

Branching and Pruning for Timeline-based Planning

Riccardo De Benedictis, Gloria Beraldo, Amedeo Cesta and Gabriella Cortellessa

Institute of Cognitive Sciences and Technologies (ISTC) - Via S. Martino della Battaglia 44, 00185 Roma (Italy) - National Research Council of Italy (CNR)

Abstract

One of the features that allowed classical planners to efficiently solve large problems is the ability their heuristics to prune large portions of the search space. These heuristics, however, by addressing classical approaches, do little to support the resolution of problems in which the temporal components are relevant and, even more so, they are not directly suitable for timeline-based approaches to automated planning. This paper takes advantage of some pruning techniques to increase the efficiency of timeline-based planning problems resolution. The elimination of some choices estimated as not very advantageous, in particular, allows the use of inference techniques to reduce the number of decisions to be made, reducing the risk of running into dead ends and, consequently, increasing the resolution efficiency of the solvers.

Keywords

Automated Planning, Timeline-based Planning, Heuristic search, Scheduling

1. Introduction

The introduction of domain independent *heuristics* within the Automated Planning [1] community immediately allowed solvers to efficiently solve large instances of complex problems. The different approaches that make up a solver's paraphernalia, range from the seminal h_{add} and h_{max} [2] to the more recent developments relying on *delete-relaxation*, like the h^{FF} heuristic [3] and the *causal graph* heuristics [4], on *landmarks*, like in [5, 6], on the *critical path*, like the h^m heuristic [7, 8] or, lastly, on *abstraction*, like in [9] or in [10, 11]. The salient aspects of such heuristics can be divided into the ability to direct the search process towards promising areas of the search space (*branching*) as well as the ability to avoid dead-ends that would require potentially expensive backtracking operations (*pruning*)[12].

While the above heuristics are significantly heterogeneous among them (although, often, they share some commonalities), they have in common the fact that they have been developed specifically for the resolution of a particular type of problem, characterized by a specific modeling language called PDDL [13], representing a natural evolution of the most long-lived STRIPS [14] formalism. Despite the PDDL, over the years, has been extended through different directions by introducing *durative-actions* and *numeric fluents* [15], *derived predicates* and *timed initial literals* [16], *continuous changes* [17], *state-trajectory constraints* and *preferences* [18] and *object-fluents*¹,

IPS-2022: 10th Italian Workshop on Planning and Scheduling, November 29, 2022, Udine, Italy

✉ riccardo.debenedictis@istc.cnr.it (R. De Benedictis); gloria.beraldo@istc.cnr.it (G. Beraldo);

amedeo.cesta@istc.cnr.it (A. Cesta); gabriella.cortellessa@istc.cnr.it (G. Cortellessa)

🆔 0000-0003-2344-4088 (R. De Benedictis); 0000-0001-8937-9739 (G. Beraldo); 0000-0002-0703-9122 (A. Cesta);

0000-0002-9835-1575 (G. Cortellessa)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

¹<http://www.plg.inf.uc3m.es/ipc2011-deterministic/attachments/Resources/kovacs-pddl-3.1-2011.pdf>

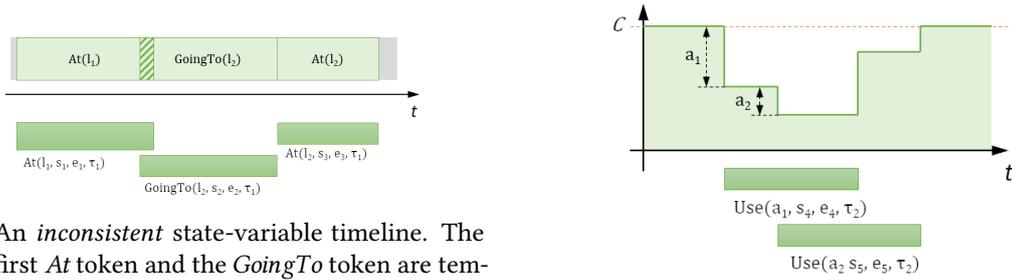
the development of heuristics for reasoning with these more expressive formal systems has remained relatively limited to a few cases (e.g., [19, 20]).

Timeline-based planning [21, 22] is an approach to automated planning which, by relying on partial-order planning [23], allows to generate plans which, during their execution, are, compared to the total order plans produced by classical planners, more easily adaptable. Analogously to the solvers reasoning upon the previous mentioned PDDL extensions, timeline-based planners have to cope with the high expressiveness of the formalisms which, despite making them particularly suited at addressing real-world applications, unavoidably leads to performance issues. In a recent work about timeline-based planning it has been shown that, thanks to the introduction of some domain-independent heuristics, the computation time could be effectively reduced [24]. The presented heuristics aim at directing the search process towards the most promising areas of the search space. As it will be shown in more detail in the following sections, however, the same data structures can also be used to partially avoid dead-ends and, consequently, potentially expensive backtracking operations.

2. Timeline-based planning

Timeline-based planning constitutes a form of deliberative reasoning which, in an integrated way, allows to carry out different forms of semantic and causal reasoning. Although this approach to planning has mostly been relegated to forms of causal reasoning in the space domain, many solvers have been proposed over the time like, for example, $\text{K}\text{T}\text{E}\text{T}$ [25], *EUROPA* [26], *ASPEN* [27], the *TRF* [28, 29] on which the *APSI* framework [30] relies and, more recently, *PLATINUM* [31]. Some theoretical works on timeline-based planning like [32, 26] were mostly dedicated to identifying connections with classical planning a-la PDDL [15]. The work on $\text{K}\text{T}\text{E}\text{T}$ and *TRF* has tried to clarify some keys underlying principles but mostly succeeded in underscoring the role of time and resource reasoning [33, 34]. The planner *CHIMP* [35] follows a Meta-CSP approach having meta-Constraints which heavily resembles timelines. The Flexible Acting and Planning Environment (*FAPE*) [36, 37] tightly integrates structures similar to timelines with acting. The Action Notation Modeling Language (*ANML*) [38] is an interesting development which combines the Hierarchical Task Network (*HTN*) [39, 40, 41] decomposition methods with the expressiveness of the timeline representation. Finally, it is worth mentioning that the timeline-based approaches have been often associated to resource managing capabilities. By leveraging on constraint-based approaches, most of the above approaches like $\text{K}\text{T}\text{E}\text{T}$ [42, 34], [43], [44] or [45] integrate planning and scheduling capabilities. Finally, [46] proposes a recent formalization of timeline-based planning.

Given the mentioned link with the heuristics we will refer, in this paper, to the timeline-based planning formalization as defined in [24]. According to this formalization, specifically, the basic building block of timeline-based planning is the *token* which, intuitively, is used to represent the single unit of information. Through their introduction and their constraining during the planning process, in particular, tokens allow to represent the different components of the high-level plans. In its most general form, a token is formally described by an expression like $n(x_0, \dots, x_i)_\chi$. In particular, n is a *predicate* symbol, x_0, \dots, x_i are its *parameters* (i.e., constants, numeric variables or object variables) and $\chi \in \{f, g\}$ is a constant representing the class of the



(a) An *inconsistent* state-variable timeline. The first *At* token and the *GoingTo* token are temporally overlapping. The inconsistency can be removed, for example, by introducing a $e_1 \leq s_2$ constraint.

(b) A *consistent* reusable-resource timeline. The overlap of tokens is allowed as long as the simultaneous use of the resource is less than its capacity.

Figure 1: Different timelines extracted by their associated tokens.

token (i.e., either a *fact* or a *goal*).

The token's parameters are constituted, in general, by the variables of a *constraint network* \mathcal{N} (refer to [47] for further details) and can be used, among other things, to represent temporal information such as the start or the end of some tasks. The semantics of the χ constant, on the contrary, is borrowed from Constraint Logic Programming (CLP) [48]. Specifically, while the facts are considered inherently true, the goals must be achieved as defined by a set of *rules*. Rules, in particular, are expressions of the form $n(x_0, \dots, x_k) \leftarrow \mathbf{r}$ where $n(x_0, \dots, x_k)$ is the *head* of the rule and \mathbf{r} is the *body* of the rule. In particular, \mathbf{r} represents the *requirement* for achieving any goal having the “form” of the head of the rule. Such requirements can be either a token, a *constraint* among tokens (possibly including the x_0, \dots, x_k variables), a *conjunction* of requirements or a *disjunction* of requirements. It is worth noting the recursive definition of requirement, which allows the definition of the body of a rule as any logical combination of tokens and constraints.

Similarly to CLP, through the application of the rules it is hence possible to establish and generate relationships among tokens. Compared to CLP, however, timelines introduce an added value: some tokens may be equipped with a special object variable τ that identifies the *timeline* affected by the token. Different tokens with the same value for the τ parameter, in particular, affect the same timeline and, depending on the nature of the timeline, might interact with each other. There can, indeed, be different types of timelines. In case of *state-variable* timelines (see Figure 1a), for example, different tokens on the same state-variable cannot temporally overlap. In case of *reusable-resource* timelines (see Figure 1b), on the contrary, tokens represent resource usages and can, hence, overlap as long as the concurrent uses remain below the resource's capacity.

Given the ingredients mentioned above we can now formally introduce the addressed planning problem. A *timeline-based planning problem*, specifically, is a triple $\mathcal{P} = (\mathbf{O}, \mathcal{R}, \mathbf{r})$, where \mathbf{O} is a set of typed objects, needed for instantiating the initial domains of the constraint network variables and, consequently, the tokens' parameters, \mathcal{R} is a set of rules and \mathbf{r} is a requirement. Intuitively, a solution to such a problem should be described by a set of tokens whose parameters assume values so as to guarantee the satisfaction of all the constraints imposed by the problem's

requirement, by the application of the rules, as well as by the cumulative constraints imposed by the timelines. Unfortunately, the previous definition, although intuitive, is not easily translatable into a reasoning process which guarantees its achievement starting from the definition of the planning problem. For this reason, just like common partial-order planners, timeline-based planners often rely on the concepts of *flaw* and *resolver*. The planner, in particular, internally maintains a data structure, called *token network*, which represents a partial plan $\pi = (\mathcal{T}, \mathcal{N})$, where \mathcal{T} is a set of tokens whose parameters are constrained by the constraint network \mathcal{N} . During the resolution process, the reasoner incrementally refines the current token network π by identifying its flaws and by solving them through the application of resolvers, while maintaining consistent the constraints of \mathcal{N} .

There can be, in general, different types of flaws, each resolvable by applying the corresponding resolvers. The achievement of a goal, for example, can take place either through the application of a rule or through a *unification* with either a fact or another already achieved goal with the same predicate (i.e., the parameters of the current goal and the token with which is unifying are constrained to be pairwise equal). In case of disjunctions, introduced either in the initial problem or by the application of a rule, a disjunct must be chosen. The domain of all the variables that make up the token parameters must be reduced to a single allowed value. Finally, timelines must be consistent, possibly requiring the introduction of constraints which prevent not allowed overlaps. Thanks to the introduction of the flaw and resolver concepts, it is therefore possible to provide an implementable definition of solution. Specifically, a *solution* to a timeline-based planning problem is a flawless token network whose constraint network is consistent.

2.1. A Lifted Heuristic for Timeline-based Planning

Finding a solution to a timeline-based planning problem is far from simple. Choosing the *right* flaw and the *right* resolver, in particular, constitutes a crucial aspect for coping with the computational complexity and hence efficiently generating solutions. Taking a cue from classical planning heuristics, [24] describes how, by building a *causal graph* and by analyzing its topology, it is possible to estimate the costs for the resolution of the flaws and for the application of the resolvers. Flaws and resolvers, in particular, are seen as if they are, respectively, classical planning propositions and actions. Similarly to a proposition added by the positive effect of an action, in particular, the effect of applying a resolver is the resolution of a flaw. Additionally, just like the preconditions in a classical action, further flaws can be introduced in the case of the application of a rule or the choice of a disjunct in a disjunction. Starting from the initial facts, with a zero estimated resolution cost, the cost of applying a resolver can be estimated as an intrinsic cost of the resolver plus the maximum cost (h^{max} heuristic). The cost of resolving a flaw, on the other hand, is given by the minimum cost of its resolvers. Starting from the top-level goals present in the planning problem, initially estimated with infinite cost, a graph is constructed by proceeding backwards, considering all the possible resolvers for all the possible flaws. The estimated costs are updated every time a unification is found or in those cases in which the resolver does not introduce further flaws. Finally, the graph building procedure proceeds until a finite estimate cost for the top-level goals is reached.

Compared to other state-of-the-art timeline-based solvers, the above heuristics allow solving

problems up to one order of magnitude faster [24]. The most interesting aspect for the current topic, however, concerns the management of the causal constraints in the causal graph. Similar to planning models based on satisfiability [49], indeed, a set of propositional variables is assigned to flaws and to resolvers. For the sake of brevity we will use subscripts to indicate flaws (e.g., φ_0, φ_1 , etc.), resolvers (e.g., ρ_0, ρ_1 , etc.) as well as their associated propositional variables. Additionally, given a flaw φ , we refer to the set of its possible resolvers by means of $res(\varphi)$ and, by means of $cause(\varphi)$, to the set of resolvers (possibly empty, in case of the flaws of the problem's requirement) which are responsible for introducing it. Moreover, given a resolver ρ , we refer to the set of its preconditions (e.g., the set of tokens introduced by the application of a rule) by means of $precs(\rho)$ and to the flaw solved through its application by means of $eff(\rho)$. Using the above notation we can estimate the cost of a generic flaw φ as $G(\varphi) = \min_{\rho \in res(\varphi)} G(\rho)$ (1) and the cost of a generic resolver ρ as $G(\rho) = c(\rho) + \max_{\varphi \in precs(\rho)} G(\varphi)$ (2), in which $c(\rho)$ represents the intrinsic cost of the resolver ρ .

The introduction of such variables allows to constrain them so as to guarantee the satisfaction of the causal relations. Specifically, for each flaw φ_i , we guarantee that the preconditions of all the applied resolvers are satisfied ($\varphi_i = \bigwedge_{\rho_k \in cause(\varphi_i)} \rho_k$ (3)) and that at least one resolver is active whenever the flaw becomes active ($\varphi_i \Rightarrow \bigvee_{\rho_l \in res(\varphi_i)} \rho_l$ (4)). Additionally, we need a gimmick to link the presence of the tokens with the causality constraint. A further variable $\sigma \in \{inactive, active, unified\}$, in this regard, is associated to each token. A partial solution will hence consist solely of those tokens of the token network which are *active*. Moreover, in case such tokens are goals, the bodies of the associated rules must also be present within the solution. Later on, we refer to tokens by means of the σ variables (we will use subscripts to describe specific tokens, e.g., σ_0, σ_1 , etc.) and to the flaws introduced by tokens by means of the $\varphi(\sigma)$ function.

The last aspect to consider concerns the update of such variables as a consequence of the activation of a rule application resolver and of a unification resolver. Specifically, each rule application resolver ρ_a binds the σ_a variable of the goal token, whose rule has been applied, to assume the *active* value (formally, $\rho_a = [\varphi(\sigma_a) = active]$). Finally, for each unification resolver ρ_u representing the unification of a token σ_u with a target token σ_t , the constraints $\rho_u = [\sigma_u = unified]$ and $\rho_u \Rightarrow [\sigma_t = active]$ guarantee the update of the σ variables while adding $\varphi(\sigma_t)$ to the preconditions of ρ_u guarantees the operation of the heuristic.

2.2. An explanatory example

In order to better understand how the heuristics and the causality constraints work, we introduce in this section a running example of an explanatory planning problem, whose objective is to plan a physical rehabilitation session for an hypothetical user. Figure 2 shows the causal graph which is generated for the problem, whose problem requirement is constituted by the sole goal σ_0 . Estimated costs for flaws (boxes) and resolvers (circles) are on their upper right. The propositional variables that participate in the causal constraints are on their upper left. Solid (TRUE) and dashed (UNASSIGNED) contour lines are used to distinguish flaws' and resolvers' associated propositional variables' values. In the figure, in particular, the φ_0 variable, representing a flaw which is present in the problem requirement and therefore must necessarily be solved, assumes the TRUE value.

It is worth noting that, in the example, the φ_0 flaw, for achieving the σ_0 goal, can only be solved through the ρ_0 resolver, which is hence directly applied (notice the solid line) as a consequence of the propagation of the causal constraints. Since $res(\varphi_0) = \{\rho_0\}$, indeed, the expression (4) translates into $\varphi_0 \Rightarrow \rho_0$. This, in turn, forces the σ_0 goal to assume the *active* value as a consequence of $\rho_0 = [\varphi(\sigma_0) = active]$. The ρ_0 resolver, furthermore, represents the application of a rule having a *PhysicalExercise()* in the head and, in the body, a conjunction of the two σ_1 and σ_2 goals. The application of this resolver, in particular, introduces the $\varphi_1 = \varphi(\sigma_1)$ and the $\varphi_2 = \varphi(\sigma_2)$ flaws, each of which must necessarily be resolved as a consequence of the $\varphi_1 = \rho_0$ and $\varphi_2 = \rho_0$ causal constraints, from the expression (3). These flaws, in turn, can be solved through the application of the ρ_1 and of the ρ_2 resolvers which introduce, respectively, the disjunctions represented by the φ_3 and φ_4 flaws.

Proceeding backwards, the propagation of the causal constraints no longer allows to infer what is present in the current partial plan (notice the dashed lines). The resolution of the φ_3 and φ_4 flaws, in particular, constitute two choices that the planner must make during the resolution process. The φ_3 flaw, for example, can be solved either by applying the $Disj_0$ disjunct, represented by the ρ_3 resolver, or by applying the $Disj_1$ disjunct, represented by the ρ_4 resolver. The graph construction process, however, which proceeds following a breadth-first approach, has identified, in the example, a possible solution for the φ_3 flaw by applying first the ρ_3 resolver and then the ρ_7 resolver (the latter corresponding, in this simple example, to a rule with an empty body). The heuristics' estimated costs propagation procedure, hence, makes the ρ_3 resolver, with an estimated cost of 2, much more attractive than the ρ_4 resolver, with an estimated cost of ∞ . For a similar reason, the ρ_5 resolver will be preferred over the ρ_6 resolver, leading to a (possible) solution of the planning problem.

It is worth noting that, for the sake of simplicity, the tokens' parameters are not represented in the example figure. All tokens, however, are endowed with numerical variables that represent the start and the end of the associated activities, appropriately constrained according to common sense. Upper and lower body exercises, for example, represented respectively by the σ_1 and by the σ_2 tokens, will take place as part of the more general physical exercise represented by the σ_0 token. The σ_3 and by the σ_5 tokens, additionally, are endowed with their τ variables which will avoid their temporal overlapping if they will assume the same value.

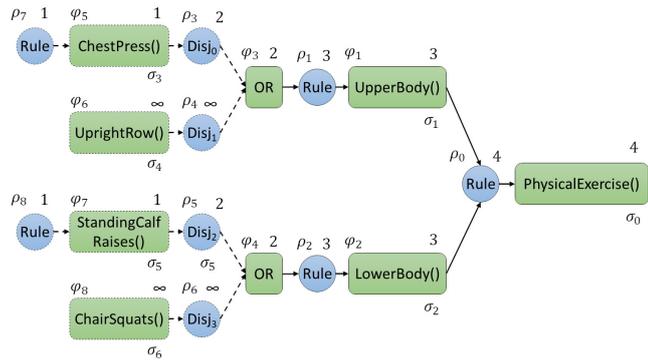


Figure 2: An example of causal graph for the planning of a physical rehabilitation session. Tokens' parameters are omitted to avoid burdening the notation.

3. Pruning the causal graph

The construction of the causal graph is, inevitably, a partial task that cannot be carried out in an exhaustive manner. Consider, for example, the case of a not so smart robotic arm which, in order to move an object from one point to another, considers an infinite number of tasks in which it takes the object and puts it back where it initially was. The graph building procedure, in particular, starts by queuing the high-level flaws, defined in the planning problem, in an expansion queue. At each step, a flaw is extracted from the queue and expanded, possibly queuing further sub-flaws, proceeding backwards in a breadth-first manner until the high-level flaws do assume a finite estimated cost. It is worth noticing that, exception made for very simple cases, when the graph building procedure halts, the queue might still contain some flaws. These flaws, however, have an estimated infinite cost. Once given way to the resolution algorithm, in particular, these flaws would be avoided as much as possible by the search algorithm. In other words, for these flaws, the graph is not able to provide any indication for their resolution.

Can we, before the resolution process starts, remove them from the graph or, somehow, forbid their choice? As we will see, the answer is yes, but we can do even more. Some of the flaws in the graph expansion queue, indeed, might constitute the preconditions for the resolution of already expanded flaws which, within the causal graph, do already have a finite estimated cost. Such flaws, which we call *deferrable*, can easily be identified by traversing the causal graph following the direction of the arrows, until a flaw with a finite estimated cost is found (hence the flaw is classified as deferrable) or a sink node is reached (hence the flaw is classified as non-deferrable). Once removed from the graph's expansion queue, in particular, these flaws can immediately be re-queued for subsequent further processing.

When the graph expansion procedure halts, in particular, there will be, in the graph's expansion queue, a set of flaws, either deferrable or not expanded yet. Instead of discouraging the search algorithm from choosing such flaws, we can forcibly prevent it by imposing constraints on the corresponding causal variables (the causal variables are forced to false). It is worth noting that these flaws, having an infinite estimated cost, would not be chosen by the search anyway. The introduction of the constraints, however, causally propagates within the causal graph leading to some benefits in terms of performance.

Consider, for example, the graph presented in Figure 2. Before the expansion of the φ_5 flaw all the estimated costs are infinite. The expansion of the φ_5 flaw, however, introduces a resolver, ρ_7 , that has no preconditions. The cost of such a resolver, in line with the h^{max} heuristic, can easily be estimated as the sole intrinsic cost of the resolver (Eq. 2). By applying the cost estimation formula, the cost update is propagated forward by assigning an estimated cost of 1 to the φ_5 flaw (Eq. 1), an estimated cost of 2 to the ρ_3 resolver, an estimated cost of 2 to the φ_3 flaw, an estimated cost of 3 to the ρ_1 resolver and an estimated cost of 3 to the φ_1 flaw. All the other flaws and resolvers maintain, at this point, an infinite estimated cost.

The φ_6 flaw is then removed from the graph's expansion queue and checked for deferrability. Since the φ_3 flaw has a finite estimated cost, the flow is recognized as deferrable and is re-queued into the graph expansion queue. It's now the turn of the φ_7 flaw, having a term similar to that of the φ_5 flaw. The cost update propagation, however, reaches, this time, the updating the cost of the high-level φ_0 flaw, determining the halting of the graph building procedure with the φ_6 and the φ_8 flaws in the graph expansion queue.

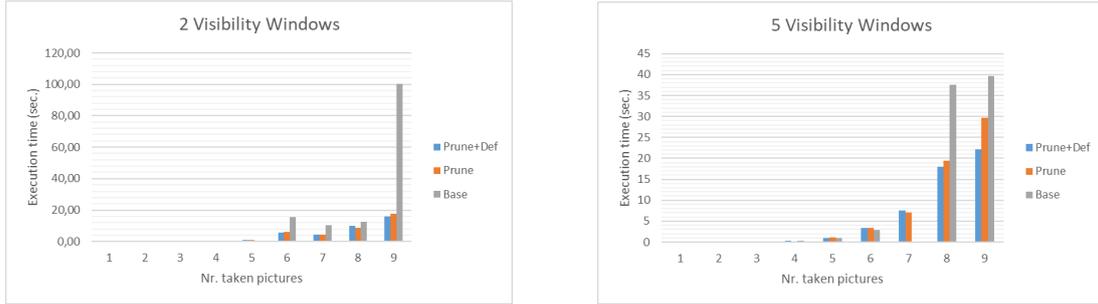


Figure 3: Different timelines extracted by their associated tokens.

The resulting graph, depicted in Figure 2, shows that the only two choices that the resolution algorithm should eventually make are the resolution of the φ_3 and of the φ_4 flaws. In solving such flaws, specifically, the heuristic-driven algorithm would choose the resolvers with the lowest estimated cost, respectively the ρ_3 and of the ρ_5 resolvers, strictly avoiding the choice of the ρ_4 and of the ρ_6 resolvers, whose estimated cost is infinite. If, however, the causal variables of the queued flaws (i.e., the φ_6 and the φ_8 flaws) are forced to false, the causal constraints (i.e., (4) and (3)) force the ρ_4 and of the ρ_6 resolvers at false false and, consequently, the ρ_3 and of the ρ_5 resolvers at true, together with the φ_5 and the φ_7 flaws and the ρ_7 and of the ρ_8 resolvers, solving the problem without the slightest need, at least from a causal point of view (a search may in fact still be required for scheduling the activities), to do any search.

4. Experimental results

In order to test the effectiveness of the proposed pruning techniques we have implemented both the procedures for pruning and for recognizing the deferrable flaws within the oRATIO² planner, testing its performance on different instances of increasing complexity on the GOAC domain. Specifically, the Goal Oriented Autonomous Controller (GOAC) was an ESA initiative aimed at defining a new generation of software autonomous controllers to support increasing levels of autonomy for robotic task achievement. In particular, the domain, initially defined in [30] and more recently cited in [50], aims at controlling a rover to take a set of pictures, store them on board and dump the pictures when a given communication channel was available. The interesting aspect of this domain is that communication can only take place within specific visibility windows that take into account the astronomical motions of the planets/satellites which, in some cases, may stand between the transmitting and receiving stations. The presence of these visibility windows, in particular, requires an explicit modeling of temporal aspects in order to adequately plan the transmission of information and can hence easily be modeled through, and solved by, timeline-based planners. The problem is made more interesting by the presence of constraints which include the available resources (e.g., memory and battery) as well as by having a distance matrix, among the possible locations, which might be not completely connected.

Figure 3 shows the execution times of different configurations of the oRATIO solver, allowing

²<https://github.com/pstlab/oRatio>

the comparison of the base solver, without pruning, pruning without recognizing the deferrable flaws and pruning with the recognition of the deferrable flaws. From the figure it is possible to see how, in general, the resolution times significantly benefit from the application of pruning on the proposed benchmarking problem, reducing by an order of magnitude the computation times in the case of the instance #9 of the problem with two visibility windows, or allowing resolution within the two minute timeout in the case of the instance #7 of the 5 visibility windows. Note how deferrable flaw recognition adds a little overhead in smaller instances, which still leads to a benefit in larger instances.

5. Conclusions

The efficiency of the planners' resolution processes is strongly influenced by the heuristics that, in some cases, guide the solution process while, in other cases, can prune the search space so as to favor the propagation of causal constraints and avoid possible dead-ends. The use of pruning techniques is not new in the case of classical planning, however, it is less explored in the case of partial-order planning and, even more so, in the case of high-expressive timeline-based planning. For this reason we have presented a pruning technique based on the construction of a lifted causal graph. The experimental results, although conducted on a single benchmarking problem, show significant benefits and are, therefore, encouraging. A more detailed experimentation on a greater number of benchmarking problems could highlight further benefits or weaknesses of the proposed approach and, therefore, will certainly be carried out in future work.

References

- [1] M. Ghallab, D. Nau, P. Traverso, *Automated Planning: Theory and Practice*, Morgan Kaufmann Publishers Inc., 2004.
- [2] B. Bonet, H. Geffner, *Planning as Heuristic Search*, *Artificial Intelligence* 129 (2001) 5–33.
- [3] J. Hoffmann, B. Nebel, *The FF Planning System: Fast Plan Generation Through Heuristic Search*, *Journal of Artificial Intelligence Research* 14 (2001) 253–302.
- [4] M. Helmert, *The Fast Downward Planning System*, *Journal of Artificial Intelligence Research* 26 (2006) 191–246.
- [5] J. Hoffmann, J. Porteous, L. Sebastia, *Ordered Landmarks in Planning*, *Journal of Artificial Intelligence Research* 22 (2004) 215–278.
- [6] J. Porteous, L. Sebastia, J. Hoffmann, *On the Extraction, Ordering, and Usage of Landmarks in Planning*, in: *Sixth European Conference on Planning*, 2014.
- [7] P. Haslum, H. Geffner, *Admissible Heuristics for Optimal Planning*, in: *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, Breckenridge, CO, USA, April 14-17, 2000, AAAI Press, 2000, pp. 140–149.
- [8] P. Haslum, B. Bonet, H. Geffner, *New Admissible Heuristics for Domain-Independent Planning*, in: *AAAI*, volume 5, 2005, pp. 9–13.
- [9] S. Edelkamp, *Planning with Pattern Databases*, in: *Sixth European Conference on Planning*, 2014.

- [10] M. Helmert, P. Haslum, J. Hoffmann, Flexible Abstraction Heuristics for Optimal Sequential Planning, in: ICAPS, 2007, pp. 176–183.
- [11] M. Helmert, P. Haslum, J. Hoffmann, R. Nissim, Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces, *Journal of the ACM (JACM)* 61 (2014) 16.
- [12] V. Vidal, H. Geffner, Branching and pruning: An optimal temporal poel planner based on constraint programming., volume 170, 2004, pp. 570–577. doi:10.1016/j.artint.2005.08.004.
- [13] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL—The Planning Domain Definition Language, 1998.
- [14] R. Fikes, N. J. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, in: IJCAI, 1971, pp. 608–620.
- [15] M. Fox, D. Long, PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains, *Journal of Artificial Intelligence Research* 20 (2003) 61–124.
- [16] S. Edelkamp, J. Hoffmann, PDDL2.2: The language for the Classical Part of the 4th International Planning Competition, Technical Report 195, Institut für Informatik, 2004.
- [17] M. Fox, D. Long, Modelling Mixed Discrete-continuous Domains for Planning, *Journal Of Artificial Intelligence Research* 27 (2006) 235–297.
- [18] A. E. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: {PDDL3} and experimental evaluation of the planners, *Artificial Intelligence* 173 (2009) 619–668. URL: <http://www.sciencedirect.com/science/article/pii/S0004370208001847>. doi:<http://dx.doi.org/10.1016/j.artint.2008.10.012>, advances in Automated Plan Generation.
- [19] W. Piotrowski, M. Fox, D. Long, D. Magazzeni, F. Mercorio, Heuristic Planning for PDDL+ Domains, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16, AAAI Press, 2016, pp. 3213–3219.
- [20] S. Franco, M. Vallati, A. Lindsay, T. L. McCluskey, Improving Planning Performance in PDDL+ Domains via Automated Predicate Reformulation, in: Computational Science – ICCS 2019, Springer International Publishing, 2019, pp. 491–498.
- [21] N. Muscettola, S. Smith, A. Cesta, D. D’Aloisi, Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture, *IEEE Control Systems* 12 (1992).
- [22] N. Muscettola, HSTS: Integrating Planning and Scheduling, in: Zweben, M. and Fox, M.S. (Ed.), *Intelligent Scheduling*, Morgan Kauffmann, 1994.
- [23] D. S. Weld, An Introduction to Least Commitment Planning, *AI Magazine* 15 (1994) 27–61.
- [24] R. De Benedictis, A. Cesta, Lifted Heuristics for Timeline-based Planning, in: ECAI-2020, 24th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 2020, pp. 498–2337.
- [25] M. Ghallab, H. Laruelle, Representation and Control in IxTeT, a Temporal Planner, in: AIPS-94. Proceedings of the 2nd Int. Conf. on AI Planning and Scheduling, 1994, pp. 61–67.
- [26] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, B. Smith, Planning in Interplanetary Space: Theory and Practice, in: AIPS-00. Proceedings of the Fifth Int. Conf. on AI Planning and Scheduling, 2000, pp. 177–186.
- [27] S. Chien, D. Tran, G. Rabideau, S. Schaffer, D. Mandl, S. Frye, Timeline-Based Space Operations Scheduling with External Constraints, in: ICAPS-10. Proc. of the 20th Int. Conf.

- on Automated Planning and Scheduling, 2010, pp. 34–41.
- [28] S. Fratini, F. Pecora, A. Cesta, Unifying Planning and Scheduling as Timelines in a Component-Based Perspective, *Archives of Control Sciences* 18 (2008) 231–271.
 - [29] A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, Developing an End-to-End Planning Application from a Timeline Representation Framework, in: *IAAI-09. Proceedings of the 21st Innovative Applications of Artificial Intelligence Conference*, Pasadena, CA, USA, 2009, pp. 66–71.
 - [30] S. Fratini, A. Cesta, R. De Benedictis, A. Orlandini, R. Rasconi, APSI-based Deliberation in Goal Oriented Autonomous Controllers, *ASTRA* 11 (2011).
 - [31] A. Umbrico, A. Cesta, M. Cialdea Mayer, A. Orlandini, Platinum: A new framework for planning and acting, in: *AI*IA 2017 Proceedings*, 2017, pp. 498–512.
 - [32] J. Frank, A. K. Jónsson, Constraint-Based Attribute and Interval Planning, *Constraints* 8 (2003) 339–364.
 - [33] A. Cesta, A. Oddi, Gaining Efficiency and Flexibility in the Simple Temporal Problem, in: L. Chittaro, S. Goodwin, H. Hamilton, A. Montanari (Eds.), *Proceedings of the Third International Workshop on Temporal Representation and Reasoning (TIME-96)*, IEEE Computer Society Press: Los Alamitos, CA, 1996, pp. 45–50.
 - [34] P. Laborie, Algorithms for propagating resource constraints in AI planning and scheduling: existing approaches and new results, *Artificial Intelligence* 143 (2003) 151–188.
 - [35] S. Stock, M. Mansouri, F. Pecora, J. Hertzberg, Hierarchical hybrid planning in a mobile service robot, in: *KI 2015 Proceedings*, 2015, pp. 309–315.
 - [36] A. Bit-Monnot, M. Ghallab, F. Ingrand, D. E. Smith, FAPE: a Constraint-based Planner for Generative and Hierarchical Temporal Planning, *arXiv preprint arXiv:2010.13121* (2020).
 - [37] F. Dvorák, A. Bit-Monnot, F. Ingrand, M. Ghallab, Plan-Space Hierarchical Planning with the Action Notation Modeling Language, in: *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, Limassol, Cyprus, 2014. URL: <https://hal.archives-ouvertes.fr/hal-01138105>.
 - [38] D. E. Smith, J. Frank, W. Cushing, The ANML language, in: *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2008.
 - [39] D. E. Wilkins, *Practical planning: extending the classical AI planning paradigm* / David E. Wilkins, Morgan Kaufmann Publishers San Mateo, Calif, 1988.
 - [40] D. S. Nau, T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, F. Yaman, SHOP2: an HTN planning system, *J. Artif. Intell. Res.* 20 (2003) 379–404.
 - [41] L. Castillo, J. Fdez-Olivares, O. García-Pérez, F. Palao, Efficiently handling temporal knowledge in an htn planner, in: *Proceedings of the Sixteenth International Conference on International Conference on Automated Planning and Scheduling, ICAPS'06*, AAAI Press, 2006, pp. 63–72.
 - [42] P. Laborie, M. Ghallab, Planning with Sharable Resource Constraints, in: *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, IJCAI'95*, Morgan Kaufmann Publishers Inc., 1995, pp. 1643–1649.
 - [43] A. Cesta, A. Oddi, S. F. Smith, A Constraint-Based Method for Project Scheduling with Time Windows, *Journal of Heuristics* 8 (2002) 109–136. URL: <https://doi.org/10.1023/A:1013617802515>. doi:10.1023/A:1013617802515.
 - [44] D. E. Smith, J. Frank, A. K. Jónsson, Bridging the Gap Between Planning and Scheduling, *Knowledge Engineering Review* (2000).

- [45] G. Verfaillie, C. Pralet, M. Lemaître, How to model planning and scheduling problems using constraint networks on timelines, *The Knowledge Engineering Review* 25 (2010) 319–336.
- [46] M. Cialdea Mayer, A. Orlandini, A. Umbrico, Planning and execution with flexible timelines: a formal account, *Acta Informatica* 53 (2016) 649–680. URL: <http://dx.doi.org/10.1007/s00236-015-0252-z>. doi:10.1007/s00236-015-0252-z.
- [47] R. Dechter, *Constraint Processing*, Elsevier Morgan Kaufmann, 2003.
- [48] K. R. Apt, M. G. Wallace, *Constraint Logic Programming Using ECLⁱPS^e*, Cambridge University Press, New York, NY, USA, 2007.
- [49] H. Kautz, B. Selman, Planning as Satisfiability, in: *ECAI*, volume 92, 1992, pp. 359–363.
- [50] A. Coles, A. Coles, M. Martinez Munoz, O. Savas, J. Delfa, T. de la Rosa, Y. E-Martín, A. García Olaya, Efficiently Reasoning with Interval Constraints in Forward Search Planning, in: *Proceedings of the Thirty Third AAAI Conference on Artificial Intelligence*, AAAI Press, 2019.