# MA-LAMA: Exploiting the Multi-Agent Nature of Temporal Planning Problems

J. CABALLERO TESTÓN*, Universidad de Alcalá, ISG, EPS, Spain
MARIA D. R-MORENO, Universidad de Alcalá, ISG, EPS, Spain and TNO, IAS, The Netherlands

**Background:** In the field of Automated Planning, the existence of multiple agents in a certain temporal setting introduces the possibility of concurrency. This severely increases the complexity of planning in multi-agent temporal scenarios, as the possible states in the search space grow exponentially.
**Objectives:** These types of domains are traditionally solved by making use of temporal reasoning techniques that do not directly address the "multi-agent nature" at their core. In contrast, we introduce MA-LAMA, a multi-agent temporal planner that makes use of multi-agent techniques to deal with the inherent complexity of multi-agent temporal scenarios.
**Methods:** We propose a sequenced framework in which several multi-agent planning techniques are applied: automatic agent detection, task decomposition, cost-informed goal assignment, and agent interaction analysis; that are aimed to reduce the search complexity when solving temporal planning tasks.
**Results:** Our results show that, in many cases, MA-LAMA outperforms other state-of-the-art temporal planners in plan cost optimization for well-known temporal domains.
**Conclusions:** These results, along with the fact that MA-LAMA does not incorporate any temporal reasoning during search, suggest that several widely considered temporal domains are best suited to be solved with multi-agent planning techniques, rather than with temporal reasoning.

**JAIR Associate Editor:** Nick Hawes

**JAIR Reference Format:**
J. Caballero Testón and Maria D. R-Moreno. 2025. MA-LAMA: Exploiting the Multi-Agent Nature of Temporal Planning Problems. *Journal of Artificial Intelligence Research* 83, Article 23 (August 2025), 41 pages. DOI: 10.1613/jair.1.18906

## 1 Introduction

Multi-agent planning (MAP) and temporal planning have traditionally been considered distinct areas of research within the broader domain of Automated Planning. Specifically, MAP solvers are commonly developed with a focus on classical non-temporal planning environments, whereas solvers dedicated to temporal planning predominantly focus on temporal and numerical reasoning. This fact has also been noted in other works, notably, the MAP Survey [53] remarks the integration of temporal dynamics into multi-agent (MA) systems as a significant ongoing challenge.

This distinction between temporal and MA systems becomes particularly relevant in scenarios where temporal systems inherently exhibit a MA structure and naturally require agent concurrency for optimal performance. However, they are typically addressed using temporal reasoning techniques, which commonly face significant

*Corresponding Author.

Authors' Contact Information: J. Caballero Testón, ORCID: 0000-0003-3099-3640, javier.caballerot@edu.uah.es, Universidad de Alcalá, ISG, EPS, Alcalá de Henares, Madrid, Spain; Maria D. R-Moreno, ORCID: 0000-0002-7024-0427, malola.rmoreno@uah.es, Universidad de Alcalá, ISG, EPS, Alcalá de Henares, Madrid, Spain and TNO, IAS, The Hague, The Netherlands.


DOI: 10.1613/jair.1.18906

computational challenges in scenarios involving numerous loosely-coupled agents. Specifically, these planners often first find a low-quality solution with minimal concurrency, making subsequent iterative searches computationally expensive, and unlikely to yield high-quality concurrent plans within reasonable time limits.

From this fact, the main idea of our work arises: exploring the applicability of MAP techniques to well-known temporal scenarios where the concurrency of multiple agents is at the core of the planning complexity. Our aim is to show that MAP techniques, rather than traditional temporal reasoning alone, can effectively address concurrency in these domains and produce high-quality plans.

In contrast to prior MAP methods, our main contribution is specifically to demonstrate that decomposition and agent-interaction analysis techniques borrowed from MAP can significantly enhance search efficiency and solution quality in temporal planning tasks, even without explicitly declared or autonomous agents. For this, we intentionally utilize "agents" as a conceptual tool that naturally emerges from domain structure, thereby enabling concurrency-oriented decomposition of temporal tasks rather than emulating autonomous agent decision-making processes.

We also contrast these results with those from conventional temporal reasoning solvers, highlighting how these approaches may differ in efficiency and effectiveness when addressing similar planning challenges. This novel approach to MA temporal planning provides deeper insights into the strengths and limitations of both MAP techniques and temporal reasoning methods in optimizing plan quality in complex cooperative planning environments.

The 2015 Competition of Distributed and Multi-Agent Planning (CoDMAP) [28] showcased a diverse array of MAP techniques, which have since been incorporated and built upon in various MA solvers. These can be classified according to several taxonomies:

(1) **Agent distribution**: where we can find (1) *single-agent planning*, where the plan is created and executed by a single agent; (2) *planning for multiple agents*, where in the solution, actions are typically assigned to different agents through the implementation of constraints; (3) *factored planning* where the planning task is decomposed into a set of independent agents and allows for distributed processing and potentially more efficient problem-solving; and (4) *planning by multiple agents*, which distributes the MAP task and put the focus on the coordination between the actuating entities.
(2) **Computational process**: solvers can be (1) *centralized*, featuring a monolithic design with a central process, or (2) *distributed*, where the planning task is shared across multiple processing units.
(3) **Plan synthesis scheme**: where (1) *threaded* solvers interleave planning and coordination, while (2) *unthreaded* solvers handle planning and coordination separately. Additionally, *unthreaded* approaches can be further categorized into non-exclusive sub-groups: (1) *post-planning coordination*, which emphasizes coordination after planning, (2) *pre-planning coordination*, where the MAP solver establishes constraints before search, and (3) *iterative response planning*, where agents receive inputs from other agents' search processes.
(4) **Communication mechanism**: if planners resort to external communication mechanisms, such as TCP/IP protocols, they exhibit (1) *external communication*, and if they simulate communication between agents, they present (2) *internal communication*.
(5) **Privacy preservation**: levels are categorized as (1) *strong*, (2) *object cardinality* or (3) *weak*, based on the extent to which each agent's sensitive information is protected.

Finally, MAP systems typically address plan quality optimization by employing cost-aware classical planners such as LAMA [45].

Our research contributes by selecting high-performing MAP techniques and adapting their algorithms for application in temporal planning contexts. Specifically, we focus on automated task decomposition, goal assignment,

and required cooperation studies. This culminates in the development of MA-LAMA, a factored, centralized, unthreaded temporal MAP system that integrates mixed pre-planning coordination and iterative response planning, internal communication and local heuristic search to effectively exploit the MA and concurrent nature of MA temporal domains. MA-LAMA represents a form of factored planning within a centralized framework, where the planning task is decomposed into agent-like structures but solved centrally, prioritizing search efficiency through decomposition. Our planner is built from the core search system of the LAMA planner [45], and effectively addresses plan optimization for loosely-coupled agents through a comprehensive approach that handles a wide range of metrics, coupled or not with the plan's *total-time*. This capability distinguishes MA-LAMA as a temporal planner, specially considering the challenges that state-of-the-art temporal solvers face in similar instances, as we will review in our experiments.

MA-LAMA does not explicitly showcase certain characteristics typically associated with classical MA systems, such as distributed decision-making, autonomy, or privacy preservation. This design decision arises directly from the central novelty of our approach: instead of developing a traditional MA system, MA-LAMA explicitly leverages MAP techniques as a means of efficiently solving temporal planning problems. Nevertheless, it should be noted that MA-LAMA does not aim for optimal solutions, as decomposition approaches can lead to non-complete exploration of the planning problem or to poor results in non-linear metrics.

Although several aspects of MA-LAMA have been discussed in previous works [8] [7], they primarily focused on the general description of agent decomposition and goal allocation methods, as well as a real-world tested domain. In contrast, this paper expands the empirical evaluation by covering a broader range of the International Planning Competition (IPC) benchmarks, assessing coverage, plan quality, and search efficiency. We also provide a more detailed description of the MA algorithms to enhance their application in temporal domains. A motivation example, the Exploration domain, is included to illustrate the internal process, and we introduce a new search method for more tightly coupled scenarios.

The paper is structured as follows. Section 2 reviews the relevant literature intersecting MAP and temporal planning, which helps to position our research. Next, Section 3 presents the LAMA planner, the foundational system upon which MA-LAMA is built. Section 4 describes the Exploration domain, which serves as a running example throughout the technical sections. In Section 5, we outline the theoretical foundations needed to understand MA-LAMA's operation. Section 6 details the planner's architecture and its key components. In Section 7, we conduct a two-part empirical evaluation of MA-LAMA. First, we evaluate MA-LAMA's coverage and plan quality performance in different domains. Subsequently, we analyze its search efficiency by comparing runtime with the quality of the generated plans in complex scenarios. Section 8 presents a critical analysis of our findings, examining MA-LAMA's strengths and limitations. Finally, section 9 summarizes the main findings of this work.

## 2 Related Work

Several lines of research address MA systems with temporal or durative aspects. From the Multi-Agent Path Finding (MAPF) community, recent works explore the formalization of MAPF problems with delay probabilities [34], the challenge of obtaining valid solutions even under unexpected delays [2], and algorithms that do not require quantization of wait and move durations [1]. Other approaches originate in the decision-making under uncertainty field, proposing methods for multirobot symbolic planning under partial knowledge [60], controller-synthesis procedures under coupling constraints and discretized time [37], and congestion avoidance policies in Markov automata [52].

To our knowledge and before implementing MA-LAMA, TFPOP [30] was the only domain-independent planner aimed for MA settings capable of handling temporal aspects such as durative actions. TFPOP is a centralized planner that exploits least commitment between loosely-coupled agents, allowing them to be flexible in terms of their execution schedule. At the same time, each agent maintains a thread of sequentially ordered actions

that follows a forward-chaining approach. As each agent can generate strong and rich local states about its own variables, TFPOP makes use of *coordination points* between them and solves a Constraint Satisfaction Problem (CSP+planning) [6] to commit in the necessary decisions to coordinate and order the actions between agents. Nonetheless, it has not been evaluated in comparison with other MAP or temporal solvers.

MA-LAMA and TFPOP both operate as centralized planners in the domains of MA and temporal planning, and also leverage *coordination points* in a similar way to make decisions on the schedule of each agent. We can begin to contextualize MA-LAMA by explaining its distinctive approach to agent coordination. Instead of solving a planning CSP+, MA-LAMA employs classical heuristic search integrated with a framework of *temporal constraints*. This strategy is implemented alongside the use of LAMA [45], which serves as the groundwork non-temporal planner for our system. This combination is specifically designed to advance our core objective of optimizing plan quality. We explore this idea in section 3.

Next, we introduce two separate subsections, each tackling one of the state-of-the-art on the MA-LAMA foundations: MAP and Temporal Planning.

## 2.1 Multi-Agent Planning

Unthreaded MAP solvers encompass a variety of approaches and systems. MAPR [5] utilizes goal allocation and plan obfuscation by independent agents to facilitate plan reuse, and CMAP, from the same authors, employs local task encryption coupled with single-agent search strategies. PMR [33] integrates simultaneous planning across all agents with plan merging and reuse. Distoplan [17] achieves optimal MAP solving by not restricting complex agent interactions, and PSM [57], which builds upon Distoplan by creating Planning State Machines: local representations that can be merged or projected. MAP-LAPKT [44] proposes that any MAP task can be manageable by a single-agent planner by utilizing an appropriate encoding. $A^{\#}$ [26] shows how $A^{*}$ can be adapted to factored planning. And DPP [36] integrates accurate public projection of MAP tasks information with *object cardinality* privacy preserving to deliver one of the best performances between the MAP solvers of its kind.

As examples of threaded solvers, we can consider several systems. Partial-order-planning (POP) based planners MAP-POP [55] [54], and its subsequent versions, FMAP and MH-FMAP [56], focus on distributed computations of the $h_{DTG}$ and $h_{Land}$ heuristics. Similarly, MADLA [51] employs a dual heuristic approach combining local and distributed versions of $h_{FF}$. Continuing, GPPP [35], presents a distributed privacy-preserving version for a *landmark* heuristic. MAFS [40] and its optimal counterpart, MAD-$A^{*}$ [39] are based in combining per-agent local estimations for each given state. And lastly, MAPlan [19] adopts a distributed and flexible strategy, employing multiple heuristics, such as $h_{FF}$ and *LM-Cut*, and demonstrating robust performance particularly in distributed settings.

Other MA techniques aim to reduce the search space through decomposition-based pruning through a symmetry score in classical planning [38]. And it has been extended for its use in numeric planning [49].

During the 2015 Competition of Distributed and Multi-Agent Planning (CoDMAP) [28] several of these already mentioned planners and techniques were showcased. Among them, the top performers were ADP [13] for coverage and CMAP-q [5] for quality centralized tracks. Whereas for distributed approaches, PSM [57] and MAPlan [19] were the overall winners.

Considering all these approaches, MA-LAMA integrates several key techniques to reduce the complexity of a given classical MA task. Task decomposition [31] and automatic agent detection from ADP [13] serve us to produce local search regions for distinct acting entities detected in the original MA task. Constraints between agents definition and *coordination points* detection from Planning First [41], along with constrained distributed planning graphs from DPGM [42], and the application of required cooperation studies [61][50] allow us to efficiently coordinate each agent plan acquisition. Finally, we borrow ideas on how to achieve satisfiability by

solving sequentially MAP tasks from $\mu$-SATPLAN [15]. MA-LAMA implements these techniques in a way that simplifies the search phase while ensuring plan quality optimization.

### 2.2 Temporal Planning

The literature on Temporal Planning is extensive, therefore, this section selectively approaches the field by focusing on the most successful temporal solvers in latest competitions and exploring planners whose methodologies are relevant to the MA-LAMA discussion.

The latest temporal competition was the IPC 2018 temporal track, where four solvers participated. These were PopCorn [47], TemPorAL [9], TFLAP [46] and CP4TP [21]. The overall winner of the track was TemPorAl, a portfolio that integrates several good performing planners from previous temporal competitions and launches them sequentially limiting their running time, but all participants are worth mentioning. CP4TP is also a portfolio, and was able to solve temporally expressive domains that TemPorAl could not. PopCorn is a planner that is able to deal with control parameters, also proposed in their work, which allow infinite domain parameters and durative-actions with non fixed duration. And finally, TFLAP operates as a forward partial-order temporal solver that employs classical heuristics without temporal reasoning, yet it remained competitive against the portfolio approaches.

There are also some notable approaches from previous IPC competitions. In the 2014 temporal track the top performers were Temporal Fast Downward (TFD) [16] and single-thread/multi-threaded versions of Yahsp3 [58]. TFD does not rely in rescheduling after search as most temporal solvers do, instead it makes use of the context-enhanced additive heuristic directly in temporal search. On the other hand, Yahsp3 performs state-space heuristic search by computing look-ahead relaxed plans. ITSAT [18] was another notable participant, a SAT-Based Temporally Expressive Planner, which is used in the CP4TP portfolio in the 2014 competition. Lastly, OPTIC [4] is a temporal planner that delivers state-of-the-art performance in problems where the objective is to optimize the plan numeric cost and preferences.

MA-LAMA addresses temporal problems by initially managing the temporal complexity using MAP techniques, followed by the implementation of cost-aware planning methods. Consequently, we do not make use of any of the described temporal reasoning approaches. This mirrors the strategy used by TFLAP [46], which also performs search using classical heuristics. In addition, MA-LAMA deals with the temporal features by borrowing the *snap-operators* paradigm, continuous numeric effects management and frontier state *temporal constraints* from POPF [11] and OPTIC [4], both from the COLIN [10] family of planners.

In this context, MA-LAMA operates as a temporal planner that utilizes temporal planning techniques specifically for adapting and translating temporal tasks to other paradigms, while also maintaining the temporal integrity of the plans during the search phase. Additionally, MA-LAMA aims to efficiently optimize any numeric function as a metric, coupled or not with the *total-time* of the plan, which is a line of work that has not been much explored within the temporal planning community.

## 3 The LAMA Planner

The LAMA planner [45] is a classical planning system built upon the concepts of heuristic forward search. It is derived from the Fast Downward solver [22] and makes use of its finite-domain variables task representation, creating multi-valued planning tasks (MPTs), also referred to as $SAS^+$ planning problems [3]. A distinctive feature of LAMA is the use of *landmarks* [43], which are propositional formulas in a planning task that are needed to be true at some point for a solution to be found.

The structure of LAMA is divided into three distinct modules, from which MA-LAMA builds upon:

- **Translation.** This module transforms the PDDL input to the MPT paradigm. To achieve this, it detects groups of mutually exclusive facts, called *invariants*. Then, the multi-valued variables are obtained by

creating one for each *invariant* and encoding which of the mutually exclusive facts is true for a given world state.

- **Knowledge Compilation.** The planner creates data structures that are key for determining the *landmarks*. Additionally, they increase efficiency when determining which operators are applicable for a certain world state and the changes in the variables values.
- **Search**. LAMA incorporates two algorithms for heuristic search: greedy best-first search and weighted A* (WA*). Additionally, two heuristic functions are used to evaluate all states, which results in two different open states queues. When choosing the next state to expand, LAMA alternates between these queues using numerical priorities. The two heuristics employed are:
  - A pseudo-heuristic based on *landmarks* [25] [48], that computes a *landmark* graph and generates an estimation for a state based on the number of still non-attained *landmarks*.
  - A cost sensitive variant of the FF heuristic [24], that generates a relaxed plan by disregarding the actions delete effects and returns the count of actions involved.

  In conjunction, LAMA implements a *Preferred Operators* system, that identifies particularly useful operators and favors expanding states that are reached through them. *Anytime search* is also used, and allows to continue the search for better solutions until search is interrupted or the search space is completely explored.

LAMA serves as a base for MA-LAMA mainly for two reasons:

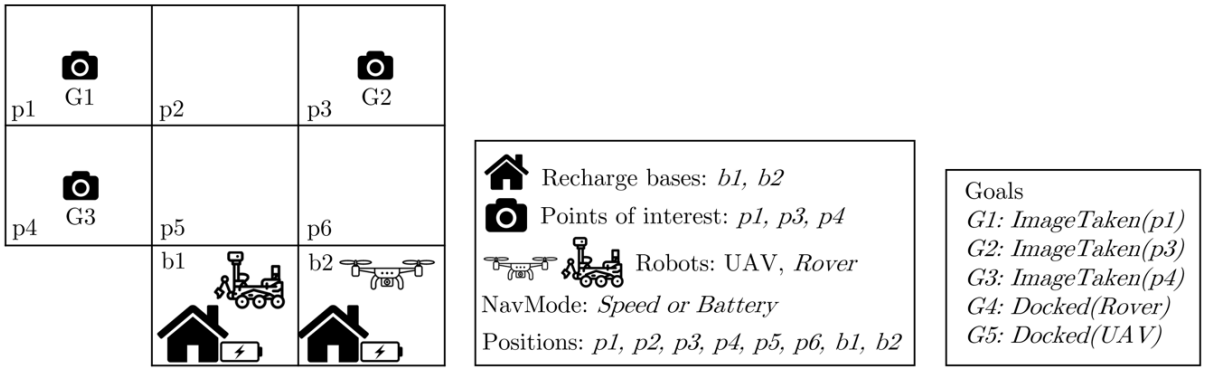
- **LAMA as a search core framework** has already been utilized in several classical MAP solvers, such as CMAP-q [5], winner of the quality CoDMAP track. Thus, given that our hypothesis tests the effectiveness of MAP techniques in addressing temporal complexity, our goal is to make minimal modifications to existing frameworks. It is important to note that, despite using LAMA as a base, most classical MAP solvers ignore action costs.
- **Cost sensitive heuristics.** The authors of LAMA pointed out a weakness in the planner's framework, consisting in the incorporation of action costs into the heuristic values, which proved to be a disadvantage in terms of solved problems. This makes sense, as in many cases, following the path of total cost optimization does not necessarily lead to potential solutions. In MA-LAMA, this issue is directly addressed by the task decomposition techniques, which results in much simpler to solve sub-tasks where introducing action costs in the heuristic estimations, combined with *anytime* iterative search from LAMA, can potentially lead to faster and better solutions, specially in temporal tasks.

## 4 The MA Temporal Exploration Domain

This section presents the MA temporal scenario that motivates the development of MA-LAMA: the Exploration domain. The domain serves two key objectives: (1) to clearly illustrate the type of MA temporal problems that MA-LAMA is designed to address, and (2) to use it as a running example during the description of the MA-LAMA framework.

The Exploration domain shares similarities with the temporal Rovers' IPC domain, where exploration agents navigate a grid-like map composed of interconnected locations, with manually defined paths shaping the structure of the world. Here, agents perform specific tasks at points of interest. However, unlike the Rovers domain, where communication requires coordination through a single channel, our setup does not impose such a limitation. Instead, the complexity in this domain arises from numeric constraints, including:

- **Recharge limitations**: agents must find an available recharge base, introducing concurrency challenges.
- **Operation modes**: agents can switch between modes that affect movement cost and time, allowing for a trade-off between energy efficiency and speed. We will refer to these modes as *NavMode S*, that prioritizes speed, and *NavMode B*, that prioritizes battery consumption.




| **Operators** | |
|---|---|
| GoTo(Robot, NavMode, PositionFrom, PositionTo) | TakeImage(Robot, Position) |
| ChangeNavMode(Robot, NavMode1, NavMode2) | Recharge(Robot, RechargeBase) |
| Dock(Robot, RechargeBase) | Undock(Robot, RechargeBase) |

Fig. 1. MA temporal Exploration problem with two heterogeneous robots: a ROVER and a UAV. Each starts docked in a recharge base, *b1* and *b2*, and must take three images in points of interest, *G1: TakeImage(p1)*, *G2: TakeImage(p3)* and *G3: TakeImage(p4)*. In the end, both robots must return to a base, *G4: Docked(ROVER)* and *G5: Docked(UAV)*.

```
:durative-action GoTo
    :parameters(?r - robot, ?n - NavMode, ?from, ?to - position)
    :duration (/ ((distance ?from ?to) (speed ?r ?n)))
    :condition (and ((at start (at ?r ?from))
                     (over all (> (avail_battery ?r) (0)))
                     (over all (notDocked ?r))
                     (over all (nav_mode ?r ?n)) ))

    :effect (and ((at start (not (at ?r ?from)))
                  (at end (visited ?to))
                  (at end (at ?r ?to))
                  (at end (decrease (avail_battery ?r) (bat_move)))
                  (at end (increase (battery ?r) (bat_move)))
                  (at end (increase (risk ?r) (p_risk ?to)))))
```

Fig. 2. *Goto* PDDL2.1 durative operator from the Exploration problem depicted in Figure 1. It includes the declaration of parameters, the duration as a function of distance and speed, conditions and effects.

The domain presents battery usage, mission risk, distance, and time through numeric variables. Each location has an associated danger value, increasing the mission risk depending on the path followed by the robots. Plan metrics may consist of any weighted combination of these fluents.

```
(:metric minimize (+ (* 0.6 (battery UAV))
                     (* 1 (battery ROVER))
                     (* 0.6 (risk UAV))
                     (* 1 (risk ROVER))))
```

Fig. 3. Quality metric in PDDL2.1 syntax, balancing *battery* usage and associated path *risk* for UAV and ROVER movements. The weights associated with each agent variables, 1 for ROVER and 0.6 for the UAV, are set to encourage the usage of the UAV by the planners. These weights are configurable during the problem definition.

Although the battery recharge constraints add complexity, these problems are generally manageable without intensive search space exploration. The real challenge lies in balancing the plan metrics to achieve optimized solutions efficiently.

To illustrate MA-LAMA's approach in subsequent sections, we created a simplified instance of the Exploration scenario, shown in Figure 1.

The depicted setup includes two exploration robots: a UAV and a ROVER, which can be *Docked* (or not) at a base (*b1* and *b2*). These robots can move prioritizing speed *(NavMode S)* or battery efficiency *(NavMode B)*. The map consists of eight positions *(p1, p2, p3, p4, p5, p6, b1, b2)*. The available operators include moving an agent (*GoTo*), take an image (*TakeImage*), change the navigation mode (*ChangeNavMode*), recharge at a base (*Recharge*), dock at a base (*Dock*), and undock from a base (*Undock*).

As a reference, Figure 2 shows the declaration of the *GoTo* operator. Its parameters specify the robot to move (*?r*), navigation mode (*?n*), and initial (*?from*) and final positions (*?to*). The operator's duration is given by a function of distance and speed, which varies according to the navigation mode (*S* or *P*). Its conditions include propositional facts that involve the robot's current position and chosen navigation mode, as well as a numeric requirement ensuring the robot has sufficient battery. The propositional effects update the world state, indicating that the robot has moved and marking a location as visited, while the numeric effects capture battery usage and associated risk. The numeric variable *bat_move* represents a simplified numeric cost, which in practice would depend on the distance traveled and navigation mode. As will be detailed in subsequent sections, MA-LAMA handles propositional and numeric conditions and effects using different mechanisms.

In the initial state, the UAV is *Docked* at *b2* and set to *NavMode S*, while the ROVER is *Docked* at *b1* and set to *NavMode S*. No images have been taken, no position has been visited and no action is happening. The goal state for the problem requires taking three images, *G1: ImageTaken(p1)*, *G2: ImageTaken(p3)*, and *G3: ImageTaken(p4)*. Upon completing their tasks, both robots must return to the *Docked* state, *G4: Docked(UAV)* and *G5: Docked(ROVER)*.

The evaluation metric in this example assigns configurable weights, as shown in Figure 3. The metric aims to minimize the overall sum, favoring UAV operations by assigning them lower penalty values compared to ROVER actions. This incentivizes UAV usage, while ROVER movements provoke higher costs. Planners must recognize this weighting scheme to generate optimal plans. Additionally, combining these two numeric variables increases the complexity of the problem from a numerical optimization perspective, as predicting optimal search paths depends on multiple conditions.

In the following section, we provide formal definitions for multi-valued planning tasks (MPTs) and other foundational concepts that are essential for understanding the MA-LAMA framework and the subsequent discussion of its operational structure.

## 5 Background

This section introduces the key technical concepts necessary for comprehending MA-LAMA's description, starting with the temporal tasks definition in PDDL2.1 semantics [20]:

**Definition 1. Temporal Planning Task**
*A **temporal planning task** is defined as* $\Lambda = \langle \rho, \vartheta, O_{inst}, O_{dur}, s_0, g \rangle$ *where:*

- $\rho$ *is a set of atomic propositional facts,*
- $\vartheta$ *is a set of real-valued numeric fluents,*
- $O_{inst}$ *is a set of grounded instantaneous operators or actions,*
- $O_{dur}$ *is a set of grounded durative operators or actions,*
- $s_0$ *is the initial state, and*
- $g$ *is the goal condition.*

The distinction between durative, $o_{dur} \in O_{dur}$, and instantaneous operators, $o_{inst} \in O_{inst}$, lies in the fact that durative operators take time to complete, $dur(o_{dur})$, and perform a state transition from $s$ to $s'$. While instant operators have preconditions, $pre(o_{inst})$, and effects, $eff(o_{inst})$; durative actions have start conditions, $startCond(o_{dur})$, end conditions, $endCond(o_{dur})$, overall $dur(o_{dur})$ conditions, $inv(o_{dur})$, start effects, $startEff(o_{dur})$, end effects $endEff(o_{dur})$ and numeric continuous effects, $numContEff(o_{dur})$.

For the *Goto* durative operator in Figure 2, the first condition belongs to $startCond(o_{dur})$, whereas the third and fourth conditions belong to $inv(o_{dur})$. The second condition is numeric, but for now is part of $inv(o_{dur})$ regardless. Regarding effects, the first effect corresponds to $startEff(o_{dur})$, second and third to $endEff(o_{dur})$, and all final three belong to $numContEff(o_{dur})$.

In order to deal with the continuous nature of these types of tasks, we make use of two techniques from the COLIN [10] family of planners.

We perform a translation from the temporal paradigm just described to the *snap-operators* paradigm. This operation encodes start and end endpoints of $o_{dur}$, $o_{\vdash}$ and $o_{\dashv}$, as instantaneous operators, with $pre(o_{\vdash}) = startCond(o_{dur})$, $eff(o_{\vdash}) = startEff(o_{dur})$, $pre(o_{\dashv}) = endCond(o_{dur})$, and $eff(o_{\dashv}) = endEff(o_{dur})$. This encoding removes the continuous effects and preconditions of durative operators while maintaining the capability to reason with concurrent operators, as it allows them to overlap. Intuitively, all invariant conditions, $inv(o_{dur})$, for a certain durative operator ($o_{dur}$) need to be continuously upheld from $o_{\vdash}$ to $o_{\dashv}$. Finally, a control variable is introduced for each $o_{dur}$ to indicate that the operator is running between the *start* and *end snap-operators*.

In the Exploration scenario, the transformation of the *Goto* operator depicted in Figure 2 produces two *snap-operators*: *start_GoTo* ($o_{\vdash}$) and *end_GoTo* ($o_{\dashv}$). Consequently, a new control variable *GoToOnGoing* is introduced with the same arguments inherited from the original *GoTo* operator.

Moreover, it is possible to establish a metric to evaluate the quality of a plan, $M$, and optimizing this metric value is added to the planners responsibilities along with solving the propositional problem. Planners vary widely in the types of metrics they support, from LAMA's implementation of total-cost to OPTIC's time-dependent continuous costs. In MA-LAMA, we define $M$ as an array of weighted numeric variables, $\{w_1 * v_1, w_2 * v_2, ..., w_n * v_n\}$ where $v_n \in \vartheta$ and $w_n$ is a real number. $v$ can also be the duration of the plan, the *total-time*. As a reference, the metric for the Exploration scenario can be seen in Figure 3.

From this temporal planning task, MA-LAMA translates it to the non-temporal multi-valued planning task (MPTs) paradigm, also referred as $SAS^+$ planning problems [3]. In order to facilitate presentation, we omit axioms and compile away conditional effects in this version of MPTs.

**Definition 2. Multi-valued planning tasks (MPTs)**
*A **multi-valued planning task (MPT)** is a 4-tuple* $\langle V, I, G, O \rangle$ *where:*

- $V$ *is a finite set of multi-valued variables* $v$*,*
- $I$ *is the initial state for* $V$*,*
- $G$ *is the goal condition and a partial state of* $V$*, and*
- $O$ *is a set of instantaneous operators or actions.*

Table 1. Multi-valued variables list derived from the problem of Figure 1. The second column shows the number of variables of each type in the MPT, and the third indicates their number of possible values. The last two columns depict the initial and goal states respectively. Non-defined goal values imply no constraints on those variables.

| Multi-Valued Variables | Quantity | Values | Initial State | Goal state |
|---|---|---|---|---|
| **UP**: UAV Position | 1 | 44 | At *b2* | |
| **UD**: UAV Docked | 1 | 6 | Docked | Docked |
| **UN**: UAV NavMode | 1 | 4 | Mode *S* | |
| **RP**: ROVER Position | 1 | 44 | At *b1* | |
| **RD**: ROVER Docked | 1 | 6 | Docked | Docked |
| **RN**: ROVER NavMode | 1 | 4 | Mode *S* | |
| **I**: ImageTaken | 8 | 2 | No images taken | *[p1, p3, p4]* |
| **V**: VisitedPosition | 8 | 2 | No visited positions | |
| **IO**: TakeImageOngoing | 16 | 2 | No *TakeImage* on-going | |
| **RO**: RechargeOngoing | 4 | 2 | No *Recharge* on-going | |

The crucial distinction between these two representations lies in MPT employing multi-valued variables, $v \in V$, contrasting with the boolean-types used in PDDL2.1. It should be noted that these propositional multi-valued variables are characterized as a set of mutually-exclusive states plus the *undefined* state.

In LAMA and MA-LAMA, the creation of the multi-valued variable set, $V$, is inherited from Fast Downward (FD) [22], although we introduce some optimizations that improve runtime for the encoding of MPTs from the PDDL *snap*-operator paradigm. These will be described in section 6.1.

For the Exploration scenario from Figure 1, the propositional multi-valued variables generated, $v \in V$, are depicted in Table 1. It shows each multi-valued numeric variable type, the total number of such variables in the MPT, their possible value sets, and how the initial and goal states are encoded. As can be seen, one variable is generated for the *Position*, *Docked (or not)*, and *NavMode* states. Each of these encapsulates all possible values (e.g. possible positions) and control variables that encode on-going state transitions (e.g. *GotoOnGoing* for the *GoTo* operator). For instance, the *UAV Position* variable contains 44 possible values: one for each possible position (8), and one for each possible position transition (36), containing both *at* and *GoToOnGoing* variable types.

Other PDDL propositional variables are directly encoded in the MPT as two state variables (*true* or *false*). This happens for all possible images (*ImageTaken*), visited positions (*Visited*), and control variables *RechargeOnGoing* and *TakeImageOnGoing*. As a consequence, for these, there will be as many variables declared in the final MPT as possible instantiations of the original PDDL boolean variable. As an example, there are 16 *TakeImageOngoing* variables: one for each robot (2) and position (8).

To properly handle all components of a temporal planning task (Definition 1), we extend the original MPT framework. Specifically, we introduce a new set of multi-valued numeric variables, *N*, to explicitly represent numeric fluents, $\vartheta$, paralleling how, in MPTs, $V$ represents propositional fluents, $\rho$. Additionally, we incorporate a relevant numeric metric, *M*, culminating in what we refer to as the extended MPT (eMPT).

**Definition 3. Extended Multi-valued planning tasks (eMPTs)**
*An **extended multi-valued planning task (eMPT)** is a 6-tuple $\langle V, I, G, O, N, M\rangle$ where:*

- *$V$, $I$, $G$ and $O$ are defined as in Definition 2.*
- *$N$, a finite set of multi-valued numeric variables, n, each defined by a real numeric value and a finite set of exclusive states, and*
- *$M$, a metric to measure the plan quality, directly assigned from the temporal task.*

This extension enables handling any numeric operation as an *effect* on $N$. Further details on how MA-LAMA calculates the possible values for each numeric variable $n$ will be discussed in section 6.1.

Additionally, to obtain a MAP task definition, we can follow the formalization of eMPTs from *Definition 3*, introducing the existence of several acting entities that can operate independently and modify the environment concurrently.

**Definition 4. Multi-Agent Planning Task (MAP)**
*A **Multi-Agent Planning Task (MAP)** is a 7-tuple* $\langle AG, V, I, G, \{O^i\}_{i=1}^{n}, N, M\rangle$ *where:*

- $V, I, G, N$ *and* $M$ *are defined as in the Definition 3,*
- $AG$ *is a finite set of n agents or independent planning entities, and*
- $\{O^i\}$ *is a finite set of operators that a certain agent has access to.*

As can be seen, the inclusion of the concept of agents also affects the operators set, $O$, in eMTPs, which are now exclusive per agent and indicate each individual entity's characteristics. In the scenarios that MA-LAMA addresses, agents, $AG$, are not declared explicitly. Instead, they will be detected as data structures that emerge from the domain structure. This is also true for the Exploration domain from Figure 1, where agents are not declared as such, nor are detected from textual information from the variable type. The process for this automatic detection of independent entities will be described in section 6.2.1.

By employing the MPT representation, it is possible to build graphs that explore the evolution of each variable within an MPT, these are *domain transition graphs* (DTGs) [27].

**Definition 5. Domain Transition Graph (DTG)**
*For an MPT* $\Phi$ *with a set of variables* $V$ *and* $v \in V$*, the **Domain Transition Graph** for* $v$*,* $DTG(v)$ *is the labeled digraph with a set of vertex* $D_v$ *that contains an arc* $(d, d')$ *iff* $d \neq d'$ *and there exists an operator,* $o \in O$*, where* $(v, d) \in \text{pre}(o)$ *including* $d =$ *undefined, and* $(v, d') \in \text{eff}(o)$.

As we have extended the definition of MPTs to eMPTs to include numeric variables, $N$, we can also define graphs that study the evolution of numeric variables, we call them **Domain Numeric Transition Graphs (DNTGs)**. Their formal definition, along with the insights that we extract from them, is discussed in section 6.2.3.

Additionally, we will make use of the *Causal Graph* (CG) [22], which is a visual representation of variable dependencies derived from available operators in a given task. The *CG* is essential for MA task decomposition techniques.

**Definition 6. Causal Graph (CG)**
*For an MPT* $\Phi$ *with a set of variables* $V$*, the **Causal Graph** of* $\Phi$*,* $\boldsymbol{CG}(\Phi)$ *is the directed graph with a set of vertex* $V$ *that contains an arc* $(v, v')$ *iff* $v \neq v'$ *and one of the following conditions is true:*

- *The domain transition graph of* $v'$ *has a transition with some condition on* $v$.
- *The set of affected variables in the effect list of some operator includes both* $v$ *and* $v'$.

Figure 4 shows the *CG* for the Exploration domain MPT from Table 1. Each node represents a multi-valued variable in the MPT and edges indicate influences between them. For example, the directed links from *RD* to *RP*, *V*, *RN*, *RO* and *IO* indicate that *ROVER Docked* variable affects when the ROVER can change positions, change its *NavMode*, recharge and take images, respectively. To increase clarity, this graph groups the variables *TakeImageOnGoing (IO)*, *RechargeOngoing (RO)*, *ImageTaken (I)* and *Visited (V)* into single nodes, although Table 1 indicates that multiple instances actually exist. This simplification does not affect the subsequent discussion of automatic agent detection, for which the *CG* is key.

Having reviewed the necessary background, we now turn to the description of MA-LAMA's architecture.

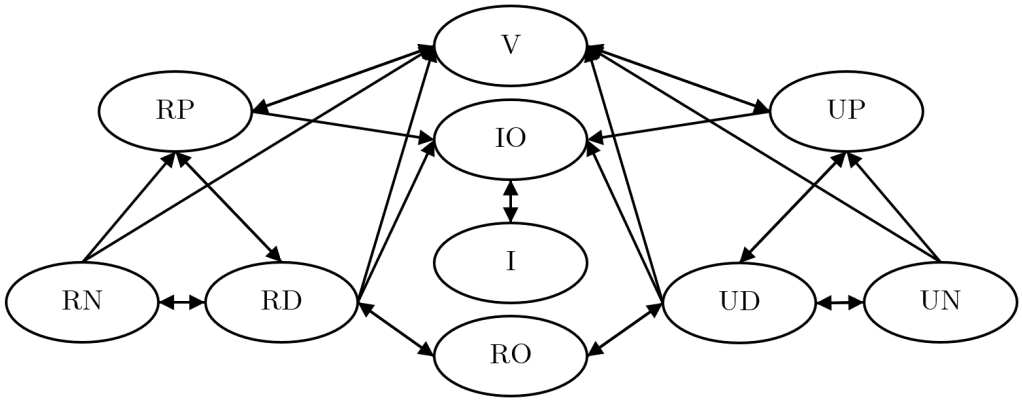



Fig. 4. Causal Graph derived from the example in Figure 1. Each node represents a multi-valued variable from Table 1.

## 6 MA-LAMA System Architecture

We now provide a detailed explanation of each component of MA-LAMA. Since MA-LAMA is based on LAMA, our planner retains the same general structure, which includes the Translation, Knowledge Compilation, and Search modules, but with significant modifications. Additionally, MA-LAMA introduces a new module called Unify.

The primary functions of each module are next described:

**Translation.** It reads the PDDL2.1 input and translates it first to the *snap-operators*, and second to the eMPT paradigm. For this step, LAMA's original MPT creation framework is used as a basis.

**Knowledge Compilation.** It receives the eMPT as input and performs transformations to it based on extracted MA information. This process involves three primary algorithms:

- *Agent Decomposition (AD)*: MA-LAMA identifies the possible agents within the eMPT using the causal graph (*CG*), following the approach described by Crosby et al. 2013. It then decomposes the task based on the independent agents.
- *Goal Categorization and Assignment (GCA)*: each goal from the eMPT is categorized into *coordination* and *cooperation* sets based on required cooperation principles [61]. Goals are then assigned to the agents identified during the AD phase, taking into account *coordination points* and metric cost estimations derived from relaxed searches. This process results in sets of linked eMPTs called *Parallel Steps*, which are solved independently and are categorized according to the nature of the goals.
- *Agent Interaction Analysis (AIA)*: it determines whether the MAP task should be addressed during search in a decomposed or assembled manner. This study relies primarily on Domain Transition Graphs (DTGs) and our newly introduced **Domain Numeric Transition Graphs (DNTGs)**.

The sequential decision-making process used by these three algorithms is intentional. Our experiments showed that interleaving the decisions from the AD and GCA did not lead to better plans. In fact, our results indicate that better solutions are closely correlated with focusing on optimal subtasks resolution during the search phase. Therefore, the AD algorithm is designed to generate simple agents and local eMPTs with the minimal necessary operators.

**Search**. The search process in MA-LAMA can be conducted using two distinct procedures:

- *Parallel Search*: it handles the *Parallel Steps*, received from the AD and GCA algorithms. These are eMPT sets that are addressed individually and in parallel by inheriting *temporal constraints* between agents. The

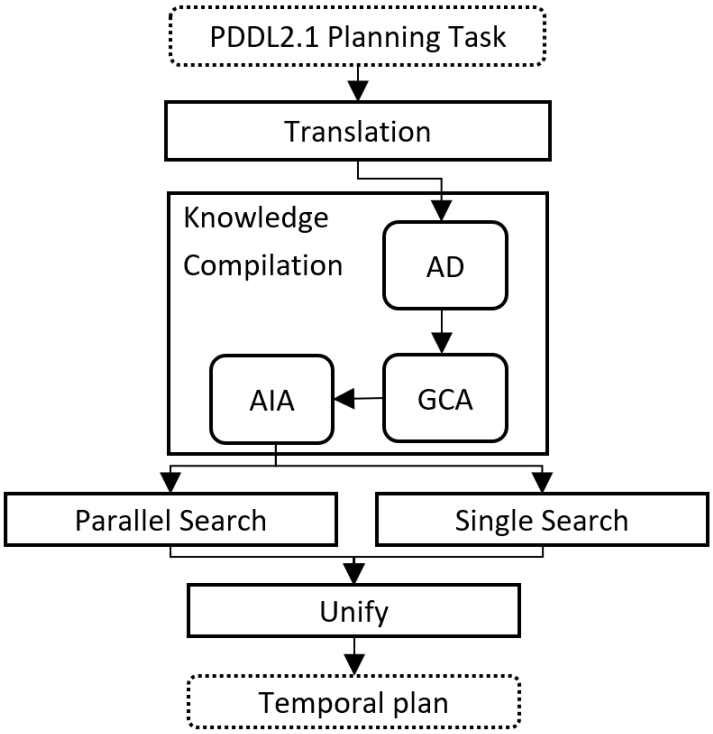



Fig. 5. MA-LAMA scheme for MA temporal planning tasks resolution. The input is a PDDL2.1 task that is translated to the *snap-operators* and eMPT paradigms in the *Translation* module. The *Knowledge Compilation* module automatically detects agents in the *Agent Decomposition (AD)* algorithm, assigns goals in the *Goal Classification and Assignment (GCA)* and decides which search mode to launch in the *Agent Interaction Analysis (AIA)*. Goals are solved in linked search processes during *Parallel Search* or in one *Single Search* for the whole eMPT. Finally, each partial plan is unified in the final *Unify* module to produce the temporal plan.

way these constraints are handled depends on whether they involve *cooperation* or *coordination Parallel Steps*. Single eMPTs are solved using two classical heuristics, $h_{Land}$ [25] [48] and $h_{FF}$ [24], within a *WA** search process. This process integrates a numeric and temporal framework to ensure soundness. The output is a partial *snap* plan for each individual *Parallel Step*.

- *Single Search*: if the task cannot be decomposed in agents or the AIA determines that it is more efficient to solve the task without decomposition, the search is initiated for the full MA temporal task. MA-LAMA uses the *WA** search with the same classical heuristics, combined with pruning techniques based on the extracted MA information. Nevertheless, the main priority of MA-LAMA is to favor *Parallel Search* whenever feasible.

**Unify**. It takes all partial plans as input and merges them concurrently, verifying numeric and temporal consistency throughout the process. Additionally, the *snap-operators* paradigm is reversed to produce the final temporal plan.

In summary, MA-LAMA automatically decomposes MA temporal tasks by identifying independent agents. Then, the search addresses the goals through linked search processes, called *Parallel Steps*, that involve agents based on metric cost estimations. Finally, all partial plans for each *Parallel Step* are assembled and the full temporal plan is produced. The MA-LAMA scheme for temporal MA tasks resolution can be seen in Figure 5.

If the decomposition of the MAP temporal task is either not feasible or deemed unnecessary by the planner, MA-LAMA's temporal and numeric framework can still address the resolution process with the *Single Search* procedure. This will be discussed in section 6.5, as the decomposition process introduces significant interactions between the *coordination points* and the landmarks leveraged by LAMA's heuristics.

Next, we will provide a detailed review of each of the components mentioned above. For each key component, we will also describe the corresponding step in the resolution process of the Exploration domain depicted in Figure 1.

## 6.1 Translation

The main goal of MA-LAMA's Translation module is to perform transformations over the PDDL2.1 task so that MA techniques can be applied. Therefore, the first operation aims to remove the temporal nature of the durative operators by translating them to the ***snap-operators* paradigm**.

This can be done in a very straight forward way, as durative operators, following the Definition 1 in Section 5, can directly be split into two *snap-operators*: a start operator, $o_{\vdash}$, and an end operator, $o_{\dashv}$, while considering at the same time the duration, *dur*, the overall conditions (including numeric conditions), *inv*, and the continuous numeric effects, *numContEff*. A binary control variable is introduced for each durative operator so that it is possible to encode that the operator is running, an approach first used in LPGP [32]. Specifically, the start operator $o_{\vdash}$ has a precondition ensuring the control variable equals *false* and sets it to *true* as an effect; consequently, the end operator $o_{\dashv}$ requires that the control variable is *true* and resets it to *false* upon completion. This modeling choice prevents durative operators from self-overlapping, and consequently limits MA-LAMA from solving tasks that require such self-overlaps.

As an example, the *Goto* durative operator depicted in Figure 2 is translated into two *snap-operators*, *start_GoTo* ($o_{\vdash}$) and *end_GoTo* ($o_{\dashv}$), and a new control variable *GoToOnGoing*. By setting *GoToOnGoing* to *true* in *start_GoTo* and reverting it to *false* in *end_GoTo*, we encode the fact that the *GoTo* action is running over its duration.

Handling the continuous numeric effects and conditions requires special attention. First, effects, *numContEff(o)*, are all of the form $v\{+=, -=, *=, /=, =\}\mathrm{w}.\vartheta + c$ where $v \in \vartheta$, "w" is a vector of constants and "c" a scalar constant. Each numeric effect is applied individually between $o_{\vdash}$ and $o_{\dashv}$ in a linear manner, meaning the numeric change is uniformly distributed throughout the duration of the operator. Note that the simultaneous presence of multiple numeric effects may indeed produce combined non-linear numeric evolutions (such as quadratic behavior). However, these combined interactions are not explicitly processed in the Translation module; rather, each numeric effect is treated individually and linearly. Later, during the search phase, numeric variable values are computed by linearly applying each individual numeric effect currently active. In the Exploration problem, the *GoTo* operator from Figure 2 presents numeric effects. An increase in the battery consumption of a given robot, *?r*, is encoded as *(battery ?r)* += $\mathrm{w}.\vartheta$, where w is a weight vector consisting of zeros except for the value corresponding to the *(bat_move)* numeric variable.

Similarly, "overall" numeric conditions, $numCond(o) \in inv(o)$, present a "continuous" numeric constraint over $v$ of the form $v\{>, <, \geq, \leq\}\mathrm{w}.\vartheta + c$. We encode that they must hold for the whole duration between $o_{\vdash}$ and $o_{\dashv}$. The *GoTo* operator in Figure 2 also presents a numeric condition for the battery available of a certain robot, *?r*, which would be encoded as *(battery ?r)* > $\mathrm{w}.\vartheta + c$, where w is a weight vector consisting of zeros, and $c$ is a scalar of value zero.

Then, MA-LAMA translates the *snap* temporal task to an **eMPT** based in the formalism of the LAMA and Fast Downward (FD) [22] MTP representation.

The main objective that must be accomplished during a translation from PDDL-like planning tasks to MPTs is an *invariant* synthesis algorithm that finds groups of mutually exclusive facts. These groups represent facts in the world that cannot happen at the same time and, and by encoding which of them is true for a certain world

state, the multi-valued variables are created. In MA-LAMA, the original LAMA *invariants* algorithm design [23] is maintained with one contribution that optimizes the multi-valued variables definition runtime by incorporating one rule:

*For every atom, $v$, that has been decided to be included in a group of mutually exclusive facts and for every $o_\vdash$ where $v \in startCond(o_\vdash)$; automatically check if the associated control variable can also be included.*

This process generates a set of propositional multi-valued variables that describe the world, the initial state and the goal state of a given *snap* temporal task. For the Exploration domain example, Table 1 summarizes this set of multi-valued variables $v \in V$ for the MPT of the problem in Figure 1.

Additionally, as a consequence of expanding the original MPTs definition to eMPTs, we must propose a definition for the multi-valued numeric variables $n \in N$.

**Definition 7. Multi-Valued Numeric Variables**
*Within an eMPT with operators, $o \in O$, a **multi-valued numeric variable** $n \in N$ is defined by a set of states, $\epsilon$. This set contains one state for each numeric effect present in numContEff(o) in which the numeric variable n is affected, plus the undefined state and the current numeric value.*

Therefore, for each numeric variable, $n$, we expand all possible values for each numeric effect, which allows us to encode the last applied numeric operation over $n$. For reference, if we consider the *GoTo* operator from Figure 2, a multi-valued numeric variable, $n \in N$, would be created for (*battery ?r*), (*avail_battery ?r*), (*risk ?r*) and, from the duration, (*total-time*). Since *?r* can be instantiated to a UAV and a ROVER, two variables would be created for each except for *total-time*. Their possible values depend on the right-hand side of the numeric effect, which results to a single possible value for (*battery ?r*) variables: (*bat_move*), and one for each possible position for (*risk ?r*) variables: (*p_risk ?to*) with *?to* $\in [p1, p2, p3, p4, p5, p6, b1, b2]$.

Furthermore, most numeric operations from *numContEff* $(o)$ and *numCond*(o) can be pre-processed before the search is resolved, reducing the need for the planner to compute the on-line value of most $n \in N$ during search. To summarize, in our eMPTs, we track each fluent numeric variable by encoding both its present value and the most recent numeric effect applied to it.

Although this representation is sufficient to encode numeric information within operators and eMPTs, tracking only the most recently applied numeric effect is not enough to correctly handle concurrent numeric effects on the same numeric variable during the search phase. This limitation reflects a deliberate design choice aimed at simplifying the integration of eMPTs into the LAMA framework. We address this issue in the description of the search process and numeric framework in section 6.3.2.

Once all aspects of the temporal task have been encoded into the eMPT, MA-LAMA is ready to start making use of MA techniques to extract information and deal with the concurrent operators search space that the eMPT still represents. Therefore, in the next subsection we review MA-LAMA's Knowledge Compilation module, which takes as an input the full *snap* task eMPT.

## 6.2 Knowledge Compilation

Once the temporal task has been translated and encoded as an eMPT, the Knowledge Compilation module of MA-LAMA is initiated. This module embodies the core concept of MA-LAMA: addressing temporal complexity through MAP techniques. It achieves this by performing a static analysis that extracts MA information from the complete eMPT and transforms it to simplify the temporal complexities of the search module.

The Knowledge Compilation module consists of three key algorithms:

- **Agent Decomposition (AD)**: it automatically detects agents within the eMPT and creates individual sub-tasks for each agent, facilitating a more efficient search process.

**Algorithm 1** Agent Decomposition (AD)

**Input**: MAP temporal task, *eMPT* $\langle V, I, G, O, N, M \rangle$, where $V$ is the multi-valued variables set, $I$ is the initial state, $G$ is the goals set, $O$ is the operators set, $N$ is the multi-valued numeric variables set and $M$ is the metric.
**Output**: Agent Decomposed Task set $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$

**Automatic Agent Detection**
$CG$ generation
$\Omega \leftarrow \{v \in V : v$ root node of $CG \backslash 2$ way cycles$\}$
$\Omega \leftarrow$ AssembleAgents($\Omega$)
**Extend Private Agent Sets**
**repeat**
  **for** $\Omega_n \in \Omega$ **do**
    $\Omega_n \leftarrow \Omega_n \cup \{v \in V : v$ only successor of $\cup\ \Omega_n\}$
  **end for**
**until** $\Omega$ can no longer be refined
$O_n \leftarrow \{o \in O, v \in \Omega_n : \exists v \in \mathit{eff}(o) \cup \mathit{pre}(o)\}$
**Extend Public Agent Sets**
**repeat**
  **for** $\Omega_n \in \Omega$ **do**
    $\Omega_n \leftarrow \Omega_n \cup \{v \in V : v$ is connected with $\cup\ \Omega_n$ in the $CG \wedge v$ not yet assigned$\}$
  **end for**
**until** every $v \in V$ is assigned
$O_n \leftarrow \{o \in O : \mathit{eff}(o) \cup \mathit{pre}(o) \in \Omega_n \wedge o$ not assigned in previous step$\}$
$N_n \leftarrow \{n \in N : n \in \mathit{contEff}(\{o \in O_n\}) \cup \mathit{numPre}(\{o \in O_n\})\}$
$I_n \leftarrow \{v \in I : v \in V_n\}$
$M_n \leftarrow \{(w * n) \in M : n \in N_n\}$
**return** $\{\Omega_n, I_n, \emptyset, O_n, N_n, M_n\}$

- **Goal Categorization and Assignment (GCA)**: it categorizes goals into cooperation and coordination sub-groups. It then generates linked search processes that involve several agents in resolving these sub-groups, referred to as Parallel Steps.
- **Agent Interaction Analysis (AIA)**: it assesses whether the MA temporal task should be addressed using the *Parallel* or *Single Search* procedures. It utilizes DTGs and DNTGs to distinguish between tightly and loosely-coupled planning tasks.

The subsequent subsections will provide a detailed review of each of these algorithms and their roles within the Knowledge Compilation module.

#### *6.2.1 Agent Decomposition.*

The Agent Decomposition algorithm encapsulates the first set of MAP techniques that MA-LAMA makes use of. It aims to detect agents within the eMPT and then, using this information, create an individual sub-task for each agent from the original MA temporal task.

From the perspective of a single agent, variable decomposition algorithms typically group domain variables into two categories: (1) *private* variables, that inform about the agent internal state, and (2) *public* variables, that describe the shared environment where the agents operate. Assembled, all *private* and *public* variable sets encompass all variables from the original temporal task.

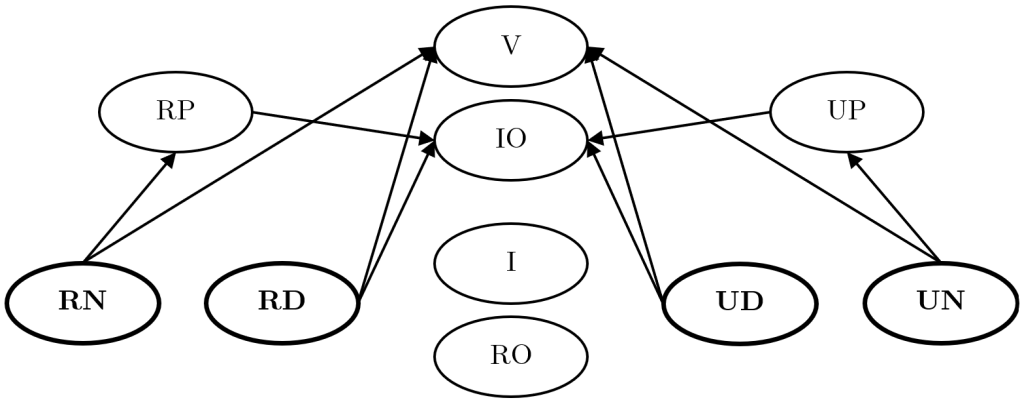



Fig. 6. *CG* with 2-way cycles derived from the original one of Figure 4. Each node represents a multi-valued variable from Table 1.

To define our task variables decomposition, we follow the *variable decomposition* definition from Crosby et al. 2013. Given a certain eMPT, a valid variable decomposition separates the full set of variables, $V$, into a set of variable groups, $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_n\}$, and a set of *public* variables, that can be empty. Crosby et al. 2013 also propose *Agent Variable Decompositions* (ADPs); which are variable decompositions without *joint operators*, which are shared between several agents, and where no internal variable for any agent can be modified by external operators. In our work, we only make use of the first definition, as the restrictions that the ADP introduces are detrimental to MA-LAMA's internal framework. We, instead, will not merge agents to avoid interactions between them and deal with *required cooperation* in a later point.

A pseudo-code overview of the MA-LAMA Agent Decomposition (AD) can be seen in Algorithm 1. The output for this module is a set of eMPTs $\langle V, I, G, O, N, M \rangle$, with all elements defined except $G$. The decomposition is achieved through mainly three stages: (1) the *Automatic Agent Detection*, that detects agents and divides the original task into sub-tasks, (2) the *Extend Private Agent Sets*, that identifies and creates sub-sets of all variables that are internal to each agent and (3) *Extend Public Agent Sets*, that completes the previous sub-sets with the remaining *public* variables. Next, these main stages are described, along with an AD execution walkthrough over the Exploration example scenario from Table 1.

The **Automatic Agent Detection** stage starts with the creation of the *CG* for the full eMPT. For the Exploration problem, the *CG* can be seen in Figure 4. The next step is the removal of 2-way cycles. 2-way cycles involve exactly two vertices connected by a pair of directed edges, one in each direction, meaning that the two nodes are mutually connected. After removing such cycles, nodes without incoming edges and with at least one remaining successor represent independent variables whose states influence other variables. We consider these "root nodes", and serve as the foundation for each acting entity.

The result of the cycles removal from the original Exploration *CG* can be seen in Figure 6, with these root nodes marked in bold: (*RN, RD, UD, UN*). Additionally Table 2 shows this root node selection in the first and second columns within the context of the whole AD algorithm. After removing the 2-way cycles from the *CG*, four distinct root nodes are identified. These nodes correspond to multi-valued variables that represent the *NavMode*, and *Docked* (or not) state of the UAV and ROVER.

Then, a process is carried out to check if some of the detected root nodes are in fact part of the same agent and, as such, should be assembled.

Table 2. Automatic AD process for the problem of Figure 1 and multi-valued variables from Table 1. Root nodes are identified from the *CG* after removing 2-way cycles, and agents are formed by grouping their private variables. Finally, these agents are expanded to include the complete variable set.

| Initial Variables | *CG* Root Nodes | Detected Agents | Extended Agents |
|---|---|---|---|
| **UP**: UAV Position<br>**UD**: UAV Docked<br>**UN**: UAV NavMode<br>**RP**: ROVER Position<br>**RD**: ROVER Docked<br>**RN**: ROVER NavMode<br>**I**: ImageTaken<br>**V**: VisitedPosition<br>**IO**: ImageTakingOngoing<br>**RO**: RechargeOngoing | UD<br>UN<br>RD<br>RN | [UD, UN] | [UD, UN, UP, I, V, IO, RO] |
| | | [RD, RN] | [RD, RN, RP, I, V, IO, RO] |

This *AssembleAgents* function is designed to only merge agents when it is absolutely essential, as we aim to produce as small and simple agents as possible. The agents assembly rule is the following:

*For two possible agents* $[v, v']$ *in a root node set* $\Omega$*, these are assembled if:*

- *there is a path between* $v$ *and* $v'$ *in the CG, or*
- $v$ *and* $v'$ *share all constant parameters in their root invariants.*

In the case of our Exploration example, the agent assembly rule identifies that all root node variables linked to the UAV or the ROVER are interconnected in the *CG*. This can be easily proved, as the *Docked* variable conditions the *NavMode* variable possible state changes. This leads to the grouping of these variables into two final agents: one for the UAV and another for the ROVER. Consequently, for both the UAV and the ROVER, an agent is created, as can be seen in the third column of Table 2.

Compared to ADP, this process results in a partial agent assembly and, therefore, a higher number of agents. As an illustrative example the DriverLog CodMAP domain is decomposed into "trucks + drivers" agents in ADP, and for MA-LAMA, single "drivers" agents are produced.

As the agents have been detected and refined, the main objective of the **Extend Private Agent Sets** stage is to create each acting entity *public* variables, operators and numeric fluents sets. The variable sets, $\Omega_n$, are conformed so that every $v \in V$ are added only if $v$ is a unique successor to the current $\Omega_n$ set. This process is iterative and only stops once no more variables can be added to any agent. Then, a set of *private* operators can be created for each agent by, for every action $o \in O$, including to the private set, $O_n$, only if there is a private variable $v \in V$ that exists in $\mathit{eff}(o)$ or $\mathit{pre}(o)$.

Subsequently, the variables, $\Omega_n$, and operators $O_n$ *private* sets for each agent are produced and, all remaining variables and actions are included in the *public* set.

Finally, the **Extend Public Agent Sets** stage completes all agent sets. First, all variables reachable by an agent in the *CG* are added to $\Omega_n$. And then operators $O$, numeric variables $N$, the initial state $I$ and the metric $M$ can all be decomposed and added to a certain agent $i$:

- For $O_i$, all operators are included following the same past rule, but using the new complete variable set $\Omega_i$.
- For $N_i$, numeric variables are added if they appear in *numContEff(o)* for all operators $o \in O_i$.
- For $I_i$, the initial state only covers $v \in \Omega_i$.
- For $M_i$, each metric element $\{w * n\}$ is included to $M_i$ if $n$ appears in *numContEff(o)* or *numCond(o)* for all actions $o \in O_i$.

In our Exploration example, the variables representing each agent's position, (*UP, RP*), remain unassigned and are *private*. Consequently, these variables are assigned exclusively to their corresponding agent. For *ImageTakingOngoing (IO)* and *RechargeOngoing (RO)*, note that we represent them as one node in the CG (Figure 4), but there are several of each in the real eMPT (Table 1). Therefore, each is assigned individually as *private* to their corresponding agent.

Finally, any variables that were not yet assigned to an agent, *ImageTaken (I)* and *VisitedPosition (V)*, are added to the *public* variable sets of the two agents, as both the UAV and ROVER can act on these variables. This can be seen in the fourth column of Table 2. Consequently, the initial state, operators, numeric variables, and metric are assigned based on whether they relate to the UAV or the ROVER.

A set of eMPTs, $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$, is the final output for the algorithm. One eMPT is created for each agent with all components conforming with the exception of goals, $G$. The next step, the Goal Categorization and Assignment algorithm, receives this set and $G$ as input.

Finally, we want to establish our algorithm's soundness and completeness, which is necessary to guarantee a reliable task decomposition.

**Theorem 1** *The Agent Decomposition algorithm described in Algorithm 1 for eMPTs is both sound and complete*

**Sketch of Proof** We base our proof on Crosby et al. 2013's Theorem 6.1, as our decomposition definitions are aligned, and our algorithms have similar main objectives. There are three main cases that need to be covered to assure soundness and completeness:

- During the *Automatic Agent Detection* stage, if identified as agents variables are also root nodes in the causal graph, a decomposition is certain to be found.
- In case a possible variable decomposition can be found, the *Automatic Agent Detection* stage always identifies at least two non-mergeable root variables.
- For the last statement, our algorithm does not include any "merge restrictions" from Crosby et al.'s 2013 ADP. Therefore, soundness is not affected by agents with links in the *CG* that have not been merged.

#### *6.2.2 Goal Categorization and Assignment.*

The GCA algorithm aims to create linked search processes that agents can solve concurrently. The Required Cooperation (RC) Analysis from Zhang et al. 2016 serves as the foundation for categorizing the goals of the full eMPT. Following their work, we say that a given eMPT presents RC if it is not necessarily 1-agent solvable. Additionally, Zhang et al. formally describes the possible interactions among agents within a planning task:

- *Type-1 RC with Heterogeneous Agents*. These arise due to specific agent characteristics or capabilities. For example, scenarios involving robotic agents equipped with different specialized tools would typically exhibit this type of interaction.
- *Type-2 RC with Homogeneous Agents*. RC interactions that may arise in two distinct situations:

(1) **Traversability interactions:** These interactions occur when agents encounter restrictions that prevent straightforward traversal of their respective causal graphs (CGs). Typical examples include resource conflicts, such as multiple agents competing for a shared or occupied location.

(2) **Causal Loops interactions:** These interactions arise when agents must perform certain actions in a logically ordered sequence to achieve collective goals. This type of interaction frequently appears in homogeneous-agent scenarios requiring coordinated concurrency.

Therefore, MA-LAMA categorizes goals in the following manner:

- *Cooperation* goals: require Type-2 RC Traversability interactions, or no RC at all, meaning the goal is theoretically attainable by multiple agents operating independently.

**Algorithm 2** Goal Categorization and Assignment (GCA)

**Input**: Agent Task Set, $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$, and goals, $G$
**Output**: Cooperation and Coordination Parallel Steps, $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$

**Find Coordination Points**
$CoorP \leftarrow \emptyset$ (Coordination Points)
**for** $\{\Phi_n, \Phi_m\} \in \binom{\Phi}{2}$ **do**
  $CoorP \leftarrow CoorP \cup \{v \in P : \exists(o_n \in O_n, o_m \in O_m) : v \in pre(o_n) \land v \in eff(o_m)\}$
**end for**
**Single Goal Relaxed Plans Obtention**
$G_{coop}, G_{coord} \leftarrow \emptyset$ (Coord and Coop goal sets)
**for** $g \in G$ **do**
  $Sol_g \leftarrow \emptyset$ (relaxed solutions set)
  **for** $\Phi_n \in \Phi$ **do**
    $Sol_g \leftarrow Sol_g \cup$ **relaxedSearch**$(\Phi_n, g)$
  **end for**
  **if** $Sol_g \neq \emptyset$ **then**
    $G_{coop} \leftarrow G_{coop} \cup \{\langle g, Sol_g \rangle\}$
  **else**
    $Sol_g \leftarrow$ **relaxedSearch**$(\Phi, g)$
    $Sol_g \leftarrow p \in CoorP \mid p \in Sol_g$
    $G_{coord} \leftarrow G_{coord} \cup g, Sol_g$
  **end if**
**end for**
**Goal Assignment and Parallel Steps Creation**
$\sigma_{coop} \leftarrow G_{coop}, MinCostAssignment(\Phi, G_{coop})$
$\sigma_{coord} \leftarrow \emptyset$
**for** $g_{coord} \in G_{coord}$ **do**
  $\sigma_n \leftarrow g_{coord} \cup \{p \in CoorP \mid p \in g_{coord}.Sol_g, A \in \Phi_n \mid A \in g_{coord}.Sol_g\}$
  $\sigma_{coord} \leftarrow \sigma_{coord} \cup \sigma_n$
**end for**
**return** $\sigma_{coop} \cup \sigma_{coord}$

- *Coordination* goals: necessitate Type-1 RC (Heterogeneous Agents) or Type-2 RC (Homogeneous Agents) Causal Loop interactions, indicating that several agents must interact to accomplish the goal.

The set of linked search processes, referred to as *Parallel Steps*, is the output of the algorithm, with each phase focused on solving a subset of the same type of goals $\{g\} \in G$.

We provide a formal description of the GCA algorithm, outlining its purpose and structure (Algorithm 2), which we divide into three distinct stages, each serving a specific purpose in the overall process of goal assignment and resolution:

- **Find Coordination Points:** this stage determines whether each goal from the full task $g \in G$ requires mandatory interactions between agents for its resolution. We extract the *public* variables that could serve as *coordination points* in our eMPTs task set $\Phi$, specifically those that are preconditions for one agent's actions and effects in another's. For the Exploration example, no states qualify as coordination points.

Table 3. Metric cost estimation results for each goal defined in Figure 1 for the UAV and ROVER agents, identified during the AD process.

| Goal | UAV Agent Cost | ROVER Agent Cost |
|---|---|---|
| **G1**: ImageTaken(p1) | 1.06 | 1.51 |
| **G2**: ImageTaken(p3) | 0.57 | 0.94 |
| **G3**: ImageTaken(p4) | 0.40 | 0.79 |
| **G4**: UAV Docked | 0 | - |
| **G5**: ROVER Docked | - | 0 |
| **Final Assignment** | [G1, G3, G4] | [G2, G5] |

This is due to the fact that the only candidates are variables within the agents' *public* sets, and neither *ImageTaken* nor *VisitedPosition* variables act as preconditions for any operator.

- **Single Goal Relaxed Plans Obtention:** in this stage, a relaxed search is launched in which each agent eMPT $\Phi_n \in \Phi$ attempts to solve each goal $g \in G$ while ignoring delete effects and all numeric conditions *numCond()*. For all these search processes, we determine the limits of each numeric variable and obtain the metric cost of the final relaxed plan by taking the worst-case cost of each numeric effect *numContEff()*.
  From this point on, goals are classified following two rules:
  – All goals that are achievable by a single agent relaxed search are considered *cooperation goals*, and MA-LAMA uses *anytime search* so that all agents that can achieve a certain goal find the most optimized relaxed plan through iterative relaxed searches.
  – In case no single agent can solve the relaxed task, then MA-LAMA considers $g$ a *coordination* goal. For this goal type, a full task eMPT relaxed search is launched that tries to determine the minimum number of agents needed for its resolution. At the same time, the variables that have been identified as possible *coordination points* are monitored and its values during agents interactions stored. These are used in the next stage to create linked search processes that are the base for the search in MA-LAMA.

  No variables are classified as *coordination points* in the Exploration problem. Therefore, all goals are directly categorized as *cooperation* goals. Following this, the algorithm proceeds with the relaxed cost-informed search for each agent-goal pair to obtain plan quality estimates. In our domain, the goals include three *ImageTaken* predicates and the requirement for both agents to end the plan *Docked* at the base.
  Table 3 shows the cost estimation results for each goal-agent pair. Each cell shows the estimated cost for the respective agent to accomplish the given goal following the metric from Figure 3, with lower values indicating higher efficiency. Goals that an agent cannot accomplish are marked with a dash ("-").
  As expected, both agents, the UAV and the ROVER, can independently achieve the *ImageTaken* goals. However, the UAV agent demonstrates more favorable plan quality, meaning it incurs a lower cost to achieve some of these goals compared to the ROVER. The final row of the table depicts the final goal assignment, which is covered in the next stage.
- **Goal Assignment and Parallel Steps Creation:** this stage aims to create the *Parallel Steps* and decompose the planning problem in a way that enhances the potential for metric cost optimization during the search process. To achieve this, MA-LAMA addresses *coordination* and *cooperation* goals differently.
  – For *cooperation* goals, a *Parallel Step*, $\sigma_1$, with all goals is created in a manner that optimizes the balance between overall metric optimization and the cooperativeness of the solution. This balance is achieved thanks to a goal allocation heuristic. It prioritizes agents with fewer goals, assigning them the next most cost-effective goal. This *cooperation Parallel Step* includes a set of goals, $G_{coop}$, and only those agents, $\Phi_n \in \Phi$, that have at least one goal assigned.

– For *coordination* goals, a *Parallel Step*, $\{\sigma_2, ..., \sigma_m\}$, is created for each single goal, $g_{coord} \in G_{coord}$, including agents that appear in the relaxed plan, $\Phi_n \in \Phi$, and a set of *coordination points* specific to each involved agent, $CoorP$.

The final row of Table 3 shows the optimized assignment of goals for the Exploration scenario, balancing individual costs and cooperative efficiency. A single *cooperation Parallel Step* is then created including two eMPTs, each one targeting the local goals associated with either the UAV or the ROVER.

Next, we want to establish whether our proposed algorithm is sound and complete, which is crucial for ensuring the reliability and effectiveness of the goal assignment.

**Theorem 2** *The algorithm presented in this subsection for goal assignment among a set of agent tasks is both sound and complete.*

**Sketch of Proof** We demonstrate that, given any valid eMPT set, we can derive a valid and solvable goal assignment:

- Initially, if decomposition is not achieved, all goals will be solved by the full task.
- Secondly, when a goal is reachable by the entire task, it is either allocated to a single capable agent based on initial relaxed searches utilizing individual agent operators or designated as a *coordination goal* assigned to multiple involved agents through a relaxed search of the full task.
- Lastly, it is shown that for *coordination* goals, the *coordination points* effectively address any remaining scenarios in the AD by integrating sets of agents linked in the $CG$.

Before proceeding, we verify the validity of the assignments from a numeric conditions perspective. If a relaxed solution is unattainable for any agent in a *Parallel Step*, we restart the process by assigning weights to each agent's metric estimation, thereby reducing reliance on constrained agents.

Once the full set of *Parallel Steps* has been completed, $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$, the *Parallel Search* can proceed. However, MA-LAMA introduces an additional step with its *Agent Interaction Analysis* algorithm, which is responsible for estimating whether the planning task should be solved with *Single* or *Parallel Search*. We cover the details of this module in the next subsection.

#### 6.2.3 *Agent Interaction Analysis.*

The *Agent Interaction Analysis* (AIA) is an integral part of the MA-LAMA framework, designed to choose between two distinct paths during the search process. After the successful execution of the Agent Decomposition (AD) and Goal Categorization and Assignment (GCA) algorithms, MA-LAMA can decide whether to solve the entire eMPT task through a *Single* or *Parallel Search*. Our aim is to estimate which option is more promising in terms of the effort required to find a solution and the optimization of the metrics.

Existing decomposition approaches, such as ADP [13], demonstrate varying performance depending on the type of domain tested. In their experiments, they compare their approach with LAMA, showing that their decomposition can be beneficial in terms of search time and expanded nodes for domains like Zenotravel, Rovers, Transport, or Satellites. However, they do not improve LAMA's performance in domains such as Depot, Driverlog, or Elevators. Since our AD algorithm follows similar principles to ADP, we intend to leverage this knowledge to choose between solving the full temporal task in a decomposed or assembled manner.

Additionally, the MA temporal planner TFPOP [30], which was introduced in section 2, has shown that loosely-coupled MA temporal domains can effectively be solved using a decomposition-based approach. With this insight and the results from ADP, we aim to differentiate between tightly and loosely-coupled domains using the AIA. This differentiation allows us to apply decomposition techniques for loosely-coupled MAP temporal tasks while employing the *Single Search* approach for tightly coupled ones. This strategy also creates notable interactions between the LAMA and MA-LAMA base heuristics and the task decomposition.

To achieve this, AIA identifies the interactions between agents produced by the AD algorithm. This analysis is based on the morphology of each agent's Domain Transition Graphs (DTGs) and our newly introduced Domain Numeric Transition Graphs (DNTGs). We previously explained that the numeric variable values, $n \in N$, from eMPTs do not encode states; rather, they represent the last numeric effect applied along with the current numeric value. Consequently, DNTGs cannot be constructed using the same procedure as DTGs. Instead, for a numeric variable $n$, we utilize the DTGs' propositional variable states to build the DNTG.

**Definition 8. Domain Numeric Transition Graphs (DNTGs)**
*For an eMPT $\Phi$ with numeric variables $n \in N$ and a set of propositional variables $v \in V$, the **Domain Numeric Transition Graph** for n, denoted as DNTG(n), consists of a single or set of labeled digraphs. Each digraph has a vertex set $D_{\{v\}}$ and contains an arc $(d, d')$ if there exists an operator that would create an arc in DTG({v}) and the operator has a numeric effect acting on n, numContEff$(n)$.*

Thus, our DNTGs encode the evolution of numeric variables across the states of propositional variables. For a given DNTG($n$), multiple digraphs may exist, each representing the evolution of $n$ across various $v \in V$. These DNTGs enable us to analyze the nature of each agent generated by the AD algorithm based on their morphological characteristics. More specifically, for two agents to be considered of the same type, they must share the following characteristics:

- The **number of single graphs** in the assembled DNTG for the numeric variables that constitute the complete eMPT metric. It informs about how agents impact the metric. For the Exploration domain example, this is the battery usage and the risk of the mission, from Figure 3. The metric DNTGs for the ROVER and the UAV can be seen in Figure 7 and are identical, since the cost transitions do not affect the presence of links between nodes. Instead, the DNTGs for each agent indicate which value transitions within each propositional variable also affect the battery usage or the mission risk. Both metric DNTGs show one single graph for the *Docked* variable and another for the *Position* variable. The nodes represent possible values that these variables can have, and edges the possible transitions between them. Non-labeled nodes are on-going operators states. Therefore, the number of single graphs for the metric DNTG is two.
- The **number of single graphs** in the DNTG for the *total-time* numeric variable, that encapsulates how each agent evolves over time. In the Exploration scenario, the same considerations stated about the metric DNTG also apply in this case. The UAV and ROVER's *total-time* DNTGs, represented in Figure 8, show one single graph for the *NavMode* variable, one for the *Docked* variable, and another for the *Position* variable. Therefore, the number of single graphs for the *total-time* DNTG is three.
- The **number of root nodes** in the *CG* with 2-way cycles removed. In the Exploration problem, these are obtained from the process depicted in Table 2, where six root nodes were found, out of which three of them correspond to each agent.

While there are other options for analyzing each DNTG's morphology, we have chosen these specific characteristics due to their simplicity in computation, which minimizes any additional runtime for MA-LAMA.

Once the agents have been categorized into types, the AIA utilizes the *coordination points* obtained from the GCA algorithm. A straightforward rule is then applied to arrive at the final decision during this step:

*All eMPTs with a valid decomposition are solved with Parallel Search, unless more than one agent from two different types presents coordination points between them.*

In the Exploration domain case, during the GCA algorithm execution, it was already determined that no *coordination points* exist between the ROVER and UAV.

The complete AIA results for the Exploration domain example are depicted in Table 4. These indicate that both the UAV and ROVER agents share the same number of root nodes in the *CG* (after removing 2-way cycles).

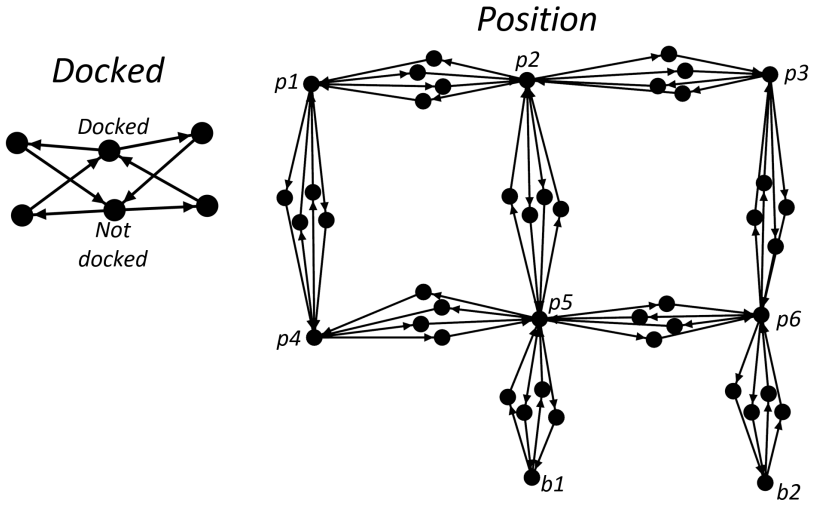



Fig. 7. Metric (risk + battery) Domain Numeric Transition Graph (DNTG) for the UAV and ROVER agents in the MA temporal Exploration problem. The DNTG depicts the variables whose state transitions impact the numeric values of risk and battery.

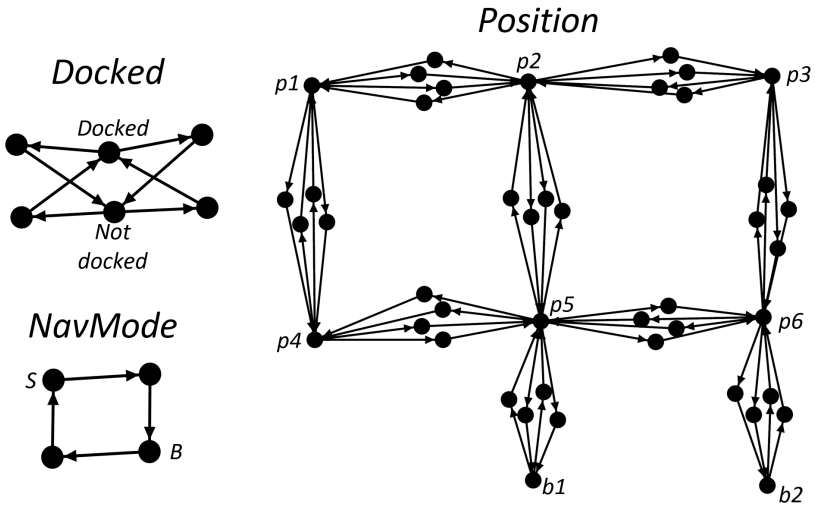



Fig. 8. *Total-time* Domain Numeric Transition Graph (DNTG) for the UAV and ROVER agents defined in the problem of Figure 1. The DNTG depicts the variables whose state transitions impact the time evolution.

They also present the same number of single graphs for the metric and *total-time* DNTGs, suggesting that the evolution of the metric and *total-time* numeric variables follows a similar pattern. Finally, no *coordination points*

Table 4. AIA results for the MA temporal Exploration problem. The UAV and ROVER agents share the same results for the *CG* and DNTG morphology analysis and do not present any *coordination point*. This classifies them as the same type, labeled as "1".

| | **UAV Agent** | **ROVER Agent** |
|---|---|---|
| Root nodes in 2-way *CG* | 3 | 3 |
| Metric DNTG single graphs | 2 | 2 |
| Total-time DNTG single graphs | 3 | 3 |
| Coordination Points | None | None |
| **Final Agent Type** | 1 | 1 |

were detected for the Exploration problem. These results indicate that only one type of agent can be considered for the search process, proceeding with the *Parallel Search* procedure.

Through the AIA, MA-LAMA prioritizes the search process established by the AD and GCA algorithms. However, if a tightly-coupled scenario is detected, it resorts to *Single Search*, which may yield beneficial interactions that will be explored further in section 6.5.

## 6.3 Search

MA-LAMA prioritizes the application of Agent Decomposition (AD) and Goal Categorization and Assignment (GCA) techniques, focusing on *Parallel Search*. This section discusses the search process that follows the execution of the AD and GCA algorithms.

The search in MA-LAMA accepts two types of input: a set of *Parallel Steps*, denoted as $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$, or a full task eMPT, ($eMPT\langle V, I, G, O, N, M\rangle$). Since these options require different management approaches, we will first outline the shared aspects of the search process before delving into the details in the subsequent subsections.

### *6.3.1 Search Process Overview.*

In all cases, search processes are initiated over eMPTs. MA-LAMA employs a *WA** forward total-order search utilizing two classical heuristics presented in section 3: the Cost-Sensitive FF/add heuristic, $h_{FF}$, and the $h_{Land}$ heuristic. Our approach is based on the observation that the key elements of temporal complexity in most MA temporal domains revolve around the principles of *cooperation* and *coordination*. Consequently, eMPTs derived from tasks decomposed by agents typically do not require complex simultaneity.

Throughout the search process, all states are evaluated using both heuristics. The decision on which state to expand next is made by alternating between the two heuristics, guided by numerical priorities. Additionally, drawing from practices in the LAMA planner, we utilize *preferred operators*, which are operators identified as likely to be effective in a specific state.

### *6.3.2 Temporal and Numeric Framework.*

A crucial aspect of these search processes is that no temporal reasoning is performed or included in the heuristic values generated by the classical heuristics. This aligns with the initial design concept of MA-LAMA, which aims to compensate for the lack of temporal information during the search with a MAP preprocessing module. However, we still need to incorporate modifications to the main LAMA search core framework to ensure that *snap* eMPTs can be solved efficiently and soundly:

- We have implemented a basic **temporal framework** designed to manage local concurrency in eMPTs. This framework integrates *temporal constraints* between start and end *snap-operators*, $o_\vdash$ and $o_\dashv$, ensuring that preconditions for new actions are met in the current frontier state. Following the formalisms of temporal expressiveness outlined by Cushing et al. (2007), MA-LAMA is capable of solving temporal

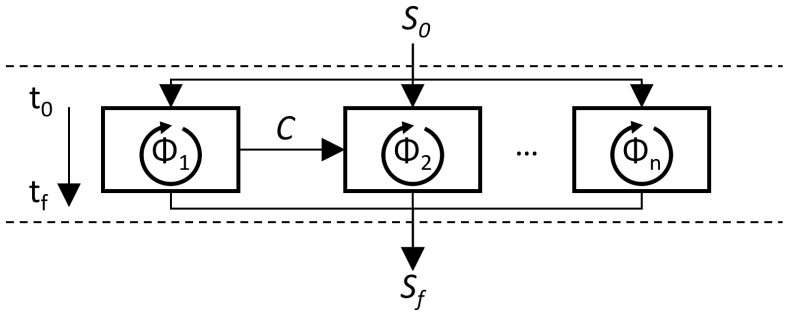



Fig. 9. Structure of a *cooperation Parallel Step* featuring iterative local searches. Agents, $\Phi_i$, derive *temporal constraints* from the previous agent. The initial state, $S_0$, and the final state, $S_f$, are collaboratively generated by all agents. The phase's total time, $t_f - t_0$, represents the duration of the longest partial plan.

planning problems encoded in a $L^{s,e}_{s,e,o}$ language following a simple Decision Epoch architecture. Its primary enhancement lies in its ability to identify specific operators that become available exclusively between a designated snap-operator pair, $o_{\vdash}$ and $o_{\dashv}$. However, as noted by Cushing et al 2007, this design renders MA-LAMA's individual search processes incomplete for temporally expressive tasks. Furthermore, MA-LAMA does not incorporate explicit temporal reasoning to determine *when* operators should be applied. These design choices prioritize search simplicity, relying on prior MAP techniques to manage temporal complexity effectively.

- Similarly, the **numeric framework** design includes the necessary components to manage continuous numeric operations and address numeric preconditions. A specific remark is required regarding the handling of concurrent numeric effects during the search. As stated in the Translation module, for each numeric variable $n \in N$ within an eMPT, we explicitly track only the current numeric value and the last numeric effect applied. However, this approach alone would incorrectly handle overlapping numeric effects on the same numeric variable, as newly applied numeric effects would overwrite previous operations. To resolve this, our numeric framework explicitly accounts for concurrent numeric effects by tracking the duration of each operator applying these effects, thus correctly managing multiple numeric effects simultaneously active within a given state.

It is important to note that our primary goal with these frameworks is to ensure temporal and numeric soundness. No numeric or temporal information is collected or utilized in the *Single* or *Parallel Search* processes.

#### *6.3.3 Parallel Search.*

After the AD and GCA algorithms are successfully completed, and the AIA estimates that it is advantageous to solve the full eMPT using *Parallel Steps*, the *Parallel Search* begins. Having previously described the heuristic search methods employed by MA-LAMA, we will now focus on how to effectively solve the *Parallel Steps*, which represent the interactions between agents within the original temporal task.

The search process starts by taking the complete array of *Parallel Steps*, $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_m\}$, each containing a set of eMPTs, $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$. For each eMPT, we initiate a multi-heuristic search that incorporates *temporal constraints*, represented as $t = \langle \Phi_n, v, t, d \rangle$, established by *public* variables. This structured search process ensures that agents' *cooperation* or *coordination* interactions are considered only at specific points in time during the search, thereby reducing the complexity and runtime of the overall process.

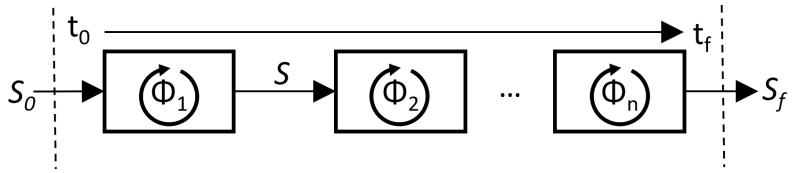



Fig. 10. Diagram of a *coordination Parallel Step* structure involving iterative local searches. Agents, $\Phi_i$, adopt the previous initial state, $S_0$, while the final state, $S_f$, is produced by the last agent. The total time for the phase, $t_f - t_0$, is calculated as the sum of all agents' local plans.

**Definition 9. Temporal Constraints**
*A **Temporal Constraint** is a 4-tuple,* $(\Phi_n, v, t, d)$*, where:*

- $\Phi_n$ *represents the affected agent,*
- $v$ *denotes a specific value that must be maintained,*
- $t$ *indicates the moment when the constraint starts, and*
- $d$ *signifies a certain duration.*

All *Parallel Steps* utilize the *temporal constraints* system, albeit for different purposes:

- For *cooperation* goals, the aim is to ensure that restrictions on *public* variables are preserved.
- For *coordination* goals, the focus is on ensuring synchronization among agents around the *coordination points*.

Consequently, *temporal constraints* are computed as state specifications or conditions for *cooperation Parallel Steps*. These conditions are encoded as $inv() = v$, starting at time $t$, for a duration of $d$, and are defined and set each time a *cooperation* agent identifies a partial solution. This list of constraints is subsequently used by agents to limit their local search graphs, and they add their own restrictions as they identify their partial solutions. This entire scheme is summarized in Figure 9.

This approach can potentially lead to dead-ends when one agent imposes *temporal constraints* on another agent, making their assigned goals unachievable. When this condition is identified, the search process for the entire *Parallel Step* is restarted, modifying the order between the constrained agent and the one that sets the constraints. Since eMPTs are relatively simple and straightforward to solve, resetting the search process for a *Parallel Step* does not consume a significant amount of time, as we will discuss in our experiments section.

In *coordination Parallel Steps*, the *coordination points* dictate the order followed by agents to obtain their respective partial plans. The *coordination* goal is achieved only with the resolution of the last agent's eMPT. The *temporal constraints*, reflecting the *coordination points* and end states achieved by previous agents, are passed down to the next agent and serve as the starting point for each local search. The structure of the *coordination Parallel Steps* is summarized in Figure 10.

#### *6.3.4 Metric Optimization in Parallel Steps.*

The approach to metric optimization varies across different types of *Parallel Steps*:

- In *cooperation Parallel Steps*, optimization relies on the quality of individual eMPT resolutions. Thus, agents with more goals are prioritized, enabling them to tackle their eMPTs with fewer *temporal constraints*, thereby enhancing overall optimization. Each individual eMPT is refined through *anytime* iterative searches, striving for optimality.

- In *coordination Parallel Steps*, although there may be limited potential for numeric optimization since only one task goal is solved, temporal concurrency can still be improved. This will be addressed further in section 6.4. Nonetheless, we aim for the optimal resolution of each local search within the *Parallel Step* through *anytime* iterative searches.

As observed, MA-LAMA incorporates another key feature from LAMA: *anytime search*. This allows MA-LAMA to continue searching for better solutions in each individual eMPT until the search space is fully explored. This process works in conjunction with cost-aware heuristics, enabling us to strive for optimal resolutions of eMPTs.

Each *Parallel Step* is deemed solved once every eMPT is resolved, producing a partial plan for each agent, $\Phi_i \in \sigma_m$, based on *snap-operators*. The final step in MA-LAMA, Unify, involves assembling all partial *snap* plans into a complete temporal plan which will be described in the next subsection.

#### 6.3.5 *Parallel Search for the Exploration Scenario.*

As decided by the AIA, MA-LAMA solves the Exploration problem through the *Parallel Search* process. In this case, the procedure only involves one *cooperation Parallel Step*. We will not detail every agent eMPT resolution partial plan that is generated, as it is not relevant for the *Parallel Steps* resolution. This *cooperation Parallel Step* involves two agents, the UAV and the ROVER. Therefore, two local eMPTs will be solved. The resolution process is the following:

(1) **UAV goals resolution**. Following Table 3, the UAV agent has more goals assigned than the ROVER, therefore, its eMPT is solved with priority. The *snap-operators* partial plan that solves *G1: ImageTaken(p1)*, *G3: ImageTaken(p4)* and *G4: UAV Docked* is produced. Additionally, the timed values of the shared *public* set variables, *VisitedPosition* and *ImageTaken*, are also passed on to the next agent as *temporal constraints*.
(2) **ROVER goals resolution**. The ROVER agent inherits the *temporal constraints* that limit its search graph at certain times. In this case, they do not restrict the search process as the variables *VisitedPosition* and *ImageTaken* are not preconditions for any ROVER operators. Therefore, the *snap-operators* partial plan is produced solving *G2: ImageTaken(p3)* and *G5: ROVER Docked*.

### 6.4 Unify

The final module in the execution of MA-LAMA involves consolidating all *snap* partial plans generated from each *Parallel Step* to produce a comprehensive temporal plan. The structure of this process is illustrated in Figure 11.

After completing the **Parallel Search**, unifying each agent's partial plan within a *Parallel Step*, $\Phi_n \in \sigma_m$, is a straightforward task, as concurrency and constraints have been managed across all scenarios, except for those involving *coordination Parallel Steps*. The complete *Parallel Step* partial plan, $\Phi_n$, is compiled by integrating each agent's local plan concurrently in *cooperation* scenarios and sequentially in *coordination* scenarios.

This integration process is particularly crucial for *coordination Parallel Steps*, as it allows for additional concurrency among agents. For a specific pair of partial plans, $\Phi_i, \Phi_j \in \sigma$, which both belong to the set solving a given *Parallel Step*, they can be combined concurrently if their respective *public* sets satisfy the condition $P_i \cap P_j = \emptyset$, meaning they do not affect the same *public* variables, and they also do not have a shared past *coordination point*. We perform this check exclusively in *coordination Parallel Steps*, as their *coordination points* have already been computed, and this condition predominantly holds between agents of this type.

A similar procedure is implemented to combine all partial plans from the *Parallel Steps*. The *public* variables of a pair of *Parallel Steps*, $\sigma_i$ and $\sigma_j$, are assessed so that if $P_i \cap P_j = \emptyset$, then both $\sigma_i$ and $\sigma_j$ can be concatenated into the overall plan concurrently. Any remaining partial plans that do not meet this criterion are merged sequentially.

During the previous *Parallel Search* process for the Exploration domain example, two eMPTs were solved, resulting in the generation of two *snap-operator* partial plans. Therefore, their respective partial plans can be

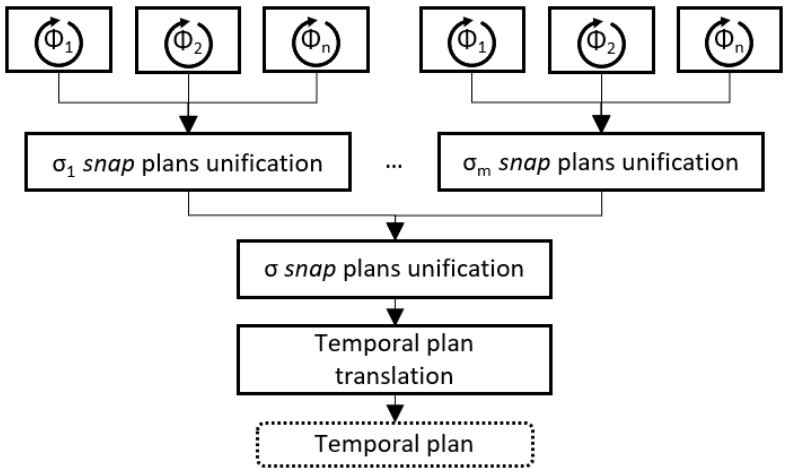



Fig. 11. Structure of the Unify module. Each eMPT, $\Phi_i$, within each *Parallel Step*, $\sigma_j$, generates a *snap* partial plan. These plans are first unified within each *Parallel Step*, and then among the entire set of *Parallel Steps*, $\sigma$, leading to the creation of a complete *snap* plan. This *snap* plan is then translated into the temporal paradigm, resulting in a complete temporal plan.

assembled concurrently since the *temporal constraints* have already been managed during the *Parallel Search*. This produces a *snap* complete plan that solves all *G1-G5* goals.

The final steps are equal for both *Single* and *Parallel Search* processes. They involve calculating the total cost of the plan, verifying the preservation of temporal, numeric, and logical constraints, and converting the *snap* operators back to the temporal planning paradigm. This translation from *snap* operators to temporal operators is straightforward and requires only changing the designation of the start operator, $o_{\dashv}$, and removing the end operator, $o_{\vdash}$, from the complete plan.

Once these processes are completed, the full temporal plan is produced, which serves as the primary output of MA-LAMA. For the Exploration problem, no further considerations are needed and the plans for the ROVER and UAV can be executed in parallel.

Before proceeding to the experimental evaluation, we will address important considerations regarding the *Single Search* process that MA-LAMA undertakes when the AD fails or when the Agent Interaction Analysis (AIA) assesses the decomposition as unpromising. Our focus will be on discussing the interactions between a heuristic based on *landmarks* and the decomposition of a MAP task.

## 6.5 Single Search Strategies: Heuristics and Landmarks

This subsection details the search process conducted by MA-LAMA for eMPTs when the Agent Decomposition (AD) fails to identify a valid agent decomposition, or when the Agent Task Selection (AIA) algorithm determines that the MA temporal eMPT has greater potential when solved as an assembled task. Although this type of search is not the primary focus of MA-LAMA, significant interactions with the landmark heuristic, $h_{Land}$, are worth noting. These interactions provide further insight into the design decisions made within the AIA algorithm.

### *6.5.1 Overview of the Single Search.*

For the *Single Search*, MA-LAMA initiates a process similar to that applied to single-agent eMPTs, as described in section 6.3.3. The heuristics $h_{Land}$ and $h_{FF}$ are utilized alongside the same numeric and temporal frameworks,

ensuring soundness is preserved across the concurrent search space of *snap* operators. The *temporal constraints* system is not needed in this case, as agent interactions are already handled during the search.

The *Single Search* also employs LAMA's *anytime* iterative search, meaning that the output will be the best plan found until the search space is exhausted or until the allowed search time is reached. This plan is based on *snap-operators* and addresses all goals from the entire eMPT.

Initially, our search approach was based on the premise that eMPTs derived from decomposed tasks are relatively straightforward and can be optimally solved with minimal effort. However, this assumption does not hold in scenarios where the AD fails or the AIA predicts a high degree of inter-agent dependency, indicating a tightly coupled domain. In such tightly coupled cases, classical heuristics can be sufficiently effective in solving the complete eMPT. As we will demonstrate in section 7, they may even outperform *Parallel Search* methods in certain instances. Thus, the current section aims to explain why specific MAP temporal tasks can be efficiently solved using classical heuristic search, even in the absence of temporal reasoning.

#### 6.5.2 *Heuristic Context.*

To understand the interactions between MA scenarios and classical heuristics, we must first provide context on the $h_{FF}$ and $h_{Land}$ heuristics.

The $h_{FF}$ heuristic is derived from a relaxation of the planning graph that ignores delete effects, allowing variables to simultaneously hold multiple values in eMPTs. Consequently, the performance of this heuristic is directly tied to the complexity of the original planning graph. Given that the propositional information introduced by inter-agent interactions only adds to the overall complexity of the planning task, the efficiency of the heuristic is expected to remain consistent across both decomposed and non-decomposed eMPT searches. We will not delve deeper into the framework of this heuristic since its value within MA-LAMA lies in its joint application alongside the $h_{Land}$ heuristic.

The $h_{Land}$ heuristic, known as the *landmarks* heuristic, is constructed around propositional formulas that must hold true at specific points during plan execution to satisfy the overall planning task goals. In the LAMA framework, and by extension in MA-LAMA, the heuristic value of a particular state is determined by the number of remaining *landmarks* that have yet to be achieved.

#### 6.5.3 *Landmarks and Task Decomposition.*

To understand the relationship between eMPT decomposition and *landmarks*, $\Psi$, we first need to clarify how landmarks are identified. Within the LAMA framework, *landmarks* are established by backtracking from the complete eMPT goals through three distinct rules for each specific *landmark*:

(1) **Shared preconditions** from the effects that can potentially first achieve $\Psi$.
(2) **Domain Transition Graph** nodes that are predecessors of $\Psi$ in cases where it is a multi-valued variable.
(3) Attempting to solve a **restricted relaxed planning graph** that lacks all possible $\Psi$ achievers, thereby obtaining all *landmarks* that can only be achieved after $\Psi$.

When applying these rules to the *coordination points* calculated in the initial phase of the GCA algorithm, it becomes clear that they meet all the criteria for classification as *landmarks*. This can be demonstrated for each rule: by definition, a *coordination point* signifies necessary interactions among agents required to achieve a goal, making it a prerequisite for other goals or *landmarks*. Specifically, *coordination points* represents necessary preconditions for other goals or landmarks (rule one), will enforce mandatory transitions between DTG nodes for multi-valued variables spanning multiple agents (rule two), and constrain the achievement of associated goals if the operators facilitating them are eliminated (rule three).

Consequently, agent task decomposition leads to a fragmentation of the propositional information available to the $h_{Land}$ heuristic, hindering its capacity to establish a *landmark* ordering that encompasses the complete knowledge required for eMPT resolution. This effect is particularly pronounced in tightly coupled scenarios,

where the complexity of interactions among agents increases. In such cases, a complete eMPT *landmark* system is often more effective for problem resolution than relying solely on local eMPT solving and assembly methods.

These interactions further explain why MA-LAMA prioritizes *Single Search* in tightly coupled scenarios through the AIA. We choose to forgo the benefits of performing the search with the *Parallel Steps* framework, allowing the heuristics to access the entire eMPT propositional information, which proves beneficial, as we will prove in the experiments.

## 7 Experiments

In this section, we present the experimental analysis of the MA-LAMA planner, evaluating its performance across a variety of domains created using PDDL2.1 [20]. To ensure compatibility and extend the range of domains where MA-LAMA can be applied, we opted to work with the standard PDDL rather than the agent-specific MA-PDDL language [29]. This choice allows MA-LAMA to operate with eMPT-defined tasks from *Definition 3*, autonomously identifying agents in each scenario. Privacy features, although possible in MA settings, are not considered here as they fall outside the scope of this study.

We test the performance of MA-LAMA through a MA temporal domains test bench, which comprises the MA temporal Exploration domain and all IPC temporal domains [12] that present MA features.

For all experiments, we compare MA-LAMA with other state-of-the-art temporal planners: OPTIC [4], TFLAP [46], Temporal Fast Downward (TFD) [16] and PopCorn [47]. Additionally, we include the temporal planner SGPlan [59], that ignores time to provide strong results for temporal domains coverage, although one note should be added about its inclusion. We follow the same approach as Benton et al. 2012 in the OPTIC planner evaluation, and include two versions of SGPlan: SGPlan 6; and SGPlan-w, which is the same planner encapsulated by a wrapper that alters the textual properties of the domain and adds a dummy action that cannot be grounded. This is done to avoid any domain-dependent behavior that SGPlan may implement for well-known temporal domains.

Notably, Yahsp3 is excluded due to its single-thread version being outperformed by TFD and its inability to handle metrics in our tests. Similarly, TemPorAl [9] and CP4TP [21] are not considered, as both operate as portfolios.

We predominately compare planners using the IPC score. It is obtained by the ratio $C^*/C$ between the cost of the cheapest discovered plan for a given planner, C, and the cost of the best plan found by all planners, $C^*$. The score for an unsolved task is 0 and greater is better in all cases.

Although the main focus is IPC performance scores in MA temporal domains, coverage results will also be discussed so that the contrast between the effort to find a solution and to find optimized solutions is noticeable.

The structure of the experimentation is the following. We first present the results for plan quality in our motivation domain. Then, the test bench of all IPC temporal domains that present MA nature follows. To enhance our understanding on the impact of using MAP techniques in temporal domains, we separate the domains in two groups:

- *Cooperation* domains. Mainly loosely-coupled environments where most of the goals should be achievable by independent agents, although *Temporal constraints* between them may still be present. These resemble the MA temporal Exploration domain characteristics that MA-LAMA is designed to solve with *Parallel Search*.
- *Coordination* domains. Mainly tightly-coupled environments and *coordination points* are typically present. In some of these cases, MA-LAMA will make use of *Single Search* to achieve a solution.

Finally, search efficiency will be studied by comparing runtime results with plan quality for selected *cooperation* and *coordination* temporal domains.

Table 5. MA temporal Exploration domain quality results using the IPC performance scores. Higher is better for all cases. The metric used is a weighted combination of battery usage and mission risk. The total is rounded.

| Prob | nº of agents | SGP | SGP-w | PopC | TFD | TFLAP | OPTIC | MA-LAMA |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 0.21 | 0.21 | 0.17 | 0.22 | 0.13 | 0.17 | **1** |
| 2 | 4 | 0.78 | 0.78 | 0.94 | 0.92 | 0.79 | 0.66 | **1** |
| 3 | 4 | 0.36 | 0.36 | 0.42 | 0.59 | 0.90 | 0.60 | **1** |
| 4 | 4 | 0.26 | 0.26 | 0.15 | 0.15 | 0.13 | 0.15 | **1** |
| 5 | 5 | 0.47 | 0.47 | 0.47 | 0.79 | 0.66 | 0.47 | **1** |
| 6 | 5 | 0.43 | 0.43 | 0.57 | 0.78 | 0.53 | 0.55 | **1** |
| 7 | 6 | 0.18 | 0.18 | 0.11 | 0.10 | 0.19 | 0.11 | **1** |
| 8 | 6 | 0.10 | 0.10 | 0.21 | 0.20 | 0.18 | 0.14 | **1** |
| 9 | 6 | 0.10 | 0.10 | 0.14 | 0.27 | 0.24 | 0.22 | **1** |
| 10 | 7 | 0.13 | 0.13 | 0.14 | 0.13 | 0.16 | 0.14 | **1** |
| 11 | 7 | 0.42 | 0.42 | 0.53 | 0.55 | 0.64 | 0.53 | **1** |
| 12 | 9 | 0.12 | 0.12 | 0.11 | 0.08 | 0.09 | 0.09 | **1** |
| **Total (12)** | | 4 | 4 | 4 | 5 | 5 | 4 | **12** |

## 7.1 MA temporal Exploration Domain Results

Our experimental evaluation for the motivation domain includes twelve problems, each progressively increasing in complexity and number of agents. The metric used is a weighted combination of battery usage and mission risk, similar to the one defined in Figure 3. In Table 5 we show the IPC score of SGPlan, SGPlan-w, PopCorn, TFD, TFLAP, OPTIC and MA-LAMA. All executions are limited to 10 minutes and 4GB of RAM. In this case, no differences are expected between vanilla and wrapped SGPlan.

For this type of domain, finding a propositional operator sequence that solves the problem should be almost trivial for any state-of-the-art temporal planner. Thus, coverage results are as expected. All planners successfully provide solutions for the twelve proposed problems, indicating that none of the instances present a significant propositional challenge. In contrast, MA-LAMA dominates in plan quality results, doubling the IPC performance score of all other planners for the Exploration domain.

For most planners, trying to find a balance between finding a solution and optimizing the risk and battery metric becomes a hard task because of their iterative search approach. Typically, the first solution found does not guide the search to promising quality paths, so instead, they rely on brute force WA* to find good solutions by extensively exploring the search space with cost pruning. This approach is ineffective when the number of possible states is high, as demonstrated in this case, where the 10 minutes time limit is insufficient to produce solutions of the same quality as MA-LAMA.

This exponential growth in the search space typically happens in scenarios that require to trace a path between positions for several agents, forcing the planners to solve a MA Path Finding (MAPF) problem.

MA-LAMA's task decomposition by agents and cost informed goal allocation techniques show strong performance in dealing with loosely-coupled temporal scenarios where the path to solve the propositional problem is not tied with the path to find high quality solutions.

We are aware that, despite the good results, the Exploration scenario fits MA-LAMA's design too well to draw conclusions about our solver's general performance in MA temporal domains. Therefore, in the next subsections we will test MA-LAMA in a wide range of MA temporal scenarios.

Table 6. Coverage (problems solved) in *cooperation* temporal domains for a total of 90 problems. Higher is better for all cases.

| Domain | SGP | SGP-w | PopC | TFD | TFLAP | OPTIC | MA-LAMA |
|---|---|---|---|---|---|---|---|
| Taxis (20) | **20** | **20** | **20** | **20** | 18 | **20** | **20** |
| Trucks (10) | **10** | 1 | **10** | **10** | **10** | **10** | **10** |
| Rovers (20) | **20** | **20** | 15 | 19 | 4 | 17 | **20** |
| Satellites (20) | **20** | **20** | 0 | 14 | **20** | 16 | **20** |
| Zenotravel(20) | **20** | **20** | 11 | **20** | 18 | 14 | **20** |
| **Total (90)** | **90** | 81 | 56 | 83 | 70 | 77 | **90** |

Table 7. IPC Performance scores (rounded to whole numbers) in *cooperation* temporal domains for a total of 90 problems. Higher is better for all cases.

| Domain | SGP | SGP-w | PopC | TFD | TFLAP | OPTIC | MA-LAMA |
|---|---|---|---|---|---|---|---|
| Taxis (20) | 7 | 7 | 6 | 13 | 17 | 13 | **19** |
| Trucks(10) | 2 | 0 | 2 | 3 | **9** | 4 | **9** |
| Rovers (20) | 14 | 14 | 6 | 12 | 1 | 9 | **20** |
| Satellites (20) | 3 | 1 | 0 | 8 | 7 | 12 | **17** |
| Zenotravel(20) | 8 | 5 | 2 | 7 | 13 | 6 | **19** |
| **Total (90)** | 34 | 27 | 16 | 43 | 47 | 44 | **84** |

## 7.2 Cooperation Multi-Agent Domains

Coverage and IPC quality score results for *cooperation* MA temporal domains can be seen in Tables 6 and 7 respectively.

For all of them, the AD finds the agent task decomposition that a human operator would choose and no *coordination points* are present. This allows MA-LAMA to solve all of them through *Parallel Search*.

In coverage results, MA-LAMA and SGPlan are the only two planners that are able to solve all instances, although all except PopCorn provide good results. Differences are specially relevant in Rovers, Satellites and Zenotravel domains, whose instances are characterized by being significantly heavier in terms of search space size compared to Exploration, Taxi and Trucks. Notably, SGPlan-w performs much worse than vanilla SGPlan in coverage for the Trucks domain.

For IPC performance scores, MA-LAMA outperforms every other tested temporal solver in all domains, and it is the only planner that does not significantly decrease the score obtained previously in coverage.

The **Taxis** domain presents a distance traveled metric for each agent. It includes a grid-like map that has similar characteristics to those described for the Exploration domain MAPF problem. All planners are able to find solutions for most instances but, when trying to optimize the distance traveled, only MA-LAMA and TFLAP are able to actually find good solutions. Notably, these two temporal solvers ignore time during search and center their efforts in cost optimization. A similar trend can be seen in **Trucks** for a similar distance based metric.

For the **Rovers** domain, the metric employed is based on weighted battery usage. Taking into account that Rovers is the most constrained *cooperation* domain tested, some observations can be made on the performance of SGPlan and TFD, being these good coverage solvers. Although Rovers cannot be considered a tightly-coupled scenario, the existence of battery-recharge limitations and shared communication resources limits the availability of the rover agents for plan quality optimization. The improvement in quality score for these planners indicates

Table 8. Coverage (problems solved) in *coordination* temporal domains for the total of 140 problems. Higher is better for all cases.

| Domain | SGP | SGP-w | PopC | TFD | TFLAP | OPTIC | MA-LAMA |
|---|---|---|---|---|---|---|---|
| Elevators (20) | **20** | **20** | **20** | **20** | **20** | **20** | **20** |
| DriverLog (20) | **20** | **20** | 12 | 15 | 14 | 15 | 16 |
| Depot (20) | **20** | **20** | 4 | 4 | 11 | 7 | 11 |
| Logistics (20) | **20** | **20** | 0 | **20** | **20** | **20** | **20** |
| Traffic-Accident (10) | **10** | **10** | **10** | **10** | **10** | **10** | **10** |
| Woodworking (20) | **19** | **19** | 8 | 14 | 7 | 8 | 12 |
| Floortile (10) | **7** | **7** | 0 | 0 | 4 | 0 | 0 |
| Storage (20) | **20** | 9 | 15 | 0 | 17 | 16 | 14 |
| **Total (140)** | **136** | 125 | 69 | 83 | 103 | 96 | 103 |

that, with the increase in the domain resolution complexity, finding any solution becomes a better option in terms of quality. This idea, that ties more constrained scenarios with better quality performance for good coverage solvers, will be expanded during the *coordination* results. In the end, MA-LAMA finds the best solution for all problems.

The **Satellites** domain metric is the *total-time* of the plan. This is relevant because OPTIC and TFD heavily reason with time during search, which should provide better quality results in *total-time* metrics based tests. Despite this, OPTIC is the only one that actually improves, and all three are outperformed by MA-LAMA's non temporal reasoning search process. SGPlan-w exhibits varying performance compared with vanilla SGPlan, resulting in slightly worse results.

Finally, the **Zenotravel** domain uses a metric that combines weighted fuel and the *total-time*, and shows results that resemble the ones from the other domains, with MA-LAMA providing significantly better solutions. We observe that SGPlan-w delivers slightly worse performance than base SGPlan again.

In general, MA-LAMA proves to be the most reliable temporal planner to solve MA *cooperation* domains, showing strong performance in coverage and the best in plan quality against the other state-of-the-art temporal solvers.

## 7.3 Coordination Multi-Agent Domains

Coverage and IPC performance score results for *coordination* MA temporal domains can be seen in Tables 8 and 9 respectively.

MA-LAMA solves Elevators and DriverLog through *Parallel Search*, detecting the existing *coordination points* and solving the *Parallel Steps* around them. For Depot, Logistics, Woodworking and Traffic-Accident, MA-LAMA employs *Single Search*, either because the AIA estimates that the decomposition is not promising or because the AD fails to find a valid agent task decomposition, which happens for Depot and Woodworking instances.

For *coordination* domains, both SGPlan and SGPlan-w dominate in coverage results, although SGPlan-w produces worse results for the Storage domain. MA-LAMA comes in second place with a similar performance compared to TFD, TFLAP and OPTIC, with PopCorn delivering the worst performance out of the six.

Regarding IPC quality performance, MA-LAMA comes on top despite solving less instances than SGPlan, but with a much closer result compared with the *cooperation* test bench. Again, OPTIC, TFLAP and TFD perform similarly and outperform PopCorn.

The **Elevators** domain metric is the *total-time* of the plan. All planners successfully solve all instances, maintaining high plan quality throughout. Despite the tight coupling of elevator agents within the domain, all

Table 9. IPC Performance scores (rounded to whole numbers) in *coordination* temporal domains for the total of 140 problems. Higher is better for all cases.

| Domain | SGP | SGP-w | PopC | TFD | TFLAP | OPTIC | MA-LAMA |
|---|---|---|---|---|---|---|---|
| Elevators (20) | **19** | **19** | 15 | 15 | 16 | 17 | **19** |
| DriverLog (20) | 9 | 4 | 4 | 4 | 9 | 7 | **16** |
| Depot (20) | **14** | **14** | 1 | 3 | 4 | 6 | 10 |
| Logistics (20) | 15 | 15 | 0 | 12 | **18** | **18** | **18** |
| Traffic-Accident (10) | 8 | 8 | 8 | 9 | 9 | 9 | **10** |
| Woodworking (20) | **15** | **15** | 4 | 13 | 7 | 7 | 11 |
| Floortile (10) | **4** | **4** | 0 | 0 | 3 | 0 | 0 |
| Storage (20) | 12 | 5 | 6 | 0 | 14 | **15** | 13 |
| **Total (140)** | 96 | 84 | 38 | 56 | 80 | 79 | **97** |

planners find solutions with relative ease until the last instances, where SGPlan and MA-LAMA outperform the other solvers. MA-LAMA's performance is notable since it is able to efficiently decompose and solve difficult instances around the *coordination points* between the elevators.

**Driverlog** implements a metric of weighted walked and driven distance. It is the only *coordination* domain where MA-LAMA MAP techniques deliver a relevant advantage over the other temporal solvers, delivering the best performance for all its solved problems. This improvement in plan quality can happen thanks to the specific domain characteristics, that, as in some *cooperation* domains, allow MA-LAMA to optimize the MAPF problem through iterative local searches within the *Parallel Search* framework. In this case, SGPlan-w delivers different solutions compared with vanilla SGPlan, which results in worse quality overall.

The **Depot** domain presents a fuel consumption metric, and proves to be challenging for MA-LAMA. The AD fails to detect a usable decomposition, trying to use crates as agents, and resorting to *Single Search* is not enough to optimize the driver distance cost from the search space. It becomes clear that MA-LAMA needs the *Parallel Steps* optimization potential to deal with these types of scenarios. SGPlan performs the best in this case, taking advantage of the previous coverage score.

The **Logistics** domain metric demands *total-time* optimization along with the distance traveled. OPTIC again benefits from its temporal reasoning, achieving a performance on par with TFLAP and MA-LAMA in plan quality. In tightly-coupled domains with little room for optimization such as this one, MA-LAMA proves that the *Single Search* is enough to deliver optimized solutions, making use of the extra information that its classical heuristics receive. It is also worth noting that SGPlan, TFLAP and MA-LAMA are consistently able to outperform the other temporal planners despite their heavy temporal reasoning.

The **Traffic-Accident** domain employs a distance traveled metric. It presents a similar domain structure compared to Logistics, but with more tightly-coupled agents. This provokes that MA-LAMA is able to deliver the best performance with the *Single Search* approach.

**Woodworking** is the most difficult *coordination* domain tested and demands *total-time* optimization. In this case, MA-LAMA's *Single Search* is not enough to deal with the scenario, performing worse than SGPlan and TFD, that benefit from solving more instances.

The **Floortile** domain evidences MA-LAMA limitations. In this case, dead-ends during the search will arise if agents are not coordinated adequately. This is a type of interaction that MA-LAMA's *coordination points* do not cover and proves to be a weakness when trying to solve the problems with *Parallel Search*. PopCorn, TFD, and OPTIC also fail to produce solutions. SGPlan, SGPlan-w, and TFLAP do solve some instances, with TFLAP

obtaining slightly better solutions for the instances that it solves. These results indicate that this domain benefits from efficient exploration of the search space, rather than heavy temporal reasoning or decomposition approaches.

Finally, the **Storage** domain also presents some challenges to MA-LAMA. The agent detection fails and produces a non-usable decomposition that resembles the one from Depot. With the increasing number of agents, the search space expands and the *Single Search* process stops being able to explore it effectively, leading to poor performance in the last problems due to scalability issues. This behavior is shared between TFLAP, OPTIC, and MA-LAMA, with TFLAP and OPTIC achieving better overall performance. Notably, SGPlan varies heavily in coverage and quality between wrapped and vanilla versions. We consider that TFD's results in this specific domain are not representative of its overall performance, as the search is not initiated correctly.

To conclude this section, MA *coordination* domains present less room for metric optimization than *cooperation* domains, which makes MA-LAMA performance more in line with the other temporal solvers. Despite this, the *coordination points* management in *Parallel Steps* and *Single Search* procedure in tightly-coupled scenarios help MA-LAMA to outperform the other state-of-the-art temporal solvers in MA temporal plan quality performance for *coordination* temporal domains.

## 7.4 Search Efficiency

In the previous section, we evaluated the performance of all planners on *cooperation* and *coordination* MA temporal benchmarks using metrics such as coverage and IPC performance. These metrics allowed us to rank the best solution each planner produced within the imposed 10-minute limit. However, they do not convey the computational effort or search efficiency, which are central to our investigation.

Given our focus on plan quality and the structural differences between MA-LAMA and other temporal planners, selecting an appropriate efficiency metric is critical. Measuring runtime would only reflect the best solution found within the time limit for planners using iterative searches. This would not be meaningful, as planners that found quick but lower-quality solutions would show shorter runtimes without reflecting actual performance. The number of expanded nodes would only indicate how extensively a planner explores the search space, without necessarily reflecting the quality of the solutions found, and would fail to encapsulate MA-LAMA's decomposition of the MA temporal task.

Therefore, we opted to compare the overall search time in relation to IPC performance. Although this approach sacrifices detailed information on individual problem instances, we deem this trade-off acceptable because it clearly illustrates the computational effort invested by each planner in obtaining high-quality solutions across domains. Figure 12 presents the IPC Quality and runtime metrics for the ten most challenging problems across two *cooperation* and two *coordination* domains. The baseline planners selected are the same from the previous section, excluding vanilla SGPlan to avoid domain-dependent behavior in efficiency comparisons. The plots clearly illustrate the quality of the obtained solutions, where higher values indicate superior performance, and the computational effort required to achieve them, with runtime increasing logarithmically from left to right.

For *cooperation* domains, we present results for Zenotravel and Rovers. In **Zenotravel** domain, MA-LAMA consistently produces high-quality plans while maintaining search times comparable to those of other planners. Only TFLAP occasionally achieves a similar level of solution quality, but generally it does so at the cost of longer search times. OPTIC, TFD, PopCorn and SGPlan-w do not present improvements in runtime results that could justify the worse plan quality.

In the **Rovers** domain, results are similar. MA-LAMA clearly dominates in plan quality while exhibiting runtime performance comparable to other planners. Although TFD occasionally produces high-quality solutions, its performance is less consistent and it incurs higher search times in some cases. Additionally, SGPlan-w and OPTIC solve certain instances quickly, but their overall plan quality remains inconsistent.

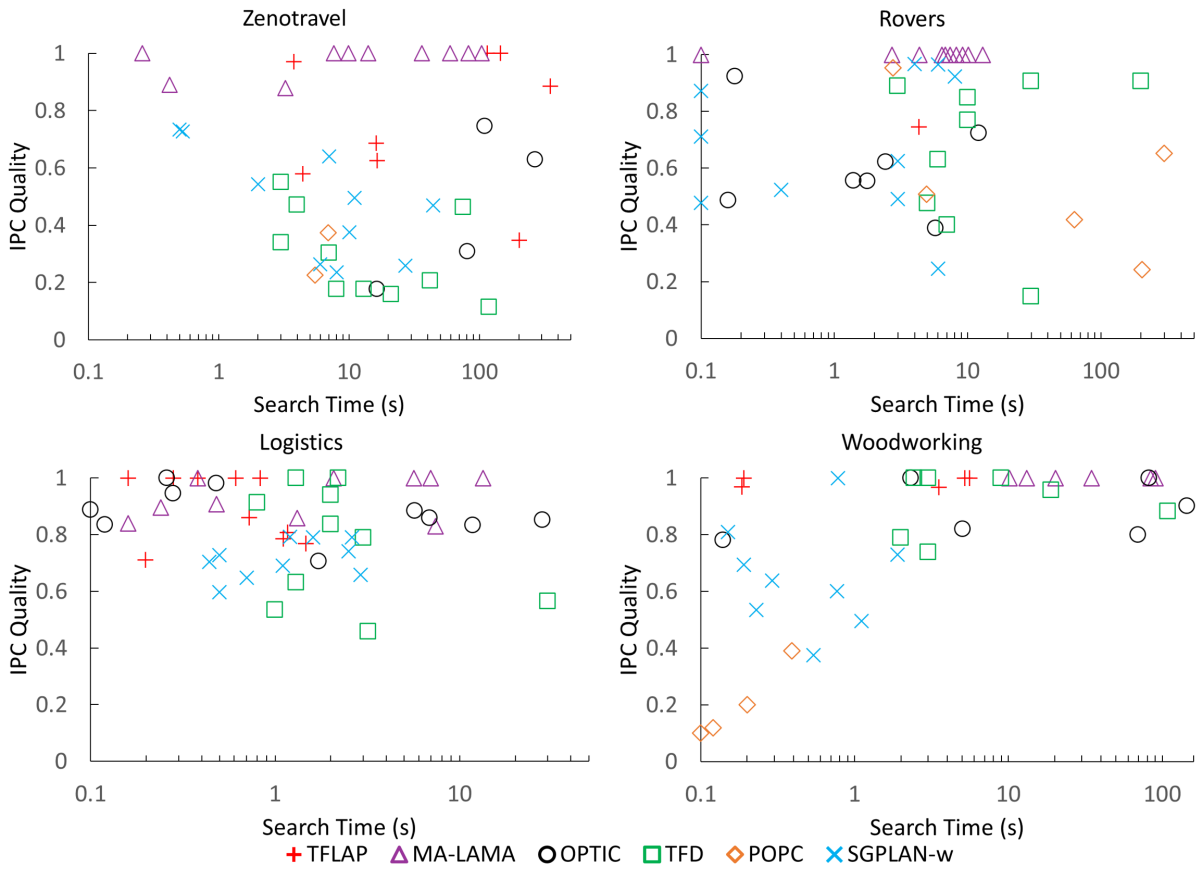



Fig. 12. IPC Performance scores and time consumed to achieve the solution for the last 10 problems of Zenotravel, Rovers, Logistics and Woodworking domains. Higher values indicate superior IPC quality. Runtime, in seconds, increases logarithmically from left to right.

For *coordination* domains, Logistics and Woodworking are used for comparison. In the **Logistics** domain, MA-LAMA, TFLAP, and OPTIC yield comparable plan quality solutions, though there are notable differences in how they achieve these results. For OPTIC and TFLAP, the highest quality plans emerge when runtimes are low, whereas MA-LAMA delivers good, but not always optimal, solutions in these scenarios. However, when TFLAP's runtime exceeds one second, its plan quality diminishes, while MA-LAMA, despite requiring more time, produces higher-quality plans. A similar trend is observed for OPTIC, although without TFLAP's advantage in speed. Consequently, MA-LAMA's and TFLAP's search efficiencies are comparable in this context, with each exhibiting a trade-off between quality and runtime that favors them under different circumstances. As in previous domains, TFD's results are inconsistent in quality, and SGPlan-w does not provide any significant runtime benefits.

Finally, in the **Woodworking** domain, each planner exhibits distinct performance characteristics. MA-LAMA consistently finds high-quality solutions for all instances it solves, but it does so with notably high runtime values. In contrast, TFD achieves superior overall quality without incurring additional runtime costs, thereby outperforming MA-LAMA in this case. OPTIC and TFLAP generate some high-quality solutions, with TFLAP once again demonstrating greater efficiency; however, both struggle with solving certain instances. SGPlan-w

delivers varied solution quality but demonstrates strong runtime performance, successfully solving almost all instances. PopCorn exhibits a similar trend in terms of quality and runtime; however, it solves significantly fewer instances in comparison. Overall, in this case TFD emerges as the most efficient planner when prioritizing solution quality, whereas TFLAP is the best option when prioritizing speed.

We can conclude that different trends emerge from *cooperation* and *coordination* domains:

- In **cooperation** scenarios, MA-LAMA consistently generates higher-quality plans without compromising search speed, demonstrating superior search efficiency overall.
- For **coordination** domains results are mixed. As we could already observe in *coordination* quality results from Table 9, MA-LAMA does not outperform other planners in all cases. This, coupled with the increase in runtime values caused by the additional inter-agent interactions, diminishes MA-LAMA's search efficiency relative to other planners. In general, MA-LAMA's efficiency excels in *coordination* domains that benefit from its *Single Search* strategy, as evidenced in the Logistics domain. However, it exhibits reduced efficiency in domains where the classical heuristics used in the *Single Search* strategy encounter difficulties, such as in the Woodworking domain, which is temporally more complex.

Therefore, MAP techniques benefit efficiency and can effectively mitigate the complexity of temporal planning problems in *cooperation* temporal domains. For *coordination* temporal domains, MAP techniques still present benefits in scenarios whose complexity does not arise from temporal interactions.

## 8 Discussion

The results presented in the previous section demonstrate that MA-LAMA outperforms state-of-the-art temporal planners in terms of coverage and plan quality in *cooperation* (or loosely-coupled) MA temporal planning domains. In *coordination* (or tightly-coupled scenarios), MA-LAMA also achieves strong coverage performance and consistently delivers high-quality plans, comparable to top-performing planners.

Although MA-LAMA shares several features with other tested solvers, it is the only temporal planner in this study that implements MA-specific techniques designed to reduce the computational complexity associated with search spaces involving concurrent acting entities. In terms of search efficiency, these prove to be effective in reducing the complexity of achieving optimized solutions for *cooperation* domains and non-temporally complex *coordination* domains. Notably, other high-performing planners in our experiments, such as TFLAP and SGPlan, alongside MA-LAMA, do not incorporate any form of temporal reasoning during the search process.

These findings indicate that the temporal domains typically used in our tests may be more amenable to resolution with MA-techniques rather than traditional temporal reasoning. This suggests that these domains are primarily MA in nature, involving concurrent operators rather than complex temporal interactions.

Furthermore, the successful application of agent task decomposition, cost-informed goal allocation, and iterative search partial plans in MA-LAMA suggests a promising direction for enhancing temporal planning through the integration of other MA planning techniques, as discussed in section 2.

The Knowledge Compilation module in MA-LAMA, which includes Agent Detection (AD), Goal Categorization and Assignment (GCA), and Agent Interaction Analysis (AIA), is instrumental in determining the techniques MA-LAMA applies to a given scenario. We propose that these types of pre-search analysis modules hold potential for improving various types of temporal planning. Previous work, such as the portfolio-based approach used by TemPorAl [9], has shown that portfolio methods can outperform traditional solvers in versatility and coverage. Incorporating a knowledge compilation module into such portfolios could, in theory, guide the selection of appropriate solvers within the portfolio, thereby enhancing efficiency.

Regarding the limitations of MA-LAMA, the most evident one is its lack of temporal reasoning capabilities. Although this is a deliberate design choice to improve efficiency in certain scenarios, it limits MA-LAMA's ability to solve temporally complex domains that require a deep understanding of agent interactions over time, such as

the Floortile domain. In these cases, the decomposition approach may fail to identify dead-ends during the search process, leading to worse performance.

Additionally, the automatic agent decomposition (AD) technique in MA-LAMA, inherited from ADP [13], has its limitations. In domains such as Depot or Storage, the detection of agents via the causal graph (CG) may produce non-intuitive decompositions when agents have tight interactions. This can provoke scalability issues, as MA-LAMA is forced to solve these scenarios with the Single Search process. MA-LAMA mitigates this issue by relaxing its agent merging rules, but a more robust agent detection (AD) technique could further improve its performance.

## 9 Conclusions

In this work, we have presented MA-LAMA, a MA temporal solver specifically designed to leverage the MA characteristics of temporal planning problems to provide high-quality solutions.

MA-LAMA achieves this through the implementation of several MA techniques that address the temporal complexity arising from agent interactions. These techniques include automatic agent detection, task decomposition, cost-informed goal allocation, and *anytime* iterative searches to solve sub-goals.

Our results demonstrate that MA-LAMA outperforms other state-of-the-art temporal planners in terms of plan quality for both loosely-coupled and tightly-coupled temporal scenarios. This indicates that MA techniques can be effectively applied to certain domains traditionally considered purely temporal.

## Acknowledgments

This study is funded by Junta de Comunidades de Castilla-La Mancha (SBPLY/24/180225/000143) and Comunidad de Madrid (PIPF-2023/COM-29922) projects.

## References

[1] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern. 2022. Multi-agent pathfinding with continuous time. *Artificial Intelligence*, 305, 103662.

[2] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig. 2020. Probabilistic robust multi-agent path finding. *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 30, 1, (June 2020), 29–37.

[3] C. Bäckström and B. Nebel. 1995. Complexity results for sas+ planning. *Computational Intelligence*, 11, (Jan. 1995), 625–656.

[4] J. Benton, A. Coles, and A. Coles. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 22. Atibaia, Brazil, (May 2012), 2–10. DOI: 10.1609/icaps.v22i1.13509.

[5] D. Borrajo and S. Fernández. 2015. Mapr and cmap. In *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP)*. Jesuralem, Israel, 1–3.

[6] R. I. Brafman and C. Domshlak. 2008. From one to many: planning for loosely coupled multi-agent systems. In *Proc. of the Int. Conf. on Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Sydney, Australia, 28–35. ISBN: 9781577353867.

[7] J. Caballero, O. Perez-Mon, M. D. R-Moreno, and J. d. O. Filho. 2023. Integral ai-based planning for management of wsns in military operations. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*. Atlanta, USA, 341–348. DOI: 10.1109/ICTAI59109.2023.00056.

[8] J. Caballero Testón and M. D. R-Moreno. 2024. Multi-agent temporal task solving and plan optimization. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 34. Banff, Canada, (June 2024), 50–58.

[9] I. Cenamor, M. Vallati, L. Chrpa, T. de la Rosa, and F. Fernández. 2018. Temporal: temporal portfolio algorithm. https://ipc2018-temporal.bitbucket.io/planner-abstracts/team1. Accessed: 2022-09-01. (2018).

[10] A. J. Coles, A. I. Coles, M. Fox, and D. Long. 2012. Colin: planning with continuous linear numeric change. *Journal of Artificial Intelligence Research*, 44, (May 2012), 1–96. DOI: 10.1613/jair.3608.

[11] A. Coles, A. Coles, M. Fox, and D. Long. 2010. Forward-chaining partial-order planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 20. Toronto, Canada, (May 2010), 42–49. DOI: 10.1609/icaps.v20i1.13403.

[12] A. Coles, C. Coles, M. Martinez, and P. Sidiropoulos. 2018. International planning competition 2018 temporal tracks. https://ipc2018-temporal.bitbucket.io/. Accessed: 2022-09-01. (2018).

[13] M. Crosby, M. Rovatsos, and R. Petrick. 2013. Automated agent decomposition for classical planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 23. Rome, Italy, (June 2013), 46–54. DOI: 10.1609/icaps.v23i1.13564.

[14] W. Cushing, S. Kambhampati, Mausam, and D. S. Weld. 2007. When is temporal planning really temporal? In *Int. Joint Conference on Artificial Intelligence*. Menlopark, California, 1852–1859.

[15] Y. Dimopoulos, M. A. Hashmi, and P. Moraitis. 2012. $\mu$-satplan: multi-agent planning as satisfiability. *Knowledge-Based Systems*, 29, 54–62. DOI: https://doi.org/10.1016/j.knosys.2011.07.019.

[16] 2012. *Using the context-enhanced additive heuristic for temporal and numeric planning. Towards Service Robots for Everyday Environments: Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments*. Springer Berlin Heidelberg, Berlin, Heidelberg, 49–64. ISBN: 978-3-642-25116-0. DOI: 10.1007/978-3-642-25116-0_6.

[17] E. Fabre, L. Jezequel, P. Haslum, and S. Thiébaux. 2010. Cost-optimal factored planning: promises and pitfalls. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 20. Toronto, Ontario, Canada, 65–72.

[18] M. Feyzbakhsh Rankooh and G. Ghassem-Sani. 2015. Itsat: an efficient sat-based temporal planner. *Journal of Artificial Intelligence Research*, 53, (July 2015), 541–632. DOI: 10.1613/jair.4697.

[19] D. Fišer, M. Štolba, and A. Komenda. 2015. Maplan. In *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP)*. Jerusalem, Israel, 8–10.

[20] M. Fox and D. Long. 2003. Pddl2.1: an extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20, (Dec. 2003), 61–124. DOI: 10.1613/jair.1129.

[21] D. Furelos-Blanco and A. Jonsson. 2018. CP4TP: A Classical Planning for Temporal Planning Portfolio. In *Temporal Track of the International Planning Competition (IPC)*. Delft, Netherlands.

[22] M. Helmert. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26, (July 2006), 191–246. DOI: 10.1613/jair.1705.

[23] M. Helmert. 2009. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence*, 173, 5, 503–535. Advances in Automated Plan Generation. DOI: https://doi.org/10.1016/j.artint.2008.10.013.

[24] J. Hoffmann and B. Nebel. 2001. The ff planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14, (May 2001), 253–302. DOI: 10.1613/jair.855.

[25] J. Hoffmann, J. Porteous, and L. Sebastia. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22, (Nov. 2004), 215–278. DOI: 10.1613/jair.1492.

[26] L. Jezequel and E. Fabre. 2012. A#: a distributed version of a* for factored planning. In *51st IEEE Conf. on Decision and Control (CDC)*. Maui, United States, 7377–7382. DOI: 10.1109/CDC.2012.6426187.

[27] P. Jonsson and C. Bäckström. 1998. State-variable planning under structural restrictions: algorithms and complexity. *Artificial Intelligence*, 100, 1, 125–176. DOI: https://doi.org/10.1016/S0004-3702(98)00003-4.

[28] A. Komenda, M. Štolba, and D. Kovács. 2016. The international competition of distributed and multiagent planners (codmap). *AI Magazine*, 37, (Oct. 2016), 109–115. DOI: 10.1609/aimag.v37i3.2658.

[29] D. L. Kovacs. 2012. A multi-agent extension of pddl3.1. In *Proc. of the Workshop on the International Planning Competition (IPC)*. Atibaia, Brazi, 19–27.

[30] J. Kvarnström. 2011. Planning for loosely coupled agents using partial order forward-chaining. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 21. Freiburg, Germany, (Mar. 2011), 138–145. DOI: 10.1609/icaps.v21i1.13462.

[31] A. L. Lansky. 1991. Localized search for multiagent planning. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*. Vol. 1. Sydney, Australia, 252–258. ISBN: 1558601600.

[32] D. Long and M. Fox. 2003. Exploiting a graphplan framework in temporal planning. In *Procs of the Int. Conf. on Automated Planning and Scheduling (ICAPS)* (ICAPS'03). AAAI Press, Trento, Italy, 52–61.

[33] N. Luis, S. Fernández, and D. Borrajo. 2020. Plan merging by reuse for multi-agent planning. *Applied Intelligence*, 50, 2, (Feb. 2020), 365–396. DOI: 10.1007/s10489-019-01429-0.

[34] H. Ma, T. K. S. Kumar, and S. Koenig. 2017. Multi-agent path finding with delay probabilities. In *Proc. of the AAAI Conf. on Artificial Intelligence* (AAAI'17). AAAI Press, San Francisco, California, USA, 3605–3612.

[35] S. Maliah, G. Shani, and R. Stern. 2014. Privacy preserving landmark detection. *Frontiers in Artificial Intelligence and Applications*, 263, (Jan. 2014), 597–602. DOI: 10.3233/978-1-61499-419-0-597.

[36] S. Maliah, G. Shani, and R. Stern. 2016. Stronger privacy preserving projections for multi-agent planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 26. London, United Kingdom, (Mar. 2016), 221–229. DOI: 10.1609/icaps.v26i1.13753.

[37] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas. 2018. On the timed temporal logic planning of coupled multi-agent systems. *Automatica*, 97, 339–345.

[38] R. Nissim, U. Apsel, and R. I. Brafman. 2012. Tunneling and decomposition-based state reduction for optimal planning. In *European Conf. on Artificial Intelligence (ECAI)*. Vol. 242. Montpellier, France, 624–629. https://api.semanticscholar.org/CorpusID:17139428.

[39] R. Nissim and R. Brafman. 2013. Distributed heuristic forward search for multi-agent systems. *Journal of Artificial Intelligence Research*, 51, (June 2013), 293–332. DOI: 10.1613/jair.4295.

[40] R. Nissim and R. Brafman. 2012. Multi-agent A* for parallel and distributed systems. In *Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*. Vol. 3. Valencia, Spain, (June 2012), 1265–1266.

[41] R. Nissim, R. I. Brafman, and C. Domshlak. 2010. A general, fully distributed multi-agent planning algorithm. In *Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (ICAPS)*. Vol. 1. Toronto, Canada, 1323–1330. ISBN: 9780982657119.

[42] D. Pellier. 2010. Distributed planning through graph merging. In *Proc of the Int. Conf. on Agents and Artificial Intelligence (ICAART)*. Vol. 2. Valencia, Spain, (Jan. 2010), 128–134.

[43] J. Porteous, L. Sebastia, and J. Hoffmann. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Proc. European Conf. on Planning*. Vol. 6. Toledo, Spain, (July 2001), 174–182.

[44] M. Ramirez, N. Lipovetzky, and C. Muise. 2015. Lightweight Automated Planning ToolKiT. http://lapkt.org/. Accessed: 2024. (2015).

[45] S. Richter and M. Westphal. 2010. The lama planner: guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39, (Sept. 2010), 127–177. DOI: 10.1613/jair.2972.

[46] O. Sapena, E. Marzal, and E. Onaindia. 2018. Tflap: a temporal forward partial-order planner. https://ipc2018-temporal.bitbucket.io/planner-abstracts/team2. Accessed: 2022-09-01. (2018).

[47] E. Savaş, M. Fox, D. Long, and D. Magazzeni. 2016. Planning using actions with control parameters. In *European Conf. on Artificial Intelligence*. Vol. 285. The Hague, The Netherlands, 1185–1193. https://api.semanticscholar.org/CorpusID:11711977.

[48] L. Sebastia, E. Onaindia, and E. Marzal. 2006. Decomposition of planning problems. *AI Communications*, 19, (Jan. 2006), 49–81.

[49] A. Shleyfman, R. Kuroiwa, and J. C. Beck. 2023. Symmetry detection and breaking in linear cost-optimal numeric planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 33. Prage, Czech Republic, (July 2023), 393–401. DOI: 10.1609/icaps.v33i1.27218.

[50] S. Sreedharan, Y. Zhang, and S. Kambhampati. 2015. A first multi-agent planner for required cooperation (marc). In *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP)*. Jesuralem, Israel, 17–20.

[51] M. Štolba and A. Komenda. 2017. The madla planner: multi-agent planning by combination of distributed and local heuristic search. *Artificial Intelligence*, 252, 175–210. DOI: https://doi.org/10.1016/j.artint.2017.08.007.

[52] C. Street, S. Pütz, M. Mühlig, N. Hawes, and B. Lacerda. 2022. Congestion-aware policy synthesis for multirobot systems. *IEEE Transactions on Robotics*, 38, 1, 262–280. DOI: 10.1109/TRO.2021.3071618.

[53] A. Torreño, E. Onaindia, A. Komenda, and M. Štolba. 2017. Cooperative multi-agent planning: a survey. *ACM Computing Surveys*, 50, 6, (Nov. 2017), 1–32. DOI: 10.1145/3128584.

[54] A. Torreño, E. Onaindia, and Ó. Sapena. 2014. A flexible coupling approach to multi-agent planning under incomplete information. *Knowledge and Information Systems*, 38, (Jan. 2014), 141–178. DOI: 10.1007/s10115-012-0569-7.

[55] A. Torreño, E. Onaindia, and Ó. Sapena. 2012. An approach to multi-agent planning with incomplete information. *Frontiers in Artificial Intelligence and Applications*, 242, (Aug. 2012), 762–767. DOI: 10.3233/978-1-61499-098-7-762.

[56] A. Torreño, O. Sapena, and E. Onaindia. 2015. Global heuristics for distributed cooperative multi-agent planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 25. Jerusalem, Israel, (Apr. 2015), 225–233. DOI: 10.1609/icaps.v25i1.13701.

[57] J. Tožička, J. Jakubuv, and A. Komenda. 2015. Psm-based planners description for codmap 2015 competition. In *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP)*. Jesuralem, Israel, (Jan. 2015), 29–32.

[58] V. Vidal. 2014. YAHSP3 and YAHSP3-MT in the international planning competition. In *The International Planning Competition*. Jerusalem, Israel, 64–65.

[59] B. Wah and C.-W. Hsu. 2006. Temporal planning using subgoal partitioning and resolution in sgplan. *Journal of Artificial Intelligence Research*, 26, (Aug. 2006), 323–369. DOI: 10.1613/jair.1918.

[60] S. Zhang, Y. Jiang, G. Sharon, and P. Stone. 2017. Multirobot symbolic planning under temporal uncertainty. In *Proc. of the Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, Sao Paulo, Brazil, 501–510.

[61] Y. Zhang, S. Sreedharan, and S. Kambhampati. 2016. A formal analysis of required cooperation in multi-agent planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*. Vol. 26. London, United Kingdom, (Mar. 2016), 335–343. DOI: 10.1609/icaps.v26i1.13770.

Received 18 April 2025; accepted 12 June 2025