Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# Multi-Robot Path Planning with Time Windows and Boolean Specifications

**Siyu LIU**
University of Le Havre
**Syrine Bouazza**
University of Le Havre
**Zhou He**
hezhou@sust.edu.cn
Shaanxi University of Science and Technology
**Dimitri Lefebvre**
University of Le Havre

**Research Article**

**Keywords:** Multi-robot path planning, Time windows, Boolean specifications, Heuristic algorithm, Beam search

**Posted Date:** November 16th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-9708502/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# Multi-Robot Path Planning with Time Windows and Boolean Specifications

Siyu Liu[1], Syrine Bouazza[1], Zhou He[2*], Dimitri Lefebvre[1]

[1]Université Le Havre Normandie, GREAH, 25 Rue Philippe Lebon, Le Havre, 76600, France.

[2]School of Electrical and Control Engineering, Shaanxi University of Science and Technology, No.6 Xuefu Road, Weiyang District, Xi'an, 710021, China.

*Corresponding author(s). E-mail(s): hezhou@sust.edu.cn;
Contributing authors: siyu.liu@etu.univ-lehavre.fr;
syrine.bouazza@univ-lehavre.fr; dimitri.lefebvre@univ-lehavre.fr;

**Abstract**

Path planning for multi-robot systems under complex task-logical specifications and strict temporal constraints poses significant combinatorial challenges. Existing approaches often handle logical and temporal requirements separately, limiting their scalability and practical applicability. To overcome this limitation, we propose a unified planning framework that seamlessly integrates Boolean specifications with time-window constraints. The problem is first formalized as a Mixed-Integer Linear Programming model that captures logical dependencies and timing requirements, including a collision avoidance module. To improve scalability for large-scale instances, we develop a Beam Search-based heuristic algorithm tailored for handling time windows. A similarity-based filtering mechanism is further incorporated to promote solution diversity and mitigate premature convergence. Experimental results demonstrate that the proposed approach achieves a desirable trade-off between computational efficiency and solution quality, offering a robust and scalable solution for multi-robot path planning under complex temporal-logical specifications.

**Keywords:** Multi-robot path planning, Time windows, Boolean specifications, Heuristic algorithm, Beam search

# 1 Introduction

Nowadays, Multi-Robot Systems (MRS) are deployed in many real-world scenarios, such as intelligent warehouses, disaster response, and smart factories (Chakraa, Guérin, Leclercq, & Lefebvre, 2023). Standard robot path planning methods typically focus on generating collision-free paths between a start and a goal region (N. Liu et al., 2025). As robot control problems become increasingly complex, path planning for complex tasks has attracted significant attention in recent years. Temporal logics, such as Linear Temporal Logic (LTL) (Kloetzer & Mahulea, 2020; Lv, Luo, Ma, Li, & Yin, 2023; Yang, Yin, Li, & Zamani, 2020; Yu, Yin, Li, & Li, 2022) and Signal Temporal Logic (STL) (Sun, Chen, Mitra, & Fan, 2022), provide rich expressiveness for complex temporal and timing requirements. However, the combinatorial complexity of satisfying nested temporal formulas often restricts them to smaller-scale applications as the system scales up. To ensure high scalability for large robot fleets, Boolean specifications (Mahulea & Kloetzer, 2017; Shi, He, Ma, Ran, & Yin, 2023; Shi, He, Tang, Liu, & Ma, 2022; Zhang, Luo, Long, Huang, & Wu, 2020) focusing on the combinatorial logic of spatial tasks effectively capture the core logical connectives of temporal specifications and significantly mitigate the computational burden.

Despite their scalability, Boolean specifications focus solely on logical correctness without considering temporal constraints. In practical applications, robots (or agents) are expected not only to satisfy high-level task specifications such as Boolean specifications but also to perform real-world tasks with strict temporal requirements, including transportation, inspection, and collaborative assembly (Nunes, Manner, Mitiche, & Gini, 2017). These tasks are often associated with specific start and end times, and failure to meet these deadlines can undermine the overall coordination and effectiveness of the system. Since such tasks are typically time-sensitive, focusing solely on logical correctness without considering temporal constraints often leads to infeasible or suboptimal execution in MRS. Therefore, integrating logical correctness with temporal feasibility has emerged as a key challenge in multi-robot path planning.

To better illustrate this need, we present a hospital disinfection scenario (Fig. 1) in which a team of robots is assigned to disinfect multiple mission-critical regions, subject to both temporal constraints and logical task dependencies. The team is initially located in the lobby (Region $\Pi_4$), and the hospital has different regions that require disinfection. Regions $\Pi_1$ and $\Pi_2$ are operating rooms where a surgery is scheduled to start at time 3. The robot needs to disinfect one of the operating rooms before time 3. The disinfection process in either operating room takes 1 hour. The task is scheduled to ensure that the robot finishes disinfecting one of the operating rooms before time 3, thus ensuring the room is ready for the surgery. Region $\Pi_3$ is the waiting area where the robot needs to disinfect the area between time 1 and 2. The disinfection process takes 1 hour, and this time window allows the robot to complete the disinfection before patients arrive, ensuring the waiting area remains hygienic. One robot must finally perform routine disinfection in the lobby (Region $\Pi_4$).

In this paper, we address the multi-robot path planning problem involving time windows and Boolean specifications. A unified framework is proposed, in which temporal constraints are integrated into the Boolean formalism. Unlike existing logic-based methods that ignore temporal constraints (Mahulea & Kloetzer, 2017; Shi et al.,

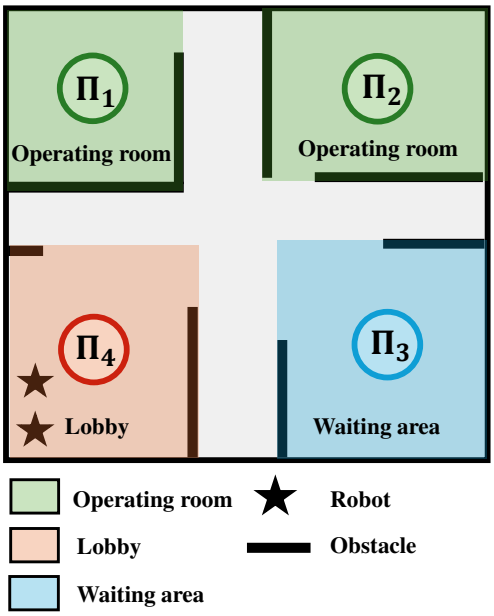



**Fig. 1**: An example of multi-robot system

2022; Zhang et al., 2020), or other approaches that lack expressive logical specifications (Alharbi, Stohy, Elhenawy, Masoud, & Khalifa, 2021; McIntire, Nunes, & Gini, 2016; Nunes et al., 2017), the proposed method simultaneously accounts for both aspects. This capability is essential for complex, time-sensitive missions with logical task dependencies. To the best of our knowledge, this is the first framework that systematically unifies Boolean specifications and time window constraints within a scalable planning paradigm for MRS. We further formalize the problem with a Mixed-Integer Linear Programming (MILP) model to compute the optimal solution. Since the problem is NP-hard and difficult to scale, we propose a heuristic algorithm based on the Beam Search (BS) algorithm. In particular, this work extends the classical BS algorithm to address temporal constraints. The extended version of BS integrates a time-unfolding process, explicitly designed to model and manage temporal information within the decision-making procedure. While most existing methods focus on the correctness of high-level task plans, they often overlook execution-level feasibility, such as collision avoidance in shared environments. In this work, we integrate a scheduling refinement into the Beam Search algorithm, where time sampling provides robots with the flexibility of in-region waiting, thereby avoiding collisions. Finally, we compare the performance and computation time of the proposed method under the effects of the number of robots, time window settings, and the number of conjunctions in the Boolean specification. MILP results are used as a benchmark. In summary, our main contributions are as follows:

- We model a problem that simultaneously addresses Boolean specifications and time window constraints for MRS path planning.

- We develop a novel extension of BS that incorporates a time-unfolding mechanism, enabling explicit temporal reasoning within the combinatorial optimization process and enhancing scalability to large-scale scenarios.
- We analyze the performance of our algorithm experimentally. As reported later, our approach can effectively generate scalable solutions. We also introduce a refinement module to improve execution safety by ensuring a collision-free execution of the path plan.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the relevant literature. Section 3 introduces the formal problem setting, followed by Section 4, which presents the MILP formulation. Section 5 describes the beam-search-based approach. Section 6 reports the experimental results and compares the performance of the proposed method with the exact MILP solution. Finally, Section 7 concludes the paper and discusses potential directions for future work.

## 2 State of the Art

In recent years, Boolean specifications have become widely used in MRS planning due to their capability to succinctly express complex logical relationships among sub-tasks. Among modeling methods, Petri nets (PNs) have become a dominant method for modeling these Boolean specifications. For example, one work formulates Boolean tasks as binary decision variables within an Integer Linear Programming framework to derive optimal robot paths (Mahulea & Kloetzer, 2017). Building upon this, the authors introduce safety constraints and labeled PNs to enhance modeling expressiveness (Shi et al., 2023). Additionally, the authors propose a structured PNs model that aims to minimize total travel distance by leveraging the inherent characteristics of PNs structures (Zhang et al., 2020). Regarding scalability, a hybrid method that combines Dijkstra's algorithm with simulated annealing demonstrates improved scalability and computational efficiency for large-scale MRS under complex Boolean constraints (Shi et al., 2022). More recently, an optimal planning approach first constructs a basis reachability graph offline and uses an online ILP to compute optimal paths in real time (He, Yuan, Ran, & Lefebvre, 2025).

Temporal constraints have been extensively studied across various domains, such as Multi-Robot Task Allocation (MRTA), as comprehensively reviewed (Nunes et al., 2017). In broader scheduling theory, problems like the Vehicle Routing Problem (VRP) and the Team Orienteering Problem (TOP) with time-related constraints have also received sustained attention (Gunawan, Lau, Vansteenwegen, & Lu, 2017; X. Liu, Chen, Por, & Ku, 2023). Meanwhile, significant efforts have also been made to address temporal constraints, specifically time windows and precedence constraints, in classical VRP frameworks and related variants. Approaches such as those presented previously (Ait Haddadene, Labadie, & Prodhon, 2016; Frey, Jungwirth, Frey, & Kolisch, 2023) have effectively handled time-constrained tasks for scheduling and routing applications, but generally lack the capability to represent rich logical task structures involving disjunctions, exclusions, or precedence relations. In multi-robot scenarios, distributed scheduling frameworks considering time windows have been explored (Barb-Ciorbea, Kloetzer, & Mahulea, 2024). Additionally, Mixed Integer

Programming-based solutions (Zhou, Che, & Yan, 2012), Constraint Programming models (Ham, 2020), and iterative auction-based heuristics like the Temporal-and Precedence-constrained Sequential Single-Item auction (McIntire et al., 2016) have been adapted for MRTA problems with temporal constraints. However, Mixed Integer Programming and Constraint Programming approaches typically require intricate formulations and quickly become computationally infeasible as the problem scales. Although heuristic algorithms offer better scalability, they often specialize in either logical constraints or temporal constraints, rarely providing a unified solution capable of handling both simultaneously.

Beyond task allocation and temporal feasibility, avoiding inter-robot collisions is a fundamental requirement when multiple agents operate in the same environment. Without effective mechanisms, task execution may fail and safety risks increase (Madridano, Al-Kaff, Martín, & de la Escalera, 2021). Previous work typically implements collision avoidance strategies at two levels: the *routing level* and the *navigation level* (Chakraa, Leclercq, Guérin, & Lefebvre, 2025; Kiadi, García, Villar, & Tan, 2023; Petitprez, Guérin, Guinand, Germain, & Kerthe, 2023; Wang, Kloetzer, Mahulea, & Silva, 2015). At the routing level, conflicts are proactively removed during planning by explicitly modeling time so that no two robots occupy the same cell (or region) simultaneously. Common mechanisms include introducing initial start-time offsets and reserving spatiotemporal resources; for example, staggering departures can reduce contention in shared areas (Wang et al., 2015). Other approaches integrate A* with constraint propagation or evolutionary coordination to identify and resolve potential conflicts at plan time (Kiadi et al., 2023). Such planning-time methods generally increase the modeling and computational complexity and are most suitable when robots can cooperate and communicate. At the navigation level, collisions are handled online through local sensing and reactive rules (e.g., repulsive-velocity or potential-field schemes), typically without changing the precomputed global route (Chakraa et al., 2025; Petitprez et al., 2023). While lightweight and scalable, purely local policies may be unsuitable with respect to global task or time-window constraints and, in adverse geometries, can lead to deadlocks (e.g., circular waiting among three or more robots).

In summary, existing methods for MRS planning still face two limitations: (i) it is difficult to simultaneously capture Boolean task specifications, time windows, and collision-avoidance constraints in a unified framework; and (ii) exact-solver pipelines face scaling bottlenecks, and problem-specific heuristics lack robust transfer across constraint combinations. To address these issues, we propose a unified framework that jointly accounts for Boolean specifications, time windows, and shared-region collision avoidance within one model: an MILP serves as the optimal benchmark, and a scalable BS-based heuristic is employed for large instances.

# 3 Problem statement

## 3.1 Environment Modeling

To formalize the problem, the environment in which the robots operate is discretized into a finite set of regions denoted by $\Omega$. Each region corresponds to a spatial location

of interest, and we write $\Omega = \{\Pi_1, \Pi_2, \ldots, \Pi_{|\Omega|}\}$[1]. A *path* in this paper is defined as a sequence of regions successively visited by a robot, together with the corresponding sequence of visiting times.

The environment is modeled as a directed and weighted graph $G = (\Omega, E)$, where the node set $\Omega$ represents the regions, and the edge set $E \subseteq \Omega \times \Omega$ captures the feasible transitions between adjacent regions. Each edge $(\Pi_i, \Pi_j) \in E$ is assigned an elementary cost $c_{i,j} \in [0, \infty)$, representing the direct travel time from region $\Pi_i$ to region $\Pi_j$. To incorporate static obstacles and environmental constraints, transitions through infeasible or non-adjacent regions are assigned an infinite cost, i.e., $c_{i,j} = \infty$ if the transition is blocked or not allowed.

For any two regions $\Pi_i, \Pi_j \in \Omega$, we compute the shortest path $P_{i,j}$ using Dijkstra's algorithm, and denote its travel time by $T(i, j)$. Note that $c_{i,j}$ denotes the direct edge cost (defined only for $(\Pi_i, \Pi_j) \in E$), while $T(i, j)$ represents the minimal travel time over all possible paths between $\Pi_i$ and $\Pi_j$, computed from the edge costs.

## 3.2 Boolean Specifications

In this paper, the considered path planning problem is to determine the path of each robot in a team of robots such that a given travel task is completed, while minimizing a weighted sum of (i) the maximum completion time among all robots, and (ii) the sum of completion times over all robots. The travel task is specified using Boolean specifications together with time-window parameters. For any region $\Pi \in \Omega$ that appears in the specification, we associate temporal parameters: $ES(\Pi)$, $LS(\Pi)$, $DU(\Pi)$, with $0 \leq ES(\Pi) \leq LS(\Pi) \leq +\infty$ and $DU(\Pi) \in \mathbb{R}_{\geq 0}$. These parameters denote the earliest and latest admissible start times and the required duration of stay, respectively.

Atomic propositions are categorized into three finite sets:

- $P_v \subseteq \Omega \times \mathbb{R}^2_{\geq 0} \times \mathbb{R}_{\geq 0}$: a tuple $(\Pi, [ES(\Pi), LS(\Pi)], DU(\Pi))$ requires that a robot enters region $\Pi$ at some time $z \in [ES(\Pi), LS(\Pi)]$ and remains there for at least $DU(\Pi)$ time units.
- $P_f \subseteq \Omega \times \mathbb{R}^2_{\geq 0}$: a pair $(\Pi, [ES(\Pi), LS(\Pi)])$ requires that a robot reaches region $\Pi$ at some time $z \in [ES(\Pi), LS(\Pi)]$ and then stays indefinitely (equivalently, $DU(\Pi) = +\infty$).
- $P_a \subseteq \Omega$: a simple reachability requirement, meaning that a robot must eventually visit region $\Pi$, without explicit time-window or duration constraints.

In the above definitions, atomic proposition $(\Pi_i, [ES, LS], DU)$ is true if and only if the region marked as $\Pi_i$ is visited by at least one robot in the specified time window $[ES, LS]$ and the visit lasts for $DU$ time units. Similarly, the atomic proposition $(\Pi_i, [ES, LS])$ is true if and only if region $\Pi_i$ is visited and then permanently occupied by at least one robot during the time interval $[ES, LS]$. However, atomic proposition $(\Pi_i)$ is false if and only if the region $\Pi_i$ is not accessed by any robots.

By combining these atomic propositions with the logical operators conjunction ($\wedge$), disjunction ($\vee$), and negation ($\neg$), the travel task $\varphi$ is defined as a general specification for MRS and consists of three sub-specifications:

[1] $|\Omega|$ denotes the number of regions in the set $\Omega$.

$$\varphi = Y \wedge A \wedge U \tag{1}$$

1) The sub-specification $Y$ represents the logical requirements for the intermediate regions:

$$\begin{cases} Y = y_1 \wedge \cdots \wedge y_m \\ y_j = \bigvee\limits_{(\Pi_i, [ES, LS], DU) \in P_{y_j}} (\Pi_i, [ES, LS], DU), \quad j = 1, \ldots, m, \end{cases} \tag{2}$$

where $m$ represents the total number of conjunctions in sub-specification $Y$, i.e., the number of terms being conjoined. Here, $P_{y_j} \subseteq P_v$ denotes the subset of atomic propositions for satisfying the $j$-th intermediate requirement.

2) The sub-specification $A$ represents the logical requirements on prohibited regions that should be avoided along the paths:

$$A = \bigwedge_{(\Pi_i) \in P_a} \neg(\Pi_i) = \neg \left( \bigvee_{(\Pi_i) \in P_a} (\Pi_i) \right). \tag{3}$$

3) The sub-specification $U$ is the logical requirements for the final regions:

$$\begin{cases} U = u_1 \wedge \cdots \wedge u_q \\ u_j = \bigvee\limits_{(\Pi_i, [ES, LS]) \in P_{u_j}} (\Pi_i, [ES, LS]), \quad j = 1, \ldots, q, \end{cases} \tag{4}$$

where $q$ represents the total number of conjunctions in sub-specification $U$. Here, $P_{u_j} \subseteq P_f$ denotes the subset of atomic propositions for the $j$-th final-state requirement.
**Example 1:** *The task illustrated in the motivational example shown in Fig. 1 can be formalized by the following Boolean formula:*

$$\varphi_1 = ((\Pi_1, [0, 3], 1) \vee (\Pi_2, [0, 3], 1)) \wedge (\Pi_3, [1, 2], 1) \wedge (\Pi_4, [0, \infty])$$

*This formula specifies that at least one robot must visit either region* $\Pi_1$ *or* $\Pi_2$ *within the time window* $[0, 3]$ *and remain there for at least 1 time unit. Subsequently, region* $\Pi_3$ *must be visited by at least one robot during the interval* $[1, 2]$*, also for a minimum duration of 1 time unit. Finally, region* $\Pi_4$ *must be visited at an arbitrary time and remain occupied indefinitely by at least one robot.*

## 3.3 Collision avoidance constraints

To ensure execution safety, we impose *region-conflict* constraints on the transit regions along the planned paths. These regions are traversed by the paths between consecutive

regions selected to satisfy the Boolean specification. Let $\mathrm{occ}(r, \Pi, t) \in \{0, 1\}$ indicate whether robot $r$ occupies a transit region $\Pi$ at time $t$. The basic constraint is

$$\mathrm{occ}(r_i, \Pi, t) + \mathrm{occ}(r_j, \Pi, t) \;\leq\; 1, \qquad \forall\, i \neq j,\ \forall\, \Pi \in \Omega,\ \forall\, t. \tag{5}$$

We further introduce a minimum safety interval $\delta > 0$ so that two robots visiting the same region must be separated by at least $\delta$ time units. If robot $r_i$ occupies $\Pi$ over $[t_i^{\mathrm{start}}, t_i^{\mathrm{end}}]$ and robot $r_j$ over $[t_j^{\mathrm{start}}, t_j^{\mathrm{end}}]$, then

$$\left(t_j^{\mathrm{start}} \;\geq\; t_i^{\mathrm{end}} + \delta\right) \;\vee\; \left(t_i^{\mathrm{start}} \;\geq\; t_j^{\mathrm{end}} + \delta\right). \tag{6}$$

Fig. 2 illustrates two robots visiting the same region $\Pi$ at nonoverlapping intervals while respecting the safety gap $\delta$.

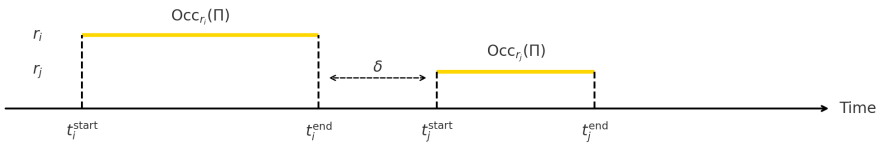



**Fig. 2**: Illustration of the region-conflict requirement with minimum safety interval $\delta$: robots $r_i$ and $r_j$ occupy region $\Pi$ at disjoint times, separated by at least $\delta$.

# 4 Mixed-Integer Linear Programming models

In this section, we solve the problem using an MILP model that provides optimal solutions as a benchmark. To establish the MILP model, we consider a set of robots denoted by $R$, where each robot $r \in R$ starts from a predefined initial region $\mathbf{\Pi}_S^r$.

Each conjunction includes logical requirements for the intermediate regions associated with a set of candidate regions. Visiting any region in this set is sufficient to satisfy the corresponding logical requirements. We define the set of regions associated with the $j$-th intermediate requirement as $\mathbf{\Pi}_{y_j} = \left\{\Pi_i \,\middle|\, (\Pi_i, [ES_{v_i}, LS_{v_i}], DU_{v_i}) \in P_{y_j}\right\}$. The union of all such sets is denoted by $\mathbf{\Pi}_Y = \bigcup_{j=1}^{m} \mathbf{\Pi}_{y_j}$. Similarly, we define the set of regions associated with the $j$-th final requirement as $\mathbf{\Pi}_{u_j} = \left\{\Pi_i \,\middle|\, (\Pi_i, [ES_{f_i}, LS_{f_i}]) \in P_{u_j}\right\}$. The union of all such final-region sets is denoted by $\mathbf{\Pi}_U = \bigcup_{j=1}^{q} \mathbf{\Pi}_{u_j}$. The logical requirements on prohibited regions form the set $\mathbf{\Pi}_a = \{\Pi \mid (\Pi) \in P_a\}$. Meanwhile, we introduce two auxiliary sets: $\mathbf{\Pi}^+ = \mathbf{\Pi}_Y \cup \mathbf{\Pi}_U$, the set of all intermediate and terminal regions; and $\mathbf{\Pi}' = \mathbf{\Pi}_Y \cup \{\mathbf{\Pi}_S^r \mid r \in R\}$, the set of all intermediate regions and robot start regions. Each region $\Pi_k \in \mathbf{\Pi}^+$ is associated with a time window $[ES(k), LS(k)]$, where $ES(k)$ is the earliest start time and

$LS(k)$ is the latest start time for visiting the region. Additionally, each region requires a duration of $DU(k)$ for its visit. For any two regions $\Pi_k$ and $\Pi_{k'}$, we define $p_{k,k'}$ as the shortest path connecting them on the region graph $G = (\Omega, E)$. The corresponding minimal travel time is denoted by $T(k, k')$, computed as the sum of elementary costs over the path $p_{k,k'}$: $T(k, k') = \sum_{(\pi,\pi')\in p_{k,k'}} c(\pi, \pi')$.

We now introduce the decision variables used in the MILP formulation:

- $o^r_{k,k'} \in \{0, 1\}, \quad \forall r \in R,\ \forall k \in \mathbf{\Pi}',\ \forall k' \in \mathbf{\Pi}^+$with $k' \neq k$: a binary variable equal to 1 if robot $r$ travels from region $\Pi_k$ to region $\Pi_{k'}$, and 0 otherwise;

- $S_k$: the actual start time of visiting region $\Pi_k$, for $k \in \mathbf{\Pi}^+$;
- $F_k$: the actual finish time at region $\Pi_k$, for $k \in \mathbf{\Pi}^+$.

In addition, we associate the following derived variables:

- $C_r \geq 0, \quad \forall r \in R$: the completion time of robot $r$;
- $C_{\max}$: the maximum completion time among all robots;
- $T_{\text{sum}}$: the sum of the completion times for all robots.

To enhance clarity, Table 1 summarizes the key notations, decision variables, and derived variables used in the MILP model. Given the above-defined objectives, parameters, and variables, the problem can be formalized as follows:

$$\min \quad \alpha\, C_{\max} + (1-\alpha)\, T_{\text{sum}} \tag{7}$$

$$\text{s.t.} \quad \sum_{k'\in\mathbf{\Pi}^+} o^r_{\mathbf{\Pi}^r_S,k'} = 1, \quad \forall\, r \in R \tag{8}$$

$$F_{\mathbf{\Pi}^r_S} = 0,\ \forall r \in R \tag{9}$$

$$\sum_{r\in R} \sum_{k\in\mathbf{\Pi}_{y_j}} \sum_{i\in\mathbf{\Pi}'} o^r_{i,k} \geq 1, \quad \forall\, j = 1, \ldots, m \tag{10}$$

$$\sum_{r\in R} \sum_{k\in\mathbf{\Pi}_{u_j}} \sum_{i\in\mathbf{\Pi}'} o^r_{i,k} \geq 1, \quad \forall\, j = 1, \ldots, q \tag{11}$$

$$\sum_{k\in\mathbf{\Pi}^+} \Big[ \sum_{i\in\mathbf{\Pi}'} o^r_{i,k} - \sum_{j\in\mathbf{\Pi}^+} o^r_{k,j} \Big] = 1, \quad \forall\, r \in R \tag{12}$$

$$\sum_{j\in\mathbf{\Pi}^+} o^r_{k,j} \leq \sum_{i\in\mathbf{\Pi}'} o^r_{i,k}, \quad \forall\, r \in R,\ \forall\, k \in \mathbf{\Pi}_Y \tag{13}$$

$$ES(k) \leq S_k \leq LS(k), \quad \forall\, k \in \mathbf{\Pi}^+ \tag{14}$$

$$S_{k_2} \geq F_{k_1} + T(k_1, k_2) - M \cdot (1 - o^r_{k_1,k_2}), \tag{15}$$

$$\forall\, r \in R,\ \forall k_1 \in \mathbf{\Pi}',\ \forall k_2 \in \mathbf{\Pi}^+ \text{s.t. } k_2 \neq k_1 \tag{16}$$

$$F_k \geq S_k + DU(k), \quad \forall\, k \in \mathbf{\Pi}_Y \tag{17}$$

$$
\begin{cases}
C_r \geq F_k - M\left(1 - \left[\sum\limits_{i \in \mathbf{\Pi}'} o^r_{i,k} - \sum\limits_{j \in \mathbf{\Pi}^+} o^r_{k,j}\right]\right), \quad \forall r \in R,\ \forall k \in \mathbf{\Pi}_Y, \\
C_r \geq S_k - M\left(1 - \sum\limits_{i \in \mathbf{\Pi}'} o^r_{i,k}\right), \quad \forall r \in R,\ \forall k \in \mathbf{\Pi}_U, \\
C_{\max} \geq C_r, \quad \forall r \in R, \\
T_{\text{sum}} = \sum\limits_{r \in R} C_r.
\end{cases} \tag{18}
$$

The objective function in Eq. (7) minimizes a weighted sum of the maximum completion time $C_{\max}$ and the total completion $T_{\text{sum}}$. Minimizing $C_{\max}$ enhances system responsiveness by reducing the overall completion time, while minimizing $T_{\text{sum}}$ encourages the robots to finish their assigned routes as early as possible. The parameter $\alpha \in [0, 1]$ controls the trade-off between the two objectives.

**Table 1**: Global notations, decision variables, and derived variables.

| Notation | Description |
|---|---|
| **Sets and Parameters** | |
| $R$ | Set of robots |
| $\mathbf{\Pi}^r_S$ | Initial region of robot $r$ |
| $\mathbf{\Pi}_a$ | Set of avoidance (prohibited) regions |
| $\mathbf{\Pi}_{y_j}$ | Set of candidate regions for the $j$-th intermediate requirement |
| $\mathbf{\Pi}_Y$ | $\mathbf{\Pi}_Y = \bigcup_{j=1}^{m} \mathbf{\Pi}_{y_j}$: union of all intermediate region sets |
| $\mathbf{\Pi}_{u_j}$ | Set of candidate regions for the $j$-th final requirement |
| $\mathbf{\Pi}_U$ | $\mathbf{\Pi}_U = \bigcup_{j=1}^{q} \mathbf{\Pi}_{u_j}$: union of all final region sets |
| $\mathbf{\Pi}^+$ | Set of all intermediate and final regions: $\mathbf{\Pi}_Y \cup \mathbf{\Pi}_U$ |
| $\mathbf{\Pi}'$ | Set of all intermediate and start regions: $\mathbf{\Pi}_Y \cup \{\mathbf{\Pi}^r_S \mid r \in R\}$ |
| $ES(k)$ | Earliest start time for region $\Pi_k$ |
| $LS(k)$ | Latest start time for region $\Pi_k$ |
| $DU(k)$ | Required duration to stay in region $\Pi_k$ |
| $T(k, k')$ | Travel time along path $P_{k,k'}$, $T(k, k') = \sum_{(\pi,\pi') \in P_{k,k'}} c(\pi, \pi')$ |
| $\alpha$ | Weight balancing makespan and total completion time in the objective |
| $\delta$ | Minimum safety time gap between consecutive visits to the same region |
| $M$, $M_{\text{col}}$ | Big-M constants used for linearization |
| **Decision Variables** | |
| $o^r_{k,k'}$ | Binary variable: 1 if robot $r$ travels from $\Pi_k$ to $\Pi_{k'}$ |
| $S_k$ | Start time of visit to region $\Pi_k$ |
| $F_k$ | Finish time of visit to region $\Pi_k$ |
| $z^p_{r,r',i,j}$ | Binary precedence variable for collision avoidance |
| **Derived Quantities** | |
| $C_r$ | Completion time of robot $r$ |
| $C_{\max}$ | Maximum completion time across all robots (makespan) |
| $T_{\text{sum}}$ | Sum of completions for all robots |
| $A_{r_i,r_{i+1}}(p)$ | Arrival instant at $\Pi_p$ along $P_{r_i,r_{i+1}}$ |
| $L_{r_i,r_{i+1}}(p)$ | Leave end instant at $\Pi_p$ along $P_{r_i,r_{i+1}}$ |

[2]Here, $m$ denotes the total number of conjunctions in sub-specification $Y$, as defined in Section 3.
[3]Here, $q$ denotes the total number of conjunctions in sub-specification $U$, as defined in Section 3.

Eq. (8) ensures each robot begins its route from its specified initial region; Eq. (9) sets $F_{\mathbf{\Pi}_S^r} = 0$ as the time origin; Eq. (10) and (11) encode the Boolean specifications, guaranteeing that at least one region from each conjunction in both intermediate and final region sets is visited; Eq. (12) guarantees that every robot ends at exactly one final region, thus preventing incomplete or infinite paths; Eq. (13) enforces continuity and consistency of robot paths by ensuring that every non-final region visited has at most one successor; Eq. (14) enforces that each region visit must start within its specified time window; Eq. (16) ensures the temporal feasibility of consecutive visits using a Big-M linearization technique. Specifically, for any transition between two regions $\Pi_1$ and $\Pi_2$ by robot $r$, the start time at $\Pi_2$ must be no earlier than the finish time at $\Pi_1$ plus the travel time $T(\Pi_1, \Pi_2)$. If the transition is inactive, the constraint is relaxed via a sufficiently large constant $M$; Eq. (17) constrains the actual finish time at each intermediate region to be no less than the sum of its start time and required service duration; Eq. (18) defines $C_{\max}$ by bounding it from below using the finish times of intermediate regions and the start times of final regions, effectively linearizing the maximum completion time. The term $T_{\text{sum}}$ explicitly captures the sum of the completion times for all robots. The parameter $\alpha$ promotes a balanced distribution of workload.

## 4.1 Collision-avoidance refinement

This refinement stage keeps the path plan fixed: the arc-selection variables $\hat{o}^r_{k,k'} \in \{0, 1\}$ obtained from the MILP (first-stage) are treated as constants, and only timing variables are adjusted to eliminate overlaps in shared regions. For each robot $r$, let

$$\widehat{O}^r \;=\; \Pi_{r_1}\,\Pi_{r_2}\,\cdots\,\Pi_{r_{n_r}},$$

be the ordered sequence of regions that satisfies the Boolean specifications, induced by the active arcs with $\hat{o}^r_{k,k'} = 1$, where $n_r = |\widehat{O}^r|$. For every consecutive pair $(\Pi_{r_i}, \Pi_{r_{i+1}})$, denote by

$$P_{r_i,r_{i+1}} \;=\; \Pi_{r_i}\,\widetilde{\Pi}^r_{i,1}\,\cdots\,\widetilde{\Pi}^r_{i,\ell_i}\,\Pi_{r_{i+1}}$$

the shortest path sequence on the region graph $G = (\Omega, E)$, where $\widetilde{\Pi}^r_{i,j}$ are intermediate transit regions and $\ell_i \geq 0$. For later use, define

$$\tilde{P}_{r_i,r_{i+1}} \;=\; \widetilde{\Pi}^r_{i,1}\,\cdots\,\widetilde{\Pi}^r_{i,\ell_i}\,\Pi_{r_{i+1}},$$

which avoids double-counting service at $\Pi_{r_i}$ when enforcing shared-region timing constraints. We write $V(\tilde{P}_{r_i,r_{i+1}})$ for the set of distinct regions visited along $\tilde{P}_{r_i,r_{i+1}}$. To ensure scalability, collision-avoidance constraints are strictly enforced on these intermediate transit regions to resolve resource contention. Intermediate and final regions ($\mathbf{\Pi}^+$) are treated as safe zones for concurrent occupancy, assuming local reactive systems manage maneuvering within these areas.

We now introduce the decision variables used in the refinement MILP:

- $S_k$: the actual start time of visiting region $\Pi_k$, for $k \in \mathbf{\Pi}^+$;
- $F_k$: the actual finish time at region $\Pi_k$, for $k \in \mathbf{\Pi}^+$;

- $z^{p}_{r,r',i,j} \in \{0,1\}, \quad \forall\, r \in R,\ \forall\, r' \in R \setminus \{r\},\ \forall\, i = 1,\dots,n_r - 1,\ \forall\, j = 1,\dots,n_{r'} - 1,\ \forall\, \Pi_p \in V(\tilde{P}_{r_i,r_{i+1}}) \cap V(\tilde{P}_{r'_j,r'_{j+1}})$: a binary precedence variable that equals 1 if and only if robot $r$ traverses region $\Pi_p$ along the path segment $P_{r_i,r_{i+1}}$ before robot $r'$ traverses $\Pi_p$ along $P_{r'_j,r'_{j+1}}$, and equals 0 otherwise.

For notational convenience, for each robot $r \in R$ and each index $i = 1,\dots,n_r - 1$, let $(\Pi_{r_i}, \Pi_{r_{i+1}})$ denote the consecutive pair in the sequence $\widehat{O}^r$. For every $\Pi_p \in V(\tilde{P}_{r_i,r_{i+1}})$, define the following derived expressions:

$$A_{r_i,r_{i+1}}(p) = \begin{cases} S_{r_{i+1}}, & \text{if } \Pi_p = \Pi_{r_{i+1}}, \\ F_{r_i} + T(r_i,p), & \text{otherwise,} \end{cases} \tag{19}$$

$$L_{r_i,r_{i+1}}(p) = \begin{cases} F_{r_{i+1}}, & \text{if } \Pi_p = \Pi_{r_{i+1}}, \\ A_{r_i,r_{i+1}}(p), & \text{otherwise,} \end{cases} \tag{20}$$

where $T(r_i,p)$ denotes the prefix travel time from $\Pi_{r_i}$ to $\Pi_p$ along the fixed shortest-path $P_{r_i,r_{i+1}}$. By construction, $A_{r_i,r_{i+1}}(p)$ and $L_{r_i,r_{i+1}}(p)$ are *shorthand* expressions in all $S_k$ and $F_k$ ($k \in \mathbf{\Pi}^+$) and in fixed path costs; they are not additional decision variables.

The refinement stage solves:

$$\min \quad \alpha\, C_{\max} + (1-\alpha)\, T_{\text{sum}} \tag{21}$$

$$\text{s.t.} \quad ES(k) \le S_k \le LS(k), \quad \forall k \in \mathbf{\Pi}^+, \tag{22}$$

$$F_k \ge S_k + DU(k), \quad \forall k \in \mathbf{\Pi}_Y, \tag{23}$$

$$S_{r_{i+1}} = F_{r_i} + T(r_i, r_{i+1}), \quad \forall r \in R, \forall i = 1,\dots,n_r - 1, \tag{24}$$

$$A_{r'_j,r'_{j+1}}(p) \ge L_{r_i,r_{i+1}}(p) + \delta - M_{\text{col}}(1 - z^{p}_{r,r',i,j}), \quad \forall r, r' \in R,\ r \ne r',$$
$$\forall i = 1,\dots,n_r - 1,\ \forall j = 1,\dots,n_{r'} - 1,\ \forall \Pi_p \in V(\tilde{P}_{r_i,r_{i+1}}) \cap V(\tilde{P}_{r'_j,r'_{j+1}}), \tag{25}$$

$$A_{r_i,r_{i+1}}(p) \ge L_{r'_j,r'_{j+1}}(p) + \delta - M_{\text{col}}\, z^{p}_{r,r',i,j}, \quad \forall r, r' \in R,\ r \ne r',$$
$$\forall i = 1,\dots,n_r - 1,\ \forall j = 1,\dots,n_{r'} - 1,\ \forall \Pi_p \in V(\tilde{P}_{r_i,r_{i+1}}) \cap V(\tilde{P}_{r'_j,r'_{j+1}}), \tag{26}$$

$$z^{p}_{r,r',i,j} + z^{p}_{r',r,j,i} = 1, \quad \forall r, r' \in R,\ r \ne r',\ \forall i = 1,\dots,n_r - 1,$$
$$\forall j = 1,\dots,n_{r'} - 1,\ \forall \Pi_p \in V(\tilde{P}_{r_i,r_{i+1}}) \cap V(\tilde{P}_{r'_j,r'_{j+1}}), \tag{27}$$

$$
\begin{cases}
C_r \geq F_k - M\left(1 - \left[\sum\limits_{i \in \mathbf{\Pi}'} \hat{o}^r_{i,k} - \sum\limits_{j \in \mathbf{\Pi}^+} \hat{o}^r_{k,j}\right]\right), & \forall r \in R,\ \forall k \in \mathbf{\Pi}_Y, \\
C_r \geq S_k - M\left(1 - \sum\limits_{i \in \mathbf{\Pi}'} \hat{o}^r_{i,k}\right), & \forall r \in R,\ \forall k \in \mathbf{\Pi}_U, \\
C_{\max} \geq C_r, & \forall r \in R, \\
T_{\text{sum}} = \sum\limits_{r \in R} C_r.
\end{cases} \tag{28}
$$

Eq. (21) is identical to Eq. (7); Eqs. (22)–(23) restate time-window, start-time, and service-duration constraints from the first stage. Eq. (24) enforces a tight arrival–entry link: since waiting is allowed only inside regions and never on edges, early arrival to the next window is realized by delaying the departure time $F_{r_i}$ at the current region. Eqs. (25)–(26) enforce collision-free traversal with minimum safety gap $\delta$ in any shared region, activated by the binary precedence variable $z^p_{r,r',i,j}$. Eq. (27) guarantees anti-symmetry so that exactly one order holds for each robot pair at each shared region. Here $M_{\text{col}}$ is a sufficiently large constant used only in collision-avoidance constraints.

# 5 Beam Search-Based Planning Algorithm

Beam Search (Steinbiss, Tran, & Ney, 1994) is a technique for searching decision trees that has demonstrated strong performance across a range of combinatorial optimization problems. However, its application in multi-robot task planning remains limited, particularly in problems involving time windows. The core idea of BS is to maintain a fixed number of candidates—referred to as the beam—and iteratively explore the space of solutions in a selective and efficient manner. At each iteration, all beam elements are expanded into feasible successors, which are then evaluated using a heuristic cost function. The final solution is selected as the feasible candidate with the lowest heuristic cost among all those generated during the search. An efficient improvement of BS, called filtered BS, is computationally simple and produces high-quality solutions (Ow & Morton, 1988).

In this study, we propose a filtered BS-based approach for solving multi-robot path planning problems with time window and Boolean specifications. The method incorporates a filtered width mechanism to limit the number of local beam elements explored at each step, thereby controlling computational complexity. We define two width parameters: the global width $K_{\text{global}}$, which limits the number of beam elements retained per iteration, and the local width $K_{\text{local}}$, which bounds the number of successors generated per state. First, we introduce the state representation and a similarity-based diversity preservation strategy. Then, we design a task-specific heuristic function based on the structural features of the problem. Finally, we present a time window handling mechanism during the beam expansion process to improve the feasibility and quality of the solutions.

## 5.1 Similarity-Based Filtering Strategy

To promote diversity during the BS and prevent premature convergence to suboptimal solutions, we integrate a similarity-based filtering mechanism. At each expansion step, multiple candidate states are generated by assigning new regions to robots. Instead of retaining all candidates, only those that are sufficiently distinct from the current beam states are preserved. By maintaining a diverse set of partial solutions, the search explores a broader solution space and reduces the risk of getting trapped in local optima.

This issue is particularly relevant in *homogeneous multi-robot systems*, where different robot-to-region assignments may result in functionally equivalent plans. Without similarity control, redundant expansions can quickly saturate the limited beam width with structurally similar solutions. Filtering out such candidates ensures more effective beam utilization and improved search quality.

Each state represents the current progress of region assignments for all robots. Formally, a state is defined as a triplet $s = (\mathcal{A}, \mathcal{T}, \mathcal{U})$, where:

- $\mathcal{A} = \{\mathcal{A}_r : r \in R\}$, where $\mathcal{A}_r$ is the ordered sequence of regions assigned to robot $r$, including the start region, with $\mathcal{A}_r[1] = \Pi^r_S$.
- $\mathcal{T} = \{\mathcal{T}_r : r \in R\}$, where $\mathcal{T}_r$ is the sequence of start times at the regions in $\mathcal{A}_r$, aligned elementwise, with $|\mathcal{T}_r| = |\mathcal{A}_r|$ and $\mathcal{T}_r[1] = 0$.
- $\mathcal{U} \subseteq \Pi^+$ is the set of unassigned regions.

To quantify similarity between states, we focus on the *region assignments* as *unordered sets*, emphasizing overlap in region coverage rather than visitation order.

**Definition 1** (Similarity Between States)**.** *Given two states $s$ and $s'$, represented by sets of assigned regions per robot, similarity is computed by constructing a complete bipartite graph between the sets. Each edge is weighted by the Jaccard index:*

$$\mathit{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

*A maximum-weight matching is solved, and the total similarity is computed as the sum of matched Jaccard indices, normalized by the number of robots:*

$$\mathrm{sim}(s, s') \in [0, 1].$$

A similarity score of 1 indicates identical region assignments (ignoring order), while 0 implies no overlap. During beam expansion, each candidate state $s'$ is compared to all existing beam states $s$. If $\mathrm{sim}(s, s') > \theta$, the candidate is discarded as redundant; otherwise, it is retained. In our implementation, $\theta$ is set empirically based on preliminary experiments to balance diversity and solution quality.

## 5.2 Cost Function

The cost function guides the BS process by prioritizing promising partial solutions, ensuring both feasible and efficient region assignments. We adopt a composite cost

function that evaluates the current travel cost and estimates the effort required to complete future assignments.

Formally, each state is assigned a cost score:

$$f(s) = g(s) + h(s),$$

where $g(s)$ is the actual incurred cost, and $h(s)$ is a heuristic estimation of the remaining cost.

The actual cost $g(s)$ reflects the current assignment progress, taking into account both the overall workload and the imbalance among robots:

$$g(s) = \alpha \cdot \max_{r \in R}\big(\max \mathcal{T}_r\big) + (1-\alpha) \cdot \sum_{r \in R}\big(\max \mathcal{T}_r\big),$$

where $\max \mathcal{T}_r$ denotes the last arrival time of robot $r$. The parameter $\alpha \in [0,1]$ balances the two objectives: minimizing the makespan to avoid workload imbalance, and minimizing the total completion to reduce cumulative effort.

The heuristic cost $h(s)$ estimates the remaining travel effort required to complete the tasks from the current state. To guarantee optimality, the heuristic function $h(s)$ should be *admissible*. To converge, the heuristic function $h(s)$ should under-estimate the actual cost (Hsaini et al., 2023).

We define three region sets for heuristic computation: $C_K$ denotes the current locations of the robots, $C_Y$ denotes unvisited intermediate regions, and $C_U$ denotes unvisited terminal regions specified by the Boolean specifications.

Given two region sets $A$ and $B$, the minimal travel time between them is defined as:

$$c^*(A,B) = \min_{\substack{k_1 \in A,\, k_2 \in B \\ k_1 \neq k_2}} T(k_1, k_2).$$

This metric is used to estimate the least travel time between different sets of regions in the heuristic.

Two heuristic estimates are then computed as:

$$h_1 = c^*(C_K, C_Y) + \sum_{k \in C_Y} c^*\left(\{k\}, C_Y \setminus \mathbf{\Pi}_{y_i}\right),$$

$$h_2 = \sum_{k \in C_Y} c^*\left(\{k\}, C_K \cup (C_Y \setminus \mathbf{\Pi}_{y_i})\right) + \sum_{u \in C_U} c^*\left(\{u\}, C_K \cup C_Y\right),$$

where $c^*(C_K, C_Y)$ represents the minimal cost from the current robot positions to the set of unvisited intermediate regions. The term $c^*(\{k\}, C_Y \setminus \mathbf{\Pi}_{y_i})$ estimates the cost from a given region $k$ to other intermediate regions not belonging to the same clause. In $h_2$, the first summation estimates the cost of visiting regions in $C_Y$ while avoiding redundant visits within the same clause, and the second summation evaluates the cost of reaching unvisited terminal regions $C_U$ from any previously visited or intermediate region.

The final heuristic cost is computed as:

$$h(s) = w_{\max} \cdot \max(h_1, h_2) + \frac{w_{\text{sum}}}{|R|} \cdot \max(h_1, h_2). \tag{29}$$

Here, the heuristic function $h(s)$ combines the maximum of $h_1$ and $h_2$, with weights reflecting their respective contributions to the makespan and the total travel effort. This formulation enables a balanced consideration of both individual task completion time and overall system efficiency.

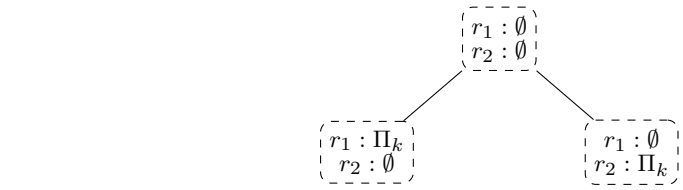
## 5.3 Time Window Handling

In the BS framework, the size of the state space during the expansion phase plays a crucial role in determining the scalability and computational efficiency of the algorithm. As such, effectively incorporating time window constraints into the expansion process is a challenge. Even in a simple scenario involving a single task, the introduction of temporal constraints can lead to a substantial increase in the number of potential execution paths.

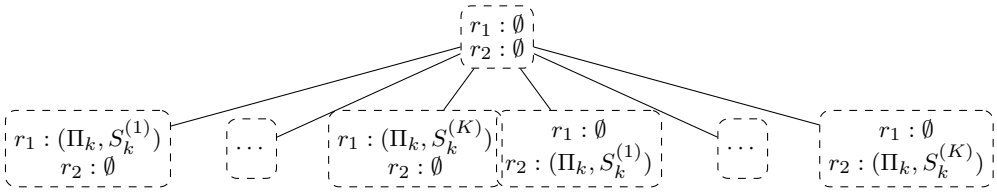
As illustrated in Figure 3, consider a path planning requirement where a region $\Pi_k$ must be visited by one of two available robots, $r_1$ or $r_2$. In the absence of time constraints, the expansion generates only two possible outcomes: either $r_1$ visits the region while $r_2$ remains idle, or the reverse. However, to more accurately model real-world scheduling environments, the continuous time window associated with region $\Pi_k$ is discretized into a finite set of sampled time points. As a result, each robot can attempt to visit $\Pi_k$ at any of the $K$ discrete time instances $\{S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(K)}\}$, yielding up to $2K$ potential successor states. This simple case clearly demonstrates the combinatorial explosion introduced by incorporating time modeling into the planning process. To balance computational efficiency and scheduling accuracy, we discretize the feasible start-time interval of each robot–region pair $(r, \Pi_k)$ into $K$ candidate points. Here, $K$ specifies the number of samples drawn from the time window during the expansion phase: When $K = 1$, the interval leads to a single value, namely the earliest feasible start time $t_r = \max\{ES(k),\, t_{\text{arr}}(r,k)\}$; when $K \geq 2$, the interval $[\max\{ES(k),\, t_{\text{arr}}(r,k)\},\, LS(k)]$ is uniformly discretized (including both endpoints) into $K$ candidates, where $t_{\text{arr}}(r,k)$ denotes the earliest time robot $r$ can start service at region $\Pi_k$ under the current partial schedule.

By restricting each $(r, \Pi_k)$ to a single feasible execution time, the number of successor states is significantly reduced, thereby improving the tractability and scalability of the BS framework. This heuristic is consistent with makespan-oriented objectives, since earlier feasible start times typically yield better scheduling outcomes. When modeling collision avoidance or introducing slack, it is common to set $K \geq 2$. The specific values of $K$ used in our experiments are reported in the experimental section.

This pseudocode demonstrates how the heuristic is used to determine the execution time for each task region and how time window constraints are enforced to ensure feasibility. Specifically, intermediate regions are grouped into subtask clusters $\mathbf{\Pi}_{y_i}$, each of which is satisfied by visiting *at least one* of its candidate regions. Once any region $\Pi_k \in \mathbf{\Pi}_{y_i}$ is assigned, the clause is marked satisfied and all regions within the




(a) Without time modeling: 2 branches




(b) With time modeling: $2K$ branches

**Fig. 3**: Comparison of state expansion without and with time modeling

same cluster are removed from the unvisited region set $\mathcal{U}$. Only when no intermediate regions remain do we proceed to assign terminal goal regions from $\mathbf{\Pi}_U$. The detailed expansion procedure with time-window sampling is presented in Algorithm 1.

## 5.4 Collision Avoidance Mechanism

In multi-robot path planning, collisions may occur when multiple robots occupy overlapping spatial regions. To ensure feasibility and collision-free execution of the planned trajectories, we introduce an explicit collision avoidance mechanism that is simple yet effective.

This mechanism is integrated into the BS framework through active collision checking. Notably, its effectiveness depends on the temporal discretization level. This temporal diversity allows robots to delay their arrival and avoid conflicts by waiting in their previous Boolean task region.

During the search, each successor produced by Expand is immediately filtered by a collision predicate IsCollisionFree against the beam from the previous expansion step (denoted by $\mathcal{B}$). Candidates that fail the test are discarded on the spot and never enter the candidate pool. This places collision checking before similarity filtering and ranking, reducing wasted evaluations downstream.

This mechanism not only leverages the temporal flexibility introduced by discretized time windows but also enforces proactive filtering of infeasible solutions. It enables temporal coordination among robots even in environments with unavoidable spatial overlap, thereby improving the planner's ability to discover feasible solutions efficiently. Algorithm 2 takes as input the current candidate state $x = (\mathcal{A}, \mathcal{T}, \mathcal{U})$; the

**Algorithm 1** EXPAND($s$): state expansion with time-window sampling

**Require:** Current state $s = (\mathcal{A}, \mathcal{T}, \mathcal{U})$; sampling parameter $K$; earliest start function $ES(k)$; latest start function $LS(k)$; service duration $DU(k)$; travel time $T(k, k')$; intermediate region sets $\{\mathbf{\Pi}_{y_j}\}_{j=1}^{m}$ and their union $\mathbf{\Pi}_Y$; final region sets $\{\mathbf{\Pi}_{u_j}\}_{j=1}^{q}$ and their union $\mathbf{\Pi}_U$

**Ensure:** Set of expanded states $\mathcal{S}'$

Initialize $\mathcal{S}' \leftarrow \varnothing$
**if** $\mathcal{U} \cap \mathbf{\Pi}_Y \neq \emptyset$ **then**
  $\mathbf{\Pi}_{\text{exp}} \leftarrow \mathcal{U} \cap \mathbf{\Pi}_Y$ // expand intermediate regions
**else**
  $\mathbf{\Pi}_{\text{exp}} \leftarrow \mathcal{U} \cap \mathbf{\Pi}_U$ // expand terminal regions
**end if**
**for** each region $\Pi_k \in \mathbf{\Pi}_{\text{exp}}$ **do**
  **for** each robot $r \in R$ **do**
    $i \leftarrow |\mathcal{A}_r|$
    $t_{\text{arr}}(r, k) \leftarrow \mathcal{T}_r[i] + DU(\mathcal{A}_r[i]) + T(\mathcal{A}_r[i], k)$
    Sample $K$ candidate start times $\{S_k^{(1)}, \ldots, S_k^{(K)}\}$ as $K$ equally spaced points from $[\max\{ES(k), t_{\text{arr}}(r, k)\}, LS(k)]$
    **for** $j = 1$ to $K$ **do**
      Update task assignment: $\mathcal{A}' \leftarrow \mathcal{A}$; $\mathcal{A}'_r \leftarrow \mathcal{A}_r \cup \{k\}$
      Update timing (arrival time): $\mathcal{T}' \leftarrow \mathcal{T}$; $\mathcal{T}'_r \leftarrow \mathcal{T}_r \cup \{S_k^{(j)}\}$
      **if** $\Pi_k \in \mathbf{\Pi}_{y_i}$ for some $i$ **then**
        $\mathcal{U}' \leftarrow \mathcal{U} \setminus \mathbf{\Pi}_{y_i}$
      **else if** $\Pi_k \in \mathbf{\Pi}_{u_i}$ for some $i$ **then**
        $\mathcal{U}' \leftarrow \mathcal{U} \setminus \mathbf{\Pi}_{u_i}$
      **end if**
      $s' \leftarrow (\mathcal{A}', \mathcal{T}', \mathcal{U}')$
      $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(s', r)\}$
    **end for**
  **end for**
**end for**
**return** $\mathcal{S}'$

robot $r$ that most recently moved in $x$; the minimum temporal safety gap $\delta$; the prespecified shortest paths $P_{k,k'}$ with travel times $T(k, k')$ for all $k, k' \in \Omega$; and the service times $DU(k)$ for all $k \in \mathbf{\Pi}_Y$. The procedure first identifies robot $r$'s latest motion segment, denoted $(a, b)$ with departure region $a$ and arrival region $b$, and along the fixed path $P_{a,b}$ computes the arrival and leave instants for all regions on that segment using the timestamps stored in $x$. It then iterates over every other robot $r' \neq r$ and each consecutive pair $(r'_i, r'_{i+1})$ on its route, evaluates the corresponding instants along $P_{r'_i, r'_{i+1}}$, and checks every shared region $w \in V(P_{a,b}) \cap V(P_{r'_i, r'_{i+1}})$. If, at any shared region, the two robots' occupancy intervals overlap within $\delta$, a vertex conflict is declared and the algorithm returns **false**; otherwise, if all intervals are separated by at least $\delta$, it returns **true**. Assuming that waiting is permitted only inside regions (i.e., no queuing on edges), this filtering step discards infeasible candidates prior to expansion and promotes temporal coordination in environments with unavoidable spatial overlap, thereby improving the efficiency of the search for feasible solutions.

## 5.5 Algorithm

The BS-based algorithm incrementally constructs region assignment plans under temporal and Boolean specifications. At each iteration, successor states are generated from the current beam members using a cost-guided expansion strategy (see Section 5.3 and Algorithm 1). Candidate states are ranked based on a composite cost function that reflects solution quality.

**Algorithm 2** IsCollisionFree$(x, r, \delta)$

**Require:** Candidate state $x = (\mathcal{A}, \mathcal{T}, \mathcal{U})$; robot $r$ (last moved in $x$); safety gap $\delta$; shortest paths $P_{k,k'}$; vertex sets $V(P_{k,k'})$ corresponding to those paths; travel times $T(k, k')$; service time $DU(k)$
**Ensure:** **true** if $x$ has no vertex conflicts; otherwise **false**

$\mathcal{A}_r = (\Pi_{r_1}, \ldots, \Pi_{r_{n_r}})$; $n_r = |\mathcal{A}_r|$; $a \leftarrow n_r - 1$; $b \leftarrow n_r$
**for** each $p \in P_{a,b}$ **do**
  **if** $p = a$ **then**
    $A_{a,b}(p) \leftarrow \mathcal{T}_r[a]$
    $L_{a,b}(p) \leftarrow \mathcal{T}_r[b] - T(a, b)$
  **else**
    **if** $p = b$ **then**
      $A_{a,b}(p) \leftarrow \mathcal{T}_r[b]$
      $L_{a,b}(p) \leftarrow A_{a,b}(p) + DU(p)$
    **else**
      $A_{a,b}(p) \leftarrow L_{a,b}(a) + T(a, p)$
      $L_{a,b}(p) \leftarrow A_{a,b}(p)$
    **end if**
  **end if**
**end for**
**for** each $r' \in R$ with $r' \neq r$ **do**
  $\mathcal{A}_{r'} = (\Pi_{r'_1}, \ldots, \Pi_{r'_{n_{r'}}})$; $n_{r'} = |\mathcal{A}_{r'}|$
  **for** $i = 1$ to $n_{r'} - 1$ **do**
    $P_{r'_i, r'_{i+1}}$ is the shortest path from $\mathcal{A}_{r'}[i]$ to $\mathcal{A}_{r'}[i+1]$
    **for** each $w \in V(P_{a,b}) \cap V(P_{r'_i, r'_{i+1}})$ **do**
      **if** $w = r'_{i+1}$ **then**
        **if** $r'_{i+1} = r'_{n_{r'}}$ and $w = r'_{n_{r'}}$ **then**
          $A_{r'_i, r'_{i+1}}(w) \leftarrow \mathcal{T}_{r'}[i+1]$
          $L_{r'_i, r'_{i+1}}(w) \leftarrow A_{r'_i, r'_{i+1}}(w) + DU(w)$
        **else**
          $A_{r'_i, r'_{i+1}}(w) \leftarrow \mathcal{T}_{r'}[i]$
          $L_{r'_i, r'_{i+1}}(w) \leftarrow \mathcal{T}_{r'}[i+1] - T(r'_i, r'_{i+1})$
        **end if**
      **else**
        $A_{r'_i, r'_{i+1}}(w) \leftarrow \big(\mathcal{T}_{r'}[i+1] - T(r'_i, r'_{i+1})\big) + T(r'_i, w)$
        $L_{r'_i, r'_{i+1}}(w) \leftarrow A_{r'_i, r'_{i+1}}(w)$
      **end if**
      $\Delta_1 \leftarrow A_{r'_i, r'_{i+1}}(w) - L_{a,b}(w)$, $\Delta_2 \leftarrow A_{a,b}(w) - L_{r'_i, r'_{i+1}}(w)$
      **if** $(\Delta_1 \leq \delta)$ **and** $(\Delta_2 \leq \delta)$ **then**
        **return** false
      **end if**
    **end for**
  **end for**
**end for**
**return** true

To maintain structural diversity, we use the similarity threshold $\theta$ that filters out candidates too similar to those already selected in the local beam (see Section 5.1). A candidate is admitted only if its similarity to all existing beam members is below $\theta$. We also discard any solutions that exhibit collisions (see Section 2). If fewer than $K_{\text{local}}$ candidates pass this filter, the beam is greedily filled with the next-best states according to $f(s)$, avoiding duplicates. In addition, we apply a global beam width $K_{\text{global}}$ to retain only the top $K_{\text{global}}$ candidates after each expansion, thus bounding the search space and promoting broader exploration.

In summary, the proposed algorithm offers a scalable, constraint-aware planning framework for region assignment in MRS. It effectively integrates Boolean task specifications with temporal feasibility and balances solution quality with beam diversity through a combination of cost-driven search and structural filtering.

**Remark 1** (Deadlock Avoidance)**.** *As a forward-search algorithm, BS does not use a global backtracking mechanism because it maintains solution diversity thanks to the beam widths $K_{global}$ and $K_{local}$. In addition, the deadlock is structurally mitigated by the operational constraints of our system: robots are permitted to wait only within their assigned regions (which act as safe zones), and they must continuously traverse the paths between these regions without stopping. This constraint ensures that the intermediate regions and edges remain transient and clear, effectively preventing the formation of cyclic waiting chains. Consequently, the search is efficiently guided by the heuristic function and the diversity maintained by the beam widths $K_{local}$ and $K_{global}$, allowing the algorithm to navigate toward feasible, deadlock-free solutions without relying on intensive backtracking.*

# 6 Experimental results

This section aims to evaluate the effectiveness and scalability of the proposed approach under different problem sizes and parameter configurations. We design two complementary solution methods: an exact MILP formulation solved by the Gurobi optimizer for small-scale problems, and a BS-based approach tailored for large-scale scenarios, which offers fast computation and high-quality approximate solutions.

All experiments were implemented using Python 3.9 with Gurobi 10.0 solving the MILP formulation, on a PC with a 12-core processor (Intel Core i7, 2.6 GHz) and 32 GB of RAM. We specifically analyze the impact of varying parameters, including the number of robots, the number of conjunctions $m$ in sub-specification $Y$, the width of time windows $LS - ES$, the random generation range of $ES$, and the duration $DU$ of the visit to intermediate regions.

Gurobi is executed with its default parameters. For the BS-based approach, the baseline configuration is set to $K_{\text{local}} = 10$, $K_{\text{global}} = 50$, and $\theta = 0.95$. These parameters were selected based on a series of preliminary tests to balance solution quality and computational efficiency. Specifically, we observed that $K_{\text{global}}$ is the primary driver of performance, where $K_{\text{global}} \in [50, 100]$ is recommended for most scenarios to achieve a favorable trade-off between optimality and runtime. In contrast, the performance is relatively insensitive to $K_{\text{local}}$ and the similarity threshold $\theta$. We recommend $K_{\text{local}} \in [5, 15]$ and $\theta \in [0.9, 0.99]$ as stable operational ranges, with $\theta = 0.95$ serving as a robust baseline value to preserve search diversity. The weighting parameters in the cost function are balanced at $\alpha = 0.5$ to equally prioritize makespan and total travel effort. We systematically analyze the influence of four key parameters on the performance of both methods:

- Number of robots;
- Number of conjunctions $m$ in sub-specification $Y$;
- Time window width $LS - ES$;

**Algorithm 3** Beam Search-Based Planning Algorithm

**Require:** Initial state $s_0 = (\mathcal{A}_0, \mathcal{T}_0, \mathcal{U}_0)$; beam widths $K_{\text{local}}$, $K_{\text{global}}$; similarity threshold $\theta$
**Ensure:** Feasible region assignment $s^*$ satisfying all constraints

Initialize beam $\mathcal{B} \leftarrow \{s_0\}$, solution pool $\mathcal{S}^* \leftarrow \emptyset$
Initialize global candidate archive $\mathcal{G} \leftarrow \emptyset$
**while** $\mathcal{B} \neq \emptyset$ **do**
  $\mathcal{C} \leftarrow \emptyset$ // Candidate states
  **for all** $s \in \mathcal{B}$ **do**
    $\widetilde{\mathcal{C}} \leftarrow \textsc{Expand}(s)$ // Algorithm 1
    **for all** $(x, r) \in \widetilde{\mathcal{C}}$ **do**
      **if** $\textsc{IsCollisionFree}(x, r, \delta)$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{x\}$
      **end if**
    **end for**
  **end for**
  Sort $\mathcal{C}$ by increasing $f(s)$
  $\mathcal{B}_{\text{new}} \leftarrow \emptyset$
  **for all** $s \in \mathcal{C}$ **do**
    **if** $\forall s' \in \mathcal{B}_{\text{new}},\ \text{sim}(s, s') < \theta$ **then**
      $\mathcal{B}_{\text{new}} \leftarrow \mathcal{B}_{\text{new}} \cup \{s\}$
      **if** $|\mathcal{B}_{\text{new}}| \geq K_{\text{local}}$ **then**
        **break**
      **end if**
    **end if**
  **end for**
  **if** $|\mathcal{B}_{\text{new}}| < K_{\text{local}}$ **then**
    Fill $\mathcal{B}_{\text{new}}$ with next-best states from $\mathcal{C}$ according to $f(s)$, skipping duplicates, until $|\mathcal{B}_{\text{new}}| = K_{\text{local}}$
  **end if**
  $\mathcal{G} \leftarrow \text{Top}K_{\text{global}}(\mathcal{G} \cup \mathcal{C})$ // Update global archive (optional)
  $\mathcal{B} \leftarrow \mathcal{B}_{\text{new}}$
  Add complete, feasible states from $\mathcal{B}$ to solution pool $\mathcal{S}^*$
**end while**
**return** $s^* \leftarrow \arg\min_{s \in \mathcal{S}^*} f(s)$

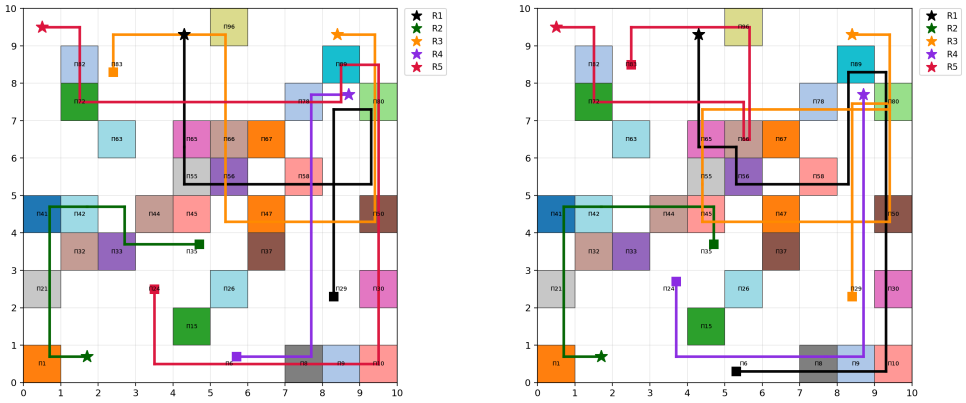
- Range of earliest start times $ES$.

When collision avoidance is disabled, we set $K = 1$ to control branching. When collision avoidance is enabled, we use $K > 1$ to provide minimal temporal slack; in our experiments we set $K = 2$. Note that $K$ only affects the BS-based approach; the MILP benchmark is independent of $K$.

In each subsection, we compare the performance of the MILP benchmark (solved by Gurobi) and the BS-based approach under different configurations, focusing on metrics such as computational time, objective value (makespan), and the optimality gap. These experiments demonstrate the strengths and complementarity of the two methods in handling multi-robot path planning problems of varying scales, validating their effectiveness as a scalable solution framework. For clarity, in the tables presented, "–" indicates that the computation time exceeded 5000 seconds and no feasible solution was returned within the time limit.

## 6.1 Case study: impact of collision avoidance

This subsection assesses, on a small instance, the effect of enabling collision-avoidance checks inside the BS-based approach on solution quality and runtime. The instance involves five robots; the numbers of conjunctions are $m = 15$ for sub-specification $Y$ and $q = 5$ for $U$ (full formulas in the appendix). To provide minimal timing slack for resolving conflicts, we set the time discretization to $K = 2$ in this subsection (elsewhere




(a) Without collision avoidance (b) With collision avoidance

**Fig. 4**: Planned robot trajectories with and without collision avoidance.

$K = 1$). For each region, we retain the earliest feasible start $S_{\min} = \max\{t_{\text{arrival}}, ES\}$ and the latest start $S_{\max} = LS$. All other hyperparameters are kept identical; the similarity filter is disabled here so that candidates differing only in timing are not discarded.

We compare two configurations: BS-based approach (without collision avoidance mechanism): no conflict checking during expansion; BS-based approach (with collision avoidance mechanism): candidates that would cause conflicts at shared regions are pruned during expansion.

**Table 2**: BS-based approach with and without collision avoidance on the case-study instance.

| Method | CPU time (s) | $C_{\max}$ | Obj. |
|---|---|---|---|
| Without collision avoidance mechanism | 29.66 | 68 | 162.0 |
| With collision avoidance mechanism | 25.82 | 68 | 162.5 |

As shown in Table 2, with $K = 2$ the makespan remains unchanged at $C_{\max} = 68$ after enabling collision checks. The composite heuristic objective increases marginally (from 162.0 to 162.5, $\approx 0.3\%$), indicating negligible degradation in solution quality. Runtime stays of the same order and is lower in this instance (from 29.66 s to 25.82 s, $\approx 13\%$), consistent with early pruning reducing branching; this speed-up is instance-dependent rather than guaranteed. Figure 4 visualizes the planned trajectories: the collision-aware plan closely matches the baseline, with minor timing shifts to satisfy safety constraints. For the quantitative comparisons in the subsequent subsections, collision avoidance is disabled and $K = 1$ is used for like-for-like comparability with the MILP benchmark.

## 6.2 Impact of the number of robots

This experiment examines the impact of increasing the number of robots on both objective performance and computational time using the MILP model solved by **Gurobi**. The number of conjunctions ($m$) and the number of conjunctions in the sub-specification $U$ ($q$) are fixed at 6 and 1, respectively. The time window width $LS-ES$ is fixed at 10, with each conjunction $y_i$ containing three disjunctions. To reduce the impact of randomness in task release times, we perform 20 runs for each robot configuration with randomly generated ES values. The reported CPU times and objective values in Table 3 are the average results across these 20 runs.

We evaluate both the objective performance and computational cost of the MILP model solved by Gurobi. As shown in Table 3, increasing the number of robots leads to a significant improvement in the objective value. The average objective value decreases from 28.85 with one robot to 24.93 with six robots, after which it remains constant despite further increases in the number of robots. This indicates that increasing the number of robots up to six significantly enhances the flexibility of feasible path combinations, allowing the system to identify lower makespan solutions. However, adding more robots beyond this threshold does not yield additional performance gains.

In contrast, the computational time required by Gurobi grows rapidly with the number of robots, increasing from 0.51 seconds for one robot to 3111.21 seconds for 50 robots. This sharp rise is primarily due to the expansion of the problem size: as the number of robots increases, the number of decision variables and constraints grows substantially, resulting in a combinatorial explosion of the search space. Consequently, the branch-and-bound tree becomes significantly larger, greatly increasing the solving time.

To address this scalability issue, we introduce the BS-based approach. As illustrated in Table 3, the BS-based approach is capable of producing near-optimal solutions in a fraction of the time required by Gurobi. Across all robot configurations, the computational time of the BS-based approach remains consistently low, ranging from 0.27 to 5.22 seconds, demonstrating excellent efficiency and stable scalability.

The efficiency of the BS-based approach stems from its use of beam width and diversity threshold control, which effectively limit the size of the search space and mitigate the combinatorial explosion caused by increasing the number of robots. Although the BS-based approach does not guarantee global optimality, experimental results show that for small-scale problems (1–10 robots), the solutions obtained by the BS-based approach deviate by less than 3% from the optimal ones computed by the MILP benchmark. For larger instances (e.g., 30–50 robots), the deviation reaches approximately 8%. Given that the exact MILP approach becomes computationally infeasible in such large-scale settings, the BS-based approach provides a practical and efficient alternative for solving large-scale multi-robot path planning problems.

## 6.3 Impact of the number of conjunctions in sub-specification $Y$

This subsection analyzes the impact of varying the number of conjunctions ($m$) in sub-specification $Y$ on solver performance. In the experiments, each conjunction contains

**Table 3**: Comparison (20 runs) of MILP (solved by Gurobi) and the BS-based approach on the effect of varying the number of robots with fixed $m = 6$ and $q = 1$.

| Number of robots | CPU time (s) | | Min. Obj. | | Max. Obj. | | Avg. Obj. | |
|---|---|---|---|---|---|---|---|---|
| | MILP | BS-based | MILP | BS-based | MILP | BS-based | MILP | BS-based |
| 1 | 0.51 | 0.27 | 23.0 | 23.0 | 34.0 | 34.0 | 28.85 | 29.65 |
| 2 | 1.18 | 0.68 | 19.0 | 19.5 | 33.5 | 34.0 | 26.43 | 27.12 |
| 3 | 3.11 | 0.92 | 18.5 | 19.0 | 33.0 | 33.0 | 25.83 | 26.20 |
| 4 | 6.79 | 1.26 | 18.5 | 19.0 | 33.0 | 34.0 | 25.75 | 26.22 |
| 5 | 14.25 | 1.54 | 18.0 | 20.0 | 33.0 | 34.0 | 25.15 | 25.83 |
| 6 | 21.89 | 1.84 | 17.5 | 18.5 | 33.0 | 34.0 | 24.93 | 25.42 |
| 7 | 31.21 | 2.09 | 17.5 | 18.0 | 33.0 | 34.0 | 24.93 | 25.52 |
| 8 | 34.08 | 2.53 | 17.5 | 17.5 | 33.0 | 34.0 | 24.93 | 25.52 |
| 9 | 49.39 | 2.86 | 17.5 | 17.5 | 33.0 | 34.0 | 24.93 | 25.52 |
| 10 | 56.69 | 1.80 | 17.5 | 17.5 | 33.0 | 35.5 | 24.93 | 25.63 |
| 15 | 106.59 | 1.90 | 17.5 | 17.5 | 33.0 | 37.0 | 24.93 | 26.53 |
| 20 | 136.22 | 1.98 | 17.5 | 17.5 | 33.0 | 38.0 | 24.93 | 26.68 |
| 25 | 188.10 | 1.72 | 17.5 | 17.5 | 33.0 | 38.0 | 24.93 | 26.68 |
| 30 | 234.60 | 2.24 | 17.5 | 17.5 | 33.0 | 38.0 | 24.93 | 27.03 |
| 35 | 293.40 | 2.83 | 17.5 | 17.5 | 33.0 | 38.0 | 24.93 | 27.08 |
| 40 | 1290.80 | 3.82 | 17.5 | 17.5 | 33.0 | 38.0 | 24.93 | 27.08 |
| 45 | 1091.00 | 4.40 | 17.5 | 17.5 | 33.0 | 38.0 | 24.93 | 27.08 |
| 50 | 3111.21 | 5.22 | 17.5 | 17.5 | 33.0 | 38.0 | 24.93 | 27.08 |

three regions, the number of robots is fixed at 3, and the number of conjunctions in sub-specification $U$ ($q$) is fixed at 1. The earliest start times ($ES$) are randomly generated within the interval $[0, 120]$, with the time window width $LS - ES$ fixed at 10. To mitigate the influence of randomness, each configuration is independently executed 20 times. The experimental results are summarized in Table 4.

From the results, it can be observed that as the number of conjunctions $m$ increases, the problem size expands rapidly, leading to a significant rise in Gurobi's computation time. When $m$ ranges from 5 to 20, Gurobi can still return solutions within a reasonable time. However, when $m > 20$, the computation time increases sharply, and at $m = 25$, Gurobi fails to return a feasible solution within the time limit. This is mainly due to the NP-hard nature of the MILP model: as $m$ increases, the number of decision variables and constraints grows exponentially, resulting in a dramatic increase in the number of nodes explored during the branch-and-bound process and a substantial drop in overall solving efficiency.

In contrast, the BS-based approach demonstrates much better scalability as $m$ increases. Although the computation time of the BS-based approach also grows with $m$, the growth rate is significantly lower than that of Gurobi. Even for large-scale problems with $m = 50$, the BS-based approach can still return solutions within an acceptable amount of time, which is largely due to its ability to limit the search space and control complexity. The good performance of the BS-based approach in large-scale problems can be attributed to its beam width limitation and diversity control mechanisms, which effectively constrain the search space size and mitigate the computational burden caused by combinatorial explosion.

Regarding solution quality, the objective value gap between the BS-based approach and the MILP benchmark slightly increases as $m$ grows. For instance, at $m = 20$, the average objective value difference between the two methods is 6.4%. For $m > 25$, since Gurobi fails to return a solution within the time limit, a direct comparison is not available. Nevertheless, based on the trend of the BS-based approach's objective values, it can be inferred that the BS-based approach continues to provide stable and practically useful near-optimal solutions even for large-scale problems.

In summary, when the number of conjunctions is small ($m \leq 20$), the MILP benchmark is able to efficiently find strictly optimal solutions, making it suitable for small to medium-sized problems. As the number of conjunctions further increases, resulting in a rapid rise in solving time or even infeasibility, the BS-based approach exhibits clear advantages. For large-scale problems ($m > 25$), the BS-based approach efficiently produces near-optimal solutions in a reasonable time.

**Table 4**: Comparison (20 runs) of MILP (solved by Gurobi) and the BS-based approach on the effect of varying the number of conjunctions $m$ for fixed $q = 1$, and 3 robots.

| $m$ | Number of regions | CPU Time (s) | | Min. Obj. | | Max. Obj. | | Avg. Obj. | |
|---|---|---|---|---|---|---|---|---|---|
| | | MILP | BS-based | MILP | BS-based | MILP | BS-based | MILP | BS-based |
| 5 | 15 | 0.06 | 0.08 | 27.0 | 34.0 | 100.0 | 100.0 | 70.05 | 70.75 |
| 10 | 30 | 0.52 | 1.46 | 44.0 | 47.0 | 107.0 | 107.0 | 83.33 | 84.43 |
| 15 | 45 | 14.74 | 12.89 | 55.5 | 60.0 | 113.0 | 118.0 | 85.63 | 89.90 |
| 20 | 60 | 180.56 | 35.70 | 68.5 | 72.0 | 115.0 | 120.5 | 88.68 | 94.43 |
| 25 | 75 | – | 100.16 | – | 79.5 | – | 128.5 | – | 105.53 |
| 30 | 90 | – | 199.70 | – | 87.0 | – | 136.0 | – | 114.40 |
| 35 | 105 | – | 538.75 | – | 93.0 | – | 144.5 | – | 125.30 |
| 40 | 120 | – | 1241.41 | – | 97.5 | – | 149.0 | – | 129.65 |
| 45 | 135 | – | 1768.49 | – | 112.5 | – | 152.5 | – | 133.35 |
| 50 | 150 | – | 2482.13 | – | 118.5 | – | 157.0 | – | 142.13 |

## 6.4 Impact of time window width ($LS - ES$)

This subsection analyzes the impact of varying the time window width ($LS - ES$) on solver performance. Two experimental settings were considered:

(i) The earliest start time ($ES$) for each region was randomly generated within the range $[0, 30]$, and the impact of varying the time window width ($LS - ES$) was tested (see Table 5). In this setting we use $m = 8$.

(ii) The latest start time ($LS$) for each region was randomly generated within the range $[20, 50]$, and the impact of varying the time window width was similarly tested (see Table 6). In this setting we use $m = 6$.

In both settings, the number of robots was 3 and $q = 1$. To ensure statistical robustness, each configuration was independently run 20 times.

As shown in Tables 5 and 6, regardless of whether $ES$ or $LS$ was fixed, changes in the time window width significantly affected the computational efficiency of Gurobi on the MILP formulation. Specifically, as the time window width increased, the solver's computational time generally increased. Although the number of variables and constraints in the initial model remained the same, after presolve, instances with wider time windows retained significantly more variables and constraints than those with narrow windows.

In narrow time window scenarios, the strict temporal constraints allow the solver to quickly eliminate many infeasible paths or decision variables during presolve, thus substantially simplifying the model and improving solving efficiency. In contrast, in wide time window scenarios, the increased temporal flexibility results in a significantly larger number of potential feasible paths and decision variables, making it difficult for presolve to effectively simplify the model. Consequently, the final model size increases, leading to higher computational complexity.

From the results under the fixed $ES$ setting (Table 5), it can be observed that as the time window width increases from 5 to 50, the average computation time of BS-based approach remains stable between 3 and 6 seconds, with the average objective value consistently around 30.5. In contrast, MILP was able to solve instances only when $LS-ES \leq 15$, and at $LS-ES = 15$, the computation time had already reached 2026.16 seconds, after which it failed to return a solution within a reasonable time, revealing significant scalability bottlenecks.

Similarly, under the fixed $LS$ setting (Table 6), as the time window width increased, the average objective value of the BS-based approach gradually decreased from about 110.40 to 30.53, while the average computation time remained stable between 4 and 6 seconds. Again, Gurobi could only solve instances when $LS - ES \leq 15$, and its computation time sharply increased as the window width expanded.

Also, the experimental results demonstrate that the BS-based approach exhibits excellent stability and scalability under different time window width conditions. Under the fixed $ES$ setting (Table 5), even as $LS - ES$ increases up to 50, the average computation time and objective value of the BS-based approach remain stable. Under the fixed $LS$ setting (Table 6), as $LS-ES$ increases, the BS-based approach achieves continuous improvements in the average objective value, while maintaining the average computation time between 4 and 6 seconds.

Overall, when the time window is narrow ($LS - ES \leq 10$), the MILP benchmark can solve the problem within acceptable time limits and provide strict optimal solutions, making it suitable for small- to medium-scale problems with tight temporal constraints. As the time window widens, however, Gurobi faces severe computational difficulties due to the rapid expansion of variables and constraints, gradually losing practical applicability.

## 6.5 Impact of random generation range of $ES$

This subsection analyzes the impact of varying the random generation range of the $ES$ on solver performance. In the experiments, the number of conjunctions ($m$) and the number of conjunctions in the sub-specification $U$ ($q$) are fixed at 6 and 1, respectively. Each conjunction $y_i$ contains three disjunctions. The time window width

**Table 5**: Comparison (20 runs) of MILP (solved by Gurobi) and the BS-based approach on the impact of time window width $LS - ES$ for randomly varying $ES$, $m = 8$, $q = 1$, and three robots.

| LS - ES | CPU Time (s) | | Min. Obj. | | Max. Obj. | | Avg. Obj. | |
|---|---|---|---|---|---|---|---|---|
| | MILP | BS-based | MILP | BS-based | MILP | BS-based | MILP | BS-based |
| 2 | 6.28 | 0.46 | 25.5 | 25.5 | 42.0 | 41.5 | 33.68 | 33.28 |
| 4 | 7.28 | 1.13 | 25.5 | 25.5 | 40.0 | 40.0 | 31.30 | 30.78 |
| 6 | 7.76 | 3.54 | 25.5 | 25.0 | 39.0 | 38.0 | 30.53 | 30.15 |
| 8 | 8.48 | 3.02 | 25.5 | 25.0 | 38.5 | 38.0 | 30.70 | 29.95 |
| 10 | 77.40 | 2.77 | 23.5 | 25.5 | 38.0 | 38.5 | 29.53 | 30.60 |
| 15 | 2026.16 | 5.62 | 22.5 | 25.5 | 38.0 | 38.0 | 29.30 | 30.60 |
| 20 | – | 5.13 | – | 25.5 | – | 38.0 | – | 30.50 |
| 25 | – | 5.26 | – | 25.5 | – | 38.0 | – | 30.50 |
| 30 | – | 3.29 | – | 25.5 | – | 38.0 | – | 30.50 |
| 35 | – | 3.87 | – | 25.5 | – | 38.0 | – | 30.50 |
| 40 | – | 6.98 | – | 25.5 | – | 38.0 | – | 30.50 |
| 45 | – | 6.42 | – | 25.5 | – | 38.0 | – | 30.50 |
| 50 | – | 5.18 | – | 25.5 | – | 38.0 | – | 30.50 |

**Table 6**: Comparison of MILP (solved by Gurobi) and the BS-based approach on the impact of time window width $LS - ES$ for randomly varying $LS$, $m = 6$, $q = 1$, and three robots.

| $LS - ES$ | CPU Time (s) | | Min. Obj. | | Max. Obj. | | Avg. Obj. | |
|---|---|---|---|---|---|---|---|---|
| | MILP | BS-based | MILP | BS-based | MILP | BS-based | MILP | BS-based |
| 2 | 0.25 | 2.86 | 101.0 | 103.0 | 116.5 | 116.5 | 108.15 | 110.40 |
| 4 | 0.63 | 3.54 | 83.0 | 83.0 | 113.5 | 113.5 | 95.00 | 97.10 |
| 6 | 1.29 | 4.20 | 77.0 | 77.0 | 87.0 | 87.0 | 82.15 | 82.75 |
| 8 | 3.00 | 5.70 | 74.0 | 75.0 | 82.0 | 86.0 | 78.65 | 79.90 |
| 10 | 13.39 | 3.44 | 72.0 | 73.0 | 79.0 | 80.0 | 75.95 | 77.35 |
| 15 | 2052.67 | 4.33 | 64.0 | 71.2 | 75.0 | 75.0 | 70.50 | 72.25 |
| 20 | – | 4.82 | – | 62.0 | – | 70.0 | – | 65.70 |
| 25 | – | 5.94 | – | 56.0 | – | 65.0 | – | 60.55 |
| 30 | – | 5.76 | – | 47.0 | – | 60.0 | – | 54.55 |
| 35 | – | 5.56 | – | 42.0 | – | 55.0 | – | 47.15 |
| 40 | – | 4.07 | – | 37.0 | – | 48.0 | – | 42.10 |
| 45 | – | 6.29 | – | 33.0 | – | 43.0 | – | 36.68 |
| 50 | – | 5.45 | – | 25.5 | – | 38.0 | – | 30.53 |

$LS - ES$ is fixed at 10. To mitigate the influence of randomness, each configuration is independently executed 20 times.

As shown in Table 7, when the $ES$ generation range is large, Gurobi is able to efficiently find the optimal solution. However, when the $ES$ generation range becomes small, the computational time required by Gurobi increases significantly. This phenomenon is mainly attributed to the substantial growth in the number of explored nodes during the branch-and-bound process.

Specifically, Gurobi enhances the lower bound by adding valid inequalities (cutting planes) to tighten the LP relaxation and constructs feasible integer solutions using heuristic methods such as greedy search, local search, and metaheuristics to improve the upper bound. In theory, a smaller feasible region should lead to more effective convergence of cutting planes. However, in this problem, as the $ES$ generation range decreases, feasible integer solutions become extremely sparse, making it difficult for heuristics to quickly find high-quality feasible solutions. Moreover, due to the increased discrepancy between the LP relaxation solution and the actual feasible integer space, cutting planes also struggle to tighten the feasible region effectively and in a timely manner. As a result, the convergence between the lower and upper bounds slows down significantly, leading to a sharp increase in the number of nodes that need to be explored, thereby substantially extending the overall solving time.

In contrast, the BS-based approach exhibits superior efficiency for smaller $ES$ generation ranges. Table 7 compares the performance of the MILP benchmark and the BS-based approach under varying $ES$ ranges. The experimental results reveal that Gurobi is highly sensitive to these changes: when the range is small (e.g., $x = 10$ or $x = 15$), it often fails to produce a solution within a reasonable time. The BS-based approach, however, remains largely unaffected by such variations, consistently delivering feasible solutions in around one second.

In terms of objective values, the results obtained by the BS-based approach are very close to those of the MILP benchmark. For instance, when $x = 20$, the average objective value is 22.20 for the BS-based approach and 21.67 for the MILP benchmark, with a difference of less than 2.5%; when $x = 30$, the gap is approximately 2.7%. Overall, in most cases, the solutions obtained by the BS-based approach deviate from those of the MILP benchmark by only 1%–4%, providing near-optimal performance while maintaining high computational efficiency.

In summary, when the $ES$ generation range is large and the number of feasible solutions is abundant, the MILP benchmark can solve the problem efficiently and deliver the optimal solution. However, when the $ES$ generation range is small, resulting in a limited number of feasible solutions and increased sparsity, Gurobi tends to encounter computational bottlenecks. In contrast, the BS-based approach demonstrates excellent stability and scalability across different $ES$ generation ranges, achieving a good balance between computational efficiency and near-optimality, thus providing a practical and scalable alternative for solving large-scale complex problems. Furthermore, in experiments varying the duration $DU$ of the visit to intermediate regions, we found that different values of this parameter did not exhibit any systematic impact on the performance results.

# 7 Conclusion

This paper addresses multi-robot path planning under Boolean task specifications and time-window constraints. We present a unified, collision-aware framework that integrates temporal constraints into the Boolean formalism and introduces a scheduler that enforces collision-avoidance at shared regions without altering the task logic. The problem is formulated as an MILP model to obtain exact solutions on small–medium

**Table 7**: Comparison (20 runs) of MILP (solved by Gurobi) and the BS-based approach on the impact of $ES$ generation range $[0, x]$ for fixed $m = 6$, $q = 1$, and 3 robots

| Range $[0, x]$ | CPU Time (s) | | Min. Obj. | | Max. Obj. | | Avg. Obj. | |
|---|---|---|---|---|---|---|---|---|
| | MILP | BS-based | MILP | BS-based | MILP | BS-based | MILP | BS-based |
| 10 | – | 0.97 | – | 16.5 | – | 22.0 | – | 18.55 |
| 15 | – | 1.06 | – | 18.5 | – | 22.5 | – | 20.75 |
| 20 | 422.66 | 1.19 | 18.0 | 19.0 | 28.0 | 28.5 | 21.67 | 22.20 |
| 25 | 234.41 | 1.13 | 17.0 | 17.0 | 28.0 | 31.5 | 22.88 | 23.63 |
| 30 | 65.75 | 1.09 | 19.5 | 19.5 | 32.0 | 34.5 | 26.38 | 27.10 |
| 35 | 1.36 | 1.15 | 19.0 | 20.5 | 34.5 | 34.5 | 28.00 | 28.70 |
| 40 | 1.31 | 1.15 | 22.5 | 23.0 | 45.0 | 45.0 | 31.03 | 31.93 |
| 45 | 0.91 | 1.16 | 22.0 | 22.0 | 48.0 | 48.0 | 32.33 | 32.53 |
| 50 | 0.72 | 1.19 | 22.0 | 22.5 | 53.0 | 53.0 | 35.95 | 36.20 |

instances, and a BS-based heuristic is developed for scalability. Experiments show that the method effectively balances logical and temporal requirements and significantly outperforms the pure MILP baseline in large-scale scenarios. Future work will focus on handling dynamic uncertainties and enabling real-time replanning in changing environments.

**Funding**

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

# References

Ait Haddadene, S.R., Labadie, N., Prodhon, C. (2016). A GRASP × ILS for the vehicle routing problem with time windows, synchronization and precedence constraints. *Expert Syst. Appl.*, *66*, 274–294, (https://doi.org/10.1016/j.eswa.2016.08.014)

Alharbi, M.G., Stohy, A., Elhenawy, M., Masoud, M., Khalifa, H.A.E.-W. (2021). Solving traveling salesman problem with time windows using hybrid pointer networks with time features. *Sustainability*, *13*(22), 12906, (https://doi.org/10.3390/su132212906)

Barb-Ciorbea, M.-A., Kloetzer, M., Mahulea, C. (2024). Planning strategy based on visibility graph and traveling salesman problem with time window constraints. *28th international conference on system theory, control and computing, icstcc* (pp. 521–526). IEEE. (https://doi.org/10.1109/ICSTCC62912.2024.10744736)

Chakraa, H., Guérin, F., Leclercq, E., Lefebvre, D. (2023). Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art. *Robot. Auton. Syst.*, *168*, 104492, (https://doi.org/10.1016/j.robot.2023.104492)

Chakraa, H., Leclercq, E., Guérin, F., Lefebvre, D. (2025). Integrating collision avoidance strategies into multi-robot task allocation for inspection. *Trans. Inst. Meas. Control*, *47*(7), 1466-1477, (https://doi.org/10.1177/01423312241295828)

Frey, C.M., Jungwirth, A., Frey, M., Kolisch, R. (2023). The vehicle routing problem with time windows and flexible delivery locations. *Eur. J. of Oper. Res.*, *308*(3), 1142-1159, (https://doi.org/10.1016/j.ejor.2022.11.029)

Gunawan, A., Lau, H.C., Vansteenwegen, P., Lu, K. (2017). Well-tuned algorithms for the team orienteering problem with time windows. *J. Oper. Res. Soc.*, *68*(8), 861–876, (https://doi.org/10.1057/s41274-017-0244-1)

Ham, A. (2020). Transfer-robot task scheduling in job shop. *J. Intell. Manuf.*, *31*, 1783–1793, (https://doi.org/10.1007/s10845-020-01537-6)

He, Z., Yuan, S., Ran, N., Lefebvre, D. (2025). Optimal path planning for multi-robot systems using Petri nets. *IEEE Robot. Autom. Lett.*, *10*(10), 9694-9701, (https://doi.org/10.1109/LRA.2025.3597901)

Hsaini, S., Ammour, R., Brenner, L., Charaf, M.E.H., Demongodin, I., Lefebvre, D. (2023). A reconfiguration method for multi-robot monitoring patrols. *Cybern. Syst.*, 1–26, (https://doi.org/10.1080/01969722.2023.2247267)

Kiadi, M., García, E., Villar, J.R., Tan, Q. (2023). A*-based co-evolutionary approach for multi-robot path planning with collision avoidance. *Cybern. Syst.*, *54*(3), 339–354, (https://doi.org/10.1080/01969722.2022.2030009)

Kloetzer, M., & Mahulea, C. (2020). Path planning for robotic teams based on LTL specifications and Petri net models. *Discrete Event Dyn. Syst.*, *30*(1), 55–79, (https://doi.org/10.1007/s10626-019-00300-1)

Liu, N., Hu, Z., Wei, M., Guo, P., Zhang, S., Zhang, A. (2025). Improved A* algorithm incorporating RRT* thought: A path planning algorithm for AGV in digitalised workshops. *Comput. Oper. Res.*, *177*, 106993, (https://doi.org/10.1016/j.cor.2025.106993)

Liu, X., Chen, Y.-L., Por, L.Y., Ku, C.S. (2023). A systematic literature review of vehicle routing problems with time windows. *Sustainability*, *15*(15), 12004,

(https://doi.org/10.3390/su151512004)

Lv, P., Luo, G., Ma, Z., Li, S., Yin, X. (2023). Optimal multi-robot path planning for cyclic tasks using Petri nets. *Control Eng. Pract.*, *138*, 105600, (https://doi.org/10.1016/j.conengprac.2023.105600)

Madridano, Á., Al-Kaff, A., Martín, D., de la Escalera, A. (2021). Trajectory planning for multi-robot systems: Methods and applications. *Expert Syst. Appl.*, *173*, 114660, (https://doi.org/10.1016/j.eswa.2021.114660)

Mahulea, C., & Kloetzer, M. (2017). Robot planning based on Boolean specifications using Petri net models. *IEEE Trans. Autom. Control*, *63*(7), 2218–2225, (https://doi.org/10.1109/TAC.2017.2760249)

McIntire, M., Nunes, E., Gini, M. (2016). Iterated multi-robot auctions for precedence-constrained task scheduling. *International conference on autonomous agents and multiagent systems, aamas* (pp. 1078–1086). IFAAMAS.

Nunes, E., Manner, M., Mitiche, H., Gini, M. (2017, 4). A taxonomy for task allocation problems with temporal and ordering constraints. *Robot. Auton. Syst.*, *90*, 55-70, (https://doi.org/10.1016/j.robot.2016.10.008)

Ow, P.S., & Morton, T.E. (1988). Filtered beam search in scheduling. *Int. J. Prod. Res.*, *26*(1), 35–62, (https://doi.org/10.1080/00207548808947840)

Petitprez, E., Guérin, F., Guinand, F., Germain, F., Kerthe, N. (2023). Decentralized coordination of a multi-UAV system for spatial planar shape formations. *Sensors*, *23*(23), 9553, (https://doi.org/10.3390/s23239553)

Shi, W., He, Z., Ma, Z., Ran, N., Yin, X. (2023). Security-preserving multi-robot path planning for Boolean specification tasks using labeled Petri nets. *IEEE Control Syst. Lett.*, *7*, 2017–2022, (https://doi.org/10.1109/LCSYS.2023.3284200)

Shi, W., He, Z., Tang, W., Liu, W., Ma, Z. (2022). Path planning of multi-robot systems with Boolean specifications based on simulated annealing. *IEEE Robot. Autom. Lett.*, *7*(3), 6091–6098, (https://doi.org/10.1109/LRA.2022.3165184)

Steinbiss, V., Tran, B.-H., Ney, H. (1994). Improvements in beam search. *3rd international conference on spoken language processing, icslp* (pp. 2143–2146). IEEE. (https://doi.org/10.21437/ICSLP.1994-538)

Sun, D., Chen, J., Mitra, S., Fan, C. (2022). Multi-agent motion planning from signal temporal logic specifications. *IEEE Robot. Autom. Lett.*, *7*(2), 3451-3458, (https://doi.org/10.1109/LRA.2022.3146951)

Wang, X., Kloetzer, M., Mahulea, C., Silva, M. (2015). Collision avoidance of mobile robots by using initial time delays. *54th ieee conference on decision and control, cdc* (pp. 324–329). (https://doi.org/10.1109/CDC.2015.7402221)

Yang, S., Yin, X., Li, S., Zamani, M. (2020). Secure-by-construction optimal path planning for linear temporal logic tasks. *59th ieee conference on decision and control, cdc* (pp. 4460–4466). (https://doi.org/10.1109/CDC42340.2020.9304153)

Yu, X., Yin, X., Li, S., Li, Z. (2022). Security-preserving multi-agent coordination for complex temporal logic tasks. *Control Eng. Pract.*, *123*, 105130, (https://doi.org/10.1016/j.conengprac.2022.105130)

Zhang, H., Luo, J., Long, J., Huang, Y., Wu, W. (2020). Multi-robot path planning using Petri nets. *International conference on verification and evaluation of computer and communication systems, vecos* (pp. 15–26). (https://doi.org/10.1007/978-3-030-65955-4_2)

Zhou, Z., Che, A., Yan, P. (2012). A mixed integer programming approach for multi-cyclic robotic flowshop scheduling with time window constraints. *Appl. Math. Model.*, *36*(8), 3621-3629, (https://doi.org/10.1016/j.apm.2011.10.032)

# Appendix A Boolean formula for the Case Study

$$
\begin{aligned}
\varphi_2 \;=\; & (\Pi_{41}, [12, 21], 5) \\
& \wedge \; \big((\Pi_{78}, [20, 27], 5) \;\vee\; (\Pi_{82}, [7, 10], 5) \;\vee\; (\Pi_{9}, [4, 10], 35)\big) \\
& \wedge \; \big((\Pi_{67}, [17, 28], 5) \;\vee\; (\Pi_{47}, [18, 22], 10) \;\vee\; (\Pi_{1}, [1, 8], 10)\big) \\
& \wedge \; \big((\Pi_{72}, [2, 13], 10) \;\vee\; (\Pi_{15}, [7, 19], 10)\big) \\
& \wedge \; (\Pi_{80}, [37, 49], 25) \\
& \wedge \; \big((\Pi_{58}, [20, 24], 10) \;\vee\; (\Pi_{10}, [47, 51], 10) \;\vee\; (\Pi_{45}, [18, 21], 10)\big) \\
& \wedge \; \big((\Pi_{33}, [40, 45], 5) \;\vee\; (\Pi_{56}, [13, 16], 10)\big) \\
& \wedge \; \big((\Pi_{37}, [8, 10], 10) \;\vee\; (\Pi_{50}, [6, 15], 5)\big) \\
& \wedge \; \big((\Pi_{32}, [3, 6], 10) \;\vee\; (\Pi_{66}, [15, 20], 10) \;\vee\; (\Pi_{44}, [11, 14], 10)\big) \qquad \text{(A1)} \\
& \wedge \; \big((\Pi_{30}, [19, 22], 5) \;\vee\; (\Pi_{65}, [1, 5], 10)\big) \\
& \wedge \; (\Pi_{8}, [17, 18], 10) \\
& \wedge \; \big((\Pi_{21}, [32, 43], 38) \;\vee\; (\Pi_{55}, [17, 20], 10)\big) \\
& \wedge \; (\Pi_{96}, [20, 43], 10) \\
& \wedge \; (\Pi_{89}, [18, 40], 10) \\
& \wedge \; \big((\Pi_{42}, [22, 24], 10) \;\vee\; (\Pi_{63}, [7, 9], 20) \;\vee\; (\Pi_{26}, [2, 4], 20)\big) \\
& \wedge \; (\Pi_{24}, [10, 100]) \;\wedge\; (\Pi_{35}, [10, 100]) \;\wedge\; (\Pi_{83}, [10, 100]) \\
& \wedge \; (\Pi_{6}, [10, 100]) \;\wedge\; (\Pi_{29}, [10, 100]).
\end{aligned}
$$