POPF2: a Forward-Chaining Partial Order Planner

Amanda Coles, Andrew Coles, Maria Fox and Derek Long
Department of Computer and Information Sciences,
University of Strathclyde, Glasgow, G1 1XH, UK
firstname.lastname@cis.strath.ac.uk

Please direct all citations for this planner to one of the following two papers:

(i) To refer to the search approach of POPF2 (or POPF), cite:
Coles, Coles, Fox and Long, *'Forward-Chaining Partial-Order Planning'*, ICAPS 2010

(ii) To refer to the cost-optimisation approach used in POPF2 (or Stochastic-POPF), cite:
Coles, Coles, Clark and Gilmore, *'Cost-Sensitive Concurrent Planning under Duration Uncertainty for Service Level Agreements'*, ICAPS 2011

Abstract

This paper gives an overview of the planner POPF2, as competing in the 2011 International Planning Competition. POPF employs forward-chaining search, expanding a partial-order rather than the conventional total-order: steps added to the plan are ordered after a subset of those in the plan so far, rather than after every step in the plan so far. POPF2 adopts this search approach, extending it in two ways. First, it has a number of changes to allow it to make fewer commitments to ordering constraints, and hence find more makespan-efficient plans. Second, it borrows the cost-optimisation approach of Stochastic-POPF, where modifications to the temporal relaxed planning graph heuristic, and the use of anytime-search, allow it to improve upon the first plan found.

This paper describes the planner POPF2, the latest revision of POPF (Coles *et al.* 2010). The aim of POPF is to preserve the benefits of partial-order plan construction, in terms of producing makespan-efficient, flexible plans; whilst avoiding explicit conflict resolution by always expanding the plan in a forwards direction.

We begin by giving an overview of how POPF works, before describing a number of modifications that further reduce the number of ordering constraints introduced during search in certain situations. We then briefly describe how anytime search is used to find plans of successive quality, through direct application of techniques used in Stochastic-POPF (Coles *et al.* 2011).

1 Background

POPF (Coles *et al.* 2010) is a temporal planner, working under the semantics of PDDL 2.1 (Fox & Long 2003). Each durative action A has:

- duration constraints, placing lower and/or upper bounds on duration of the action.
- instantaneous conditions that must hold either at the start or end of its execution;
- instantaneous propositional and numeric effects that occur at the start and/or end of its execution;
- conditions that must hold continuously `over all` its execution (in the interval between the start and the end);
- numeric effects that occur continuously throughout its execution. In POPF, we insist that such numeric effects are linear, i.e. increase or decrease a numeric variable by a constant amount per unit time.

Under these semantics, each durative action A can be thought of as two instantaneous snap-actions: $A_\vdash$, representing the start of the action, with the associated instantaneous conditions and effects; and $A_\dashv$, similarly representing the end of the action. Applying $A_\vdash$ then begins an interval during which the `over all` conditions of A must be respected, and its continuous numeric effects occur; this interval can then be terminated by applying $A_\dashv$. With this representation, one additional requirement needs to be introduced: for a state to be a goal state, no actions can be executing.

In the general case, states in a temporal planning problem can be characterised by a tuple $\langle F, V, Q, P, C\rangle$, where:

- F is the set of propositions that hold in the state.
- V is the vector of values of the task numeric variables.
- Q is the event queue: is a list of actions whose execution has started but not yet finished. For each $(a, i) \in Q$, a identifies a ground action, and i the step at which it began.
- P is the plan to reach the current state.
- C is a list of temporal constraints over the steps in P.

The temporal constraints in C arise from one of two sources. First, clearly, the duration constraints of actions must be obeyed: the amount of time between $A_\vdash$ and $A_\dashv$ must respect the duration constraint of A. Second, as in non-temporal planning, ordering constraints are needed to ensure the plan is causally sound. In the absence of continuous numeric effects (or instantaneous effects that depend on the duration of actions), these temporal constraints take the form of a Simple Temporal Problem (Dechter, Meiri, & Pearl 1991) (STP), and POPF employs an incremental STP solver (Cesta & Oddi 1996) to check that the temporal constraints at each state are satisfied. In the presence of either or both of these, then if the continuous numeric change is linear, a Linear Programming (LP) encoding is used, where additional constraints encode the relationship between the times at which actions are scheduled and the values of numeric variables at each point in the plan, with the LP constrained to ensure preconditions are met.

2 POPF Forwards Partial-Order Expansion

The search in POPF is based around the idea of expanding a partial-order plan in a forwards direction. Simply, when applying a snap-action in a given state S, it is ordered only after a subset of the actions in the plan to reach S: those with which there is some sort of causal interaction in terms of facts or numeric variables. This is in contrast to its predecessor, COLIN (Coles *et al.* 2009a), where each new snap-action would be ordered after *all* the actions in the plan to reach S.

To support partial-order expansion, the general case definition of a state given in Section 1 is augmented with additional information, recording which steps in the plan interact with a given fact p or numeric variable v. For full details, including how POPF supports continuous numeric effects, we refer the reader to (Coles *et al.* 2010). In the case where all effects are instantaneous, as is the case in the competition, the state annotations can be summarised as follows:

- $F^+(p)$ ($F^-(p)$) records the step in the plan that most recently had an add effect (resp. delete effect) on the fact p;
- $FP(p)$, a set of pairs, each $\langle i, d\rangle$, records preconditions involving p. For each pair, i is a step index in the plan, and d is 0 or ϵ:
  - If $d = 0$, then p can be deleted in parallel to step i. This arises where step i is the end of an action with an `over all` condition on p: under the PDDL 2.1 semantics, deleting p at this point would not be mutually exclusive with ending the action requiring p.
  - In all other cases (start or end conditions of temporal actions, or the preconditions of non-temporal actions), $d = \epsilon$, and hence p can only be deleted epsilon after step i.
- $V^{\mathit{eff}}(v)$ records the step in the plan that most recently had a numeric effect upon the variable v;
- $VP(v)$ is a set, recording which steps in the plan depend on v. A step i depends on v in one of three cases:
  1. i has a precondition referring to v
  2. i has an effect whose outcome depends on the current value of v (e.g. assigning $w = v$);
  3. i is the start of an action whose duration depends on v.

When an action is applied, as step i of a plan, these annotations are first used to ensure the preconditions of the action are met:

- To satisfy a propositional precondition p, we add the ordering constraint $t(i) - t(F^+(p)) \geq t$. If step i is the start of an action, and p is only needed as an `over all` condition of the action, then $t = 0$. In all other cases, $t = \epsilon$: the effect must complete strictly before step i.
- To satisfy a numeric precondition referring to the variable v, we add the ordering constraint $t(i) - t(V^{eff}(v)) \geq t$ (where, again, the value of t depends on the nature of the precondition).
- If i is the start of an action, with a duration constraint referring to v, then $t(i) - t(V^{eff}(v)) \geq \epsilon$.

Following this, the annotations are used and updated to reflect the effects of the action:

- For a propositional delete effect on p, prior to setting $F^-(p) = i$, we add the ordering constraints:

$$t(i) - t(F^+(p)) \geq \epsilon$$
$$\forall_{\langle j,d\rangle \in FP(p)}\, t(i) - t(j) \geq d$$

- For a propositional add effect on p, we add the ordering constraint $t(i) - t(F^-(p)) \geq \epsilon$, and then set $F^+(p) = i$. (As a special case, if $F^-(p) = i$ then no ordering constraint is added, as the action is adding a fact it has just deleted.)
- For a numeric effect referring to v, $t(i) - t(V^{\mathit{eff}}(v)) \geq \epsilon$;
- For a numeric effect acting on v, prior to setting $V^{\mathit{eff}}(v) = i$ and $VP(v) = \emptyset$, we add the ordering constraints:

$$t(i) - t(V^{\mathit{eff}}(v)) \geq \epsilon$$
$$\forall_{j \in VP(v)}\, t(i) - t(j) \geq \epsilon$$

POPF exploits the approach introduced in (Coles *et al.* 2009b) where if an action A is 'compression-safe', $A_{\dashv}$ is immediately added to the plan whenever $A_{\vdash}$ is applied. An action is considered compression safe in this setting if its end preconditions are subsumed by the action's `over all` conditions, and the only end effects are to add propositions (i.e. no numeric effects or delete effects). The state annotations ensure immediately adding $A_{\dashv}$ does not have a catastrophic effect on makespan: actions will only be ordered after $A_{\dashv}$ if they require one of its effects, or violate one of the action's `over all` conditions, which are both circumstances under which they would previously have had to follow $A_{\dashv}$.

3 Introducing Fewer Ordering Constraints

The treatment of numeric variables in the search approach taken in POPF, as described in Section 2, is quite limited. In the interests of generality, effects on numeric variables are totally ordered, and steps requiring a value of v (but not changing its value) are scheduled to occur after all the steps prior to them that modified v, and before all the steps following that modify v. The rationale behind this is that the ordering constraints ensure the value of v is known at every relevant point, and the interleaving of actions with effects and/or preconditions on v cannot be changed in such a way that renders the plan invalid.

To seek to reduce the number of ordering constraints introduced through the use of numeric preconditions and effects, POPF2 performs static analyses on the problem structure to identify patterns of numeric behaviour that can be handled more favourably. The remainder of this section will go through these cases.

3.1 Metric-Tracking Variables with Order-Independent Effects

In various domains, the metric cost function to seek to minimise when planning comprises a number of 'metric-tracking variables'. These are only ever modified by the effects of actions, and the correctness of the plan does not depend on their values: they never appear in preconditions and duration constraints, nor are their values used as a basis for numeric effects. (Indeed, if these variables did not appear in the metric function, they would be irrelevant and could be disregarded entirely.) One example of such a metric-tracking variable can be found in the Time formulation of the ZenoTravel domain from the 2002 International Planning Competition (Long & Fox 2003). Here, the variable `total-fuel-used` is updated by the `fly` and `zoom` actions to record how much fuel has been used so far when constructing the plan. In the problem file set for this domain, the metric is then to minimise a weighted sum of `total-fuel-used` and plan makespan.

If the final value of a metric-tracking variable v does not depend on the order in which the effects upon it are applied, then there is no need to totally order actions affecting v: it suffices that applying all the relevant effects will yield a value of v, which can then be used when determining the quality of the plan found. In other words, the effects on v are 'order independent'. Order-independence can be guaranteed if all effects on v can be written in the form:

$$v \mathrel{+}= c + w_0.v_0 + w_1.v_1 + ... + w_n.v_n$$

...where $c, w_0..w_n \in \Re$, and each $v_i \in [v_0..v_n]$ denotes a state numeric variable. Through the definition of a metric-tracking variable (specifically, that the value of v cannot be used as the basis of a numeric effect), we know that $v \notin [v_0..v_n]$. So long as appropriate ordering constraints are added for each v_i (as discussed in Section 2), each effect on v will then have a known value at the point of being introduced, and the sum of these effect values gives the net effect on v by the plan.

In POPF2, once static analysis has identified a metric-tracking variable v, effects on v will update its value without adding ordering constraints; and when a goal state has been found, the value of v is then available for use in calculating the plan metric. Thus, returning to ZenoTravel, the `total-fuel-used` is increased by the constant-valued effect of each `fly`/`zoom` action, without insisting that these effects are totally ordered.

3.2 As-Needed Ordering after Beneficial Effects

For a given numeric variable v, larger (smaller) values of v may always be preferable, in terms of how the actions in the domain interact with v. Larger values of v are always preferable if:

1. all preconditions (or goals) on v are of the form $v\{\geq, >\}\mathbf{w}.\mathbf{v} + c$, i.e. a larger value of v is more likely to meet the condition;
2. no action has a duration constraint depending on v;
3. the value of v is never used as the basis for a numeric effect.

The first of these is key: for meeting preconditions or goals, a larger v is better. The latter two ensure there are no circumstances in which this might not be the case, and are introduced for simplicity: more sophisticated analyses may be able to relax these, but we leave this to future work. To identify where smaller values of v are preferable, conditions 1 is altered such that all preconditions and goals on v must use a $\leq$ or $<$ operator (rather than $\geq$ or $>$). For the remainder of this subsection, in the interests of clarity, we will discuss only the case where larger is preferable.

If larger values of v are preferable, then we can conclude that increase effects on v are beneficial, and decrease effects on v are not. Returning to the treatment of numeric effects in POPF, the effects on such a variable v are totally ordered, with the most recent effect on v in a given state denoted $V^{\mathit{eff}}(v)$. An action at step i with a precondition on v is then ordered after $V^{\mathit{eff}}(v)$, i.e. after all previous effects on v. In the absence of any assignment effects on v, we can do a little better than this: rather than ordering step i after all previous increase effects on v and all previous decrease effects on v, we order it after all previous decrease effects, and *some* increase effects — enough to satisfy the precondition[1].

In POPF2 the state annotations in POPF are extended to support this. For a variable v where larger values are preferable, the state now also contains $V^{inc}(v)$: a queue of step–effect pairs, each $\langle j, c\rangle$. These correspond to beneficial effects on v: that step j has a (calculated) increase effect $v \mathrel{+}= c$. Preconditions and effects on v interact with $V^{inc}(v)$, and the existing annotations $V^{\mathit{eff}}(v)$ and $VP(v)$, as follows:

- For a new step i with a decrease effect on v , calculated as $v \mathrel{-}= c$, the effect is handled as before: i is ordered after each of $VP(v)$ and $V^{\mathit{eff}}(v)$, $V^{\mathit{eff}}(v) = i$, and the recorded value of v is decreased by c.
- For a new step i with an increase effect on v, calculated as $v \mathrel{+}= c$ (based on the values of the variable in the state), a pair $\langle i, c\rangle$ is added to $V^{inc}(v)$, and the recorded value of v is increased by c.
- For a new step i with precondition $v\{\geq, >\}k$, the ordering constraints are determined according to Algorithm 1, ordering the step after all decrease effects, and some of the increase effects (avoiding ordering i after those later in $V^{inc}(v)$) .

For this approach to be reasonable, the effects in $V^{inc}(v)$ must occur in chronological order. Otherwise, woefully inefficient ordering constraints could be introduced, using far later actions to satisfy the precondition than was necessary. As such, we only apply this special-case reasoning to the case where once a given plan step i has been added to the plan, and its minimum timestamp $t(i)$ found, step i will occur at $t(i)$ in all states subsequently reached by extending

[1]It could, in theory, be possible to satisfy a precondition on v by ordering step i before some of the existing decrease effects, but this would contradict the ethos of POPF: the partial-order is only ever expanded in a forwards direction, ordering new steps after existing ones, to avoid the issues of conflict resolution.

Algorithm 1: Ordering after beneficial increase effects on an as-needed basis

```
Data: a step index $i$; a numeric precondition $v \geq k$; the
      recorded value value of $v$ in $S$, $S[v]$; annotations
      $V^{\mathit{eff}}(v)$, $VP(v)$ and $V^{inc}(v)$
1  $C \leftarrow \{t(i) - t(V^{\mathit{eff}}(v)) \geq \epsilon\}$;
2  $\mathit{residual} \leftarrow S[v]$;
3  $\mathit{remaining} \leftarrow V^{inc}(v)$;
4  while $\mathit{residual} \geq k \wedge \mathit{remaining} \neq \emptyset$ do
5      $\langle j, c\rangle \leftarrow$ the back element of $\mathit{remaining}$;
6      $\mathit{residual} \leftarrow \mathit{residual} - c$;
7      if $\mathit{residual} \geq k$ then
8          remove back element of $\mathit{remaining}$;
9  for each $\langle j, c\rangle \in \mathit{remaining}$ do
10     $C \leftarrow C \cup \{t(i) - t(j) \geq \epsilon\}$;
11 $VP(v) \leftarrow VP(v) \cup \{i\}$;
12 return additional temporal constraints C, and the
   updated annotations
```

this plan. Then, it suffices to order the elements in $V^{inc}(v)$, each $\langle j, c\rangle$, according to $t(j)$, in ascending order from smallest $t(j)$ to largest $t(j)$. The necessary guarantee about $t(i)$ being fixed once the step is added to the plan can only be made if the domain does not require the starts and ends of actions to be coordinated, i.e. if the domain does not contain required concurrency (Cushing *et al.* 2007). For our purposes, we detect that a domain has no required concurrency by observing that all its actions are compression-safe.

4 Special Cases of Compression-Safe Action Detection

POPF inherited the basic notation of compression-safety introduced in (Coles *et al.* 2009b). As noted earlier in this paper, a durative action is 'basically' compression safe if:

- Its end effects only add propositions, i.e. it has no end numeric effects or end propositional delete effects;
- Its `at end` preconditions are a subset of its `over all` conditions, and hence ending the action does not require facts that are not already true through virtue of it executing.

This analysis is somewhat basic: scrutiny of domains will reveal actions that are not considered to be compression-safe according to this definition, but are compression-safe *with respect to the current problem.* Two such cases that we determine analytically in POPF2 are detailed below.

4.1 Compression Safety of Some End Numeric Effects

The intuition behind the general-case definition of compression safety used in POPFis to isolate actions where the end effects are only ever a good idea. As POPF does not support negative preconditions, adding a proposition is only ever beneficial: it does not preclude any actions from taking place. For numeric effects, though, it is not always clear whether a numeric effect is beneficial, so in the general case, an action cannot be compression safe if it has an end numeric effect.

In Section 3.1 we discussed the case where order-independent effects on metric-tracking variables need not be explicitly ordered. We can exploit this to relax the definition of compression safety. Simply, if an action has an end numeric effect v+=c, $c \in \Re$ on a metric-tracking variable upon which all effects are order independent then we can move that effect can be moved to the start of the action. The effect was inevitably going to occur, once the action had started; and the order in which it occurs (with respect to other effects on v) is irrelevant. This is a prime example of where actions can violate the basic definition of compression safety, but are compression safe in the current problem due to the nature of the variables upon which their end numeric effects act.

In Section 3.2, we discussed the case where certain numeric effects can be identified as being beneficial; specifically, those increasing (decreasing) a variable v, where larger (smaller) values of v are definitely preferable. We can also use this here to relax the constraints that determine whether an action is compression safe, by allowing compression-safe actions to have end numeric effects which are definitely beneficial. There is an additional consideration we must make, however: there is a risk that by deeming an action to be compression safe, and hence adding its end to the plan as soon as its start is added, we preclude concurrent activity that was previously possible. For instance, the action A, with start effect v-=c and end effect v+=c, can be applied concurrently alongside itself in a plan ordered:

$$[A_{\vdash}, A_{\vdash}, A_{\dashv}, A_{\dashv}]$$

The total order arises due to each having an effect on v, leading to each updating $V^{\mathit{eff}}(v)$. The action A is typical of the actions involving catalysts in the Pathways domain (Gerevini *et al.* 2009), where the amount of available catalyst is decreased at the start of the action, but then increased at the end. Allowing A to occur in parallel to itself allows the resulting compounds to be obtained sooner, subject to sufficient catalyst being available.

If larger values of v are preferable, A is in theory compression-safe. Exploiting this as in the propositional case, we would then add $A_{\dashv}$ to the plan immediately, as step $i + 1$, whenever $A_{\vdash}$ is added as step i. In POPF, this would result in $V^{\mathit{eff}}(v) = (i + 1)$, forcing the second copy of A to start after $A_{\dashv}$ rather than being able to start after $A_{\vdash}$, as in the total-order fragment above.

To address this potential issue, we only mark actions with end numeric effects as being compression safe if the domain does not contain required concurrency (Cushing *et al.* 2007), and we then exploit this in combination with the 'as-needed' ordering constraint approach described in Section 3.2. Again, for our purposes, we detect that a domain has no required concurrency if all its actions are compression-safe, though there is, ostensibly, a circular argument here: an end numeric effect is compression-safe if all actions are compression-safe. To address this, we first loop over the actions, marking them as compression safe (according to the basic definition of POPF) or hypothetically compression safe (subject to all other actions being compression safe). Then, if there is one non-compression-safe action, the hypothetically compression-safe actions are marked as being non-compression-safe. Otherwise, they are considered to be compression-safe, as the increase-effect-queue described in Section 3.2 is sufficient to preserve opportunities for concurrency.

4.2 The Over-All, End-Delete-Effect Idiom

In general, in the absence of negative preconditions (as in POPF) delete effects are never beneficial: deleting a fact can only preclude actions from being applied. As such, in the basic definition of compression safety, end delete effects are prohibited. But, consider two durative actions A and B, each with condition `(over all (p))` and effect `(at end (not (p)))`, and an instantaneous action C with precondition `(over all (p))` and effect `(at end (not (p)))`. It is clear that:

- $A_\dashv$, $B_\dashv$ and C are mutually exclusive (all deleting the fact `p`), so cannot occur at the same time;
- Neither $A_\dashv$ nor C can fall within the execution of B, or as it would violate the `over all` condition. (Similarly, neither $B_\dashv$ nor C can fall within the execution of A.)

In common between all three of these actions is the notion that `p` is required for some amount of time (instantaneously, in the case of C) but then inevitably destroyed. If this idiom covers all the uses of `p` in the problem, then we can allow end delete effects on `p` to be considered to be compression safe. In effect, deleting `p` can be moved to the start of the actions such as A or B. There is no point maintaining `p` throughout actions such as A, as no action referring to `p` can be applied. Such an action would either:

- follow the pattern of C, immediately deleting `p` and hence violating the active `over all` condition of A;
- follow the pattern of B, thereby leading to a guaranteed future conflict between either the `over all` condition of A and the effect of $B_\dashv$, or the `over all` condition of B and the effect of $A_\dashv$.

5 Anytime Search

In its original form, POPF terminated after the first plan found. A derivative of POPF, Stochastic-POPF (Coles *et al.* 2011), has recently extended this to both search in domains where action durations are uncertain, but also to seek to minimise plan cost. The techniques of Stochastic-POPFcan be applied in here, in a deterministic setting, unaltered. For full details, we refer the reader to the paper on Stochastic-POPF, but we will sketch the approach here. The search algorithm can be summarised as follows:

- Search begins with an upper-bound on acceptable plan quality of ∞
- Attempts to find a plan using enforced hill-climbing (EHC). If a plan is found, it is stored, and the upper-bound on acceptable plan quality is then set to the quality of this plan;
- Irrespective of whether a plan is found by EHC, then search using WA* (where $W = 5$, $g(n)$ is the plan length to node n and $h(n)$ is its heuristic value). If the variables used to record plan cost are monotonically worsening, then nodes in the search space are discarded if the cost of the plan to reach that state equals or exceeds the acceptable upper-bound on plan quality. This is similar to MIPS-XXL (Edelkamp, Jabbar, & Nazih 2006), where each time a new best solution plan is found, an additional goal is added to ensure the next plan is of better quality; but the planner does not start search from the initial state each time a new best plan is found.

Stochastic-POPF also contains an updated temporal relaxed planning graph (RPG) heuristic, where admissible estimates on the cost of reaching each fact are maintained. The approach taken is based on the costed RPG of Sapa (Do & Kambhampati 2003), extended to handle the case where certain costs are only relevant to achieving facts that only appear as goals. This further supports state pruning: if the cost of reaching the goals from a given state would definitely lead to a plan being found that is worse than the incumbent, the state can be pruned. extended to perform pruning (rather than preferring

Acknowledgements

The authors would like to thank Allan Clark and Stephen Gilmore for motivating the development of Stochastic-POPF, and David Smith for helpful discussions regarding a problem-specific notion of compression-safe actions. This work was supported by SICSA, and EPSRC fellowship EP/H029001/1.

References

Cesta, A., and Oddi, A. 1996. Gaining Efficiency and Flexibility in the Simple Temporal Problem. In *Proceedings of the Third International Workshop on Temporal Representation and Reasoning (TIME-96)*.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2009a. Temporal planning in domains with linear processes. In *IJCAI '09*.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2009b. Extending the Use of Inference in Temporal Planning as Forwards Search. In *Proc. ICAPS*.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *ICAPS*.

Coles, A. J.; Coles, A. I.; Clark, A.; and Gilmore, S. T. 2011. Cost-sensitive concurrent planning under duration uncertainty for service level agreements. In *ICAPS*.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. 2007. When is temporal planning *really* temporal planning? In *Proc. of Int. Joint Conf. on AI (IJCAI)*, 1852–1859.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49:61–95.

Do, M. B., and Kambhampati, S. 2003. Sapa: Multi-objective Heuristic Metric Temporal Planner. *JAIR* 20:155–194.

Edelkamp, S.; Jabbar, S.; and Nazih, M. 2006. Large-Scale Optimal PDDL3 Planning with MIPS-XXL. In *IPC5 booklet, ICAPS*.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension of PDDL for Expressing Temporal Planning Domains. *JAIR* 20:61–124.

Gerevini, A. E.; Long, D.; Haslum, P.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic Planning in the Fifth International Planning Competition: PDDL3 and Experimental Evaluation of the Planners. *AIJ*.

Long, D., and Fox, M. 2003. The 3rd International Planning Competition: Results and Analysis. *J. of Art. Int. Res.* 20:1–59.