Multi-Agent Temporal Task Solving and Plan Optimization

J. Caballero Testón[1], **Maria D. R-Moreno**[1,2]

[1]Universidad de Alcalá, ISG, EPS, Alcalá de Henares, Spain
[2]TNO, IAS, The Hague, The Netherlands
javier.caballerot@edu.uah.es, malola.rmoreno@uah.es

Abstract

Several multi-agent techniques are utilized to reduce the complexity of classical planning tasks, however, their applicability to temporal planning domains is a currently open line of study in the field of Automated Planning.

In this paper, we present MA-LAMA, a factored, centralized, unthreated, satisfying, multi-agent temporal planner, that exploits the 'multi-agent nature' of temporal domains to perform plan optimization.

In MA-LAMA, temporal tasks are translated to the constrained *snap-actions* paradigm, and an automatic agent decomposition, goal assignment, and required cooperation analysis are carried out to build independent search steps, called Search Phases. These Search Phases are then solved by consecutive agent local searches, using classical heuristics and *temporal constraints*.

Experiments show that MA-LAMA is able to solve a wide range of classical and temporal multi-agent domains, performing significantly better in plan quality than other state-of-the-art temporal planners.

Introduction

In numerous scenarios the systems we plan for are naturally viewed as multi-agent (MA) systems, and, additionally, MA systems inherently tend to require some degree of concurrency between agents to operate efficiently. Some examples of temporal MA planning (MAP) scenarios that present an "MA nature" can be found in the International Planning Competition (IPC) benchmark domains, e.g. Rovers and Satellites, with independent homogeneous agents; and Elevators or Logistics, where interaction between agents is required.

In contrast, Automated Planning literature has classically considered temporal and MAP as two individual lines of work. This can be viewed in the MAP Survey (Torreño et al. 2017), where it is stated that the handling of scenarios derived from temporal MA systems is an open point that needs to be addressed, as MAP solvers tend to focus on classical planning domains.

In that survey, they also present some classifications for MAP solvers in terms of their taxonomy: *threaded*, with interleaved planning and coordination, and *unthreaded*, dealing with planning and coordination separately. Their computational structure: *centralized*, with a monolithic design and a central process, and *distributed*, sharing the planning task across multiple processing units. And their privacy preservation: providing *strong*, *object cardinality* or *weak* privacy, depending on the extent to which each agent's sensitive information is preserved. For plan quality optimization (one of the main points of interest of this work), MAP systems tend to make use of cost-aware classical planners, like LAMA (Richter and Westphal 2010).

MAP techniques used across all types of MA solvers are different from those used in temporal planners, which tend to revolve around time and numeric reasoning to deal with the inherent complexity of concurrent actions search spaces (Rintanen 2007). Our objective with this paper is to test the MAP techniques' effectiveness in dealing with temporal complexity, especially in areas where time reasoning-based planners struggle, as not coupled with makespan plan quality optimization.

Therefore, our contribution involves the application of automated task decomposition, goal assignment, and required cooperation MAP techniques to temporal tasks. This results in a planning algorithm that exploits the MA and concurrent nature of temporal MA domains: MA-LAMA, a factored, centralized, unthreaded temporal MAP system that operates with mixed pre-planning coordination and iterative response planning, internal communication, and local heuristic search.

The paper is structured as follows. The next section presents the related literature. Then MA-LAMA is presented, first broadly and after, in a detailed per-component view. Finally, we provide an empirical evaluation of the planner, followed by the conclusions and future work.

Related Work

Literature on both MAP and temporal planning is extensive as separate lines of work. To our knowledge, TFPOP (Brafman and Domshlak 2008) is the only MA planner that is able to deal both with time and durative-actions. It follows a centralized scheme, producing non-linear plans that maintain a thread of sequentially ordered actions per agent, exploiting the concept of *coordination points* for loosely coupled agents and through CSP+planning. However, it was not compared to any other MAP solver.

Regarding MA planning, some examples of threaded MA planners are: partial-order-planning (POP) based planners, as MH-FMAP (Torreño, Sapena, and Onaindia 2015), which computes distributed versions of h_{DTG} and h_{Land} heuristics; MAD-A^* (Nissim and Brafman 2013), an optimal solver based on local agent evaluation of each state; GPPP (Maliah, Shani, and Stern 2014), which builds a relaxed public plan before each local planning stage; and MAPlan (Fišer, Štolba, and Komenda 2015), which follows a flexible distributed implementation of local agent state search methods, as well as both local and global heuristics, as h_{FF} and *LM-Cut*, showing strong performance especially with distributed computation agents.

For unthreaded MA planners, some approaches are: PMR (Luis, Fernández, and Borrajo 2020), based on plan merging and reuse with simultaneous planning by all agents; CMAP (Borrajo and Fernández 2015), with weak privacy preservation in agents assembly and single-agent search; Distoplan (Fabre et al. 2010), an optimal planner that exploits independence between agents, not limiting their possible interactions beforehand; PSM (Tožička, Jakubuv, and Komenda 2015), which expands Distoplan and introduces Planning State Machines: agent local task representations that can be merged or projected; $A^\#$ (Jezequel and Fabre 2012), with cost informed and constrained factored planning following A^* search; and DPP (Maliah, Shani, and Stern 2016), one of the best performing unthreaded MAP solvers through accurate public projection of MAP task information with object cardinality privacy-preserving.

Other MA techniques include symmetry score based task decomposition for classical (Nissim, Apsel, and Brafman 2012) and numeric planning (Shleyfman, Kuroiwa, and Beck 2023).

In our case, MA-LAMA deals with MAP solving with a traditional approach, considering sequential total-order planning. We also make use of several MAP techniques not yet mentioned, such as: task decomposition into local search regions (Lansky 1991), exploiting loosely coupled agents from TFPOP, *coordination points* detection and constraints definition from Planning First (Nissim, Brafman, and Domshlak 2010), distributed planning graphs with coordination constraints from DPGM (Pellier 2010), required cooperation (Zhang, Sreedharan, and Kambhampati 2016), also used in the MARC planner (Sreedharan, Zhang, and Kambhampati 2015); satisfiability through sequential MAP task solving from μ-SATPLAN (Dimopoulos, Hashmi, and Moraitis 2012), and automatic MAP agent decomposition from ADP (Crosby, Rovatsos, and Petrick 2013).

Several of these MA planners, and others, participated in the 2015 Competition of Distributed and Multi-Agent Planning (CoDMAP) (Komenda, Štolba, and Kovács 2016), being the top performers ADP in coverage and CMAP-q in quality for the centralized track, and PSM and MAPlan for the distributed track overall.

Regarding temporal planning, several temporal planners incorporate techniques that MA-LAMA makes use of, such as the *snap-actions* paradigm, continuous numeric effects treatment and temporal frontier state constraints from OPTIC (Benton, Coles, and Coles 2012), from the COLIN (Coles et al. 2012) family of planners; and TFLAP (Sapena, Marzal, and Onaindia 2018) multi-heuristic search based on h_{FF} and h_{Land}, which had good performance in the 2018 IPC temporal track.

Other participants in this competition were POPCORN (Savaş et al. 2016), which is able to operate with control parameters, TemPorAl (Cenamor et al. 2018), a portfolio that was the top performer in the competition, and CP4TP (Furelos-Blanco and Jonsson 2018), another portfolio. From 2014 IPC temporal track, notable participants were: IT-SAT (Feyzbakhsh Rankooh and Ghassem-Sani 2015), a SAT-Based Temporally Expressive Planner; YAHSP3 (Vidal 2014), which computes lookahead relaxed plans and uses them in state-space heuristic search; and Temporal FD (TFD) (Eyerich, Mattmüller, and Röger 2012), that uses the context-enhanced additive heuristic over a temporal search space.

In contrast, MA-LAMA opens a new way to study temporal domains, as we aim to deal with the temporal complexity of concurrent actions search spaces with MAP techniques, and only make use of temporal techniques to maintain temporal and numeric soundness.

Background

Following PDDL2.1 semantics (Fox and Long 2003), we define the input for our planning algorithm as:

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

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

Instantaneous actions, $a_{inst} \in O_{inst}$, and durative actions, $a_{dur} \in O_{dur}$, differ from each other in the fact that durative actions take time, $dur(a_{dur})$, to perform a state transition from s to s'. Instantaneous actions preconditions, $pre(a_{inst})$, and effects, $eff(a_{inst})$ are expanded to start conditions, $startCond(a_{dur})$, end conditions, $endCond(a_{dur})$, over all $dur(a_{dur})$ conditions, $inv(a_{dur})$, start effects, $startEff(a_{dur})$, end effects $endEff(a_{dur})$ and numeric effects, $contEff(a_{dur})$.

Start and end endpoints of a_{dur}, $a_{\vdash}$ and $a_{\dashv}$, can be encoded as instantaneous actions, with $pre(a_{\vdash}) = startCond(a_{dur})$, $eff(a_{\vdash}) = startEff(a_{dur})$, $pre(a_{\dashv}) = endCond(a_{dur})$, and $eff(a_{\dashv}) = endEff(a_{dur})$. This decomposition produces *snap actions*, allowing planners to reason with concurrent operators, as it allows them to overlap. The invariant condition, $inv(adur)$, for a durative action (a_{dur}) must be maintained throughout the open interval between $a_{\vdash}$ and $a_{\dashv}$.

For the numeric effects, we impose the same restrictions as the COLIN family of planners: the contribution of any durative action to the rate of change of each numeric fluent,

$\upsilon \in \vartheta$, remains constant, so the rate of change of a certain variable, $\delta\upsilon$, is only modified when the *snap actions* are applied.

Additionally, a metric, M, can be defined to determine the quality of a plan, and it would be the planner's duty to find which plan achieves a higher optimization of that metric. Different planners support a wide range of metric formulations, as the *total-cost* implementation in LAMA (Richter and Westphal 2010) and time-dependent continuous costs in OPTIC. In our case, we define M as a set of weighted numeric variables, $\{w_1 * \upsilon_1, w_2 * \upsilon_2, ..., w_n * \upsilon_n\}$ where $\upsilon_n \in \vartheta$ and w_n is a real number. υ can also be the duration of the plan, the *total-time*.

From this input, MA-LAMA translates it and utilises internally the multi-valued planning tasks (MPTs) representation, also referred as SAS^+ planning problems (Bäckström and Nebel 1995). For ease of presentation, we consider a simplified version of MPTs that omits axioms and with conditional effects compiled away.

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

- *V is a finite set of multi-valued variables υ,*
- *I is the initial state for V,*
- *G is the goal condition and a partial state of V, and*
- *A is a set of instantaneous actions.*

The key difference between the two planning task representations is that variables in an MPT are multi-valued, rather than standard booleans in PDDL2.1. However, for our work we need to extend the MPT definition to fully encompass all the temporal task components, thus, we include the set of multi-valued numeric variables, *N*, and the metric, *M*, resulting in our extended MPT (eMPT) definition:

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

- *V, I, G, A are defined as in the MPT Definition.*
- *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 allows us to deal with any numeric operation as an *effect* over N. We will expand on how to calculate the possible values for each numeric variable n in the MA-LAMA details section.

The MPT representation also allows us to build the *causal graph* (CG), which represents dependencies among variables according to the available actions and is the root of the MA task decomposition techniques.

Definition 4. Causal Graphs
For a MPT Φ with a set of variables V, the Causal Graph of Φ, $CG(\Phi)$ is the directed graph with a set of vertex V that contains an arc (υ, υ') iff $\upsilon \neq \upsilon'$ and there exists an action that can affect the value of υ', requiring a precondition that specifies the value of υ.

MA-LAMA Overview

MA-LAMA is a satisfying temporal planner that utilizes MA task decomposition and required cooperation techniques to deal with the temporal complexity of concurrent action search spaces. Additionally, it is designed to deliver fast and highly optimized solutions for MAP tasks.

Let us provide a comprehensive overview of the internal functioning of MA-LAMA, shown in Figure 1. The first step involves a translation of the temporal task, where the durative actions are transformed to *snap actions*, adding the same control predicates between $a_\vdash$ and $a_\dashv$ actions as the COLIN planner (Coles et al. 2012). Then, the *snap* temporal task is encoded into an eMPT and two MA algorithms take place:

- *Agent Decomposition (AD)*: following the ideas of ADP (Crosby, Rovatsos, and Petrick 2013), MA-LAMA decomposes the eMPT in terms of mostly independent entities, called agents.
- *Goal Categorization and Assignment (GCA)*: *coordination points* are computed following the principles of required cooperation (Zhang, Sreedharan, and Kambhampati 2016), and the task goals are categorized into *cooperation* and *coordination* subsets. Then, the goals are assigned to agents based on metric estimations, creating individual eMPTs to be solved subsequently and in groups, in what we call *Search Phases*.

We impose sequenced decision-making between these two algorithms, as our findings indicate that making use in the AD of the information obtained in the GCA does not guarantee to lead the planner to better solutions. Instead, we found that these are linked with near-optimal solving of local eMPTs, and thus, the AD is designed to prioritize smaller agents and, subsequently, simpler eMPTs.

Several eMPTs are solved for each Search Phase, taking as an input the *temporal constraints* imposed by the previous agent eMPT and, similarly, Search Phases inherit the initial state from the solution of the previously solved one.

For each eMPT, we launch a modified version of the LAMA planner, with numeric, temporal and constraints frameworks built on top. Each eMPT is solved by a *WA** search using two classical heuristics: h_{FF} (Hoffmann and Nebel 2001) and h_{Land} (Hoffmann, Porteous, and Sebastia 2004) (Sebastia, Onaindia, and Marzal 2006).

All eMPTs in each Search Phase generate a partial plan based on *snap actions*, so, after all phases are solved, we need to launch a Unify module that translates them to the temporal paradigm and assembles them, checking their temporal soundness and producing the full temporal plan.

Finally, MA-LAMA preserves *weak* privacy if a decomposition is found by the AD.

MA-LAMA Insight

MA-LAMA makes use of the LAMA MPT representation, inherited from Fast Downward (FD) (Helmert 2006), which only needs to be modified to support the snap task numeric

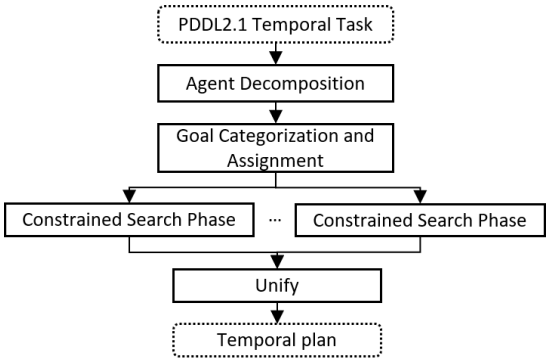

Figure 1: MA-LAMA general structure. The input is a PDDL2.1 temporal task that is decomposed to launch several constrained Search Phases. All partial plans are unified at the end to produce a full temporal plan as output.

conditions, *numCond*, and any form of *contEff* during the temporal task instantiation and multi-valued variables computation through the invariant search.

Additionally, as we have expanded the original MPTs definition to eMPTs, we need to determine the possible values for the multi-valued numeric variables, $n \in N$. For each numeric variable n, the set of states, ϵ is obtained by: $\forall a \in A \rightarrow \epsilon = \epsilon \cup \{n \in N : n \in \textit{contEff}(a)\}$, plus the *undefined state,* u. *contEff*(a) and *numCond*(a) are expanded so that all numeric operations that can be solved before the search are solved. Thus, in our eMPTs and for each fluent numeric variable, we encode the current numeric value and last applied numeric effect.

Next, we will review in detail each MA-LAMA execution module, starting with Agent Decomposition (AD) and Goal Categorization and Assignment (GCA), and continuing with Constrained Search and Unify, as shown in Figure 1.

Agent Decomposition

Most variables decomposition divides the domain variables between *private*, variables (that cannot be changed by and are not required by other agents) and *public* (representing the environment in which the agents operate), P.

We borrow from Crosby et al. (2013) their *variable decomposition* definition. In short, a variable decomposition of an eMPT divides it into a set of variable groups, $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_n\}$, and a set of public variables, P, that can be empty. In their work, they also define *Agent Variable Decompositions* (ADP); variable decompositions where there are no *joint actions* and no external actions that can affect any agent internal variables. We choose not to impose this restriction and, therefore, we obtain not merged single agents and will deal with *required cooperation* in a later stage.

Algorithm 1 gives a pseudo-code overview of the MA-LAMA Agent Decomposition (AD) module, which produces a set of eMPTs $\langle V, I, G, A, N, M\rangle$, where all components are set except G.

We consider three main stages in the AD module:

Algorithm 1: Agent Decomposition (AD)

Input: $eMPT\ \langle V, I, G, A, N, M\rangle$
Output: Agent Decomposed Task set $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$

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

1) Find Possible Agents starts with the ADP basis: removing 2-way cycles from the CG and taking resulting root nodes that still have one successor left as possible agents. Then, the algorithms differ, as we aim for different objectives with the decomposition: minimize local agent search space for MA-LAMA, and minimize mandatory agent coordination in ADP.

We run an *Assemble Agents* step before expanding all agent sets so that possible agents are more refined and really coupled agents are merged. The full assembly step can be summarized in one rule:

For two possible agents $[\upsilon, \upsilon']$ *in a root node set* Ω, υ *and* υ' *are assembled if there is a path between* υ *and* υ' *in the* CG.

2) Extend Private Agent Sets Agents sets are then expanded so that every $\upsilon \in V$ that is the only successor of an agent set, Ω_n, is added to it. This process is repeated until all sets can no longer be refined, and a set of actions is added to each agent such that, for every action $a \in A$, it is added to the private set of actions of a certain agent, A_n, if there exists a variable υ that exists in *eff*(a) or *pre*(a).

This results in a *private* set of variables, Ω_n and actions A_n for each agent; therefore, the rest of the variables and actions are assumed *public*, P.

3) Extend Public Agent Sets Agent variable sets are completed with all the variables that are reachable in the CG for a certain agent, n: actions not yet assigned are added to

Ω_n following the same past rule, but using the new complete variable set; numeric variables N and the metric M components are added to N_n and M_n if they appear in the *conteff(a)* or *numCond(a)* for every $a \in A_n$; and the initial state I values are added to an agent n if they appear in V_n.

Finally, the output of the algorithm is a set of tasks, $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$, one for each agent and without any goal assigned. This set is the input for the Goal Categorization and Assignment stage.

Theorem 1 *The algorithm presented in this sub-section for finding an agent decomposition given an eMPT is both sound and complete*

Sketch of Proof We rely on Crosby et al. (2013) Theorem 6.1 Sketch of Proof, as we share most of the decomposition definition with them. First, if variables identified during the "Find Possible Agents" stage also serve as root nodes in the causal graph, it is guaranteed that a decomposition will be produced. Second, if a decomposition exists, the "Find Possible Agents" stage will always identify a minimum of two root variables that will not be merged. Finally, as we do not impose the "merge restriction" to our definition, not merged linked in the *CG* agents do not affect soundness.

Goal Categorization and Assignment

Algorithm 2 shows the Goal Categorization and Assignment (GCA). The objective is to further exploit the MA nature of the domains by studying how the goals from the original temporal task, G, can be assigned to the agent eMPT set, Φ, to achieve optimized solutions.

The basis of the GCA algorithm is the Required Cooperation (RC) Analysis (Zhang, Sreedharan, and Kambhampati 2016), in which they formally describe the possible agent interactions within an MPT. From now on, we will refer to goals that require Type-1 RC (Heterogeneous Agents) or Type-2 RC (Homogeneous Agents) Causal Loops interactions as *coordination* goals and goals that require Type-2 RC Traversability interactions, or no RC at all, as *cooperation* goals. The output of the GCA is a set of *Search Phases*, each one aiming to solve a goal subset of *coordination* or *cooperation* $\{g\} \in G$.

We consider three main stages for the GCA module:

1) Coordination Points Variables *Coordination points* are certain points in an agent plan where it possibly influences or is influenced by other agents. Following this idea, we obtain variables that may be *coordination points* in our Φ by extracting from P all variables that are both a precondition in one agent's actions and an effect in another agent's actions.

2) Single Goal Relaxed Plans Obtention Then, we launch a relaxed (ignore delete effects and ignore *numCond()*) search for each goal $g \in G$ and agent eMPT $\Phi_n \in \Phi$ and calculate a metric value for the relaxed solution, computing numeric variables limits and using the worst-case scenario for fluents in *contEff()*.

If a solution is found for any of Φ_n, then the goal is considered a *cooperation* goal, and our aim is then changed to find the most optimized relaxed solution through iterative relaxed searches and for each agent that can achieve it.

If a solution is not found, g requires *coordination*, and a relaxed search is launched for the same g with the full task eMPT, trying to minimize the number of used agents in the relaxed solution. The values for variables selected as possible *coordination points* are stored for later use in the Search Steps creation. The optimization of all relaxed search processes takes place for a configurable amount of time or until the complete search space has been explored.

3) Goal Assignment and Search Phases Creation We deal with *coordination* and *cooperation* goals in different ways:

- for *cooperation* goals, all goals are assigned in a way that minimizes the total sum of all relaxed metrics in a single Search Phase, σ_1, where agents involved, $\Phi_n \in \Phi$, are those with at least one goal assigned, and
- for *coordination* goals, one Search Phase is created for each, $\{\sigma_2, ..., \sigma_n\}$, where agents involved are those that appear in the relaxed plan, additionally, the *coordination points* for each agent are also assigned as goals.

We check if the assignments are valid from a numeric conditions perspective before continuing. If there is no relaxed solution for one of the agents in a Search Phase, the process is restarted assigning weights to each agent metric estimation, lowering the use of constrained agents.

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 Search Phases, $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$

Coordination Points Variables
$CoorP \leftarrow \emptyset$ (Coordination Points)
for $\{\Phi_n, \Phi_m\} \in \binom{\Phi}{2}$ **do**
  $CoorP \leftarrow CoorP \cup \{v \in P : \exists(a_n \in A_n, a_m \in A_m) : v \in$ *pre*$(a_n) \wedge v \in$ *eff*$(a_m)\}$
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 $\Phi \in$ **do**
    $Sol_g \leftarrow Sol_g \cup$ **relaxedSearch**(Φ_n, g)
  end for
  if $Sol_g \neq \emptyset$ **then**
    $G_{coop} \leftarrow G_{coop} \cup g$, Sol_g
  else
    $Sol_g \leftarrow$ **relaxedSearch**(Φ, g)
    $Sol_g \leftarrow CoorP$ **in** Sol_g
    $G_{coord} \leftarrow G_{coord} \cup g$, Sol_g
  end if
end for
Goal Assignment and Search Phases Creation
$\sigma_{coop} \leftarrow G_{coop}, MinCostAssignment(\Phi, G_{coop})$
$\sigma_{coord} \leftarrow \emptyset$
for $g_{coord} \in G_{coord}$ **do**
  $\sigma_n \leftarrow g_{coord} \cup CoorP$ **in** $g_{coord}.Sol_g$, Φ_n **in** $g_{coord}.Sol_g$
  $\sigma_{coord} \leftarrow \sigma_{coord} \cup \sigma_n$
end for
return $\sigma_{coop} \cup \sigma_{coord}$

Theorem 2 *The algorithm presented in this sub-section for goal assignment among an agent task set is both sound and complete.*

Sketch of Proof We demonstrate that, given any valid eMPT set, a valid and solvable goal assignment is generated. First, if no decomposition has been found, all goals are assigned to the full task. Second, if a certain goal can be attained by the full task, it is assigned to an agent that can also attain it, as our first relaxed searches only make use of each single agent operator, or it is marked as a *coordination* goal and is assigned to a group of involved agents through a full task relaxed-search. Finally, it can be demonstrated that *coordination points* for *coordination* goals cover the remaining case from the AD, involving merging linked agent sets in the causal graph.

Constrained Search

The Constrained Search process receives as an input the full set of Search Phases, $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$, each with a set of eMPT, $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_n\}$. We launch a multi-heuristic constrained search over each eMPT, inheriting *temporal constraints*, (Φ_n, v, t, d), set by the public variables, P, between each agent search, so that interactions between agents are only considered when necessary.

Definition 5. Temporal Constraints
A Temporal Constraint, (Φ_n, v, t, d), states that the agent, Φ, sets a certain value, v, at the moment, t, for the a certain duration, d.

We will now divide the search description in two: first, each individual search characteristics are explained, and then, details on solving whole Search Phases are provided.

1) Search details For each eMPT, We launch a *WA** forward total-order search with two classical heuristics, h_{FF} (Hoffmann and Nebel 2001), Cost-Sensitive FF/add variant, and h_Land (Hoffmann, Porteous, and Sebastia 2004) (Sebastia, Onaindia, and Marzal 2006). This choice is based on the fact that most MA temporal domains revolve their temporal complexity over the concepts of *cooperation* and *coordination*, so the eMPTs that we solve at this point tend to not require complex necessary simultaneity.

During search, all states are evaluated concerning both heuristics, and, when choosing which state to expand, the search algorithm alternates between both based on numerical priorities. Inherited from the LAMA planner, we also make use of *preferred-operators*, which represent operators that are estimated to be useful in a given state.

A temporal framework is introduced to deal with local concurrency, incorporating constraints among snap-actions to guarantee that the preconditions for the new actions are satisfied in the frontier state, as well as keep track of the makespan and running actions' start-end times for each state.

We follow the same principles as the numeric framework, including the necessary mechanisms to be able to deal with continuous numeric operations and numeric preconditions.

Note that our only aim with these frameworks is to guarantee temporal and numeric soundness.

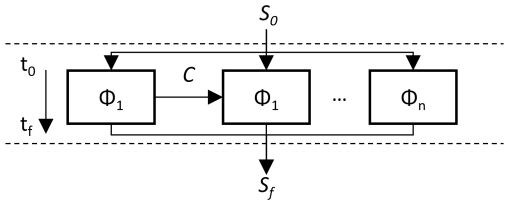

Figure 2: A Cooperation Search Phase: each agent, Φ_n, produces *temporal constraints* for the next. All agents share the initial state, S_0, and produce a combined final state, S_f. The phase makespan, t, is the longest agent plan makespan.

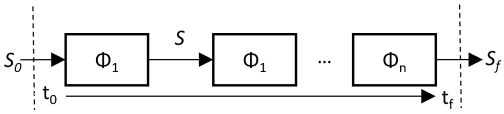

Figure 3: Coordination Search Phase structure, where each agent, Φ_n, produces the initial state, S, for the next. The final state, S_f is produced by the last agent. The makespan of the phase, t, is the sum of all agent local plans.

2) Solving Search Phases All Search Phases need a *temporal constraints* system for two main purposes:

- for *cooperation* goals, to assure that restrictions over variables $v \in P$ are preserved, and
- for *coordination* goals, to synchronize the agents around the *coordination points*.

In practice, *temporal constraints* are used as conditions in *cooperation* Search Phases (see Figure 2), computing them as *inv()* = v, starting at time, t, for the duration, d. When a *cooperation* agent finds a solution, it also computes a set of Temporal Constraints, containing all times a variable $v \in P$ value was required or changed. Consecutive agents use this list as limitations to their own local search graphs and add their own restrictions when they find a solution.

We solve *coordination* Search Phases, Figure 3, following the agent order dictated by the *coordination points*, obtaining the *coordination* goal at the end. Each agent inherits from the last a set of *temporal constraints* representing the already obtained subgoals and end states, which serve as the initial state for each local search.

Metric optimization in each Search Phase type is achieved differently. First, a *cooperation* Search Phase will be as optimized as its individual eMPTs are. Since several goals are achieved per agent, the ones with more goals are prioritized, so they solve their eMPTs less restricted by *temporal constraints*. *Coordination* Search Phases eMPTs generally only solve one goal, so they are not as promising in terms of numeric optimization. On the other hand, temporal concurrency can still be improved, and it is handled in the next step.

Unify

The final step in MA-LAMA execution consists of the unification of all partial plans for each Search Phase to obtain a full temporal plan.

Domain	MA-L	CMAP	ADP		
Blocksworld	**20**	19 (1)	**20**		
Depot	**17 (3)**	**17 (3)**	16 (4)		
DriverLog	**20**	19 (1)	**20**		
Elevators	**20**	18 (2)	**20**		
Logistics	**20**	19 (1)	**20**		
Rovers	**20**	**20**	**20**		
Satellites	**20**	**20**	**20**		
Sokoban	14 (6)	13 (7)	**17 (3)**		
Taxi	**20**	**20**	**20**		
Wireless	6 (14)	5 (15)	**9 (11)**		
Woodwork	16 (4)	15 (5)	**20**		
Zenotravel	**20**	19 (1)	**20**		
Total	213	204	**222**		
Domain	MA-L	TFLAP	OPTIC	TFD	POPC
Rovers	**11**	1 (10)	8 (3)	10 (1)	6 (5)
Satellite	**11**	**11**	7 (4)	5 (6)	0 (11)
Zenotravel	**16**	14 (2)	8 (8)	**16**	7 (9)
Logistics	**20**	**20**	**20**	**20**	0 (20)
Taxi	**20**	18 (2)	**20**	**20**	**20**
Total	**78**	54	63	71	33

Table 1: Coverage results (not solved domains in parenthesis) for CoDMAP non-temporal (up) and CoDMAP and IPC temporal (down) MA domains. All executions are limited to 10 minutes and 4GB of RAM.

The unification process of each agent in a Search Phase, $\Phi_n \in \sigma_j$, *snap* partial plan, is simple, as we have already dealt with concurrency and constraints in all cases but in between *coordination* Search Phases. The partial plans for each Search Phase are obtained by assembling each Φ_n partial plan concurrently for *cooperation* Search Phases, and consecutively in *coordination* Search Phases.

To combine all partial plans, we first check for each *coordination* Search Phase pair, σ_n, σ_m, and the variables from P that are affected in their respective *snap* partial plans, P_n, P_m. If $P_n \cap P_m = \emptyset$, then both σ_n and σ_m are added to the complete temporal plan concurrently.

Finally, all remaining partial plans are combined consecutively. During this process, we also calculate the total cost of the final plan; check that the temporal, numeric, and logic constraints soundness is maintained; and change the *snap* actions plan paradigm to temporal.

Experimentation

Our experimentation is divided into two sections. First, we study the coverage results of MA-LAMA in temporal and non-temporal domains against other classical and temporal solvers, and second, we analyze MA-LAMA quality performance against other state-of-the-art temporal planners.

Coverage for Classical and Temporal Tasks

We first check the coverage of MA-LAMA in classical and temporal domains, results can be seen in Table 1.

The initial MA-LAMA benchmark analysis is performed under classical planning, aiming to obtain the only feasible comparison with other MA planners. We study the CoDMAP domains against the ADP-legacy and CMAP-q planners. Those are the winners for the competition coverage and quality tracks, and both share internal functionalities with MA-LAMA: ADP-legacy, which shares the root of the agent decomposition, and CMAP-q, which obfuscates the domain and uses the LAMA planner during the search.

We obtain a similar coverage performance to ADP-legacy when an agent decomposition is found, and to CMAP-q if the decomposition is invalid (does not match the competition original domain). Compared to ADP, the slightly worse coverage performance of MA-LAMA is explained by this, as we solve fewer domains when no decomposition is found.

For MA temporal domains, we compare against other state-of-the-art temporal planners, already mentioned in the related work section: OPTIC, TFLAP, TFD and POPCORN. We exclude Yahsp3, as its single-thread version is outperformed by TFD and does not work with metrics in our tests, and both TemPorAl and CP4TP, as they are portfolios. The domains we chose are Rovers, Satellites, and Zenotravel from IPC, and adapted Taxis and Logistics from CodMAP to make them temporal, as Logistics presents required cooperation and Taxis allows us to check a less complex scenario.

In this case, MA-LAMA outperforms all planners, especially for the IPC domains, where only TFLAP in Satellites and TFD in Zenotravel are able to match MA-LAMA. Regarding the agent decomposition, we obtain the ones a human operator would set: planes in Zenotravel, rovers in Rovers, satellites in Satellites, planes and trucks in Logistics, and taxis in Taxis. Additionally, only Logistics contained *coordination* goals, but this did not affect coverage performance.

We can conclude that the AD and GCA algorithms are suitable to deal with MA temporal domains and that MA-LAMA is able to solve a wide range of problems in MA classical domains, and improves state-of-the-art coverage performance for the most complex MA temporal domains.

Plan Quality Performance

In this section, we will study the plan quality performance in MA temporal domains, the main focus of our planner. For the reasons we outlined in the previous point, we launched the same five planners: OPTIC, TFLAP, TFD, POPCORN, and MA-LAMA. Results are detailed in Table 2. We consider problems once the MA nature becomes relevant.

MA-LAMA dominates in Rovers (weighted battery) and Zenotravel (weighted fuel + makespan) domains, delivering better quality plans in all instances for Rovers and almost all for Zenotravel, where TFLAP also wins in some problems. These two domains are launched with metrics that are not completely coupled with the makespan of the plan, and the two best-performing planners (TFLAP only in Zenotravel) make use of classical heuristics and do not reason with time, meaning that this can be an advantage when optimizing metrics not coupled with the makespan.

For Satellites (makespan), results are mixed, as MA-LAMA gives better solutions in four problems and OPTIC and TFD in three. Planners that reason with time perform better with the makespan metric, but as instances get more complex, MA-LAMA deals better with them and delivers better solutions in the last problems.

Domain	Planner	9	10	11	12	13	14	15	16	17	18	19	20
Rovers (w battery)	MA-L		**48.6**	**19.0**	**89.0**	**155.8**	**79.8**	**102.3**	**77.0**	**135.3**	**33.2**	**313.4**	**388.6**
	POPC		130.2	37.2	93.4	-	-	156.8	184.3	-	136.6	-	-
	TFD		101.1	47.2	100	172.1	94.1	215.2	100.2	214.6	224.4	346.0	-
	TFLAP		-	-	-	-	-	137.3	-	-	-	-	-
	OPTIC		99.9	38.8	96.5	215.6	143.8	184.7	198.7	217.4	-	-	-
Satellites (makespan)	MA-L		134.5	205.9	230.3	132.9	129.1	**192.2**	**120.2**	93.5	153.6	**247.9**	**555.9**
	TFD		-	-	262.5	**104.1**	**87.7**	-	-	-	**93.9**	283.4	-
	TFLAP		217.6	479.4	385.4	634.9	361.0	462.9	370.2	385.5	296.3	509.5	948.8
	OPTIC		**115.7**	**150.9**	**171.8**	149.1	108.6	-	-	**83.7**	100.6	-	-
Zenotravel (makespan + w fuel)	MA-L	**35.9**	237.7	**76.7**	103.2	123.7	**268.3**	182.6	**210.3**	**324.4**	**154.6**	**373.4**	**398.2**
	POPC	719.7	-	313.7	213.5	-	-	-	-	-	-	-	-
	TFD	652.0	268.7	208.2	146.4	194.7	1464.6	242.8	574.4	1756.1	880.9	1674.4	3301.3
	TFLAP	214.0	**196.1**	81.8	**81.9**	**92.6**	391.0	**133.0**	303.7	504.8	414.6	-	-
	OPTIC	131.0	283.3	-	400.3	130.1	291.7	-	-	-	-	-	-
Logistics (w fuel)	MA-L	**177.0**	168.0	192.0	**247.0**	188.0	263.0	**231.9**	**210.0**	**170.5**	308.4	266.2	**197.0**
	TFD	**177.0**	168.0	304.0	**247.0**	168.0	521.0	434.4	332.4	216.2	290.0	**221.0**	235.2
	TFLAP	**177.0**	**158.0**	**172.0**	**247.0**	**158.0**	**239.0**	287.0	244.0	217.0	**264.8**	311.0	256.0
	OPTIC	187.0	**158.0**	206.0	**247.0**	178.0	253.0	272.4	213.9	192.8	318.0	313.2	229.2

Table 2: Plan quality for IPC benchmarks (Rovers, Satellites and Zenotravel), and temporal CodMAP domains (Logistics and Taxi), limited to 10 minutes and 4GB of RAM. Absence of a planner indicates that it solved no problems. "w" means weighted.

CodMAP logistics, which includes mandatory *coordination* goals, is significantly less complex than the IPC domains, as several planners find optimal (or near optimal) solutions for several instances. In these cases, MA-LAMA depends on the GCA algorithm to deliver the best solution, as a non-optimal goal assignment results in a non-optimal final solution. Similarly to the previous case, as problems incorporate more agents and variables, all planners begin to struggle to optimize the solutions, and the MA nature of MA-LAMA proves to be an advantage.

Lastly, we study the search time for each planner in the Rovers domain, shown in Figure 4. POPCORN and TFLAP are excluded since they do not solve most problems. In less complex instances, the search time for all planners is similar, however, once the number of agents increases, OPTIC stops finding solutions and TFD notably increases its search time. In contrast, MA-LAMA search time is stable through the whole domain, notably improving search time in instances with high agent counts, and only displaying peaks when the individual agent eMPTs are harder, as in *p13*.

To conclude, experiments infer that the MA techniques in MA-LAMA perform suitable decompositions for MA temporal domains and that MA-LAMA delivers better plan quality performance than other state-of-the-art temporal solvers in the most complex problems for all tested domains.

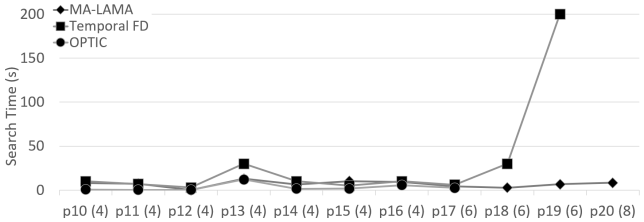

Figure 4: Search Time in seconds for each problem (n^o *of agents*) in Rovers IPC domain.

Conclusions and Future Directions

This paper introduced MA-LAMA, a multi-agent planner optimized for temporal domains, leveraging concurrent actions search spaces and employing Agent Decomposition (AD) and Goal Categorization and Assignment (GCA) algorithms for the effective decomposition of tasks. This approach notably enhances plan quality and coverage in complex MA temporal domains, surpassing existing state-of-the-art temporal planners.

However, MA-LAMA's current limitations include its inability to incorporate temporal or numeric information during search, affecting performance in domains lacking clear decomposition. Future enhancements could integrate techniques like symmetry-based decompositions or search time reasoning to broaden its applicability across a wider range of MA domains. Furthermore, the AD algorithm's potential expansion to more comprehensively address MA domain challenges presents a promising avenue for research.

The exploration of MA-LAMA alongside other planners in a portfolio approach, predicated on a pre-search analysis of domain structures, could further optimize planner selection and performance across varied scenarios. As we continue to refine MA-LAMA, addressing these limitations and exploring new integration strategies will be pivotal in advancing MA planning efficiency and effectiveness.

Acknowledgments

This work is supported by JCLM project SBPLY/19/180501/000024 and TNO.

References

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. *Proc. of the*

Int. Conf. on Automated Planning and Scheduling, 22(1): 2–10.

Borrajo, D.; and Fernández, S. 2015. MAPR and CMAP. *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15)*, 1–3.

Brafman, R. I.; and Domshlak, C. 2008. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. *Proc. of the Int. Conf. on Int. Conf. on Automated Planning and Scheduling*, 28–35.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS+ Planning. *Computational Intelligence*, 11: 625–656.

Cenamor, I.; Vallati, M.; Chrpa, L.; de la Rosa, T.; and Fernández, F. 2018. TemPoRal: Temporal Portfolio Algorithm. https://ipc2018-temporal.bitbucket.io/planner-abstracts/team1. Accessed: 2022-09-01.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2012. COLIN: Planning with Continuous Linear Numeric Change. *Journal of Artificial Intelligence Research*, 44: 1–96.

Crosby, M.; Rovatsos, M.; and Petrick, R. 2013. Automated Agent Decomposition for Classical Planning. *Proc. of the Int. Conf. on Automated Planning and Scheduling*, 23(1): 46–54.

Dimopoulos, Y.; Hashmi, M. A.; and Moraitis, P. 2012. μ-SATPLAN: Multi-agent planning as satisfiability. *Knowledge-Based Systems*, 29: 54–62. Artificial Intelligence 2010.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2012. *Using the Context-Enhanced Additive Heuristic for Temporal and Numeric Planning*, 49–64. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-25116-0.

Fabre, E.; Jezequel, L.; Haslum, P.; and Thiébaux, S. 2010. Cost-Optimal Factored Planning: Promises and Pitfalls. *Proc. of the Int. Conf. on Int. Conf. on Automated Planning and Scheduling*, 65–72.

Feyzbakhsh Rankooh, M.; and Ghassem-Sani, G. 2015. ITSAT: An efficient SAT-based temporal planner. *Journal of Artificial Intelligence Research*, 53: 541–632.

Fišer, D.; Štolba, M.; and Komenda, A. 2015. MAPlan. *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15)*, 8–10.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20: 61–124.

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

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research*, 22: 215–278.

Jezequel, L.; and Fabre, E. 2012. A#: A distributed version of A* for factored planning. *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 7377–7382.

Komenda, A.; Štolba, M.; and Kovács, D. 2016. The International Competition of Distributed and Multiagent Planners (CoDMAP). *AI Magazine*, 37: 109–115.

Lansky, A. L. 1991. Localized Search for Multiagent Planning. *Proc. of the Int. Joint Conf. on Artificial Intelligence*, 1: 252–258.

Luis, N.; Fernández, S.; and Borrajo, D. 2020. Plan Merging by Reuse for Multi-Agent Planning. *Applied Intelligence*, 50(2): 365–396.

Maliah, S.; Shani, G.; and Stern, R. 2014. Privacy preserving landmark detection. *Frontiers in Artificial Intelligence and Applications*, 263: 597–602.

Maliah, S.; Shani, G.; and Stern, R. 2016. Stronger Privacy Preserving Projections for Multi-Agent Planning. *Proc. of the Int. Conf. on Automated Planning and Scheduling*, 26(1): 221–229.

Nissim, R.; Apsel, U.; and Brafman, R. I. 2012. Tunneling and Decomposition-Based State Reduction for Optimal Planning. *European Conf. on Artificial Intelligence*.

Nissim, R.; and Brafman, R. 2013. Distributed Heuristic Forward Search for Multi-Agent Systems. *Journal of Artificial Intelligence Research*, 51.

Nissim, R.; Brafman, R. I.; and Domshlak, C. 2010. A General, Fully Distributed Multi-Agent Planning Algorithm. *Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems*, 1: 1323–1330.

Pellier, D. 2010. Distributed Planning Through Graph Merging. *Proc of the Int. Conf. on Agents and Artificial Intelligence*, 2: 128–134.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Rintanen, J. 2007. Complexity of Concurrent Temporal Planning. *Proc. of the Int. Conf. on Automated Planning and Scheduling*, 280–287.

Sapena, O.; Marzal, E.; and Onaindia, E. 2018. TFLAP: a temporal forward partial-order planner. https://ipc2018-temporal.bitbucket.io/planner-abstracts/team2. Accessed: 2022-09-01.

Savaş, E.; Fox, M.; Long, D.; and Magazzeni, D. 2016. Planning Using Actions with Control Parameters. *European Conf. on Artificial Intelligence*.

Sebastia, L.; Onaindia, E.; and Marzal, E. 2006. Decomposition of planning problems. *AI Communications*, 19: 49–81.

Shleyfman, A.; Kuroiwa, R.; and Beck, J. C. 2023. Symmetry Detection and Breaking in Linear Cost-Optimal Numeric Planning. *Proc. of the Int. Conf. on Automated Planning and Scheduling*, 33(1): 393–401.

Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2015. A first multi-agent planner for required cooperation (MARC). *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15)*, 17–20.

Torreño, A.; Onaindia, E.; Komenda, A.; and Štolba, M. 2017. Cooperative Multi-Agent Planning: A Survey. *ACM Computing Surveys*, 50(6): 1–32.

Torreño, A.; Sapena, O.; and Onaindia, E. 2015. Global Heuristics for Distributed Cooperative Multi-Agent Planning. *Proc. of the Int. Conf. on Automated Planning and Scheduling*, 25(1): 225–233.

Tožička, J.; Jakubuv, J.; and Komenda, A. 2015. PSM-based planners description for CoDMAP 2015 competition. *Proc. of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15)*, 29–32.

Vidal, V. 2014. YAHSP3 and YAHSP3-MT in the International Planning Competition. In *The International Planning Competition*, 64–65.

Zhang, Y.; Sreedharan, S.; and Kambhampati, S. 2016. A Formal Analysis of Required Cooperation in Multi-Agent Planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 26(1): 335–343.