Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)
Early Career Track

AI Planning for Hybrid Systems

Enrico Scala
University of Brescia
enrico.scala@unibs.it

Abstract

When planning the tasks of some physical entities that need to perform actions in the world (e.g., a Robot) it is necessary to take into account quite complex models for ensuring that the plan is actually executable. Indeed the state of these systems evolves according to potentially non-linear dynamics where interdependent discrete and continuous changes happen over the entire course of the task. Systems of this kind are typically compactly represented in planning using languages mixing propositional logic and mathematics. However, these languages are still poorly understood and exploited. What are the difficulties for planning in these settings? How can we build systems that can scale up over realistically sized problems? What are the domains which can benefit from these languages? This short paper shows the main two ingredients that are needed to build a heuristic search planner, outline the main impact that such techniques have on application, and provide some open challenges. These models and relative planners hold the promise to deliver explainable AI solutions that do not rely on large amounts of data.

1 Introduction

Automated Planning is one of the fundamental pillars of Artificial Intelligence. Provided some world model, AI Planning enables an agent operating in some environment to achieve a goal by executing actions. There are various classes of models that can be adopted [Ghallab *et al.*, 2004] varying on the accurateness of representation. When we are planning the tasks of some physical entities however, we want these models to be rather accurate. This increases the likelihood of success; i.e., at execution time, the planned actions becomes actually executable, and the goal reached at the end. Modern AI Planning languages provide quite a range of modeling features to address these questions. For instance, in PDDL+ [Fox and Long, 2006], it is possible to describe how a system behaves according to a set of differential equations, and explicitly indicates how the agents' actions affect the dynamics when executed. This kind of representation is as accurate as the actual physics describing the phenomena is.

While these powerful languages provide a nice account from a knowledge representation perspective, the underlying decidability issues coming from the introduction of numeric state variables [Helmert, 2002] make them quite challenging. So much, that other researchers propose black-box algorithms based on Reinforcement Learning (RL) [Corso *et al.*, 2021], instead. However, despite the success of RL and machine learning we argue that it is still interesting to look at approaches that solve these problems mainly by reasoning, in a model-based manner. This is crucial if we lack data and want to have explanation guarantees [Goebel *et al.*, 2018]

In this paper I share my personal experience on solving planning problems for hybrid systems using a model-based approach that does heuristic search. I show the two needed ingredients: 1. The design of a best-first search algorithm that can work with representations with spontaneous/controllable numeric changes and general numeric expressions 2. The design of relaxation-based heuristics that are sensitive to the hybrid (numeric and logical) structure of the problem. We show how planning in hybrid systems can be cast as a sequential decision problem provided some time discretisation. Then we see what planning heuristics can be defined in a way to be able to effectively guide search. Then we outlook at the impacts that these techniques have had on applications and systems, and conclude with a number of open challenges.

2 Planning Problems over Hybrid Systems

A hybrid system consists of a continuous and a discrete dynamics that model the evolution of a controllable physical system such as a robotic system or a production plant [Henzinger, 2000; Doyen *et al.*, 2018]. The continuous part of the model is commonly expressed using time-derivatives over state variables. For instance, we can say that the robot position evolves over time according to speed, i.e., $\dot{x} = v$, and that speed itself evolves according to acceleration, i.e., $\dot{v} = a$.

Hybrid systems are represented by the AI Planning community using a formalism based on first-order-logic, and centred around the idea of an agent that can perform actions in the world, which itself behaves according to processes and events. The standard de facto language used to express such problems is PDDL+ [Fox and Long, 2006; Haslum *et al.*, 2019], a (much) extended version of STRIPS [Fikes and Nilsson, 1971], the historical planning language proposed since the very early ages of the Artificial Intelli-

gence field. Intuitively, in PDDL+, processes dictate the differential equations governing the system, events formalises discrete changes that can happen in the environment; ultimately, actions formalises what the agent can do. Processes, events and actions are mathematically represented as pairs of condition and effects. Conditions are first-order formulas over numeric and propositional terms. For instance, assume that the truth of a Boolean variable t models whether task t is done, and x and y the coordinate of an agent operating in an environment. The formula $(x^2 + y^2 < 5) \rightarrow (t = \top)$ can be used to model that when an agent occupies any position in the circle $x^2 + y^2 = 5$, then task t has to be already done. Effects are what formalises changes. In processes, effects are sets of time-derivative functions. In events and actions, effects are assignments of the form $x := \xi$ where ξ is a mathematical expressions over the state variables. When an event is triggered, or an action is executed, the state is transformed according to given effects. Whereas processes and effects cause state transitions as soon as their conditions are met, actions are agent's decision, and the associated conditions represent when such decisions can be taken. A planning problem in the context of a hybrid system is the task of looking for a timed plan of actions given some specifics on the initial state and a goal to be reached, which is itself a formula that needs to be satisfied. The validity of a plan depends on a number of intertwined conditions, which makes the problem quite involved. First and foremost, all actions have to be applicable in each state at the time in which they are planned, and the goal reached at some instant in the future. Therefore, we need to check whether an action condition or goal is satisfied at some time instant and at the same time consider the exact state of the system at that instant, whose valuation depends on the effect of what prescribed by the processes and events, and the effects of all previous actions. For instance, consider to have a process modeling that position x of the agent changes by 1 unit every second. After ten seconds the position goes from, say 0 to 10. Now imagine to have an action that switches a flag initially set to false to true (necessary in the goal), and imagine such an action being executable when $x \geq 4 \wedge x \leq 6$. Executables and goal reaching plans are only those where the action is applied at a time within 4 and 6 seconds.

Most successful planners focuses on discrete-time [Percassi *et al.*, 2023]. In other words, differential equations are approximated through simulation and planners involved in a discretise and validate loop [Penna *et al.*, 2009]. Under this setting, next section shows how these problems can be solved through a best-first search approach. Intuitively, best-first search incrementally looks into the future branching over time passing or action decisions. This breaks the loop of dependencies between processes, actions and events in the search for a valid plan leading to the goal.

3 Best-First Search in Hybrid Systems

Best-First Search (BFS) is a general approach for searching solutions to combinatorial problems. BFS is widely used in automated planning and serves as a search engine in many state-of-the-art planning systems (e.g., LAMA [Richter and Westphal, 2010], various incarnations of Fast Downward [Helmert, 2006], FF [Hoffmann and Nebel, 2001]). BFS algorithm operates alternating between two main operations on a frontier holding the possible futures that can be reached through some chain of actions. These two operations are: node expansion and node generation.

During the expansion phase, the BFS algorithm selects a node from the frontier, evaluates it based on a certain metric, and checks if it represents a goal state. If it does, the algorithm can just terminate reporting the path all the way to such a state. During the generation phase instead, a BFS algorithm is responsible for generating those successor nodes associated with states reachable from the node just popped from the frontier. Generally, BFS exploits a successor function that establishes what can be reached by some action. The distinguishing factor among different BFS algorithms lies in how the frontier evolves over time. This refers to the order in which nodes are selected for expansion, and can be summarised by saying that a BFS algorithm takes an input a function f mapping a state to a numeric value. At expansion time, BFS pops the node with the lowest f-value. f can be instantiated in different ways: distance from the initial state, estimated distance to the goal, or the algorithm itself can be made more sophisticated with multiple frontiers [Richter and Westphal, 2010]. A specific instance of a BFS algorithm is A* [Dechter and Pearl, 1985; Felner, 2011]. In A*, each node n is evaluated using a function $f(n) = g(n)+h(n)$, where $g(n)$ represents the cost from the initial state to node n, and $h(n)$, the heuristic function, is an estimate of the distance from node n to the goal.

In the planning context, BFS is often set as a forward search: start from the initial state and continue until a state satisfying the goal conditions is reached. For each state, the actions applicable in that state define the successor function. In planning for hybrid systems, it is possible to apply the aforementioned BFS algorithms using the so called theory of waiting [McDermott, 2003]. The idea is quite simple and follows a very similar schema to what used for classical planning. Indeed, at each state of the search, we have a complete observability of the world, and therefore can determine each variable value. With this, we can check whether a state satisfies a formula by a simple propagation of the values. Indeed, assuming a tree-shaped formula we can evaluate it bottom-up by substituting each variable x with the value held by x in a state, and apply well known concepts from logic to determine whether a node of the tree evaluates to true or false. With this mechanism, from each state we know all applicable actions (those having the root condition true), and can then generate a number of successor nodes, each associated to the state resulting from the execution of a given applicable action. Since hybrid problems are temporal problems, we need to have a way to see what happens if we let time move forward. This is handle by i) decorating a state s with a time variable that specifies when s is valid ii) generating a time passing successor node that represents the system waiting for some δ time. This successor node will be associated to a state computed by updating all variables affected by the processes and events. For instance, let p_0 and p_1 be two processes stating that the two variables x_0 and x_1 evolve according to $\dot{x_0} = z_0$ $\dot{x_1} = z_1$, if we assume a discretisation $\delta = 1$, the search generates a

successor with x_0 and x_1 incremented by $z_0 \times \delta$, and $z_1 \times \delta$.

Note that, for a given set of processes, there is an exponential number of processes that can be possibly true, but a BFS only explores subsets reachable and relevant for the goal. Yet, the downside of this schema is in the approximation of the time-derivatives. The dynamics is computed via simulation rather than exact computation. Even though we can make the simulation quite fine, e.g., $\delta = 0.0001$, a better approach is to use a differential integrator; work is yet still needed to understand the actual benefit of this approach in terms of generality, accurateness and effectiveness [Ramírez *et al.*, 2017].

The BFS performance depends on whether the search focuses expansions along some feasible trajectory to the goal. This can be tamed ordering the frontier using a heuristic function h defined on a relaxation of the problem. In practice we can associate a heuristic value to each node and order the expansion to prefer those nodes associated to states closer to the goal. We can use heuristic either in a greedy fashion (i.e., $f(n) = h(n)$) or resort to A^*. Next we show how to compute heuristics systematically by transformation in sequential numeric planning.

4 Heuristics with Numeric Changes and Goals

The first step towards the construction of a relaxation-based heuristic function is based on the observation that a hybrid planning problem interpreted over a discrete timeline can be relaxed into a numeric planning problem, as those that can be expressed using PDDL2.1 [Fox and Long, 2003]. PDDL2.1 only gives the possibility to express actions in a non temporal setting and so without processes and events. A relaxation for this problem can be designed by assuming that processes and events are controllable units by the agent at her will. This is known as a process to action relaxation. This expedient lets us use any relaxation-based heuristic for numeric planning problems under the aforementioned transformation, and exploit relative estimates for guiding the search for a plan for the un-relaxed hybrid system.

Relaxation-based heuristics/relaxations for sequential numeric planning can be split in three categories: constraints-based heuristics [Piacentini *et al.*, 2018b], subgoaling-based relaxations (e.g., [Scala *et al.*, 2020a]) and interval-based heuristics [Aldinger and Nebel, 2017; Scala *et al.*, 2016; Hoffmann, 2003]. Constraints-based heuristics and subgoaling relaxations exploit a causal and quantitative relationship that can be defined between numeric conditions and actions. We can capture the effort needed by some action to reach a given numeric condition with a simple mathematical operation. This enables the use of the concept of an achiever, a notion widely used in classical heuristics. For instance, if we are given a goal $x + y \geq 5$ and are considering an action that increases x by 2 and decreases y by 1 unit, we need to execute such an action 5 times if we want to make the condition satisfied in a state where the sum between x and y is 0. Interestingly enough, if we consider fractional executions of the actions, the optimisation problem to look for the cheapest achievers among a set of possible ones can be decided in polynomial time [Scala *et al.*, 2020a], and is such that optimal solutions include those where we only take a single achiever. This very simple observation turned out to be quite effective in designing heuristics aimed at approximating how many times an action is needed to achieve a single goal (which can be either a numeric or a Boolean one). Indeed, achievers can then be used to estimate the cost of reaching a goal in a recursive setting, i.e., by regressing the conditions against the achieving action and then add the cost of its precondition. We can then estimate the cost to get to the goal by accumulating the costs through chains of achievers that given a conjunctive condition either pessimistically sum the cost of each conjuncts in a condition, leading to inadmissible estimates such as h_{add}, optimistically take the most expensive one, leading to admissible estimates such as h_{max} [Haslum and Geffner, 2000; Scala *et al.*, 2020a], or find some inadmissible (h_{mrp} [Scala *et al.*, 2020b]) or admissible tradeoff between these two extremes; for admissible heuristics, a fruitful approach combines landmarks h_{lmcut} [Kuroiwa *et al.*, 2021] and linear programs [Piacentini *et al.*, 2018b; Kuroiwa *et al.*, 2022b].

However, the eager reader can recognize that subgoaling only applies when the problem present "simple" dynamics. That is, dynamics where all effects of actions are constant and additive and conditions are linear. In other words, the action contribution does not depend on the state in which it is applied. For instance, situations like $x+ = y$ with y being a controllable variable set by some other action are not supported. There are approaches extending such schemata to linear dynamics [Li *et al.*, 2018; Kuroiwa *et al.*, 2022a], and it is likely that even more complex functions could be supported. But how to do so systematically is not clear yet.

The interval-based relaxation approach tackles this problem by proposing an alternative view that, instead of looking at achievers, look at reachable relaxed states. It does so by reinterpreting the semantics of numeric planning. Instead of treating states as specific points, interval-based relaxation considers them as sets of intervals, one for each variable. In other words, each variable is associated with an interval that represents its possible values. When applying an action, the interval for each variable is expanded to include both the original value and the potential value resulting from executing that action in the state. The expansion of the interval is done using the concept of convex union between intervals. For example, let us consider a scenario where we have intervals $[0, 3]$, $[1, 2]$, and $[2, 2]$ representing the possible values of variable x, y and z. Suppose we have action a increasing x by $y \times z$. To compute the next value of x, interval analysis specifies that the multiplication between y and z results in the interval $[1, 4]$. This interval optimistically represents the potential values that can be obtained through the multiplication. We then add this interval to the previous value of x, resulting in $[1, 7]$. By taking the convex union of all intervals, we obtain $[0, 7]$ as the range of possible values for x if action a is executed. Interval-based-relaxation relaxes satisfiability, too. To check the satisfiability of a condition (necessary to check for action applicability and goal states) each term of the formulas is evaluated against the intervals of the variables it involves. For instance, let us consider the numeric term $y \times z \geq 8$, where the intervals associated to the variables in state s are as follows $y = [0, 5]$ and $z = [1, 2]$. Under interval-based re-

laxation, the condition is considered satisfied if there exists at least one value within the evaluated interval satisfying the inequality. Indeed, interval $[0, 10]$ does include a value greater than 8, so the condition is satisfied.

Interval-based relaxation enables the construction of a polynomial time reachability procedure and heuristics that are guaranteed to terminate and always provide a solution if the un-relaxed problem has one. For instance, for a class of acyclic numeric planning problem one can use h_{ff} adapted for numeric tasks [Hoffmann, 2003], while for general numeric planning problems one can use h_{aibr} [Scala *et al.*, 2016]. The only requirement for h_{aibr} is to have associated interval analysis operations [Moore *et al.*, 2009] for each mathematical expression of interest. Interval analysis allows a very large class of mathematical expressions, including transcendental and trigonometric functions.

5 Applications and Planning Systems

The advancement of planning systems that can generate plans for hybrid systems is already gaining some momentum from an application standpoint, and we provide just a short list for space reasons below. A seminal work in this direction is the one by [Fox *et al.*, 2012]. In particular, the authors show an architecture based on a PDDL+ formulation to find policies for a battery switching problem. Their results show that it is possible to substantial increase the lifetime of the batteries with the generated policy. The employed PDDL+ planner was UpMurphi [Penna *et al.*, 2009]. Another example is the recent work by [Kiam *et al.*, 2020] that shows how a PDDL+ based approach can be used for planning a multi-UAVs scenario; the PDDL+ aims at providing a formal account for the synchronisation of a fleet of UAVs, and uses a local search procedure to distribute goals among the agents. The PDDL+ formulation not only models the shared tasks, but also make sure that the complex dynamics of the UAVs is in check. Finally, there is a very recent work by [Aineto *et al.*, 2023] that shows how to use a PDDL+ approach to find counter-examples in safety verification problems coming from challenging industrial tasks. The counter-example yields the trajectory of decisions leading to a violation of a temporal constraints. The work shows that PDDL+ is not only viable but also competitive to falsification techniques based on simulations and RL over black-boxes. These two latter applications were used exploiting ENHSP [Scala *et al.*, 2016]. PDDL+ has also found application in classical control problems and physics based games such as angry birds. In this context, the work by [Piotrowski *et al.*, 2023] uses Hydra to generate actions in a sophisticated agent-based architecture. Other researchers have used PDDL+, too, e.g., for traffic control [Vallati *et al.*, 2016], or in-station train dispatching [Cardellini *et al.*, 2021].

Interestingly, all the aforementioned planners can be understood as instances of BFS algorithm applied in a forward search manner over the transition system induced by the discretised hybrid system. UpMurphi does indeed a Breath-First-Search that is an instance of BFS with priority given to states close to the initial state ($f(n) = d(n)$ with $d(n)$ being the distance from initial state). ENHSP also does a BFS equipped with different relaxation-based heuristics and pruning techniques. Hydra is similar to both planners in the spirit, but instead of adopting domain independent heuristics, focuses on the development of a system that can easily be configured with custom heuristic specific for the problem at hand. Note that BFS plus some heuristic is not the only approach to PDDL+; for instance, others approaches use compilation into SMT (e.g., [Cashmore *et al.*, 2020]) or other constraints-based formulation, for example into MIP [Piacentini *et al.*, 2018a]. Yet, to the best of our knowledge, none of these planners has been used into practical applications yet.

6 (Many) Open Challenges

Planning for hybrid systems still holds many open challenges. A non exhaustive list on some research direction follows.

First and foremost, we lack a deeper understanding on the computational complexity of many practical fragments, even if we restrict ourselves to the discrete-time case. A big source of complexity in planning for hybrid systems comes from the adoption of unbounded numeric and continuous variables; it is hard to have procedures that are guaranteed to terminate over a state space that can get infinite. Yet, for several realistic scenarios, it is more than reasonable to still use numeric variables but have bounds on the possible values they can ever reach. Although we are very recently starting to see initial studies of these settings [Gnad *et al.*, 2023; Gigante and Scala, 2023], work is still needed at least to i) understand how these results transfer into practice through new algorithms and heuristics that exploit these assumptions for better reasoning ii) extend these works for the temporal case. In this latter regard, the available results using time in planning are restricted to states only allowing Boolean variables [Rintanen, 07; Gigante *et al.*, 2022].

Secondly, as outlined in Section 4, the current relaxation-based frameworks provide solutions mainly for two extreme situations only. Subgoaling-based relaxation and constraints-based heuristics [Scala *et al.*, 2020a; Kuroiwa *et al.*, 2021; Piacentini *et al.*, 2018b] put emphasis on very restricted formulations where actions can at most have linear effects and numeric terms only involve linear conditions. Interval-based relaxation [Hoffmann, 2003; Scala *et al.*, 2016; Aldinger and Nebel, 2017] on the other hand, does extend the framework for very general numeric expressions, yet is practical less efficient than the subgoaling. How to mix the them effectively is still an open question. An interesting direction is also the use of neural network to learn heuristic functions, a line of research very active in other areas (e.g., [Toyer *et al.*, 2020; Ferber *et al.*, 2020]).

Third, we lack work aimed at studying the adoption of trajectory constraints in hybrid systems represented in PDDL+. [Aineto *et al.*, 2023] observed that many verification problems require planners to be able to reason using signal temporal logic, which can be understood as enforcing trajectory constraints. This subject is at the moment studied mostly for classical planning problems [De Giacomo *et al.*, 2014; Bacchus and Kabanza, 1998; Baier and McIlraith, 2006; Bonassi *et al.*, 2021; Bonassi *et al.*, 2022; Bonassi *et al.*, 2023], but completely missing when we have temporal and numeric information to deal with.

Acknowledgments

This work has only been possible thanks to the people I have worked with, and it is really not possible to cite them all. But a special thank is due to Patrik Haslum, Miquel Ramirez and Sylvie Thiebaux who introduced me to heuristic search, and Diego Aineto, Nicola Gigante, Andrea Micheli, Eva Onainda, Francesco Percassi, Ivan Serina, Mauro Vallati with whom I more recently started to share interest for all sort of temporal and hybrid planning problems. A big thank goes of course to Pietro Torasso who introduced me to AI, and all the people I work with from the University of Brescia.

References

[Aineto *et al.*, 2023] Diego Aineto, Enrico Scala, Eva Onaindia, and Ivan Serina. Falsification of cyber-physical systems using pddl+ planning. In *ICAPS*, page to appear. AAAI Press, 2023.

[Aldinger and Nebel, 2017] Johannes Aldinger and Bernhard Nebel. Interval based relaxation heuristics for numeric planning with action costs. In *KI*, volume 10505 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 2017.

[Bacchus and Kabanza, 1998] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):5–27, 1998.

[Baier and McIlraith, 2006] Jorge A Baier and Sheila A McIlraith. Planning with first-order temporally extended goals using heuristic search. In *National Conference on Artificial Intelligence, (AAAI-06)*, 2006.

[Bonassi *et al.*, 2021] Luigi Bonassi, Alfonso Emilio Gerevini, Francesco Percassi, and Enrico Scala. On planning with qualitative state-trajectory constraints in PDDL3 by compiling them away. In *ICAPS*, pages 46–50. AAAI Press, 2021.

[Bonassi *et al.*, 2022] Luigi Bonassi, Alfonso Emilio Gerevini, and Enrico Scala. Planning with qualitative action-trajectory constraints in PDDL. In *IJCAI*, pages 4606–4613. ijcai.org, 2022.

[Bonassi *et al.*, 2023] Luigi Bonassi, Giuseppe De Giacomo, Marco Favorito, Francesco Fuggitti, Alfonso Emilio Gerevini, and Enrico Scala. Planning for temporally extended goals in pure-past linear temporal logic. In *ICAPS*, page to appear. AAAI Press, 2023.

[Cardellini *et al.*, 2021] Matteo Cardellini, Marco Maratea, Mauro Vallati, Gianluca Boleto, and Luca Oneto. Instation train dispatching: A PDDL+ planning approach. In *ICAPS*, pages 450–458. AAAI Press, 2021.

[Cashmore *et al.*, 2020] Michael Cashmore, Daniele Magazzeni, and Parisa Zehtabi. Planning for hybrid systems via satisfiability modulo theories. *Journal of Artificial Intelligence Research*, 67:235–283, 2020.

[Corso *et al.*, 2021] Anthony Corso, Robert J. Moss, Mark Koren, Ritchie Lee, and Mykel J. Kochenderfer. A survey of algorithms for black-box safety validation of cyber-physical systems. *J. Artif. Intell. Res.*, 72:377–428, 2021.

[De Giacomo *et al.*, 2014] Giuseppe De Giacomo, Riccardo De Masellis, and Marco Montali. Reasoning on ltl on finite traces: Insensitivity to infiniteness. In *AAAI*, pages 1027–1033, 2014.

[Dechter and Pearl, 1985] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a*. *J. ACM*, 32(3):505–536, 1985.

[Doyen *et al.*, 2018] Laurent Doyen, Goran Frehse, George J. Pappas, and André Platzer. Verification of Hybrid Systems. In *Handbook of Model Checking*, pages 1047–1110. Springer, 2018.

[Felner, 2011] Ariel Felner. Position paper: Dijkstra's algorithm versus uniform cost search or a case against dijkstra's algorithm. In *SOCS*. AAAI Press, 2011.

[Ferber *et al.*, 2020] Patrick Ferber, Malte Helmert, and Jörg Hoffmann. Neural network heuristics for classical planning: A study of hyperparameter space. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2346–2353. IOS Press, 2020.

[Fikes and Nilsson, 1971] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[Fox and Long, 2006] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research (JAIR)*, 27:235–297, 2006.

[Fox *et al.*, 2012] Maria Fox, Derek Long, and Daniele Magazzeni. Plan-based policies for efficient multiple battery load management. *J. Artif. Intell. Res.*, 44:335–382, 2012.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[Gigante and Scala, 2023] Nicola Gigante and Enrico Scala. On the compilability of bounded numeric planning. In *IJCAI*, page to appear. ijcai.org, 2023.

[Gigante *et al.*, 2022] Nicola Gigante, Andrea Micheli, Angelo Montanari, and Enrico Scala. Decidability and complexity of action-based temporal planning over dense time. *Artif. Intell.*, 307:103686, 2022.

[Gnad *et al.*, 2023] Daniel Gnad, Malte Helmert, Peter Jonsson, and Alexander Shleyfman. Planning over integers: Compilations and undecidability. In *ICAPS*, page to appear. AAAI Press, 2023.

[Goebel *et al.*, 2018] Randy Goebel, Ajay Chander, Katharina Holzinger, Freddy Lécué, Zeynep Akata, Simone Stumpf, Peter Kieseberg, and Andreas Holzinger. Explainable AI: the new 42? In *CD-MAKE*, Lecture Notes in Computer Science, pages 295–303. Springer, 2018.

[Haslum and Geffner, 2000] Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In *AIPS*, pages 140–149. AAAI, 2000.

[Haslum *et al.*, 2019] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2019.

[Helmert, 2002] Malte Helmert. Decidability and undecidability results for planning with numerical state variables. In *Proc,. of International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, pages 44–53, 2002.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Henzinger, 2000] Thomas A. Henzinger. The Theory of Hybrid Automata. In *Verification of Digital and Hybrid Systems*, pages 265–292. Springer Berlin Heidelberg, 2000.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.*, 14:253–302, 2001.

[Hoffmann, 2003] Jörg Hoffmann. The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *J. Artif. Intell. Res.*, 20:291–341, 2003.

[Kiam *et al.*, 2020] Jane Jean Kiam, Enrico Scala, Miquel Ramírez Javega, and Axel Schulte. An ai-based planning framework for HAPS in a time-varying environment. In *ICAPS*, pages 412–420. AAAI Press, 2020.

[Kuroiwa *et al.*, 2021] Ryo Kuroiwa, Alexander Shleyfman, Chiara Piacentini, Margarita P Castro, and J Christopher Beck. Lm-cut and operator counting heuristics for optimal numeric planning with simple conditions. In *ICAPS*, volume 31, pages 210–218, 2021.

[Kuroiwa *et al.*, 2022a] Ryo Kuroiwa, Alexander Shleyfman, and J. Christopher Beck. Lm-cut heuristics for optimal linear numeric planning. In *ICAPS*, pages 203–212. AAAI Press, 2022.

[Kuroiwa *et al.*, 2022b] Ryo Kuroiwa, Alexander Shleyfman, Chiara Piacentini, Margarita P. Castro, and J. Christopher Beck. The lm-cut heuristic family for optimal numeric planning with simple conditions. *J. Artif. Intell. Res.*, 75:1477–1548, 2022.

[Li *et al.*, 2018] Dongxu Li, Enrico Scala, Patrik Haslum, and Sergiy Bogomolov. Effect-abstraction based relaxation for linear numeric planning. In *IJCAI*, pages 4787–4793. ijcai.org, 2018.

[McDermott, 2003] Drew V. McDermott. Reasoning about autonomous processes in an estimated-regression planner. In *ICAPS*, pages 143–152, 2003.

[Moore *et al.*, 2009] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.

[Penna *et al.*, 2009] Giuseppe Della Penna, Daniele Magazzeni, Fabio Mercorio, and Benedetto Intrigila. UPMurphi: A Tool for Universal Planning on PDDL+ Problems. In *ICAPS*, 2009.

[Percassi *et al.*, 2023] Francesco Percassi, Enrico Scala, and Mauro Vallati. A practical approach to discretised PDDL+ problems by translation to numeric planning. *J. Artif. Intell. Res.*, 76:115–162, 2023.

[Piacentini *et al.*, 2018a] Chiara Piacentini, Margarita P. Castro, André Augusto Ciré, and J. Christopher Beck. Compiling optimal numeric planning to mixed integer linear programming. In *ICAPS*, pages 383–387. AAAI Press, 2018.

[Piacentini *et al.*, 2018b] Chiara Piacentini, Margarita P. Castro, André Augusto Ciré, and J. Christopher Beck. Linear and integer programming-based heuristics for cost-optimal numeric planning. In *AAAI*, pages 6254–6261. AAAI Press, 2018.

[Piotrowski *et al.*, 2023] Wiktor Piotrowski, Yoni Sher, Sachin Grover, Roni Stern, and Shiwali Mohan. Heuristic search for physics-based problems: Angry birds in PDDL+. *CoRR*, abs/2303.16967, 2023.

[Ramírez *et al.*, 2017] Miquel Ramírez, Enrico Scala, Patrik Haslum, and Sylvie Thiébaux. Numerical integration and dynamic discretization in heuristic search planning over hybrid domains. *CoRR*, abs/1703.04232, 2017.

[Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.*, 39:127–177, 2010.

[Rintanen, 07] Jussi Rintanen. Complexity of concurrent temporal planning. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS-07)*, page 287, 07.

[Scala *et al.*, 2016] Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramírez. Interval-based relaxation for general numeric planning. In *Proceedings of ECAI*, volume 285, pages 655–663, 2016.

[Scala *et al.*, 2020a] Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramirez. Subgoaling techniques for satisficing and optimal numeric planning. *Journal of Artificial Intelligence Research*, 68:691–752, 2020.

[Scala *et al.*, 2020b] Enrico Scala, Alessandro Saetti, Ivan Serina, and Alfonso Emilio Gerevini. Search-guidance mechanisms for numeric planning through subgoaling relaxation. In *ICAPS*, 2020.

[Toyer *et al.*, 2020] Sam Toyer, Sylvie Thiébaux, Felipe W. Trevizan, and Lexing Xie. Asnets: Deep learning for generalised planning. *J. Artif. Intell. Res.*, 68:1–68, 2020.

[Vallati *et al.*, 2016] Mauro Vallati, Daniele Magazzeni, Bart De Schutter, Lukás Chrpa, and Thomas Leo McCluskey. Efficient macroscopic urban traffic models for reducing congestion: A PDDL+ planning approach. In *Proceedings of AAAI*, pages 3188–3194, 2016.