Thesis Abstract: Constructing Heuristics for PDDL+ Planning Domains

Wiktor Piotrowski
Supervised by: **Daniele Magazzeni** and **Maria Fox**
Department of Informatics
King's College London
United Kingdom

Abstract

Planning with hybrid domains modelled in PDDL+ has been gaining research interest in the Automated Planning community in recent years. Hybrid domain models capture a more accurate representation of real world problems, that involve continuous processes, than is possible using discrete systems. However, solving problems represented as PDDL+ domains is very challenging due to the construction of complex system dynamics, including non-linear processes and events, and vast search spaces.

The main focus of my PhD is to mitigate these challenges by developing domain-independent heuristics for planning in hybrid domains modelled in PDDL+. This is a very real issue as only a handful of planners can cope with hybrid domains and, fewer still with the full set of PDDL+ features and non-linear behaviour.

1 Introduction

Over the years, Automated Planning research has been continuously attempting to solve the most advanced and complex planning problems. The standard modelling language, PDDL (McDermott et al. 1998), has been evolving to accommodate new concepts and operations, enabling research to tackle problems more accurately representing real-world scenarios. Recent versions of the language, PDDL2.1 and PDDL+ (Fox and Long 2003; 2006), enabled the most accurate standardised way yet, of defining hybrid problems as planning domains.

Planning with PDDL+ has been gaining research interest in the Automated Planning community in recent years. Hybrid domains are models of systems which exhibit both continuous and discrete behaviour. They are amongst the most advanced models of systems and the resulting problems are notoriously difficult for planners to cope with due to non-linear behaviour and immense search spaces.

My research aims mitigate these issues by developing domain-independent heuristics able to reason with nonlinear system dynamics and PDDL+ features such as processes and events. These heuristics are being implemented in UPMurphi as a proof of concept.

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

We begin by outlining the related work done in the area of hybrid domains and PDDL+ planning in section 2. We discuss the relevance of the research problem and motivation for tackling it in section 3. Section 4 describes our methodology for dealing with hybrid domains. We then outline the contribution made and ongoing research in section 5. Section 6 describes the future research. Section 7 concludes the thesis summary.

2 Related Work

Various techniques and tools have been proposed to deal with hybrid domains (Penberthy and Weld 1994; McDermott 2003; Li and Williams 2008; Coles et al. 2012; Shin and Davis 2005). Nevertheless, none of these approaches are able to handle the full set of PDDL+ features, namely non-linear domains with processes and events. More recent approaches in this direction have been proposed by (Bogomolov et al. 2014), where the close relationship between hybrid planning domains and hybrid automata is explored. (Bryce et al. 2015) use dReach with a SMT solver to handle hybrid domains. However, dReach does not use PDDL+, and cannot handle exogenous events.

On the other hand, many works have been proposed in the model checking and control communities to handle hybrid systems. Some examples include (Cimatti et al. 2015; Cavada et al. 2014; Tabuada, Pappas, and Lima 2002; Maly et al. 2013), sampling-based planners (Karaman et al. 2011; Lahijanian, Kavraki, and Vardi 2014). Another related direction is *falsification* of hybrid systems (i.e., guiding the search towards the error states, that can be easily cast as a planning problem) (Plaku, Kavraki, and Vardi 2013; Cimatti et al. 1997). However, while all these works aim to address a similar problem, they cannot be used to handle PDDL+ models. Some recent works (Bogomolov et al. 2014; 2015) are trying to define a formal translation between PDDL+ and standard hybrid automata, but so far only an over-approximation has been defined, that allows the use of those tools only for proving plan non-existence.

To date, the only viable approach in this direction is PDDL+ planning via *discretisation*. UPMurphi (Della Penna, Magazzeni, and Mercorio 2012), which implements the discretise and validate approach, is able to deal with the full range of PDDL+ features. The main drawback of UPMurphi is the lack of heuristics, and this strongly limits its

scalability. However, UPMurphi was successfully used in the multiple-battery management domain (Fox, Long, and Magazzeni 2012), and more recently for urban traffic control (Vallati et al. 2016). In both cases, a domain-*specific* heuristic was used.

3 Problem Statement & Motivation

Automated Planning is a crucial part of a multitude of systems in almost every domain of science and technology. However, in certain cases the complexity or scale of the systems has outgrown the capabilities of the modelling language rendering the planning domain either too inaccurate or too cumbersome to express.

When first introduced, PDDL+ allowed new, more complex problems, closely resembling real-world scenarios, to be modelled. Though planning is now able to express complex hybrid domains, solving these problems is very challenging due to nonlinear behaviour, state explosion and continuous variables rendering the reachability problem undecidable. As described in Section 2, various planning tools using different approaches have been developed over the past years to tackle problems set in hybrid domains. However, the vast majority cannot deal with the full set of PDDL+ features and/or nonlinear behaviour. This significantly limits the relevance of Automated Planning for a wide range of applications since only restricted and/or downscaled hybrid models can be handled. As a result, some classes of planning problems, relevant to today's science and technology, are being solved using domain-specific heuristics or approaches from outside planning altogether (Mixed-Integer Programming, Genetic Algorithms, etc.).

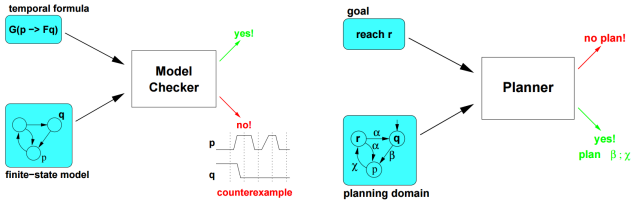
In addition to the inability to reason with some PDDL+ features and/or non-linearity, current planning tools scale poorly when presented with larger problem instances. This is due to the absence or poor performance of heuristics in the presence of vast search space, exogenous processes and events, and non-linearity. Currently, heuristics applied to hybrid systems have either been developed for a different subclass of problems (e.g. PDDL2.1), or to reason with only a subset of the features expressible in PDDL+.

The main motivation of this research is to advance PDDL+ planning to tackle larger and more complex problems by addressing the apparent lack of efficient domain-independent heuristics devised specifically for hybrid domains. To significantly increase the performance of planners in hybrid domains, heuristics should be designed to directly reason with the complex system dynamics, and PDDL+ features, i.e. processes and events.

4 Methodology

Reasoning with PDDL+ features and complex, often nonlinear, system dynamics is a challenging objective for Automated Planning tools. As shown in Section 2, all current approaches have drawbacks significantly limiting their performance and capabilities, often rendering them inadequate for the complex problems at hand.

Our approach of coping with PDDL+ domains combines two successful paradigms, ***planning as model checking*** and



(a) Model Checking (b) Planning

Figure 1: Similarity of Model Checking and Automated Planning

Discretise & Validate.

Planning as model checking (Cimatti et al. 1997; Bogomolov et al. 2014) is an approach applying model checking methods to finding the goal state in Automated Planning. Model checking and planning have striking similarities meaning that methods from one field can be exploited to improve performance in the other field. Searching for a goal in planning (Fig.1a) can be seen as searching for an error state in model checking(Fig.1b). Analogously, the error trace in model checking corresponds to a trajectory to the goal state in planning. Planning as model checking has been successfully used in various scenarios, and is gaining more research interest (multiple publications and workshops at top conferences).

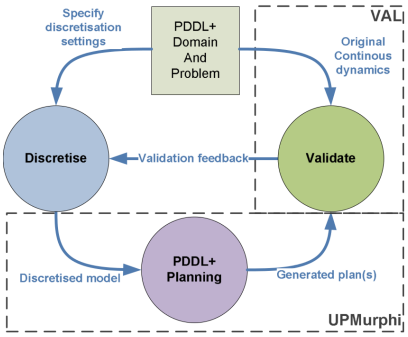


Figure 2: The Discretise & Validate process

As mentioned before, planning via discretisation is, to date, the only viable approach to PDDL+ planning. Our approach is based on the Discretise & Validate approach (Della Penna, Magazzeni, and Mercorio 2012; Della Penna et al. 2009). It approximates the continuous dynamics of systems in a discretised model with uniform time steps and step functions. Using a discretised model and a finite-time horizon ensures a finite number of states in the search for a solution. Solutions to the discretised problem are validated against the original continuous model using VAL (Howey, Long, and Fox 2004). If the plan at a certain discretisation is not valid, the discretisation can be refined and the process iterates. Discretise & Validate technique has been the basis of UPMurphi's success in the planning domain. An outline of the Discretise & Validate process is shown in Fig. 2.

5 Contributions

This section describes my contribution to date. Our development resulted in DiNo, a new heuristic planner designed for PDDL+ domains equipped with informed search algorithms and Staged Relaxed Planning Graph+ (SRPG+), a new domain-independent heuristic based on the Temporal Relaxed Planning Graph. Our publication describing DiNo is currently under review for IJCAI 2016.

5.1 DiNo

We introduced DiNo, a new planner for PDDL+ problems with mixed discrete-continuous non-linear dynamics. DiNo is built on UPMurphi, and uses the planning-as-model-checking paradigm and relies on the Discretise & Validate approach to handle continuous change and non-linearity. Though UPMurphi has been successful, it scales poorly due to exhaustive uninformed search algorithm, DiNo compensates for the lack of heuristics and shows significant improvement over its predecessor.

DiNo uses a novel relaxation-based domain-independent heuristic for PDDL+, Staged Relaxed Planning Graph+ (SRPG+). The heuristic guides the Enforced Hill-Climbing algorithm (Hoffmann and Nebel 2001). In DiNo we also exploit the deferred heuristic evaluation (Richter and Westphal 2010) for completeness (in a discretised search space with a finite horizon). States generated through non-helpful actions are considered (inserted into the queue) but they are not heuristically evaluated, instead they are assigned the heuristic value of their parent state (i.e. the are deemed no better than their parent state).

DiNo is currently the only heuristic planner capable of handling non-linear system dynamics combined with the full PDDL+ feature set.

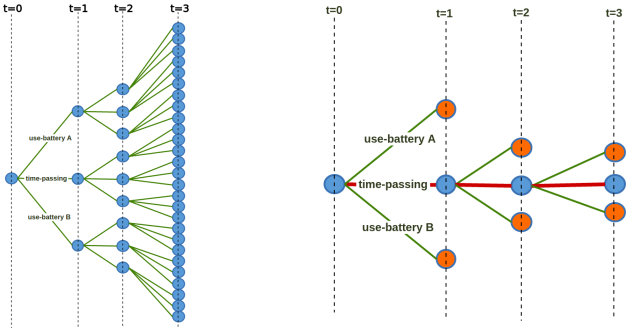
5.2 SRPG+

This section describes the Staged Relaxed Planning Graph+ (SRPG+) domain-independent heuristic designed for PDDL+ domains and implemented in DiNo.

The SRPG+ heuristic follows from Propositional (Hoffmann and Nebel 2001), Numeric (Hoffmann 2003; 2002) and Temporal RPGs (Coles et al. 2012; 2008; Coles and Coles 2013). The original problem is relaxed and does not account for the delete effects of actions on propositional facts. Numeric variables are represented as upper and lower bounds which are the theoretical highest and lowest values each variable can take at the given fact layer. Each layer is time-stamped to keep track of the time at which it occurs.

The Staged Relaxed Planning Graph+, however, extends the capability of its RPG predecessors by tracking processes and events to more accurately capture the continuous and discrete evolution of the system.

Apart from the inclusion of processes and events, the Staged RPG significantly differs from the Temporal RPG in time-handling. The SRPG explicitly represents ***every*** fact layer with the corresponding time clock, and in this sense the RPG is "staged", as the finite set of fact layers are separated by the discretised time step (Δt). In contrast, the TRPG takes time constraints into account by time-stamping each fact layer at the earliest possible occurrence of a happening. Only fact layers where values are directly affected by actions are contained in the TRPG.



(a) UPMurphi (b) DiNo

Figure 3: Branching of search trees (Blue states are explored, orange are visited. Red edges correspond to helpful actions)

Time Passing The time-passing action plays an important role as it propagates the search in the discretised timeline. During the normal expansion of the Staged Relaxed Planning Graph, the time-passing is one of the Δ-actions and is applied at each fact layer. Time-passing can be recognised as a helpful action (Hoffmann and Nebel 2001) when its effects achieve some goal conditions (or intermediate goal facts). However, if at a time t there are no helpful actions available to the planner, time-passing is assigned highest priority and used as a helpful action. This allows the search to quickly manage states at time t where no happenings of interest are likely to occur.

This is the key innovation with respect to the standard search in the discretised timeline performed, e.g., by UPMurphi. Indeed, the main drawback of UPMurphi is in that it needs to expand the states at each time step, even during the *idle periods*, i.e., when no interesting interactions or effects can happen. Conversely, SRPG+ allows DiNo to identify time-passing as a helpful action during *idle periods* and thus advance time, mitigating the state explosions.

An illustrative example is shown in Figure 3, that compares the branching of the search in UPMurphi (Fig. 3a) and DiNo (Fig. 3b) when planning with a Solar Rover domain. The domain is described in detail in Section 5. Here we highlight that the planner can decide to use two batteries, but the goal can only be achieved thanks to a Timed Initial Literal ((Edelkamp and Hoffmann 2004)) that is triggered only late in the plan. UPMurphi has no information about the future TIL, therefore it tries to use the batteries at each time step. On the contrary, DiNo recognises the time-passing as a helpful action, and this prunes the state space dramatically.

Processes and Events The SPRG+ heuristic improves on the Temporal Relaxed Planning Graph and extends its functionality to reason with information gained from PDDL+ features, namely the processes and events.

As the SRPG+ heuristic is tailored for PDDL+ domains, it takes into account processes and events. In the SRPG,

the continuous effects of processes are handled in the same manner as durative action effects, i.e. at each action layer, the numeric variables upper and lower bounds are updated based on the time-step functions used in the discretisation to approximate the continuous dynamics of the domain.

Events are checked immediately after processes and their effects are relaxed as for the instantaneous actions. The events can be divided into "good" and "bad" categories. "Good" events aid in finding the goal whereas "bad" events either hinder or completely disallow reaching the goal. Currently, DiNo is agnostic about this distinction. However, as a direct consequence of the SRPG+ behaviour, DiNo exploits good events and ignores the bad ones. Future work will explore the possibility of inferring more information about good and bad events from the domain.

5.3 Evaluation of DiNo

In this section the performance of DiNo is evaluated on PDDL+ benchmark domains. Note that the only planner able to deal with the same class of problems is UPMurphi, which is also the most interesting competitor as it can highlight the benefits of the proposed heuristic. For sake of completeness, where possible, a comparison with other planners able to handle (sub-class of) PDDL+ features is presented, namely POPF (Coles et al. 2010; Coles and Coles 2013) and dReach (Bryce et al. 2015).

For the experimental evaluation, two benchmark domains were considered: generator and car. We also developed two further domains for the evaluation to highlight specific aspects of DiNo: Solar Rover shows how DiNo handles TILs, and Powered Descent further tests its non-linear capabilities.

Generator The domain models a diesel-powered generator which has to be refueled to run for a given duration without overflowing or running dry. We evaluate DiNo on both the linear and non-linear versions of the problem. The non-linear generator models fuel flow rate using Torricelli's Law which has been previously encoded in PDDL by (Howey and Long 2003). In both variants, we increase the number of tanks available to the planner while decreasing the initial generator fuel level for each subsequent problem.

Solar Rover We developed the Solar Rover domain to test the limits and potentially overwhelm discretisation-based planners, as finding a solution to this problem relies on a TIL that is triggered only late in the plan. The task revolves around a planetary rover transmitting data which requires a certain amount of energy. To generate enough energy the rover can choose to use its batteries or gain energy through its solar panels. The goal can only be reached through a sunshine event which is triggered by a TIL at a certain time. The set of problem instances for this domain has the trigger fact become true at an increasingly further time point (50 to 1000 time units). In the non-linear variant of the domain, the TIL triggers a process charging the rover's battery at an exponential rate.

Powered Descent We developed a new domain which models a powered spacecraft landing on a given celestial body. The vehicle gains velocity due to the force of gravity. The available action is to fire thrusters to decrease its velocity. The thrust action duration is flexible and depends on the available propellant mass. The force of thrust is calculated via Tsiolkovsky rocket equation (Turner 2008). The goal is to make a controlled landing from the initial altitude within a given time-frame. The spacecraft has been modelled after the Lunar Descent Module used in NASA's Apollo missions.

Car The Car domain (Fox and Long 2006) shows that DiNo does not perform well on all types of problems, the heuristic cannot extract enough information from the domain and as a result loses out to UPMurphi by approximately one order of magnitude. This variant of the Car domain has its overall duration and acceleration limited, and the problems are set with increasing bounds on the acceleration (corresponding to the problem number).

Due to lack of heuristic information extracted from the domain, DiNo reverts to a blind Breadth-First search and, in the end, explores the same number of states as UPMurphi. The results for the Car domain in Table 1 show the overhead generated by the SRPG+ heuristic in DiNo.

Overall, the results show that DiNo holds a significant advantage in performance over UPMurphi and other competitors in most test domains.

6 Future Research

This section describes the ongoing research and future plans for my PhD. The focus is on defining and developing new heuristics as well as promoting the PDDL+ planning in the AI community.

6.1 Temporal Pattern Database

After developing the Staged relaxed Planning Graph+, we decided to examine another successful class of heuristics - Pattern Databases (PDB). The pattern database is a look-up table indexed by a subset of the state and containing a precomputed heuristic value that reflects the cost of solving the corresponding subproblem. Each state explored during concrete search is assigned the abstract cost of its corresponding abstract state in the PDB, as the heuristic value.

The key element to a high-performing PDB is the abstraction selection. In planning, PDBs have been applied to propositional domains where the abstraction would obscure part of each state's variable set ((Edelkamp 2002; Haslum et al. 2007; Edelkamp 2014)). On the other hand, research in PDBs in model checking has concentrated on abstracting continuous variables ((Bogomolov et al. 2013)).

We build on research conducted in both fields to develop Temporal Pattern Database (TPDB), a new heuristic method with novel features enabling tackling complex problems with non-linear dynamics and full PDDL+ feature set. Our heuristic simultaneously handles full PDDL+ domains and prunes a substantial part of the search space. Processes and events are accounted for by default, their effects are automatically applied when building the TPDB.

The abstraction we devised is two fold: time abstraction and state abstraction. Combining the two abstractions manages to keep the TPDB efficient and reasonable in size.

Time abstraction is a function which increases the discretised time step (Δt) for use in the abstract state space in the

	LINEAR GENERATOR				NON-LINEAR GENERATOR		LINEAR SOLAR ROVER		NON-LINEAR SOLAR ROVER		POWERED DESCENT		CAR	
PROBLEM	**DiNo**	**POPF**	**dReach**	**UPMurphi**	**DiNo**	**UPMurphi**	**DiNo**	**UPMurphi**	**DiNo**	**UPMurphi**	**DiNo**	**UPMurphi**	**DiNo**	**UPMurphi**
1	0.34	0.01	2.87	140.50	3.62	X	0.70	203.26	1.10	288.94	0.68	0.18	1.74	0.22
2	0.40	0.01	X	X	0.78	X	0.92	X	2.58	X	1.04	0.74	4.56	0.30
3	0.50	0.05	X	X	2.86	X	1.26	X	4.74	X	1.88	2.98	8.26	0.42
4	0.60	0.41	X	X	59.62	X	1.52	X	7.10	X	3.52	7.18	10.28	0.54
5	0.74	6.25	X	X	1051.84	X	1.80	X	9.58	X	2.88	30.08	14.16	0.66
6	0.88	120.49	X	X	X	X	2.04	X	12.86	X	3.14	126.08	15.78	0.68
7	1.00	X	X	X	X	X	2.28	X	16.48	X	5.26	322.16	17.08	0.72
8	1.16	X	X	X	X	X	2.64	X	21.38	X	3.82	879.52	18.90	0.72
9	1.38	X	X	X	X	X	2.98	X	26.74	X	1.58	974.60	19.30	0.76
10	2.00	X	X	X	X	X	3.30	X	29.90	X	2.26	X	19.50	0.78
11	1.84	X	X	X	N/A	N/A	3.50	X	35.96	X	11.24	X	N/A	N/A
12	2.06	X	X	X	N/A	N/A	3.74	X	42.54	X	42.24	X	N/A	N/A
13	2.32	X	X	X	N/A	N/A	4.00	X	48.06	X	14.90	X	N/A	N/A
14	2.46	X	X	X	N/A	N/A	4.38	X	55.46	X	61.94	X	N/A	N/A
15	2.88	X	X	X	N/A	N/A	5.20	X	62.84	X	19.86	X	N/A	N/A
16	2.94	X	X	X	N/A	N/A	5.40	X	74.50	X	80.28	X	N/A	N/A
17	3.42	X	X	X	N/A	N/A	5.08	X	86.96	X	2.94	X	N/A	N/A
18	3.54	X	X	X	N/A	N/A	5.64	X	95.66	X	2234.88	X	N/A	N/A
19	3.76	X	X	X	N/A	N/A	6.12	X	102.86	X	X	X	N/A	N/A
20	4.26	X	X	X	N/A	N/A	6.02	X	117.48	X	X	X	N/A	N/A

Table 1: Run time to find a valid solution (in seconds) ("X" - planner ran out of memory, "N/A" - problem not tested)

TPDB. This allows the TPDB to store fewer states and keep its size manageable. However, choosing the correct time abstraction has to be a compromise between the size of the TPDB and the precision. Too coarse abstraction can miss the adverse events occurring between the abstract state time points and cause significant back-tracking.

State abstraction is a function which reduces the precision of states' continuous variables. This method compensates for the discrepancies between the values of continuous variables in concrete and abstract states. Due to the varied discretisation, real variables in abstract states in the TPDB (generated using the abstracted time step) can differ from their corresponding variables in concrete states (achieved using the concrete time step Δt). Choosing the precision for the state abstraction is crucial for the Temporal Pattern Database. On the one hand, choosing a coarser precision for real variables will shrink the size of the TPDB (each abstract state will correspond to a larger number of concrete states). On the other hand, choosing finer precision will make the heuristic estimates more accurate. When choosing the precision value, one should aim to balance the two aspects.

The Temporal Pattern Database is a structure which maps abstract state-action pairs to the length of the shortest trajectory to an abstract goal state. A TPDB is built by executing the applicable actions under time abstraction to generate the subsequent abstract states until a the abstract goal state is found, or the finite temporal horizon T is reached (meaning the bounded abstract problem is unsolvable and the horizon should be increased, or the discretisation refined). The abstract distances stored in the TPDB are used in the concrete search as the heuristic estimate for each considered state.

We are currently in the process of implementing the Temporal Pattern Database into UPMurphi to evaluate our concept. PDBs have proven to be a high performing approach for both model checking and propositional planning. We believe that transforming this approach to reason with PDDL+ domains can generate a very efficient and powerful heuristic.

6.2 PDDL+ Benchmarks

Conducting novel research is obviously crucial to a PhD but helping in identifying and mitigating the shortcomings of one's field of study is just as important.

One of the most pertinent inconsistencies in PDDL+ planning was the use of domains for evaluation. Using inconsistent domains for empirical evaluation makes comparison of planner performances difficult or impossible. To date, no standardised set of benchmark domains exists. Despite the Generator and Car domains being thought of as benchmark domains, multiple variants of them exist and are being used by different planning research groups around the world. We concluded that a set of benchmark PDDL+ domains needs to be readily available for the community to provide a fair, unbiased comparison with competing planners, and began compiling the test suite.

First, we collected the standard hybrid domains used in planning and model checking literature. Examples of these include Generator, Car,and Battery Management. To further expand the suite, we began developing our own PDDL+ models. The top priority for our research is to diversify the domains to account for various classes of problems that can be encoded in PDDL+.

7 Conclusion

Efficient, high-performing heuristics are an integral and essential part of Automated Planning. With the PDDL+ planning area increasingly gaining research interest, it is very important to continue developing advanced heuristics to match the requirements of emerging hybrid domains. Heuristics need to mitigate vast search spaces but also reason with complex system dynamics to estimate the evolution of the system to a satisfactory extent. On the other hand, it is crucial to continue inventing novel PDDL+ domains to further test planning tools and set up a benchmark suite to allow fair comparison between competitor planners.

We have presented the Staged Relaxed Planning Graph+, a domain-independent heuristic developed entirely for PDDL+ planning domains implemented in DiNo, the first heuristic planner capable of handling the full PDDL+ feature set and non-linear system dynamics. We have shown that reasoning directly with processes and events can produce advantages in performance of a planner.

Pattern Database heuristics proved successful in model checking and classical planning. Our current research on the Temporal Pattern Database is built upon these approaches and shows promise for complex PDDL+ domains. The immediate future will be dominated by our efforts to implement the TPDB heuristic into UPMurphi and empirically evaluate its performance on PDDL+ domains.

References

Bogomolov, S.; Donzé, A.; Frehse, G.; Grosu, R.; Johnson, T. T.; Ladan, H.; Podelski, A.; and Wehrle, M. 2013. Abstraction-based guided search for hybrid systems. In *Model Checking Software*. Springer. 117–134.

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle, M. 2014. Planning as Model Checking in Hybrid Domains. In *AAAI*. AAAI Press.

Bogomolov, S.; Magazzeni, D.; Minopoli, S.; and Wehrle, M. 2015. PDDL+ planning with hybrid automata: Foundations of translating must behavior. In *ICAPS*, 42–46.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P. 2015. SMT-Based Nonlinear PDDL+ Planning. In *AAAI*, 3247–3253.

Cavada, R.; Cimatti, A.; Dorigatti, M.; Griggio, A.; Mariotti, A.; Micheli, A.; Mover, S.; Roveri, M.; and Tonetta, S. 2014. The nuXmv symbolic model checker. In *CAV*, 334–342.

Cimatti, A.; Giunchiglia, E.; Giunchiglia, F.; and Traverso, P. 1997. Planning via model checking: A decision procedure for ar. In *Recent Advances in AI planning*. Springer. 130–142.

Cimatti, A.; Griggio, A.; Mover, S.; and Tonetta, S. 2015. HyComp: An SMT-based model checker for hybrid systems. In *ETAPS*, 52–67.

Coles, A., and Coles, A. 2013. PDDL+ Planning with Events and Linear Processes. *PCD 2013* 35.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. Planning with Problems Requiring Temporal Coordination. In *AAAI*, 892–897.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *ICAPS*, 42–49.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2012. COLIN: Planning with Continuous Linear Numeric Change. *J. Artif. Intell. Res.* 44:1–96.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: A Tool for Universal Planning on PDDL+ Problems. In *ICAPS*. AAAI.

Della Penna, G.; Magazzeni, D.; and Mercorio, F. 2012. A Universal Planning System for Hybrid Domains. *Appl. Intell.* 36(4):932–959.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The language for the classical part of the 4th international planning competition. *IPC at ICAPS*.

Edelkamp, S. 2002. Symbolic pattern databases in heuristic search planning. In *AIPS*, 274–283.

Edelkamp, S. 2014. Planning with pattern databases. In *Sixth European Conference on Planning*.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.* 20:61–124.

Fox, M., and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Artif. Intell. Res.* 27:235–297.

Fox, M.; Long, D.; and Magazzeni, D. 2012. Plan-based Policies for Efficient Multiple Battery Load Management. *J. Artif. Intell. Res.* 44:335–382.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI*, volume 7, 1007–1012.

Hoffmann, J., and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *J. Artif. Intell. Res.* 14:253–302.

Hoffmann, J. 2002. Extending FF to Numerical State Variables. In *ECAI*, 571–575. Citeseer.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating"Ignoring Delete Lists"to Numeric State Variables. *J. Artif. Intell. Res.* 20:291–341.

Howey, R., and Long, D. 2003. VAL's progress: The automatic validation tool for PDDL2. 1 used in the international planning competition. In *IPC at ICAPS*.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning Using PDDL. In *ICTAI*, 294–301. IEEE.

Karaman, S.; Walter, M. R.; Perez, A.; Frazzoli, E.; and Teller, S. J. 2011. Anytime motion planning using the RRT. In *IEEE-ICRA*.

Lahijanian, M.; Kavraki, L. E.; and Vardi, M. Y. 2014. A sampling-based strategy planner for nondeterministic hybrid systems. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 3005–3012.

Li, H. X., and Williams, B. C. 2008. Generative Planning for Hybrid Systems Based on Flow Tubes. In *ICAPS*, 206–213.

Maly, M. R.; Lahijanian, M.; Kavraki, L. E.; Kress-Gazit, H.; and Vardi, M. Y. 2013. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *HSCC*, 353–362.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL - The Planning Domain Definition Language.

McDermott, D. V. 2003. Reasoning about Autonomous Processes in an Estimated-Regression Planner. In *ICAPS*, 143–152.

Penberthy, J. S., and Weld, D. S. 1994. Temporal Planning with Continuous Change. In *AAAI*, 1010–1015.

Plaku, E.; Kavraki, L. E.; and Vardi, M. Y. 2013. Falsification of LTL safety properties in hybrid systems. *STTT* 15(4):305–320.

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.* 39(1):127–177.

Shin, J.-A., and Davis, E. 2005. Processes and Continuous Change in a SAT-based Planner. *Artif. Intell.* 166(1-2):194–253.

Tabuada, P.; Pappas, G. J.; and Lima, P. U. 2002. Composing abstractions of hybrid systems. In *HSCC*, 436–450.

Turner, M. J. 2008. *Rocket and spacecraft propulsion: principles, practice and new developments*. Springer Science & Business Media.

Vallati, M.; Magazzeni, D.; Schutter, B. D.; Chrpa, L.; and Mccluskey, T. L. 2016. Efficient Macroscopic Urban Traffic Models for Reducing Congestion: A PDDL+ Planning Approach. In *AAAI*. AAAI Press.