A Normal Form for Classical Planning Tasks

Florian Pommerening and **Malte Helmert**
University of Basel
Basel, Switzerland
{florian.pommerening,malte.helmert}@unibas.ch

Abstract

We describe *transition normal form* (TNF) for classical planning tasks, where there is a unique goal state and variables occur in an operator precondition iff they appear in the effect. Tasks can be efficiently converted to TNF, all common planning heuristics are invariant under the transformation, and tasks in normal form are easier to study theoretically.

Introduction

The design and study of classical planning techniques such as heuristics or search algorithms is often considerably simpler when focusing on planning tasks with a restricted structure. Two assumptions that are frequently useful are that operators only change variables for which they have a defined precondition and that there exists a unique goal state (e.g., Bäckström 2014; Bonet 2013; Eyerich and Helmert 2013; Sievers, Ortlieb, and Helmert 2012; van den Briel et al. 2007; Zhang, Wang, and Xie 2014).

There exists a well-known transformation to achieve such a "normal form", but it can increase the size of the task exponentially. We introduce an alternative transformation that only increases the task size by a small constant factor. We also analyze the influence the transformation has on many existing planning techniques, showing that the transformation preserves many important theoretical properties while often allowing considerably simpler presentations. This makes tasks in normal form an attractive tool for planning researchers: they are simpler to work with than unrestricted tasks, yet retain their full expressiveness.

Transition Normal Form

We write *planning tasks* as tuples $\Pi = \langle \mathcal{V}, \mathcal{O}, s_{\mathrm{I}}, s_\star \rangle$.

Partial states s over the finite-domain *variables* $\mathcal{V}$ map a subset $\mathit{vars}(s) \subseteq \mathcal{V}$ to values in their domain; *states* are partial states with $\mathit{vars}(s) = \mathcal{V}$. Partial states s and s' are *consistent* if $s(V) = s'(V)$ for all $V \in \mathit{vars}(s) \cap \mathit{vars}(s')$. We call variable/value pairs $V \mapsto v$ *facts* and interchangeably view (partial) states as functions or fact sets. The *initial state* s_{I} is a state and the *goal description* $s_\star$ is a partial state.

Operators o in the finite set $\mathcal{O}$ are associated with a cost $\mathit{cost}(o)$ and partial states $\mathit{pre}(o)$ (preconditions) and $\mathit{eff}(o)$ (effects). An operator o is applicable in state s if $\mathit{pre}(o)$ is consistent with s. Applying o in s results in a state that is consistent with $\mathit{eff}(o)$ and agrees with s on all variables $V \notin \mathit{vars}(\mathit{eff}(o))$. A *plan* for Π is a sequence of operators that are iteratively applicable starting in s_{I} and result in a state consistent with $s_\star$.

Definition 1. *A planning task* Π *is in* transition normal form *(TNF) if* $\mathit{vars}(\mathit{pre}(o)) = \mathit{vars}(\mathit{eff}(o))$ *for all operators* o *of* Π *and the goal of* Π *is a fully defined state.*

The main characteristic of planning tasks in TNF is that each operator defines a unique value transition on all variables it refers to. We additionally require a unique goal state because this is also often convenient and can be easily accomplished together with the unique transition property.

For TNF to be generally useful, we need a way to transform general planning tasks to ones in TNF that are *equivalent* in a formal sense. Ensuring that precondition variables also occur as effect variables is trivial: whenever this is not the case for a given precondition fact f, we can add it as an effect without affecting the semantics of the planning task.

Ensuring the opposite, that effect variables V of an operator o are also precondition variables, is trickier. A "folklore" transformation "multiplies out" variables, creating copies of o with each possible fact for V as a precondition. However, this transformation increases task size exponentially in the worst case: if an operator has m variables occurring in the effect but not the precondition and each of these can take on n different values, the conversion produces n^m copies.

We now present an alternative conversion that can only lead to a modest increase in task size.

Definition 2. *Let* Π *be a planning task. Its* transition normalization *TNF*(Π) *is the task obtained from* Π *as follows:*

- *Add a fresh value* $\mathbf{u}$ *to the domain of each variable.*
- *For all facts* $V \mapsto v$ *with* $v \neq \mathbf{u}$*, add a* forgetting operator *with precondition* $\{V \mapsto v\}$*, effect* $\{V \mapsto \mathbf{u}\}$ *and cost* 0*.*
- *For all variables* V *and operators* o*:*
  - *If* V *occurs in the precondition but not the effect of* o*, add* $V \mapsto \mathit{pre}(o)[V]$ *to the effect of* o*.*
  - *If* V *occurs in the effect but not the precondition of* o*, add* $V \mapsto \mathbf{u}$ *to the precondition of* o*.*
- *For all variables* V *not occurring in the goal, add* $V \mapsto \mathbf{u}$ *to the goal.*

The intuition behind the transformation is that we may "forget" the value of a variable V (= set it to $\mathbf{u}$) at any time at no cost. This allows us to require a defined value for V (namely, $\mathbf{u}$) in *TNF*(Π) where Π mandates no defined value.

It is easy to see that *TNF*(Π) is in TNF and that its representation size in a reasonable encoding is at worst twice that of Π. We now show that *TNF*(Π) and Π are equivalent.

Theorem 1. *Every plan π for Π can be converted into a plan π' for TNF(Π) with the same cost in time $O(k|\pi| + |s_\star|)$, where k is the maximal number of effects of an operator, and conversely in time $O(|\pi'|)$.*

Proof: To convert π into π', insert the necessary forgetting operators in front of each operator o: if o is executed in state s, insert operators to forget $V \mapsto s[V]$ for all variables V on which o has an effect but no precondition. Also append such operators to the end of the plan for each goal of the form $V \mapsto \mathbf{u}$ in *TNF*(Π). To convert a plan for *TNF*(Π) into a plan for Π, simply drop all forgetting operators. □

As hinted in the introduction, TNF is useful because many planning approaches can be described more simply while losing none of their power when restricting attention to TNF. In the remainder of this paper, we demonstrate this observation for a large number of example approaches. Many of these examples relate to *distance heuristics*. A concise description of many of the heuristics we consider is given by Helmert and Domshlak (2009).

Delete Relaxation and Critical Paths

Many planning heuristics can be understood as computing distances in a simplified version of the given planning task Π. Important examples include the *optimal delete-relaxation heuristic* h^+ (Hoffmann 2005), which ignores delete effects, and the *critical path heuristics* h^m ($m \geq 1$), which estimate the cost of reaching a state by recursively estimating the cost of subgoals with up to m facts (Haslum and Geffner 2000). A well-known special case of the latter is the maximum heuristic $h^{\max} = h^1$. Transition normalization does not negatively affect these heuristics.

Theorem 2. *Let Π be a planning task. For all states s of Π, the values of $h^+(s)$ and of $h^m(s)$ are the same in Π and TNF(Π).*

Proof sketch: Delete relaxation can be seen as a change in semantics on planning tasks that does not affect the syntax. A version of Theorem 1 for this modified semantics can be proved analogously. This is sufficient to show that $h^+(s)$ is the same in Π and *TNF*(Π) since h^+ is based on optimal goal distances.

The h^m heuristic assigns a value to each set of facts F that intuitively measures the cost to reach a state containing the facts in F considering only the cost of the hardest subset whose size does not exceed m. The facts $V \mapsto \mathbf{u}$ can always be reached free of cost and do not allow reaching any other state more cheaply than in the original task. The h^m cost of a partial state F in *TNF*(Π) is thus the h^m cost of F without all facts $V \mapsto \mathbf{u}$ in Π. □

Landmarks

A (disjunctive action) *landmark* is a set of operators from which at least one has to be used in any plan. Several planning systems, such as *LAMA* (Richter and Westphal 2010), and heuristics, such as *LM-cut* (Helmert and Domshlak 2009) and *cost-partitioned landmarks* (Karpas and Domshlak 2009), are based on this concept. Transition normalization preserves the landmarks of a planning task.

Theorem 3. *If L is a landmark of planning task Π, then L is a landmark of TNF(Π). If L' is a landmark of TNF(Π) that does not include any forgetting operators, then L' is a landmark of Π.*

Proof: This is a direct consequence of the relationship between the plans of Π and *TNF*(Π) shown in Theorem 1. □

The *LM-cut method* is a landmark-based heuristic that computes landmarks based on $h^{\max}$ values of planning tasks. Together with the result for $h^{\max}$ shown in Theorem 2, it is not difficult to show that transition normalization preserves LM-cut heuristic values.

Domain Transition Graphs and Abstractions

Many planning techniques make use of *domain transition graphs* (DTGs), which model the effect of operators on individual variables of the planning task (Jonsson and Bäckström 1998). The DTG of variable V is a digraph with nodes $dom(V)$. It has an arc from v to v' labeled by operator o if $pre(o)[V] = v$ and $eff(o)[V] = v'$ or if $V \notin vars(pre(o))$ and $eff(o)[V] = v'$.

For tasks in TNF the latter case cannot occur, and hence every operator o mentioning a variable V induces exactly one arc in the DTG of V, from $pre(o)[V]$ to $eff(o)[V]$. Hence, a definition of DTGs for tasks in TNF is simpler and more directly communicates the underlying intent.

Conversion to TNF can also make domain transition graphs considerably *smaller*: if variable V has n values and there are n operators affecting it, each without a precondition on V, then the DTG of V in Π has n^2 transitions, while the DTG of V in *TNF*(Π) has only $2n$ transitions: n from the original values of V to $\mathbf{u}$ and one for each operator. Intuitively, *TNF*(Π) represents the concept "operators with no defined precondition value can be applied everywhere" only *once*, rather than once for every such operator.

One application of DTGs where this can lead to practical performance benefits is within *merge-and-shrink* (M&S) abstractions (e.g., Helmert et al. 2014). These are based on the manipulation of explicitly represented abstract transition systems, which are initialized to the atomic abstractions (= DTGs) of the planning task. Avoiding a potential quadratic increase in DTG size can be crucial for limiting the size of M&S abstractions, as it is well-known that the number of *abstract transitions*, not the number of abstract states, is the main limiting factor of the approach.

A similar example is the DTG-based landmark test used by LAMA (Richter and Westphal 2010). Using transition-normalized tasks finds the same landmarks while improving worst-case complexity for a landmark test from $O(n^2)$ to $O(n)$ due to the more favorable bound on DTG size.

Incremental Ranking and Zobrist Hashing

Also within the area of abstraction heuristics, Sievers, Ortlieb, and Helmert (2012) show how a *pattern database* (PDB) for a planning task can be efficiently computed using a perfect hash function that assigns a unique *rank* to each abstract state. The efficient implementation crucially relies on avoiding unnecessary *unranking* and *ranking* operations. Given a state s represented by its perfect hash value (rank) and an operator o, we want to efficiently determine the rank of the successor s' of s reached via o. With operators in TNF, this can be done by precomputing a constant *offset* for o and then computing the rank of s' as the rank of s plus the offset, without the need for unranking and ranking operations to incorporate the effects of the operator. This advantage of TNF is compelling enough that Sievers et al. use the exponential transformation to compile operators into normal form.

A closely related application of TNF is within *Zobrist hashing*. Botea et al. (2005) report that planners based on state-space search can spend up to 35% of their runtime on calculating hash values for states in some domains. They use a *Zobrist hash function* (Zobrist 1970; 1990) to speed up the computation. For operators in TNF (and not in the general case), the Zobrist hash of a state s' reached from state s via operator o can be computed incrementally as $\mathit{hash}(s) \oplus c(o)$, where $c(o)$ depends only on operator o and can be easily precomputed. (Here, $\oplus$ denotes the binary XOR operation.)

Regression

Most current search-based planning algorithms solve planning tasks by searching the state space in a forward direction (*progression*), starting from the initial state. However, a well-established and equally plausible approach is to perform a *backward* search (*regression*) from the goal towards the initial state (e.g., Bonet and Geffner 2001; Alcázar et al. 2013).

In the general case, the search nodes of a regression search are associated with *partial* states s', which correspond to all states that agree with s' on $\mathit{vars}(s')$. Regression is often seen as more complicated than progression, and several recent papers discuss the theoretical and practical complexities involved in regression for SAS$^+$-like planning tasks at length (Alcázar et al. 2013; Eyerich and Helmert 2013). With unrestricted planning tasks, an operator o is *regressable* in a partial state s' iff

- for all $V \in \mathit{vars}(\mathit{eff}(o))$: $V \notin \mathit{vars}(s')$ or $\mathit{eff}(o)[V] = s'[V]$, and
- for all $V \in \mathit{vars}(\mathit{pre}(o)) \setminus \mathit{vars}(\mathit{eff}(o))$: $V \notin \mathit{vars}(s')$ or $\mathit{pre}(o)[V] = s'[V]$.

The regression of o in s' is then defined as the partial state s with $\mathit{vars}(s) = (\mathit{vars}(s') \setminus \mathit{vars}(\mathit{eff}(o))) \cup \mathit{vars}(\mathit{pre}(o))$ and

$$s[V] = \begin{cases} \mathit{pre}(o)[V] & \text{if } V \in \mathit{vars}(\mathit{pre}(o)) \\ s'[V] & \text{otherwise} \end{cases}$$

for all $V \in \mathit{vars}(s)$.

Compared to progression, these definitions are rather involved. However, for tasks in TNF, regression can be defined in a way that is perfectly analogous to progression: regression for a task in TNF is exactly the same as progression with the roles of s_I and $s_\star$ swapped and with $\mathit{pre}(o)$ and $\mathit{eff}(o)$ swapped in each operator o. Hence with TNF, backward search is conceptually as simple as forward search.

In practical planning algorithms based on regression, this observation is not necessarily all that useful: the fact that general regression must reason about partial states is a boon as much as a burden because it enables dominance pruning techniques that are critical for strong performance. Therefore, conversion to TNF might incur a significant performance penalty despite the simpler theory.

However, the simple relationship between progression and regression in TNF is very useful for *reasoning* about regression, which is useful in a large number of planning topics such as invariant synthesis (e.g., Rintanen 2008), relevance analysis (e.g., Haslum 2007) and reachability analysis (e.g., Hoffmann and Nebel 2001). There are also applications of regression where dominance pruning is not critical for performance, such as the seeding of perimeter PDBs (Eyerich and Helmert 2013). Here, TNF can considerably simplify efficient implementations.

State-Equation Heuristic

The state-equation heuristic (van den Briel et al. 2007; Bonet 2013) observes that some operators *produce* a given fact (change it from false to true) while others *consume* it (change it from true to false). There must be a certain balance between how often a given fact is produced and consumed, and this balance can be encoded with numerical constraints (linear programs) from which heuristic values are extracted by generic constraint optimization techniques.

What sounds like a clean and simple concept involves substantial complications in practice. Without knowing the state in which an operator o is applied, it is in general not possible to tell if it produces (or consumes) a given fact. An operator with precondition $V \mapsto v$ and effect $V \mapsto v'$ (with $v \neq v'$) will always consume $V \mapsto v$ and always produce $V \mapsto v'$. However, if an operator only has the effect on V but not the precondition, it *may* produce $V \mapsto v'$, but only if $s[V] \neq v'$ in the state s in which the operator is applied. Similarly, it *may* consume the current value of V, but we cannot know what this value is from the operator description alone. Similar vagaries arise from variables whose value is unspecified in the goal.

As a consequence of these complications, the actual constraints used by the state-equation heuristic are not very obvious from the intuitive description of the idea:

$$\text{Minimize } \sum_{o \in \mathcal{O}} \mathsf{Y}_o \, \mathit{cost}(o) \text{ subject to}$$

$$\mathsf{Y}_o \geq 0 \quad \text{for all } o \in \mathcal{O}$$

$$[f \in s_\mathrm{I}] + \sum_{o \in AP_f} \mathsf{Y}_o + \sum_{o \in SP_f} \mathsf{Y}_o \geq [f \in s_\star] + \sum_{o \in AC_f} \mathsf{Y}_o$$

$$\text{for all facts } f$$

where AP_f, SP_f and AC_f are the sets (whose nontrivial definitions we omit for brevity) of operators that *always produce*, *sometimes produce* and *always consume* fact f.

For tasks in TNF these complications disappear. Every operator defines a well-defined value transition on the variables it mentions, so we can simply count the number of *consumers* of f (operators with precondition f) and *producers* of f (operators with effect f) directly. We also have no uncertainty regarding the goal state, and consequently we can always use equations instead of inequalities:

$$\text{Minimize} \sum_{o \in \mathcal{O}} \mathsf{Y}_o \, \mathit{cost}(o) \text{ subject to}$$

$$\mathsf{Y}_o \geq 0 \quad \text{for all } o \in \mathcal{O}$$

$$[f \in s_{\mathrm{I}}] + \sum_{o \in \mathcal{O}: f \in \mathit{eff}(o)} \mathsf{Y}_o = [f \in s_\star] + \sum_{o \in \mathcal{O}: f \in \mathit{pre}(o)} \mathsf{Y}_o \quad \text{for all facts } f$$

Similar to the heuristics considered previously, it turns out that the state-equation heuristic value for task Π is identical to the heuristic estimate in $\mathit{TNF}(\Pi)$ for which we can use the simpler and cleaner constraint system. (We omit the proof of this statement because we do not believe it is necessary for the main point we want to make: that working with planning tasks in transition normal form can make life substantially easier for the planning researcher and thus be a valuable aid in the design and analysis of planning techniques.)

Potential Heuristics

Pommerening et al. (2015) recently introduced *potential heuristics*, which are parameterized by an assignment of numerical values to facts of the planning task. The value assigned to a fact is called its *potential*, and the heuristic value of a state is the sum of potentials for all facts it contains. In a preprocessing step, the potentials are determined by a linear program (LP) that encodes the requirement that the resulting heuristic is consistent and admissible. Every feasible solution of this LP yields an admissible potential heuristic, and we can choose any objective function to bias the choice of heuristic towards ones that are expected to be informative.

The paper by Pommerening et al. only defines the LP for tasks in TNF and refers to a technical report (Pommerening et al. 2014) for the details of the construction in the general case because the transition-normalized case is much easier to understand and conveys the intuition of the idea much better. Here we show that the LP for a general task (adapted from the technical report) is equivalent to the LP given in the paper for the task in TNF. This means that the simpler definition for tasks in TNF is sufficient and can be used for general tasks by means of transition normalization.

Definition 3. *(Pommerening et al. 2014) Consider a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_{\mathrm{I}}, s_\star \rangle$. The potential function for facts of Π has to satisfy the* general fact potential constraints *over the unrestricted LP variables P_f for every fact f and Max_V for every variable V:*

$$\mathsf{P}_{V \mapsto v} \leq \mathsf{Max}_V \text{ for all facts } V \mapsto v$$

$$\sum_{V \in \mathcal{V}} \mathit{maxpot}(V, s_\star) \leq 0$$

$$\sum_{(V \mapsto v) \in \mathit{eff}(o)} (\mathit{maxpot}(V, \mathit{pre}(o)) - \mathsf{P}_{V \mapsto v}) \leq \mathit{cost}(o)$$

$$\text{for all } o \in \mathcal{O}$$

where maxpot is a function mapping a variable V and a partial state p to the LP variable that represents the maximal potential for a fact of V that is consistent with p:

$$\mathit{maxpot}(V, p) = \begin{cases} \mathsf{P}_{V \mapsto v} & \text{if } (V \mapsto v) \in p \\ \mathsf{Max}_V & \text{otherwise} \end{cases}$$

For tasks in TNF, the definition is much simpler:

Definition 4. *(Pommerening et al. 2015) Consider a planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, s_{\mathrm{I}}, s_\star \rangle$ in TNF. The* TNF potential function *for facts of Π has to satisfy the* TNF fact potential constraints *over the unrestricted LP variables P_f for every fact f:*

$$\sum_{f \in s_\star} \mathsf{P}_f \leq 0$$

$$\sum_{f \in \mathit{pre}(o)} \mathsf{P}_f - \sum_{f \in \mathit{eff}(o)} \mathsf{P}_f \leq \mathit{cost}(o) \quad \text{for all } o \in \mathcal{O}$$

Theorem 4. *Let Π be a planning task. The general fact potential constraints of Π are equivalent to the TNF fact potential constraints of $\mathit{TNF}(\Pi)$.*

Proof: Transition normalization introduces a new value $\mathbf{u}$ for every variable V and a "forget operator" for every fact $V \mapsto v$. The new operator leads to the constraint $\mathsf{P}_{V \mapsto v} - \mathsf{P}_{V \mapsto \mathbf{u}} \leq \mathit{cost}(o) = 0$, or equivalently $\mathsf{P}_{V \mapsto v} \leq \mathsf{P}_{V \mapsto \mathbf{u}}$. Setting $\mathsf{P}_{V \mapsto \mathbf{u}} = \mathsf{Max}_V$ shows the theorem. □

Conclusion

We introduced *transition normal form*, a normal form for planning tasks that has been informally considered before (usually as a simplifying assumption, rather than an actual normal form to convert to) without having received any detailed attention. We showed that general planning tasks can be converted to transition normal form easily with a very mild increase in representation size, unlike the worst-case exponential transformations described in the literature. The normal form is unobtrusive in the sense that the process of normalization does not appear to lose information for any of the commonly considered techniques in heuristic planning.

The *raison d'être* of transition normal form is that it greatly simplifies core concepts of a surprisingly large number of planning approaches, especially those related to domain transition graphs, regression, and any concepts that emphasize the flow of values of finite-domain state variables (as in the recently suggested state equation and potential heuristics). Several techniques that look technically involved in the general case look crisp and clear for planning tasks in transition normal form.

Transition normal form will not revolutionize the world of classical planning, but we believe it can make many planning researchers' lives considerably easier at least some of the time, and it deserves to be widely known.

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNSF) as part of the project "Automated Reformulation and Pruning in Factored State Spaces" (ARAP).

References

Alcázar, V.; Borrajo, D.; Fernández, S.; and Fuentetaja, R. 2013. Revisiting regression in planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2254–2260.

Bäckström, C. 2014. Parameterising the complexity of planning by the number of paths in the domain-transition graphs. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 33–38. IOS Press.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1):5–33.

Bonet, B. 2013. An admissible heuristic for SAS$^+$ planning obtained from the state equation. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2268–2274.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research* 24:581–621.

Eyerich, P., and Helmert, M. 2013. Stronger abstraction heuristics through perimeter search. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 303–307. AAAI Press.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, 140–149. AAAI Press.

Haslum, P. 2007. Reducing accidental complexity in planning problems. In Veloso, M. M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 1898–1903.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM* 61(3):16:1–63.

Hoffmann, J., and Nebel, B. 2001. RIFO revisited: Detecting relaxed irrelevance. In Cesta, A., and Borrajo, D., eds., *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 127–135. AAAI Press.

Hoffmann, J. 2005. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research* 24:685–758.

Jonsson, P., and Bäckström, C. 1998. State-variable planning under structural restrictions: Algorithms and complexity. *Artificial Intelligence* 100(1–2):125–176.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1728–1733.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2014. From non-negative to general operator cost partitioning: Proof details. Technical Report CS-2014-005, University of Basel, Department of Mathematics and Computer Science.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From non-negative to general operator cost partitioning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3335–3341. AAAI Press.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Rintanen, J. 2008. Regression for classical and nondeterministic planning. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, 568–572.

Sievers, S.; Ortlieb, M.; and Helmert, M. 2012. Efficient implementation of pattern database heuristics for classical planning. In Borrajo, D.; Felner, A.; Korf, R.; Likhachev, M.; Linares López, C.; Ruml, W.; and Sturtevant, N., eds., *Proceedings of the Fifth Annual Symposium on Combinatorial Search (SoCS 2012)*, 105–111. AAAI Press.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-based heuristic for optimal planning. In Bessiere, C., ed., *Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *Lecture Notes in Computer Science*, 651–665. Springer-Verlag.

Zhang, L.; Wang, C.-J.; and Xie, J.-Y. 2014. Cost optimal planning with LP-based multi-valued landmark heuristic. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2014)*, 509–516. IFAAMAS.

Zobrist, A. L. 1970. A new hashing method with application for game playing. Technical Report 88, Computer Science Department, The University of Wisconsin, Madison, WI, USA.

Zobrist, A. L. 1990. A new hashing method with application for game playing. *ICCA Journal* 13(2):69–73.