duration. However, existing approaches, such as the Conflict Based Search with Time Uncertainty (CBS-TU) method, assume that uncertainty only arises while agents move between locations, overlooking uncertainty induced by executing tasks. In many real-world applications, agents must not only navigate but also perform tasks of uncertain duration at specific locations, leading to additional temporal dependencies.

In this paper, we introduce Multi-Agent Path Finding with Tasks under Time Uncertainty (MAPF-TTU), a new extension of MAPF-TU in which agents must execute tasks of uncertain duration at intermediate goals. We first analyze the computational complexity of MAPF-TU and MAPF-TTU. We then propose two complementary solution approaches. The first extends CBS-TU to explicitly reason about uncertain task execution during planning. The second transforms an MAPF-TU solution into a Simple Temporal Network with Uncertainty (STNU) and computes a robust execution schedule through Strong Controllability.

Finally, we experimentally evaluate both approaches under varying levels of traversal and task time uncertainty, highlighting the tradeoff between solution quality and scalability.

2012 ACM Subject Classification Computing methodologies → Multi-agent planning; Computing methodologies → Planning under uncertainty; Computing methodologies → Temporal reasoning

Keywords and phrases Multi-Agent Path Finding, Temporal Uncertainty, Conflict Based Search, Simple Temporal Network

Digital Object Identifier 10.4230/OASICS.TIME.2026.13

Supplementary Material *Software*: https://github.com/ajdin14200/MAPF_TTU_TIME [31]
archived at `swb:1:dir:b90831099c338304e218af5159b6add58ed219b1`

Funding This work was supported by government funding administered by the French National Research Agency (ANR) under the France 2030 programme, reference number ANR-23-DMRO-0006.

1 Introduction

Multi-Agent Pathfinding (MAPF) is an NP-hard problem that consists of computing collision-free paths for a set of agents that must reach predefined goal locations from their initial positions [32]. MAPF has been extensively studied due to its relevance in many real-world applications, including transportation and logistics for warehouse automation [37, 18], robotics [29, 14], and video games [28]. Classical formulations of MAPF typically assume either discrete timesteps or deterministic continuous-time actions [36, 2], allowing the motion of the agents to be synchronized precisely and straightforwardly checked for conflicts.



© Alexandre Albore, Christophe Grand, Noëlie Ramuzat, and Ajdin Sumic;
licensed under Creative Commons License CC-BY 4.0

33rd International Symposium on Temporal Representation and Reasoning (TIME 2026).

Editors: AndreA Orlandini and Sophie Pinchinat; Article No. 13; pp. 13:1–13:21

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Over the last decade, a large body of work has focused on developing increasingly efficient MAPF solvers, including search-based methods, compilation approaches, prioritized planning, and hybrid techniques. Among these approaches, Conflict-Based Search (CBS) and its extensions have become particularly influential due to their balance between optimality and scalability [27, 7, 19]. CBS decomposes the problem into a high-level search that resolves conflicts between agents and a low-level search that computes individual paths under constraints. Numerous variants have subsequently extended CBS to continuous time, kinematic constraints, and execution robustness [1, 2, 8, 4, 11].

In contrast, the Prioritized Inheritance Backtracking (PIBT) algorithm is a well-known sub-optimal approach designed for large-scale MAPF instances involving thousands of agents [24]. Its main strength lies in its high computational efficiency and scalability, although this comes at the cost of lower solution quality compared to optimal approaches such as CBS.

However, the deterministic assumptions underlying classical MAPF significantly limit its applicability in realistic environments, where traversal durations may vary due to exogenous factors. Several works therefore focused on generating more robust execution strategies, either by guaranteeing feasibility under bounded delays [3, 15, 22], dynamically adapting agent priorities [30, 5], or repairing plans online during execution [16, 9, 13, 6, 20, 21].

In such settings, action durations are more naturally modeled as uncertain but bounded values rather than fixed ones. Multi-Agent Pathfinding with Time Uncertainty (MAPF-TU) extends MAPF by associating temporal intervals with edge traversals in the navigation graph [26]. Reasoning under temporal uncertainty fundamentally changes the nature of the problem. In deterministic MAPF, conflicts can be checked at precise timesteps, whereas MAPF-TU requires considering all possible realizations of uncertain durations. Two agents may conflict whenever their occupancy intervals overlap under some admissible execution scenario. Consequently, MAPF-TU requires safe plans that remain collision-free despite uncertain execution outcomes. Existing approaches typically rely on interval-based reasoning and robust conflict detection to guarantee safe execution [26, 33]. In particular, CBS_TU [26] extends CBS to reason directly over interval-valued plans and conflicts.

Although MAPF-TU considerably improves the realism of MAPF, existing formulations mainly focus on uncertainty induced by navigation actions, namely uncertain edge traversal durations. This assumption is sufficient when agents only move between locations, but it becomes inadequate in many practical applications where agents must also perform tasks at specific locations. In numerous real-world domains, these tasks may themselves exhibit uncertain durations.

Consider, for example, an automated warehouse where robots retrieve items from shelves. Uncertainty may arise not only from navigation times due to congestion, but also from picking operations because of object accessibility or grasping conditions.

Despite its practical importance, this second source of uncertainty has received little attention in the MAPF literature. Existing approaches generally assume that agents instantly complete tasks upon reaching a goal location, or that task durations are deterministic [38]. However, these assumptions may therefore lead to unrealistic schedules at execution, leading to online adaptation to avoid collisions.

In this work, we introduce a new paradigm called Multi-Agent Path Finding with Time and Task Duration Uncertainty (MAPF-TTU). In MAPF-TTU, each agent is assigned an ordered sequence of goals. Every goal except the final one requires the execution of a task of uncertain duration before the agent can proceed. Task durations are represented as temporal intervals similarly to uncertain traversal times. Consequently, uncertainty may arise both while moving between locations and while executing tasks at a location.

Introducing uncertain task durations substantially increases the complexity of temporal reasoning in MAPF. Unlike traversal uncertainty, task uncertainty occurs while an agent remains at a fixed location, as it can wait indefinitely before starting a task of uncertain duration or leaving. This introduces new temporal dependencies and conflicts, since an agent may occupy a shared resource for an uncertain duration while executing a task.

To address these challenges, we propose two complementary approaches. The first is an optimal extension of CBS_TU, called CBS_TTU, that explicitly reasons about uncertain task durations during planning. This extension augments the low-level search with explicit task execution states and interval-valued task actions, allowing the planner to compute robust solutions that account simultaneously for uncertain navigation and uncertain task execution.

Our second, suboptimal approach adopts a decoupled temporal reasoning perspective. Instead of integrating all temporal uncertainty directly into the MAPF search, we first compute a classical MAPF-TU solution and subsequently transform the resulting execution plan into a Simple Temporal Network with Uncertainty (STNU) [34]. In this transformation, traversal uncertainties and task execution uncertainties are modeled as *contingent* temporal constraints, while precedence relations induced by collisions are represented as *requirement* constraints. This enables the computation of a feasible execution policy that remains valid under all realizations of uncertainty.

The remainder of the paper is organized as follows. We first review the necessary background on MAPF, MAPF-TU, Temporal Plan Graphs, and Simple Temporal Networks with Uncertainty. We then formally define MAPF-TTU and present our two approaches. Subsequently, we provide an experimental evaluation comparing the approaches under varying levels of traversal and task uncertainty. Finally, we discuss the implications of our results and outline directions for future research on temporally robust multi-agent coordination.

2 Background

2.1 Multi-Agent Path Finding

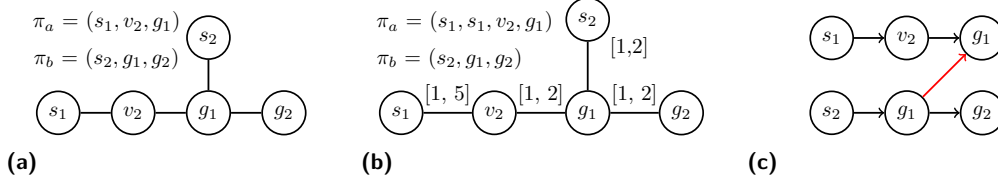
Multi-Agent Path Finding is the problem of computing collision-free paths for a set of agents moving in a shared environment from predefined start locations to goal locations [32, 29]. MAPF is generally defined as a tuple $\langle G, A, s, g \rangle$, where $G = (V, E)$ is a graph representing the environment, where vertices correspond to locations and edges represent feasible movements between adjacent locations. $A = \{a_i\}_{i \in \mathbb{N}}$ is the set of agents, while $s : A \rightarrow V$ and $g : A \rightarrow V$, respectively, maps each agent to its start and goal location.

At each timestep, an agent executes an action: it either moves to an adjacent vertex, or waits at its current location. A plan for an agent $a \in A$ is a sequence of vertices $\pi_a = (v_1, \dots, v_k)$, such that $v_1 = s(a)$ and $v_k = g(a)$. Consecutive vertices are either connected by an edge in E (move action) or identical (wait action).

The objective of MAPF is to compute a set of plans $\Pi = \{\pi_1, \dots, \pi_n\}$ such that no collisions occur between agents. Two main types of conflicts are generally considered: a *vertex conflict* occurs when two agents a and b occupy the same vertex $v \in V$ at the same timestep t and is defined as a tuple $\langle a, b, v, t \rangle$, an *edge conflict* occurs when two agents traverse the same edge $(u, v) \in E$ in opposite direction at the same timestep t and is defined as a tuple $\langle a, b, (u, v), t \rangle$. A solution to MAPF is therefore a set of collision-free plans satisfying all agents' goals.

From an MAPF solution, a directed acyclic graph can be constructed, called Temporal Plan Graph (TPG) [16], which captures the precedence relations required to preserve collision-free execution. In a TPG, vertices represent events in the agents' plans, while edges

encode temporal dependencies between these events, including both intra-agent sequencing constraints and inter-agent precedence constraints induced by conflict resolution. TPGs support online adaptation, rescheduling, and temporal reasoning during execution while preserving the safety guarantees of the original MAPF solution [30, 5, 16, 9]. Figure 1 illustrates an MAPF example with two agents and the corresponding optimal solution minimizing the makespan, transformed as a TPG.



■ **Figure 1** Figure a illustrates an MAPF problem with two agents a and b , starting respectively at nodes s_1 and s_2 and finishing at nodes g_1 and g_2 , respectively, with π_a and π_b the optimal collision-free solutions. Figure b illustrates the MAPF-TU version of the example in Figure a, where the optimal collision-free solution now requires agent a to wait one timestep on node s_1 to avoid potentially colliding with agent b in g_1 at timestep 2. Figure c illustrates the corresponding TPG of the optimal solution of Figure a with the inter-agent precedence highlighted in red, i.e., agent b (below) must pass by node g_1 before agent a (top). This TPG is also valid for Figure b.

2.2 Multi-Agent Path Finding with Time Uncertainty

The MAPF with Time Uncertainty (MAPF-TU) problem is an MAPF problem where travel duration between two positions is uncertain and can assume values within a fixed interval [26].

► **Definition 1** (MAPF-TU). *We define the Multi-Agent Path Finding problem with Time Uncertainty as a tuple $\mathcal{M} = \langle G, A, s, g, w^-, w^+ \rangle$ where:*

- $\langle G, A, s, g \rangle$ is an MAPF instance;
- $w^- : E \rightarrow \mathbb{R}^+$ is the function assigning the minimal possible duration to traverse an edge.
- $w^+ : E \rightarrow \mathbb{R}^+$ is the function returning the maximal possible duration to traverse an edge.

In addition, we refer by $[w^-(e), w^+(e)]$ the non-deterministic time interval that bounds the duration of traversing an edge $e \in E$: traversing e multiple times may lead to different travel duration each time. Hence, $w^-(e)$ and $w^+(e)$ are the lower and upper bounds of the durations associated to e .

Unlike deterministic MAPF, the exact arrival time of an agent at a vertex is no longer known precisely. Instead, each visit is associated with a temporal interval representing all possible arrival timesteps induced by uncertainty propagation, as shown in Figure 1b.

► **Definition 2** (Potential Presence). *Given an agent a and a plan $\pi_a = (v_1, \dots, v_n)$, the potential presence of a at vertex $v_k \in \pi_a$, denoted $\tau(\pi_a, v_k)$, is the interval of possible timesteps during which agent a may occupy v_k when following π_a . The potential presence is defined similarly for an edge.*

Reasoning under temporal uncertainty fundamentally changes conflict detection. In deterministic MAPF, conflicts are checked at exact timesteps, whereas MAPF-TU requires considering all possible realizations of uncertain durations.

► **Definition 3** (Potential Conflict). *Given two agents a and b , their plans π_a and π_b are in potential conflict on a vertex or edge x if $\tau(\pi_a, x) \cap \tau(\pi_b, x) \neq \emptyset$.*

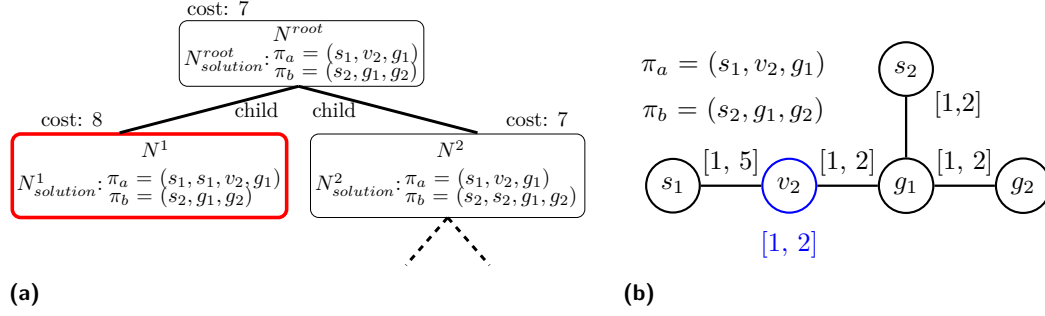


Figure 2 Figure a illustrates the execution of CBS on the provided MAPF-TU example in Figure 1b. Figure b illustrates the MAPF-TTU version of the MAPF-TU example in Figure 1b, where now agent a must perform a task at the intermediate goal node v_2 of uncertain duration $[1, 2]$. The optimal collision-free solution is given with π_a and π_b . The corresponding TPG of this solution is given in Figure 1c.

A potential conflict therefore exists whenever two potential presences overlap on the same vertex/edge. A valid MAPF-TU solution is a set of collision-free plans Π for all admissible realizations of traversal durations.

Several approaches have been proposed to solve MAPF-TU. Shahar et al. [26] extended CBS to uncertain traversal times through CBS_TU, which reasons directly over interval-valued conflicts while preserving completeness and optimality guarantees. More recently, Švancara et al. [33] proposed policy-based approaches capable of adapting agent actions dynamically according to observed traversal durations during execution.

2.3 Conflict-Based Search algorithm

CBS maintains a search tree, called *Constraint Tree* (CT), where each node represents a set of constraints and the associated solution. At each iteration, it selects the most promising node in the tree. If the node's solution has no conflicts, it is returned as the final plan. Otherwise, a conflict is resolved by branching the search tree on new nodes, each adding a constraint that forbids one side of the conflict. For each side of the conflict, a new constraint is added, and a constrained path is replanned for the involved agent. By systematically exploring the space of constraints in this way, CBS guarantees completeness and optimality. Nodes are selected according to their cost, usually defined by the makespan of the agents' plan. The first node of the tree is called *root node* and has no constraints, thus its solution contains the optimal individual path for each agent. In CBS, a low-level process computes new paths by finding the shortest path to the goal (e.g., using A*), whilst taking into consideration the set of constraints and a heuristic to guide the search. In MAPF-TU, the heuristic considers either the lower or the upper bound duration for each edge. In the present implementation, the latter option was used exclusively in order not to overestimate the delay too much.

Figure 2a shows the execution of CBS on the example in Figure 1b. The initial path computed for the agents is shown in the root CT node (N^{root}). This solution contains a vertex conflict on g_1 at $t = 2$. CBS_TU will resolve the conflict and creates two CT child nodes N^1 and N^2 , where N^1 is the collision-free solution. Please note that the search process will select N^2 first due to a lower cost and iterate until it select N^1 as the most promising node in the CT.

2.4 Simple Temporal Network with Uncertainty

A *Simple Temporal Network* (STN) is a tuple $\langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is a set of real-value variables called timepoints $\{\nu_1, \dots, \nu_n\}$, and $\mathcal{E}$ is a set of temporal constraints between timepoints called *requirements* [12]. Specifically, each requirement $r \in \mathcal{E}$ is of the form $\nu_y - \nu_x \in [l, u]$, with $\nu_x, \nu_y \in \mathcal{V}$, $l \in \mathbb{R} \cup \{-\infty\}$, and $u \in \mathbb{R} \cup \{+\infty\}$; l and u represent the minimal and maximal possible temporal distance between ν_x and ν_y . A requirement is represented as an edge $\nu_x \xrightarrow{[l, u]} \nu_y$. A reference timepoint ν_z is generally included in $\mathcal{V}$ as a fixed temporal anchor assumed to be the first executed timepoint of the network, i.e., ν_z will take a value such that each implicit constraint $\nu_z \leq \nu_x$ for each $\nu_x \in \mathcal{V}$ is satisfied. An STN is satisfiable if a timepoint assignment (schedule) exists that satisfies all the constraints. We say that a controller executes an STN when it schedules its timepoints. STNs have proven to be useful for MAPF by transforming TPGs into STNs to support execution flexibility and online adaptation in dynamic environment [16].

A *Simple Temporal Network with Uncertainty* (STNU) is an extension of the STN in which a subset of constraints is distinguished, with the values (duration) of these constraints being decided by external entities (e.g., *Nature*) and the controller can only observe them [34]. A STNU $\mathcal{X} = \langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$ is a tuple where $\mathcal{V}$ is now divided into controllable ($\mathcal{V}_c$) and uncontrollable ($\mathcal{V}_u$) timepoints. $\mathcal{C}$ is a set of (uncontrollable) contingent links each of the form $\nu_y - \nu_x \in [l, u]$, where $\nu_x \in \mathcal{V}_c, \nu_y \in \mathcal{V}_u$, and $0 < l \leq u$. The value (duration) $\omega = \nu_y - \nu_x$ is set by an external entity before or during the execution. Once ν_x is executed, the value of ν_y is $\nu_x + \omega$. A contingent link is depicted as $\nu_x \xrightarrow{[l, u]} \nu_y$. A *schedule* for a STNU is a mapping $\delta: \mathcal{V} \rightarrow \mathbb{R}$ from timepoints to real values.

► **Definition 4** (Situation, Projection and Solution). *Given a STNU $\mathcal{X} = \langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$, the situations of $\mathcal{X}$ are the set of tuples Ω defined as the Cartesian product of contingent domains: $\Omega = \times_{c_k \in \mathcal{C}} [l_k, u_k]$. Each situation $\omega = (\omega_1, \dots, \omega_n) \in \Omega$ represents one possible complete set of values for the durations of the contingent links of $\mathcal{X}$. A projection $\mathcal{X}^\omega = (\mathcal{V}, \mathcal{E} \cup \mathcal{C}^\omega)$ of $\mathcal{X}$ is an STN, where $\mathcal{C}^\omega$ replaces each contingent link c_k with the rigid requirement induced by ω_k . A solution of $\mathcal{X}^\omega$ is a schedule δ_ω satisfying all the constraints.*

In STNUs, consistency needs to be redefined: a network is now said to be *controllable* if there exists a schedule that satisfies all requirement constraints in any possible projection.

Three levels of controllability exist [34], depending on when the contingent durations are known: *Strong Controllability* (SC) assumes that observation at execution is not possible (as in conformant planning), hence, a single common schedule δ satisfying all constraints in all situations is required. *Weak Controllability* (WC) assumes all contingent durations will be known just before execution (oracle), hence, one need to guarantee a consistent schedule for each situation. *Dynamic Controllability* (DC) assumes that contingent durations are observable at execution, hence, the schedule values assignment depends on past observations only, regardless of the contingent durations still to be observed. In this paper, we will focus on the case of Strong Controllability, of which we recall the formal definition below.

► **Definition 5** (Strong Controllability). *A STNU $\mathcal{X}$ is said to be strongly controllable (SC) if there exists a schedule δ , called a strong schedule, such that for all situations ω , δ is a solution of $\mathcal{X}^\omega$.*

STNUs provide a natural formalism for reasoning about controllable and uncontrollable temporal events, making them particularly suitable for modeling uncertain traversal and task durations.

3 Multi-Agent Path Finding with Tasks under Time Uncertainty

We consider now an extension of MAPF-TU where agents must perform some task of uncertain but bounded duration in intermediate locations. To do so, we will identify those vertices where a durative task takes place as *intermediate goal locations*: they are specific to the goal assigned to each agent, and the solution plan must take them into account. We call such extension Multi-agent Path Finding with Tasks under Time Uncertainty (MAPF-TTU) and is defined as follows:

► **Definition 6** (MAPF-TTU). A MAPF-TTU is a tuple $\mathcal{P} = \langle G, A, s, g, w^-, w^+, M, t, \psi^-, \psi^+ \rangle$ where:

- $\langle G, A, s, g, w^-, w^+ \rangle$ is an MAPF-TU instance as in Definition 1.
- M is a set of intermediate goal locations $M \subseteq V$.
- $t : A \rightarrow M_a$ a function mapping agents to sets of intermediate locations, s.t. $t(a) = M_a$ is a set of intermediate goal locations assigned to agent a , where $M_a \subseteq M$
- $\psi^- : V \rightarrow \mathbb{R}^+$ is the function assigning the minimal possible duration on staying on an intermediate goal location $v \in M$;
- $\psi^+ : V \rightarrow \mathbb{R}^+$ is the function returning the maximal possible duration on staying on an intermediate goal location $v \in M$.

In addition, we refer by $[\psi^-(v), \psi^+(v)]$ the non-deterministic time interval that bounds the duration of staying in a vertex $v \in M$. Hence, $\psi^-(v)$ and $\psi^+(v)$ are the lower and upper bounds of the durations associated to v .

When an agent reaches an intermediate goal location, it must remain at the corresponding vertex during the entire task execution before continuing its plan. Consequently, uncertainty may arise both while traversing edges and while executing tasks at specific locations. Traversal uncertainty and task uncertainty are modeled independently through the functions w and ψ .

► **Definition 7** (MAPF-TTU single-agent plan). A single-agent plan for an agent a is a sequence of nodes $\pi_a = (v_1, \dots, v_n)$ that moves a from its start vertex $v_1 = s(a)$ to its goal vertex $v_n = g(a)$ such that: $\forall v \in M_a, v \in \pi_a$.

We assume that the task associated with an intermediate goal is executed only once upon the first visit of the corresponding agent.

The notion of potential presence is extended to account for both uncertain traversal durations and uncertain task durations. Intuitively, the uncertainty accumulated at a vertex corresponds to the sum of all previous traversal and task uncertainties along the path.

► **Definition 8** (MAPF-TTU Potential Presence). Given an agent a and its plan $\pi_a = (v_1, \dots, v_n)$. The potential presence of a at vertex $v_k \in \pi_a$, denoted $\tau(\pi_a, v_k)$, is defined as an interval in which a might be in vertex v_k , after traversing the edge (v_{k-1}, v_k) , with $(v_{k-1}) \in \pi_a$ s.t:

$$\begin{cases} [0, 0] & \text{if } k = 1 \\ [\tau^- + \psi^-(v_k) + w^-(v_{k-1}, v_k), \tau^+ + \psi^+(v_k) + w^+(v_{k-1}, v_k)] & \text{if } v_k \in M_a \\ [\tau^- + w^-(v_{k-1}, v_k), \tau^+ + w^+(v_{k-1}, v_k)] & \text{Otherwise} \end{cases}$$

where $\tau(\pi_a, v_{k-1}) = [\tau^-, \tau^+]$.

The potential presence is similarly defined for an edge.

We consider a set of plans to be solution for a MAPF-TTU if each agent visits its intermediate goal locations (see Definition 7) for a duration that is compatible with the potential presence (see Definition 8), in which no potential conflicts occur.

3.1 Complexity of MAPF-TTU

In this section, we analyze the computational complexity of MAPF-TTU. We first study the complexity of MAPF-TU, which, to the best of our knowledge, has not been explicitly characterized in the literature, although it can naturally be conjectured to be at least as hard as classical MAPF. We then show that introducing uncertain task durations preserves this computational hardness.

► **Definition 9.** *The MAPF-TU decision problem is defined by:*

Input: *An MAPF-TU instance $\mathcal{M} = \langle G, A, s, g, w^-, w^+ \rangle$ and a rational number κ .*

Question: *Is there a safe solution for $\mathcal{M}$ with cost at most κ ?*

► **Proposition 10.** *MAPF-TU is NP-hard.*

Proof. The NP-hardness of MAPF-TU follows directly from classical MAPF, which is NP-hard [32]. Given an MAPF instance, we can construct an equivalent MAPF-TU instance by setting: $w^-(e) = w^+(e) = 1$, for every $e \in E$. ◀

► **Definition 11.** *The MAPF-TTU decision problem is defined by:*

Input: *An MAPF-TTU instance $\mathcal{P} = \langle G, A, s, g, w^-, w^+, M, t, \psi^-, \psi^+ \rangle$ and a rational number κ .*

Question: *Is there a safe solution for $\mathcal{P}$ with cost at most κ ?*

► **Proposition 12.** *MAPF-TTU is NP-hard.*

Proof. The NP-hardness of MAPF-TTU follows immediately from the one of MAPF-TU given in Proposition 10. Any MAPF-TU instance can be transformed into an equivalent MAPF-TTU instance by setting: $M = \emptyset$, i.e., by not having any intermediate goals. The resulting MAPF-TTU instance is therefore equivalent to the original MAPF-TU instance. Since MAPF-TU is NP-hard, it follows that MAPF-TTU is also NP-hard. ◀

4 The CBS_TTU Algorithm

To solve MAPF-TTU optimally, we extend the CBS_TU algorithm proposed in [26]. The resulting algorithm, called CBS_TTU, preserves the overall structure of CBS_TU while extending the low-level search to explicitly reason about uncertain task execution.

Similarly to CBS_TU, the algorithm maintains a Constraint Tree. The high-level search remains unchanged: the algorithm iteratively selects the lowest-cost CT node, detects potential conflicts through interval overlaps, and resolves them by generating child nodes with additional constraints.

The main modification made here concerns the low-level planner. In addition to movement and wait actions, agents may now execute tasks at intermediate goal locations of uncertain duration: $[\psi^-(v), \psi^+(v)]$. During this action, the agent remains at the same location until the task is completed before continuing toward subsequent goals.

The low-level state representation is therefore augmented with a task execution status indicating whether the current sub-goal action has already been performed. Potential conflicts are detected exactly as in CBS_TU using interval overlap tests on potential presence intervals. However, these intervals may now originate both from uncertain edge traversals and uncertain task executions.

Since CBS_TTU preserves the same high-level conflict resolution mechanism as CBS_TU while extending the low-level search with additional uncertain actions, the completeness and optimality guarantees of CBS_TU are maintained.

5 The CBS_TU_STNU Algorithm

In this section, we present our STNU-based approach for MAPF-TTU called CBS_TU_STNU. We first solve the MAPF-TTU using CBS_TU to find collision free paths under uncertain edge traversal durations, then we use STNUs in order to compute a schedule by incorporating uncertain task durations to the given MAPF-TU solution.

Our method builds upon the approach proposed in [16], which extends Temporal Plan Graphs to Simple Temporal Networks in order to accommodate kinematic constraints. We further extend this approach to MAPF-TTU, with the aim of reasoning explicitly about temporal uncertainty, which arises in our benchmarks both from edge traversal and task execution durations.

5.1 Extending Temporal Plan Graphs to MAPF-TTU

Extending TPGs to MAPF-TTU is straightforward. Similarly to the classical MAPF setting, the TPG captures the precedence relations that must be preserved during execution in order to guarantee the absence of collisions. The main difference resides in the fact that events are now associated with potential presence intervals induced by uncertain traversal and task durations.

Building the TPG of a MAPF-TTU solution is shown in Algorithm 1. More precisely, each visit of an agent to a vertex in its plan is represented by a TPG node. Type-1 edges encode the sequential order of actions for each agent, while type-2 edges represent inter-agent precedence relations induced by conflict avoidance [16]. Since MAPF-TU solutions are collision-free, the potential presence intervals of two conflicting visits cannot overlap. Therefore, one interval precedes the other, which directly determines the corresponding precedence relation in the TPG.

The resulting graph compactly represents both the ordering of individual plans and the temporal dependencies required for safe execution under uncertainty. The TPG of the MAPF-TTU solution in Figure 2b is given in Figure 1c.

5.2 Extending Temporal Plan Graphs to STNUs

Given a TPG, we construct a STNU $\mathcal{X} = \langle \mathcal{V}, \mathcal{E}, \mathcal{C} \rangle$ capturing both controllable precedence relations and uncontrollable durations induced by traversal and task execution uncertainty.

Let η_i^k denote the k -th node of agent a_i in the TPG. For each node η_i^k , we decompose it into arrival, task-execution, and leaving events. We introduce two STNU timepoints: an uncontrollable timepoint $\nu_A(\eta_i^k) \in \mathcal{V}_u$ and a controllable timepoint $\nu_L(\eta_i^k) \in \mathcal{V}_c$, representing respectively the arrival and the leaving time of agent a_i at the corresponding location. However, if η_i^k corresponds to intermediate goal location $v \in M_{a_i}$, we additionally introduce two timepoints: a controllable timepoint $\nu_{Ts}(\eta_i^k) \in \mathcal{V}_c$ and an uncontrollable timepoint $\nu_{Te}(\eta_i^k) \in \mathcal{V}_u$, to represent the start and end time of the task, respectively.

An agent a_i can wait indefinitely before leaving the corresponding location or before starting a task of uncertain duration at the same location; we model this behavior by the following constraints:

$$\begin{cases} \nu_A(\eta_i^k) \xrightarrow{[0, +\infty]} \nu_{Ts}(\eta_i^k) \xrightarrow{[\psi^-(v), \psi^+(v)]} \nu_{Te}(\eta_i^k) \xrightarrow{[0, +\infty]} \nu_L(\eta_i^k) & \text{if } \eta_i^k \text{ corresponds to a } v \in M_{a_i} \\ \nu_A(\eta_i^k) \xrightarrow{[0, +\infty]} \nu_L(\eta_i^k) & \text{Otherwise} \end{cases}$$

We also introduce two reference timepoints ν_z and ν_F , respectively representing the beginning and the end of the global execution. For each agent a_i , let η_i^1 and η_i^n denote respectively the first and last node in its TPG path. We add the requirement constraints:

Algorithm 1 BuildTPGFromMAPFTU.

Input: An MAPF-TU solution $\Pi = \{\pi_1, \dots, \pi_n\}$
Output: A Temporal Plan Graph $G_T = (V_T, E_T)$

```

1  $V_T \leftarrow \emptyset, E_T \leftarrow \emptyset;$ 
2 foreach agent  $a_i$  do
3   foreach visit  $k$  in  $\pi_i$  do
4     create a TPG node  $\eta_i^k$ ;
5     annotate  $\eta_i^k$  with its vertex and potential presence interval;
6      $V_T \leftarrow V_T \cup \{\eta_i^k\};$ 
7     if  $k > 1$  then
8        $E_T \leftarrow E_T \cup \{(\eta_i^{k-1}, \eta_i^k)\};$  // intra-agent edge
9 foreach pair of agents  $a_i, a_j$  with  $i \neq j$  do
10  foreach nodes  $\eta_i^k, \eta_j^m$  visiting the same vertex do
11    if the interval of  $\eta_i^k$  precedes the interval of  $\eta_j^m$  then
12       $E_T \leftarrow E_T \cup \{(\eta_i^k, \eta_j^m)\};$  // inter-agent edge
13    else if the interval of  $\eta_j^m$  precedes the interval of  $\eta_i^k$  then
14       $E_T \leftarrow E_T \cup \{(\eta_j^m, \eta_i^k)\};$  // inter-agent edge
15 return  $G_T = (V_T, E_T);$ 

```

$\nu_z \xrightarrow{[0,0]} \nu_A(\eta_i^1)$ and $\nu_L(\eta_i^n) \xrightarrow{[0,+\infty]} \nu_F$. The first constraint synchronizes the start of every agent with the global reference timepoint, while the second ensures that ν_F occurs after all agents have completed their plans.

For every type-1 edge (η_i^k, η_i^{k+1}) in the TPG, let $e \in E$ be the corresponding traversal edge. We connect the leaving timepoint of η_i^k to the arrival timepoint of η_i^{k+1} by a dotted arrow as follows: $\nu_L(\eta_i^k) \xrightarrow{[w_-(e), w_+(e)]} \nu_A(\eta_i^{k+1})$

Finally, each type-2 edge (η_i^k, η_j^m) , representing a precedence relation induced by conflict avoidance, is translated into the requirement constraint: $\nu_L(\eta_i^k) \xrightarrow{[1,+\infty]} \nu_A(\eta_j^m)$. Please note that the lower bound defines the temporal duration safety between the moment agent a_i leaves the corresponding location and the moment agent a_j arrives at the same location.

The resulting STNU therefore captures both the temporal flexibility of the execution and the uncontrollable durations arising from navigation and task execution uncertainty. Figure 3 illustrates an example of transforming a TPG into a STNU.

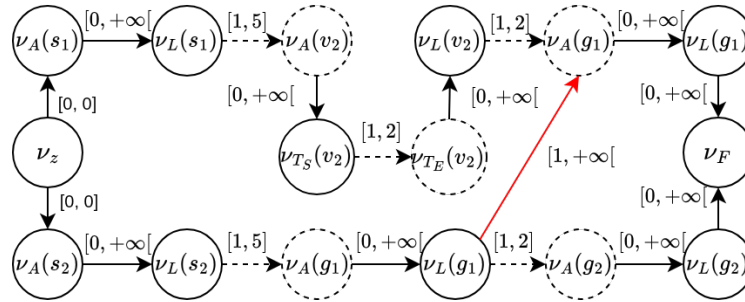


Figure 3 The transformation of the TPG from Figure 1c in a STNU. Dotted nodes depict uncontrollable timepoints of contingent links (dotted arrows), while the highlighted constraint shows the precedence relation on node g_1 .

5.3 Solving Strong Controllability

To find a solution for MAPF-TTU, we use the CBS_TU algorithm to compute a collision-free path between intermediate goals for each agent, and transform this solution into a TPG and a STNU. Then, we find a solution by looking for an assignment of the timepoints under the semantic of Strong Controllability. To do so, we use the polynomial size linear encoding of [23]. In the following, we describe the linear encoding for checking Strong Controllability.

For a requirement $\nu_x \xrightarrow{[l, u]} \nu_y$, there are three cases in which an uncontrollable timepoint is involved: (1) $\nu_y \in \mathcal{V}_u$, (2) $\nu_x \in \mathcal{V}_u$, and (3) $\nu_x, \nu_y \in \mathcal{V}_u$. A requirement in case (1) can be encoded by the linear inequality $l \leq (\nu_k + \omega_y) - \nu_x \leq u$, where ω_y represents the uncontrollable duration of the contingent link $\nu_k \xrightarrow{[l', u']} \nu_y$ with ν_y as endpoint. Other cases can be encoded similarly.

By Definition 5, a strong schedule must satisfy this system regardless of the durations $\omega_1, \dots, \omega_n$. It is shown in [34] that for a schedule to be a strong schedule, it suffices for it to satisfy the system when each duration takes one of its worst possible values, which are the lower and upper bounds of the corresponding contingent link. For example, to satisfy $l \leq (\nu_k + \omega_y) - \nu_x \leq u$ for every $\omega_y \in [l', u']$, it suffices to satisfy $l \leq (\nu_k + l') - \nu_x$ and $(\nu_k + u') - \nu_x \leq u$. This is the reason why it results in a system of linear inequalities of polynomial size.

To find the best solution, i.e., the one with the lowest makespan, we slightly modify the encoding by adding an optimization criteria on ν_F : $\text{Min}(\nu_F)$.

► **Proposition 13** (CBS_TU_STNU Completeness). *CBS_TU_STNU is complete but not optimal. Completeness follows from the completeness of CBS_TU and from the structure of the constructed STNU. Indeed, all precedence constraints introduced in the STNU are lower-bounded constraints of the form $[l, +\infty]$. Consequently, contingent delays induced by uncertain traversal and task durations can only postpone events without ever violating temporal upper bounds. Therefore, the resulting STNU is always strongly controllable.*

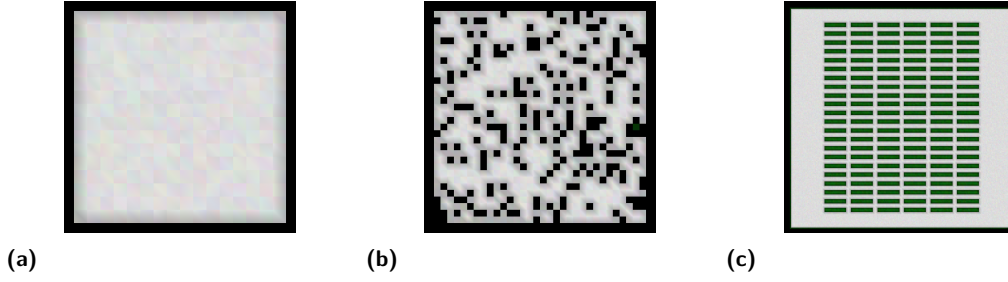
However, the approach is not optimal since temporal optimization is performed only after fixing the MAPF-TU plan and its associated precedence relations. As a result, the STNU can optimize execution times only within the flexibility of this fixed ordering, potentially excluding globally optimal MAPF-TTU solutions.

Although the resulting STNU is always strongly controllable due to the absence of restrictive upper-bound precedence constraints, the optimization process remains useful to minimize execution makespan while preserving robustness under uncertainty.

5.4 Perspective on Dynamic Controllability

Švancara et al. [33] assume that observation during execution is possible and propose a policy-based execution framework yielding solutions of improved quality for MAPF-TU. Similarly, Dynamic Controllability (DC) in STNUs allows the scheduling of controllable timepoints to depend on observations made during execution. Applying DC to MAPF-TTU could therefore enable agents to adapt their execution decisions online according to observed traversal and task durations, reducing the conservativeness of fixed robust schedules obtained under Strong Controllability (SC).

However, unlike SC, which only requires computing a fixed schedule expressible through linear constraints, DC involves observation-dependent decisions and typically requires disjunctive reasoning. As a result, efficient schedule optimization methods for DC generally rely on Mixed Integer or Non-Linear Programming approaches, making the optimization



■ **Figure 4** Figure the illustrates a the empty, b the random, and c the warehouse maps.

problem significantly more challenging [10, 35]. Consequently, finding an optimal dynamically controllable execution policy may become harder than computing the collision-free paths themselves. In this work, we therefore restrict ourselves to Strong Controllability, as MAPF applications do not always guarantee online observation capabilities.

6 Experimental Evaluation

In this section, we empirically evaluate the effectiveness of our proposed methods `CBS_TTU` and `CBS_TU_STNU`. We implemented ¹ our methods on top of the framework of [26]. Several extensions of `CBS` have been proposed to improve its efficiency while preserving completeness and optimality [7]:

- *Bypassing* nodes allow the search to skip unnecessary branching when a child node resolves a conflict without increasing the solution cost. Thus reducing the size of the constraint tree.
- *Prioritized conflicts resolution* classify conflict in three types. A conflict is *cardinal* if resolving it necessarily increases the cost of both child CT node, *semi-cardinal* if it increases the cost of only one child CT node, and *non-cardinal* otherwise. Prioritizing cardinal conflicts improves `CBS` despite overburdening the low-level search.

We particularly focus on prioritized conflict resolution, as planning paths across multiple sequential goals inherently overburdens the low-level search of `CBS`. Consequently for both methods, we evaluate a version with the two improvements (respectively called `CBS_TTU_PC` and `CBS_TU_STNU_PC`) and a version without prioritizing conflict resolution (called `CBS_TTU` and `CBS_TU_STNU`, respectively).

6.1 Environment and Benchmark

We conducted the experiments on a 20-core Intel(R) Xeon(R) CPU E5-2660 at 2.6 GHz, with 64GB RAM, running Ubuntu 18, with time-out for each run set to 300 seconds. We implemented the algorithm in Python 3.14. We compare our approach on three different types of maps from [29]: a classical empty map (i.e., without obstacles) of size 16×16 (see Figure 4a), a random map of size 32×32 (see Figure 4b), a warehouse map of size 84×170 (see Figure 4c).

We adapt the generator from [26] to generate MAPF-TTU instances. In our experiments we randomly assigned start and end positions and considered a sequence of two intermediate goals for each agent. For simplicity, the current implementation assumes that each agent

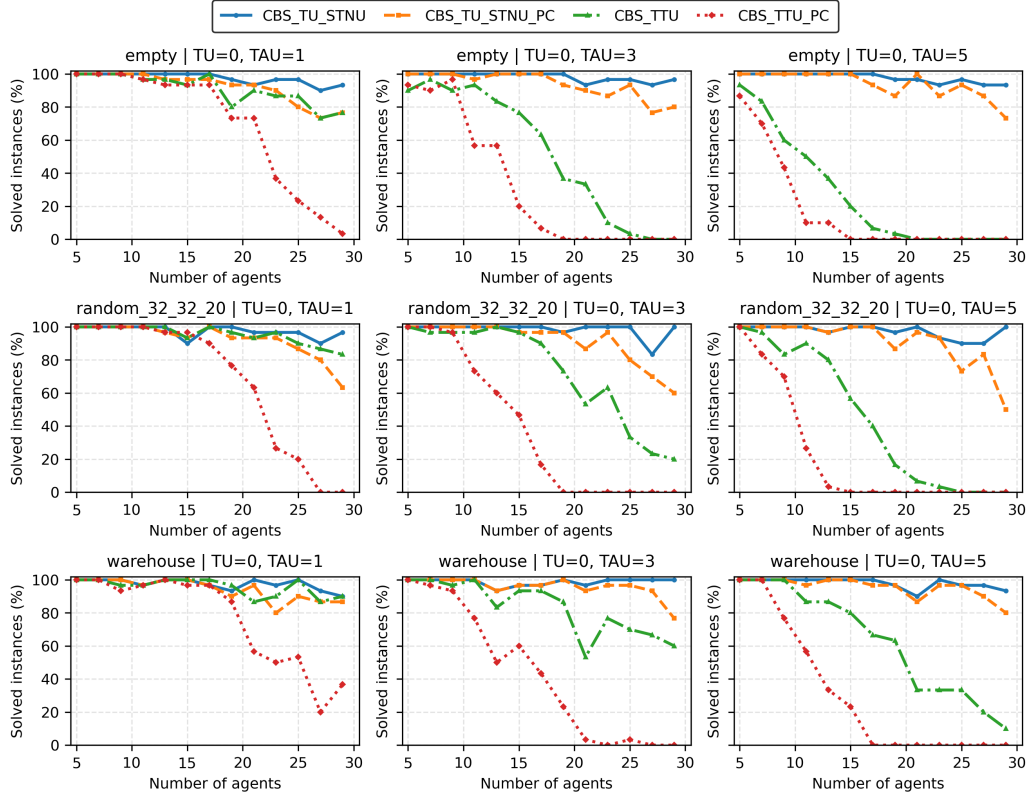
¹ The code is available at: https://github.com/ajdin14200/MAPF_TTU_TIME#

must visit its goals in the specified ordering, and does not wait before starting a task. This is a classical setting in Multi-Agent Pick-up and Delivery problems where agents must pick-up and deliver packages [25]. Please note that we extend CBS_TU so to find collision free paths for a set of goals, instead of a single goal as in the classical implementation of CBS_TU.

We distinguish the edge traversals and task executions types of uncertainty, but generate them similarly using the generator. For a given value of uncertainty U , we set $\omega^-(e)$ (resp. $\psi^-(e)$) to be a value chosen uniformly at random from the range $[1, U + 1]$, and set $\omega^+(e)$ (resp. $\psi^+(e)$) to be a value chosen uniformly at random from the range $[\omega^-(e), U + 1]$ (resp. $[\psi^-(e), U + 1]$). Thus, the parameter U allows control over the amount of uncertainty in the generated experiments, where $U = 0$ means no uncertainty. In our experimental evaluation we consider $U \in \{1, 3, 5\}$. For simplicity, we refer by TU the temporal uncertainty from traversing edges, and TAU the one from performing tasks.

For each combination of TU and TAU, we progressively increase the number of agents until the methods reach the timeout limit, generating 30 instances for each agent count.

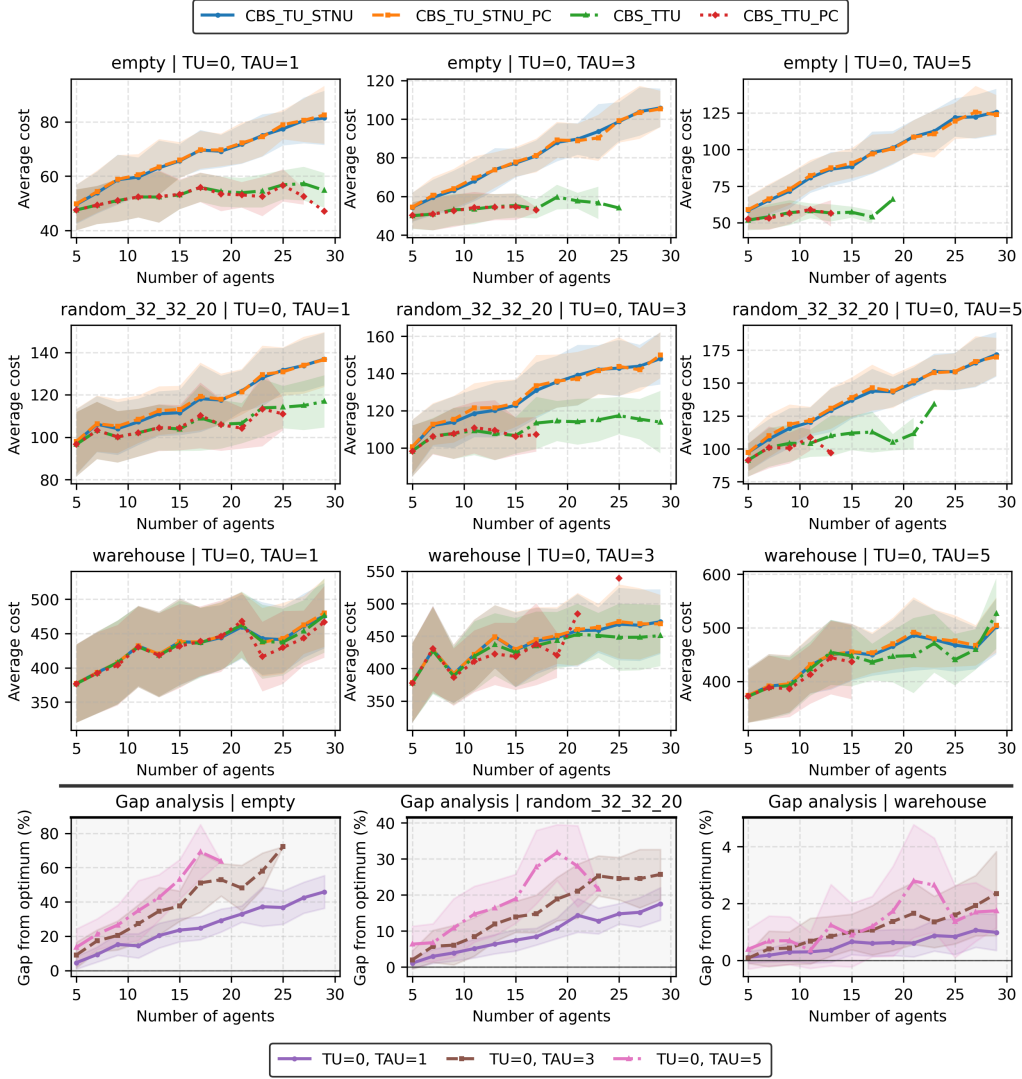
Regarding the Strong Controllability encoding, we replace the upper bound $+\infty$ with a sufficiently large constant value and solved the optimization problem using CPLEX [17] as backend solver.



■ **Figure 5** Comparison of the methods with $TU = 0$ on coverage.

6.2 Evaluation with Task Duration Uncertainty

We first evaluate the impact of task uncertainty independently from traversal uncertainty by fixing traversal uncertainty $TU = 0$, and varying task uncertainty in 1,3,5. This experiment aims to isolate the complexity introduced by uncertain task execution durations. In particular, we analyze how increasing task uncertainty affects the scalability of both methods.

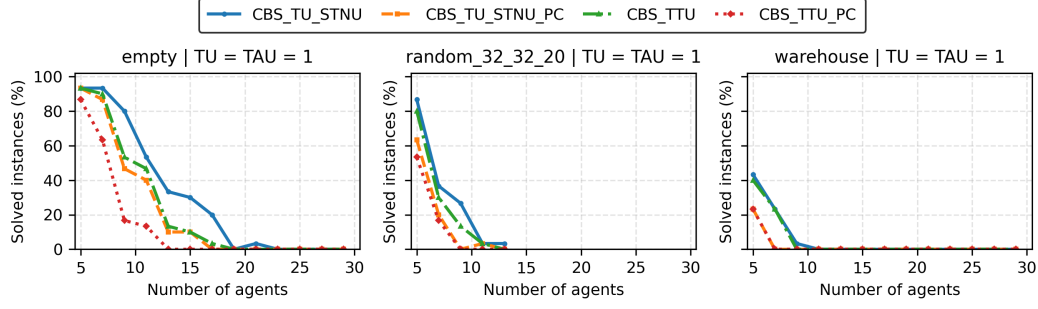


■ **Figure 6** Comparison of the methods with $TU = 0$ on cost (i.e., makespan).

Figure 5 reports the coverage (in percentage) of the different methods with respect to the number of agents. Overall, the STNU-based approaches solve more instances than CBS_TTU. Surprisingly, we observe that the prioritized conflict resolution reduces performance, suggesting that this improvement is not well suited to MAPF-TTU.

Figure 6 presents the average solution cost obtained by each method, with the corresponding standard deviation. As expected, the STNU-based approaches generate lower-quality solutions compared to the optimal methods. To quantify this difference, we compute the percentage deviation of suboptimal methods from the optimal solution cost.

The results show that CBS_TU_STNU consistently achieves a slightly smaller optimality gap than CBS_TU_STNU_PC. Consequently, the last row of Figure 6 reports the average percentage gap achieved by CBS_TU_STNU for each map, with one plot per map configuration. Interestingly, on the warehouse map, the STNU-based approach produces solutions that remain very close to the optimum.



■ **Figure 7** Comparison of the methods with $TU = TAU = 1$ on coverage.

6.3 Evaluation with Edge and Task Duration Uncertainty

We evaluate both approaches under simultaneous traversal and task uncertainty, varying both values identically in $\{1, 3, 5\}$. This evaluation studies the cumulative impact of both uncertainty sources on planning complexity and execution robustness. Combining them substantially increases potential temporal interactions between agents, as traversal uncertainty affects movement while task uncertainty impacts resource occupancy at fixed locations.

Figure 7 shows the coverage (in percentage) of the methods according to the number of agents only where $TU = TAU = 1$. We observed that adding temporal uncertainty on edges drastically impact the efficiency of the methods. For the case where $TU = TAU = 3$ and $TU = TAU = 5$, the methods show their limits under 10 agents. Overall, these results highlight that finding an optimal solution for MAPF-TTU is challenging as CBS does not scale at all.

6.4 Extension to PIBT

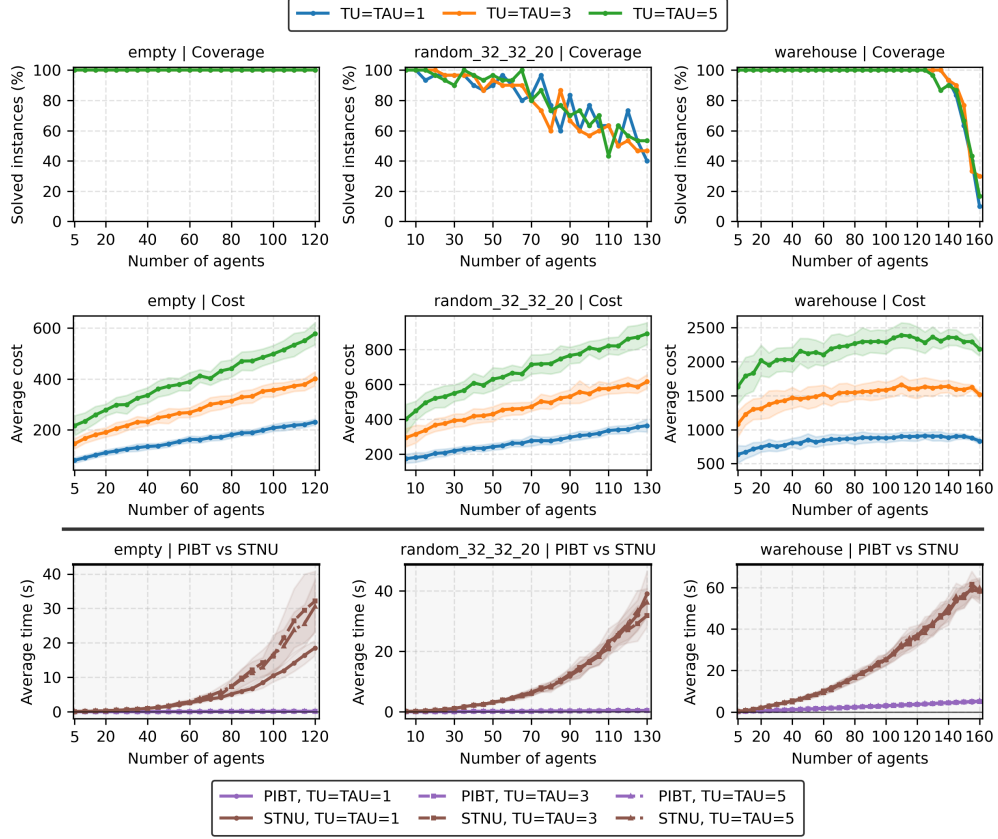
To further evaluate our STNU-based framework, we extend the experiments by replacing CBS_TU with PIBT to generate collision-free agent paths. We recall that PIBT is a suboptimal approach designed for large-scale MAPF instances involving thousands of agents [24]. Consequently, this approach, called PIBT_STNU, is overly conservative, as temporal uncertainty (both TU and TAU) is handled entirely at the STNU level rather than during the path finding process itself.

Figure 8 highlights the improvement in terms of coverage and scalability achieved by PIBT_STNU (first row), despite generating solutions of higher cost (second row). Furthermore, the third row presents a comparison of the average execution time of PIBT with that of computing an optimal strong schedule (STNU). This comparison indicates that the latter is more time-consuming.

7 Conclusion

In this paper, we introduced MAPF-TTU, a new extension of MAPF-TU in which agents must execute tasks of uncertain duration at intermediate locations. Unlike classical MAPF-TU, where uncertainty only affects traversal durations, MAPF-TTU additionally models uncertainty arising from task execution, leading to new temporal dependencies and resource conflicts between agents.

To solve this problem, we proposed two complementary approaches. First, we introduced CBS_TTU, an extension of CBS_TU that explicitly reasons about uncertain task execution during planning. Second, we proposed CBS_TU_STNU, a STNU-based approach that transforms an MAPF-TU solution into a temporally flexible execution model and optimizes its execution



■ **Figure 8** Results of PIBT_STNU on coverage (first row), cost (second row), and execution time (third row).

under uncertainty using Strong Controllability. We additionally analyzed the computational complexity of MAPF-TU and MAPF-TTU, and showed how temporal reasoning techniques can be leveraged to robustly execute MAPF plans under both traversal and task uncertainty.

Future work includes investigating Dynamic Controllability and online execution monitoring to reduce the conservativeness of Strong Controllability schedules through observation during execution. Another promising direction is the tighter integration of temporal reasoning within scalable suboptimal MAPF algorithms such as PIBT, in order to better balance solution quality, robustness, and scalability under traversal and task uncertainty. More generally, extending MAPF-TTU to online and continuously evolving coordination settings, where agents repeatedly receive new tasks during execution (i.e., lifelong settings), constitutes an important perspective for robust multi-agent coordination in dynamic environments.

References

- 1 Anton Andreychuk. Multi-agent path finding with kinematic constraints via conflict based search. In Sergei O. Kuznetsov, Aleksandr I. Panov, and Konstantin S. Yakovlev, editors, *Proceedings of the Russian Conference on Artificial Intelligence (RCAI)*, Lecture Notes in Computer Science, pages 29–45, Moscow, Russia, 2020. Springer. doi:10.1007/978-3-030-59535-7_3.
- 2 Anton Andreychuk, Konstantin Yakovlev, Pavel Surynek, Dor Atzmon, and Roni Stern. Multi-agent pathfinding with continuous time. *Artificial Intelligence*, 305:103662, 2022. doi:10.1016/J.ARTINT.2022.103662.

- 3 Dor Atzmon, Roni Stern, Ariel Felner, Glenn Wagner, Roman Barták, and Neng-Fa Zhou. Robust Multi-Agent Path Finding and Executing. *Journal of Artificial Intelligence Research*, 67:549–579, 2020. doi:10.1613/JAIR.1.11734.
- 4 Yifan Bai, Shruti Kotpalliwar, Christoforos Kanellakis, and George Nikolakopoulos. Multi-agent Path Planning Based on Conflict-Based Search (CBS) variations for Heterogeneous Robots. *Journal of Intelligent Robotic Systems*, 111(1):26, 2025. doi:10.1007/S10846-025-02229-0.
- 5 Alexander Berndt, Niels van Duijkeren, Luigi Palmieri, and Tamás Keviczky. A feedback scheme to reorder a multi-agent execution schedule by persistently optimizing a switchable action dependency graph. In Eva Onaindia Ronen I. Brafman and Shashank Shekha, editors, *Proceedings of the ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)*, 2020. arXiv:2010.05254.
- 6 Alexander Berndt, Niels van Duijkeren, Luigi Palmieri, Alexander Kleiner, and Tamás Keviczky. Receding horizon re-ordering of multi-agent execution schedules. *IEEE Transactions on Robotics*, 40:1356–1372, 2024. doi:10.1109/TR0.2023.3344051.
- 7 Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, Oded Betzalel, David Tolpin, and Solomon Eyal Shimony. ICBS: the improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the Annual Symposium on Combinatorial Search (SoCS)*, pages 223–225, Ein Gedi, the Dead Sea, Israel, 2015. AAAI Press. doi:10.1609/SoCS.V6I1.18343.
- 8 Yu Quan Chong, Jiaoyang Li, and Katia P. Sycara. Optimal task assignment and path planning using conflict-based search with precedence and temporal constraints. In Mehdi Dastani, Jaime Simão Sichman, Natasha Alechina, and Virginia Dignum, editors, *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 2210–2212, Auckland, New Zealand, 2024. International Foundation for Autonomous Agents and Multiagent Systems. doi:10.5555/3635637.3663110.
- 9 Adem Coskun, Jason M. O’Kane, and Marco Valtorta. Deadlock-free online plan repair in multi-robot coordination with disturbances. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference*, North Miami Beach, Florida, USA, 2021. doi:10.32473/FLAIRS.V34I1.128371.
- 10 Jing Cui, Peng Yu, Cheng Fang, Patrik Haslum, and Brian Charles Williams. Optimising bounds in simple temporal networks with uncertainty under dynamic controllability constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 52–60, Jerusalem, Israel, 2015. AAAI Press. URL: <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10614>.
- 11 Guilherme Dautt, Alleff Dymytry, Mariana Kolberg, and Renan Maffei. k-robust conflict-based search with continuous time for multi-robot coordination. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10260–10266, Abu Dhabi, UAE, 2024. IEEE. doi:10.1109/IR058592.2024.10801503.
- 12 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.
- 13 Ying Feng, Aditty Paul, Zhe Chen, and Jiaoyang Li. A real-time rescheduling algorithm for multi-robot plan execution. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 201–209, Banff, Alberta, Canada, 2024. AAAI Press. doi:10.1609/ICAPS.V34I1.31477.
- 14 A. Gautier, A. Stephens, B. Lacerda, N. Hawes, and M. Wooldridge. Negotiated path planning for non-cooperative multi-robot systems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 472–480, Auckland, New Zealand, 2022. International Foundation for Autonomous Agents and Multiagent Systems.
- 15 Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph W. Durham, and Nora Ayanian. Persistent and robust execution of MAPF schedules in warehouses. *IEEE Robotics and Automation Letters*, 4(2):1125–1131, 2019. doi:10.1109/LRA.2019.2894217.

- 16 Wolfgang Hönig, T. K. Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 477–485, London, UK, 2016. AAAI Press. URL: <http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13183>.
- 17 IBM. ILOG CPLEX Optimization Studio. <https://www.ibm.com/products/ilog-cplex-optimization-studio>, 2024.
- 18 Tim Tsz-Kit Lau and Biswa Sengupta. The Multi-Agent Pickup and Delivery Problem: MAPF, MARL and Its Warehouse Applications, 2022. doi:10.48550/arXiv.2203.07092.
- 19 Jiaoyang Li, Ariel Felner, Eli Boyarski, Hang Ma, and Sven Koenig. Improved heuristics for multi-agent path finding with conflict-based search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2019, pages 442–449, Macao, China, 2019. doi:10.24963/IJCAI.2019/63.
- 20 Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11272–11281, virtual, 2021. AAAI Press. doi:10.1609/AAAI.V35I13.17344.
- 21 Yihao Liu, Xueyan Tang, Wentong Cai, and Jingning Li. Multi-agent path execution with uncertainty. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 64–72, Kananaskis, Alberta, Canada, 2024. AAAI Press. doi:10.1609/SoCS.V17I1.31543.
- 22 Hang Ma, T. K. Satish Kumar, and Sven Koenig. Multi-agent path finding with delay probabilities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3605–3612, San Francisco, California, USA, 2017. AAAI Press. doi:10.1609/AAAI.V31I1.11035.
- 23 Frédéric Maris, Aidin Sumic, Thierry Vidal, and Bruno Zanuttini. Study of the computational complexity of the repair problem on simple temporal networks with uncertainty. In *Journées d’Intelligence Artificielle Fondamentale et Journées Francophones sur la Planification, la Décision et l’Apprentissage pour la conduite de systèmes, JIAF-JFPDA*, pages 101–109, Dijon, France, 2025. URL: <https://hal.science/hal-05189745v2/document>.
- 24 Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310:103752, 2022. doi:10.1016/J.ARTINT.2022.103752.
- 25 Oren Salzman and Roni Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1711–1715, Auckland, New Zealand, 2020. International Foundation for Autonomous Agents and Multiagent Systems. doi:10.5555/3398761.3398959.
- 26 Tomer Shahr, Shashank Shekhar, Dor Atzmon, Abdallah Saffidine, Brendan Juba, and Roni Stern. Safe multi-agent pathfinding with time uncertainty. *Journal of Artificial Intelligence Research*, 70:923–954, 2021. doi:10.1613/JAIR.1.12397.
- 27 Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. doi:10.1016/J.ARTINT.2014.11.006.
- 28 Devon Sigurdson, Vadim Bulitko, William Yeoh, Carlos Hernández, and Sven Koenig. Multi-agent pathfinding with real-time heuristic search. In *IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, Maastricht, Netherlands, 2018. doi:10.1109/CIG.2018.8490436.
- 29 Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Bartak. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 151–158, Napa, California, USA, 2019. doi:10.1609/SOCS.V10I1.18510.

- 30 Yifan Su, Rishi Veerapaneni, and Jiaoyang Li. Bidirectional temporal plan graph: Enabling switchable passing orders for more efficient multi-agent path finding plan execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 17559–17566, Vancouver, Canada, 2024. AAAI Press. doi:10.1609/AAAI.V38I16.29706.
- 31 Ajdin Sumic and Noëlie Ramuzat. Multi-agent Path Finding with Tasks under Time Uncertainty. Software, version 1.0., ANR-23-DMRO-0006, swhId: swh:1:dir:b90831099c338304e218af5159b6add58ed219b1 (visited on 2026-09-14). URL: https://github.com/ajdin14200/MAPF_TTU_TIME, doi:10.4230/artifacts.27850.
- 32 Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1261–1263, Atlanta, Georgia, USA, 2010. doi:10.1609/AAAI.V24I1.7767.
- 33 Jiří Švancara, David Zahrádka, Mrinalini Subramanian, Roman Barták, and Miroslav Kulich. Generating safe policies for multi-agent path finding with temporal uncertainty. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, volume 3, pages 1238–1244, Porto, Portugal, 2025.
- 34 Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):23–45, 1999. doi:10.1080/095281399146607.
- 35 Benjamin W. Wah and Dong Xin. Optimization of bounds in temporal flexible plans with dynamic controllability. *International Journal of Artificial Intelligence Tools*, 16(1):17–44, 2007. doi:10.1142/S0218213007003187.
- 36 Thayne T Walker, Nathan R Sturtevant, and Ariel Felner. Extended increasing cost tree search for non-unit cost domains. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 18, pages 534–540, 2018. doi:10.24963/IJCAI.2018/74.
- 37 Peter R. Wurman, Raffaello D’Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.
- 38 Yuanhang Zhang, Hesheng Wang, and Zhongqiang Ren. A short summary of multi-agent combinatorial path finding with heterogeneous task duration (extended abstract). In Ariel Felner and Jiaoyang Li, editors, *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 301–302, Kananaskis, Alberta, Canada, 2024. AAAI Press. doi:10.1609/SoCS.V17I1.31591.

A Appendix

In the experimental section we have shown plots of the most relevant results. In this appendix, the remaining results are shared.

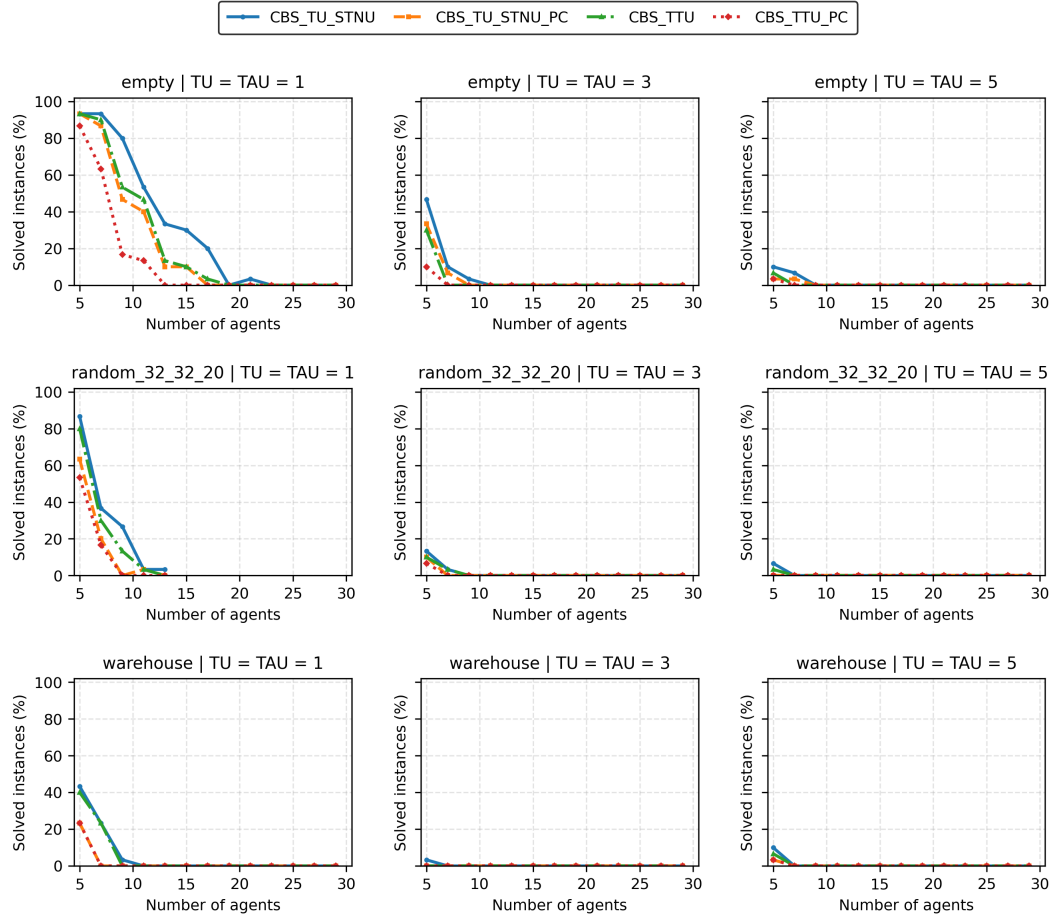
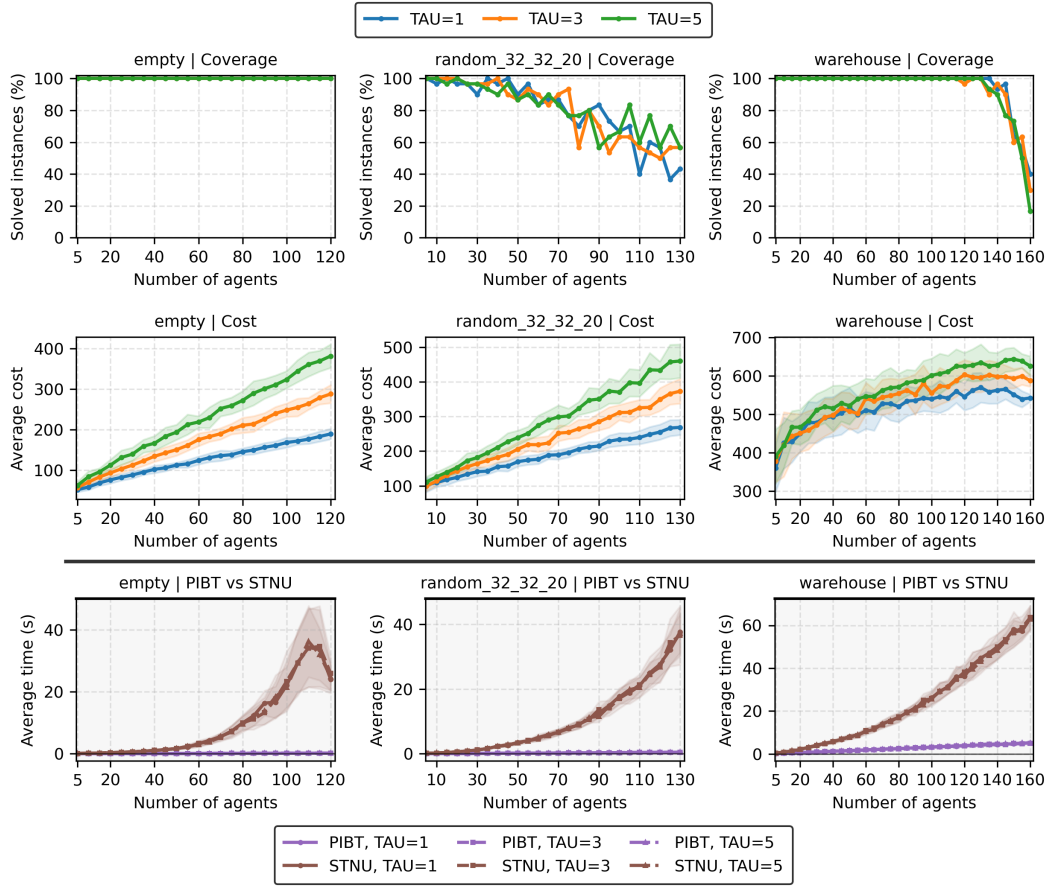


Figure 9 Complete comparison results of CBS_TTU and CBS_TU_STNU on coverage with TU and TAU in $\{1, 3, 5\}$.



■ **Figure 10** Results of PIBT with $TU = 0$.