Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Fact-Alternating Mutex Groups for Classical Planning (Extended Abstract)*

Daniel Fišer and **Antonín Komenda**
Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech Republic
danfis@danfis.cz, antonin.komenda@fel.cvut.cz

Abstract

Mutex groups are defined in the context of STRIPS planning as sets of facts out of which, maximally, one can be true in any state reachable from the initial state. This work provides a complexity analysis showing that inference of mutex groups is as hard as planning itself (PSPACE-Complete) and it also shows a tight relationship between mutex groups and graph cliques. Furthermore, we propose a new type of mutex group called a fact-alternating mutex group (fam-group) of which inference is NP-Complete. We introduce an algorithm for the inference of fam-groups based on integer linear programming that is complete with respect to the maximal fam-groups and we demonstrate that fam-groups can be beneficial in the translation of planning tasks into finite domain representation, for the detection of dead-end state and for the pruning of spurious operators. The experimental evaluation of the pruning algorithm shows a substantial increase in a number of solved tasks in domains from the optimal deterministic track of the last two planning competitions (IPC 2011 and 2014).

1 Introduction

State invariants in domain-independent planning are certain intrinsic properties of a particular planning task that hold in all states reachable from the initial state. State invariants tell something about the internal structure of the problem, which can be further utilized in the process of solving the task. A rather scarce literature on the inference of state invariants include methods using different "guess, check, and repair" approaches [Gerevini and Schubert, 1998; 2000; Helmert, 2009; Rintanen, 2000], inference based on sampling of a state space [Mukherji and Schubert, 2005; 2006], a reachability analysis generalized to h^m heuristics [Haslum and Geffner, 2000; Haslum, 2009; Alcázar and Torralba, 2015], or an iterative weakening of more general invariants formed as a disjunction of possibly negated facts [Rintanen, 2008].

In this work, we are particularly interested in the inference of state invariants called mutex groups consisting of facts that are pairwise mutually exclusive. Therefore, a mutex group states that any reachable state can contain at most one fact from the mutex group.

The most straightforward application of mutex groups is in the translation to finite domain representation (FDR, or SAS^+) [Bäckström and Nebel, 1995; Edelkamp and Helmert, 1999; Helmert, 2009]. Given a set of mutex groups, the FDR can be constructed by creating variables from those mutex groups that cover all facts. A special value "none of those" can be added to some variables, if needed, to cover a situation where none of the facts from the invariant is present in the state.

State invariants (including mutex invariants) are also critical for improving the performance of SAT planners [Kautz and Selman, 1992; Sideris and Dimopoulos, 2010].

Exploration of the state space in a symbolic search with Binary Decision Diagrams (BDDs) is not carried out through the expansion of single states but rather by construction of BDDs representing sets of states, which potentially provides an exponential saving in memory consumption. State invariants encoded as BDDs can be used for the pruning of unreachable states during search and also during the preprocessing of the planning task for pruning operators that generate dead-end states [Torralba and Alcázar, 2013]. The connection between state invariants and dead-end states was recently studied by Lipovetzky *et al.* [2016].

This work is aimed mainly at the analysis and inference of mutex groups in the context of STRIPS planning [Fikes and Nilsson, 1971]. We introduce a new type of mutex group called the fact-alternating mutex group and we discuss its relation to the general mutex group and to the mutexes inferred by the heuristic h^m [Haslum and Geffner, 2000]. We also discuss the properties of fact-alternating mutex groups, in particular their connection to dead-end states.

We provide a complexity analysis showing that the inference of the maximum sized mutex group is PSPACE-Complete whereas inferring the maximum sized fact-alternating mutex group is NP-Complete. The complexity analysis leads to a novel inference algorithm that is complete with respect to maximal fact-alternating mutex groups.

This paper is an extended abstract of the previously published work [Fišer and Komenda, 2018], so it contains only

*This paper is an extended abstract of an article in the Journal of Artificial Intelligence Research [Fišer and Komenda, 2018].

the main results without any proofs. Detailed explanations and proofs can be found in the original paper.

2 Mutex Groups

Definition 1. A STRIPS **planning task** Π is specified by a tuple $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal} \rangle$, where $\mathcal{F} = \{f_1, ..., f_n\}$ is a set of facts, and $\mathcal{O} = \{o_1, ..., o_m\}$ is a set of grounded operators. A **state** $s \subseteq \mathcal{F}$ is a set of facts, $s_{init} \subseteq \mathcal{F}$ is an **initial state** and $s_{goal} \subseteq \mathcal{F}$ is a **goal** specification. An **operator** o is a triple $o = \langle \text{pre}(o), \text{add}(o), \text{del}(o) \rangle$, where $\text{pre}(o) \subseteq \mathcal{F}$ is a set of preconditions of the operator o, and $\text{add}(o) \subseteq \mathcal{F}$ and $\text{del}(o) \subseteq \mathcal{F}$ are sets of add and delete effects, respectively. All operators are well-formed, i.e., $\text{add}(o) \cap \text{del}(o) = \emptyset$ and $\text{pre}(o) \cap \text{add}(o) = \emptyset$. An operator o is **applicable** in a state s if $\text{pre}(o) \subseteq s$. The **resulting state** of applying an applicable operator o in a state s is the state $o[s] = (s \setminus \text{del}(o)) \cup \text{add}(o)$. A state s is a **goal state** iff $s_{goal} \subseteq s$.

A sequence of operators $\pi = \langle o_1, ..., o_n \rangle$ is applicable in a state s_0 if there are states $s_1, ..., s_n$ such that o_i is applicable in s_{i-1} and $s_i = o_i[s_{i-1}]$ for $1 \leq i \leq n$. The resulting state of this application is $\pi[s_0] = s_n$. A state s is called a **reachable state** if there exists an applicable operator sequence π such that $\pi[s_{init}] = s$. A set of all reachable states is denoted by $\mathcal{R}$. An operator o is called a **reachable operator** iff it is applicable in some reachable state. A state s is called a **dead-end state** iff $s_{goal} \not\subseteq s$ and there does not exist any applicable operator sequence π such that $s_{goal} \subseteq \pi[s]$.

Definition 2. A **mutex** $M \subseteq \mathcal{F}$ is a set of facts such that for every reachable state $s \in \mathcal{R}$ it holds that $M \not\subseteq s$.

Definition 3. A **mutex group** $M \subseteq \mathcal{F}$ is a set of facts such that for every reachable state $s \in \mathcal{R}$ it holds that $|M \cap s| \leq 1$. A mutex group that is not a subset of any other mutex group is called a **maximal mutex group**.

A mutex and a mutex group are both defined as invariants with respect to all states reachable from the initial state by a sequence of operators. A mutex invariant states that certain facts cannot be part of the same reachable state at the same time. So for example, a mutex $\{f_1, f_2, f_3\}$ states that there is no reachable state containing all three facts, but a state containing $\{f_1, f_2\}$, or $\{f_1, f_3\}$, or $\{f_2, f_3\}$ can still be reachable.

A mutex group is defined as a set of facts out of which, maximally, one can be true in any reachable state, i.e., the facts from a mutex group are pairwise mutex. It is also easy to see that every mutex containing exactly two facts (mutex pair) is also a mutex group, because if two facts cannot both be part of the same reachable state, then, at most, one of them can be a part of any reachable state, which is exactly the definition of a mutex group

Proposition 4. *Let $M \subseteq \mathcal{F}$ denote a set of facts such that $|M| \geq 2$ and let $\mathcal{D}^M$ denote a set of all pairs of all facts from M, i.e., $\mathcal{D}^M = \{p \mid p \subseteq M, |p| = 2\}$. M is a mutex group iff every $P \in \mathcal{D}^M$ is a mutex group.*

It is well known that determining existence of a plan of a STRIPS planning task is PSPACE-Complete [Bylander, 1994]. And as it turns out, finding the maximum sized mutex group is also PSPACE-Complete.

Theorem 5. *Let MAXIMUM-MUTEX-GROUP denote the following decision problem: Given a planning task Π and an integer k, does Π contain a mutex group of size at least k? MAXIMUM-MUTEX-GROUP is PSPACE-Complete.*

Since the facts from a mutex group are pairwise mutex, we can infer mutex groups in the following way. Given a set of mutex pairs, we construct a graph where each node corresponds to a fact and each edge connects two facts that form a mutex pair. In such a graph, any set of nodes that is a clique (a complete induced subgraph) corresponds to a set of facts that is a mutex group.

Therefore it follows that if we somehow obtain a complete set of mutex pairs, we can determine the maximum sized mutex group simply by finding the maximum sized clique which is NP-Complete [Karp, 1972]. However, finding the maximum sized mutex group is as hard as planning itself, therefore, it follows that inference of a complete set of mutex pairs is also as hard as finding a plan.

In the next section we introduce a novel subclass of mutex groups of which inference is NP-Complete.

3 Fact-Alternating Mutex Groups

Definition 6. A **fact-alternating mutex group** (fam-group) $M \subseteq \mathcal{F}$ is a set of facts such that $|M \cap s_{init}| \leq 1$ and $|M \cap \text{add}(o)| \leq |M \cap \text{pre}(o) \cap \text{del}(o)|$ for every operator $o \in \mathcal{O}$. A fam-group that is not a subset of any other fam-group is called a **maximal fam-group**.

Proposition 7. *Every fam-group is a mutex group.*

The name fact-alternating mutex group was chosen to stress its interesting property, which lies in the mechanism by which facts from a fam-group appear and disappear in particular states after the application of the operators. Consider a fam-group M and a state s that does not contain any fact from M ($M \cap s = \emptyset$). Now we can ask whether any following state $\pi[s]$ can contain any fact from M. The answer is that it cannot because any operator o applicable in s that could add a new fact from M to the following state $o[s]$ would need to have a fact from M in its precondition ($M \cap \text{pre}(o) \neq \emptyset$) which is in contradiction with the assumption that s contains no fact from M. So it follows that facts from each particular fam-group alternate between each other as new states are created and once the facts disappear from the state they cannot ever reappear again in any following state.

Proposition 8. *Let M denote a fact-alternating mutex group and let s denote a state. If $|M \cap s| = 0$, then for every operator sequence π applicable in s it holds that $|M \cap \pi[s]| = 0$.*

Proposition 8 can be used for the detection of dead-end states. A dead-end state is a state from which it is impossible to reach any goal state by a sequence of applied operators. Consider a fam-group M having a non-empty intersection with the goal ($|M \cap s_{goal}| \geq 1$) and a reachable state s that does not contain any fact from M ($|M \cap s| = 0$). Such a state must be clearly a dead-end state, because it follows from Proposition 8 that all states reachable from s, including the goal states, cannot contain any fact from M, which is formally stated in the following simple corollary of Proposition 8.

Corollary 9. *Let $M \subseteq \mathcal{F}$ denote a set of facts and let s denote a state. If M is a fam-group and $|M \cap s_{goal}| \geq 1$ and $|M \cap s| = 0$, then s is a dead-end state.*

The operators that have more than one fact from some mutex group (and, therefore, also from some fam-group) in its preconditions cannot be applicable in any reachable state. Similarly, the operators with add effects containing more than one fact from some mutex group (fam-group) are also unreachable, because the resulting state would be in contradiction with the mutex group (fam-group).[1] Such operators can be safely removed from the planning task. These two simple rules are not limited to the fact-alternating mutex groups, but they can be used with any type of mutex group.

However, fam-groups provide one additional method for pruning superfluous operators. Consider a fam-group M having a non-empty intersection with the goal and an operator o that does not have any fact from M in its add effects, but it has a non-empty intersection with its preconditions, delete effects, and the fam-group M. The resulting state of the application of the operator o would not contain any fact from the fam-group M. Therefore, such a state would be a dead-end state for the reasons already explained. This means that the operator o can be safely removed from the planning task because it can only produce dead-end states. In other words, the states resulting from the application of the operator are not useful in finding a plan and, therefore, the operator itself is not useful too.

Corollary 10. *Let $M \subseteq \mathcal{F}$ denote a set of facts, let s denote a state and let $o \in \mathcal{O}$ denote an operator applicable in s. If M is a fam-group and $|M \cap s_{goal}| \geq 1$ and $|M \cap pre(o) \cap del(o)| \geq 1$ and $|M \cap add(o)| = 0$, then $o[s]$ is a dead-end state.*

Since fam-groups are not defined using restrictions over all reachable states, but only over the initial state and all operators, it is easy to see that given a set of facts, we can check whether this set is a fam-group in a polynomial time. This suggests that the inference of the maximum sized fam-group is not as hard as the inference of the maximum sized mutex group—and indeed, it is only NP-Complete.

Theorem 11. *Let MAXIMUM-FAM-GROUP denote the following decision problem: Given a planning task Π and an integer k, does Π contain a fam-group of size at least k? MAXIMUM-FAM-GROUP is NP-Complete.*

Considering the tight relationship between graph cliques and mutex groups, it is not surprising that the maximum possible number of mutex groups is exponential in the number of facts. However, the same holds for fam-groups and this number can be expressed exactly as is done in the following proposition.

Proposition 12. *Let $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal} \rangle$ denote a planning task and let $n = |\mathcal{F}|$ denote a number of facts in Π. The maximum possible number $\mu(n)$ and $\mu_{fa}(n)$ of maximal mutex groups and maximal fam-groups, respectively, for $n \geq 2$, is the following:*

$$\mu(n) = \mu_{fa}(n) = \begin{cases} 3^{n/3}, & \text{if } n \bmod 3 = 0; \\ \frac{4}{3} \cdot 3^{n/3}, & \text{if } n \bmod 3 = 1; \\ 2 \cdot 3^{n/3}, & \text{if } n \bmod 3 = 2. \end{cases}$$

4 h^2 Mutex Pairs and Fact-Alternating Mutex Groups

The h^m heuristic [Haslum and Geffner, 2000] is able to produce a set of mutexes as its side effect. Specifically, the h^2 heuristic is the most common method for generating pairs of facts that cannot hold together in any reachable state, i.e., mutex pairs. The mutex pairs generated by h^2 will be referred to as h^2-mutexes from now on. We already know that given a set of mutex pairs (and therefore also h^2-mutexes) we can infer mutex groups using an algorithm for finding cliques in a graph. An interesting question now is, what is the relationship between fam-groups and h^2-mutexes?

The formal definition of an h^2-mutex below is based on an alternative characterization of the h^m heuristic using a modified planning task introduced by Haslum [2009].

Definition 13. Let $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal} \rangle$ denote a planning task. The planning task $\Pi^2 = \langle \Phi, \Omega, \psi_{init}, \{\} \rangle$ consists of a set of facts $\Phi = \{\phi_c \mid c \subseteq \mathcal{F}, |c| \leq 2\}$, a set of operators Ω, an initial state $\psi_{init} = \{\phi_c \mid c \subseteq s_{init}, |c| \leq 2\}$, and an empty goal specification. For each operator $o \in \mathcal{O}$ and for each subset of facts $g \subseteq \mathcal{F}$ such that $|g| \leq 1$ and g is disjoint with add(o) and del(o), the planning task Π^2 contains an operator $\omega_{o,g} \in \Omega$ with: pre$(\omega_{o,g}) = \{\phi_c \mid c \subseteq (\text{pre}(o) \cup g), |c| \leq 2\}$, add$(\omega_{o,g}) = \{\phi_c \mid c \subseteq (\text{add}(o) \cup g), c \cap \text{add}(o) \neq \emptyset, |c| \leq 2\}$, del$(\omega_{o,g}) = \emptyset$.

Let Ψ denote a set of all reachable states in Π^2. A pair of facts $\{f_1, f_2\} \subseteq \mathcal{F}$ such that $f_1 \neq f_2$ is an **h^2-mutex** iff for every reachable state $\psi \in \Psi$, it holds that $\phi_{\{f_1,f_2\}} \notin \psi$.

The definition is altered so that it contains empty goal specification, because it is not necessary for inference of mutex pairs, and it is formulated specifically for h^2, not for a general h^m. The Π^2 is an ordinary STRIPS planning task, but we used Greek letters to describe its parts to prevent confusion with the original planning task Π.

Now once we have formally defined mutex pairs generated by h^2, we find that h^2 always produces a (possibly non-strict) superset of decomposition of all fam-groups. More precisely, any mutex pair that is a subset of a fam-group must be an h^2-mutex, but not the other way around. This also means that if we infer h^2-mutexes and use an algorithm for listing maximal cliques to join the h^2-mutexes into larger mutex groups, then the resulting mutex groups will be non-strict supersets of fam-groups. However, the mutex groups created from h^2-mutexes do not necessarily have the same properties regarding detection of dead-end states as fam-groups described in Proposition 8, Corollary 9 and Corollary 10.

Theorem 14. *Let $M \subseteq \mathcal{F}$ denote a set of facts such that $|M| \geq 2$ and let $H = \{p \mid p \subseteq M, |p| = 2\}$. If M is a fam-group, then every $h \in H$ is an h^2-mutex.*

[1] This is a special case of disambiguation proposed by Alcázar *et al.* [2013].

Input: Planning task $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal} \rangle$
Output: A set of fam-groups $\mathcal{M}$
Initialize ILP with constraints according to Equations 1 and 2;
Set objective function of ILP to maximize $\sum_{f_i \in \mathcal{F}} x_i$;
$M \leftarrow \emptyset$;
Solve ILP and if a solution was found, save the resulting fam-group into M;
while $|M| \geq 1$ **do**
  Add M to the output set $\mathcal{M}$;
  Add constraint according to Equation (3) using M;
  $M \leftarrow \emptyset$;
  Solve ILP and if a solution was found, save the resulting fam-group into M;
end

Algorithm 1: Inference of fam-groups using ILP.

5 Inference of FAM-Groups

In this section, we describe an algorithm for the inference of fam-groups that is complete with respect to maximal fam-groups. The main part of the algorithm consists of an integer linear program (ILP) based on the definition of fam-groups (Definition 6) rewritten into a set of constraints. The ILP is constructed in the following way.

Each variable x_i of the ILP corresponds to a fact $f_i \in \mathcal{F}$ from the planning task. Variables can acquire binary values 0 or 1 only, 0 meaning that the corresponding fact is not present in the fam-group and 1 meaning the corresponding fact is part of the fam-group. Definition 6 can be rewritten into ILP constraints as follows:

$$\sum_{f_i \in s_{init}} x_i \leq 1, \tag{1}$$

$$\forall o \in \mathcal{O} : \sum_{f_i \in \text{add}(o)} x_i \leq \sum_{f_i \in \text{del}(o) \cap \text{pre}(o)} x_i. \tag{2}$$

Equation (1) corresponds to the restriction on the initial state, and Equation (2) to the restrictions on the operators. The objective function of the ILP is to maximize $\sum_{f_i \in \mathcal{F}} x_i$, which enforces the inference of the maximal fam-group.

The solution to this ILP is only one fam-group, so the ILP is solved repeatedly, each time with added constraints excluding already inferred fam-groups. The constraint excluding a fam-group M and all its subsets from the next solution is the following:

$$\sum_{f_i \notin M} x_i \geq 1. \tag{3}$$

The whole algorithm is encapsulated in Algorithm 1. First, the ILP constraints are constructed according to Equations 1 and 2, which ensures that the solutions of the ILP will be fam-groups. Then, in turn, a maximal fam-group is inferred through the ILP solution and consequently removed from future solutions using the added constraint corresponding to Equation (3). The cycle continues until the inferred fam-groups consist of, at least, one fact.

Theorem 15. *Algorithm 1 is complete with respect to the maximal fact-alternating mutex groups.*

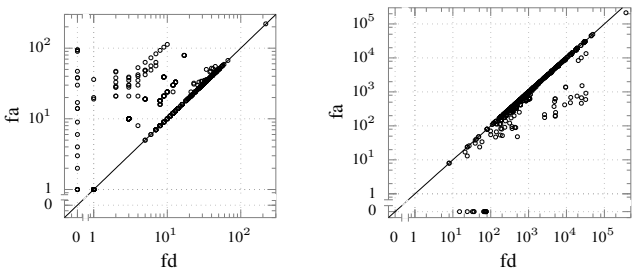

Figure 1: Comparison of `fa` and `fd` in terms of the number of inferred mutex groups (left) and the number of operators after pruning (right) for each problem from the dataset.

6 Experimental Results

The inference of fam-groups was experimentally evaluated on all domains from the optimal deterministic track of the International Planning Competition (IPC) 2011 and 2014 that do not contain any conditional effects after grounding (using Intel Xeon E5-4617 2.9GHz, 8 GB RAM). Here we present a singificantly reduced results from the original paper. We compare the implementation of Algorithm 1 (`fa`) and the state-of-the-art algorithm for inference of mutex groups from the Fast Downward planner (`fd`) [Helmert, 2006].

Every mutex group that was generated by `fd` was also generated by `fa` or it was a subset of some mutex group generated by `fa`. Overall, `fa` generated a richer set of mutex groups in 202 out of 540 problems (10 281 vs. 14 012). However, `fa` was much slower than `fd`. `fd` computed mutex groups under 1 second for almost all problems, but `fa` spent at average 6 seconds on a problem. Fig. 1 (left) shows a scatter plot of the number of inferred mutex groups in each problem for `fa` and `fd`.

We also compared the algorithms in terms of their pruning power. Mutex groups can be used for detection of operators that are either not reachable because their preconditions cannot be met, or the operators can generate only dead-end state, so they cannot be part of any plan. Also in this case `fa` provides better results than `fd` as can be seen on the scatter plot on Fig. 1 (right).

7 Conclusion

This paper is focused on the state invariants called mutex groups in the context of STRIPS planning. We provide the complexity analysis showing that the inference of the maximum sized mutex group is PSPACE-Complete. We introduce a new weaker type of mutex group called a fact-alternating mutex group (fam-group) and we show that the inference of the maximum sized fam-group is NP-Complete. Moreover, we describe an algorithm for inference of fam-groups that is complete with respect to the maximal fam-groups.

We prove that the h^2 variant of h^m heuristics [Haslum and Geffner, 2000] generates mutex pairs that are a superset of a pair decomposition of fam-groups. However, we also show that fam-groups has certain unique properties regarding detection of dead-end states, which is experimentally evaluated.

Acknowledgements

This Research was funded by the Czech Science Foundation (project no. 18-07252S).

References

[Alcázar and Torralba, 2015] Vidal Alcázar and Álvaro Torralba. A reminder about the importance of computing and exploiting invariants in planning. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 2–6, 2015.

[Alcázar *et al.*, 2013] Vidal Alcázar, Daniel Borrajo, Susana Fernández, and Raquel Fuentetaja. Revisiting regression in planning. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2254–2260, 2013.

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11:625–656, 1995.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artif. Intell.*, 69(1-2):165–204, 1994.

[Edelkamp and Helmert, 1999] Stefan Edelkamp and Malte Helmert. Exhibiting knowledge in planning problems to minimize state encoding length. In *Recent Advances in AI Planning, 5th European Conference on Planning (ECP)*, pages 135–147, 1999.

[Fikes and Nilsson, 1971] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971.

[Fišer and Komenda, 2018] Daniel Fišer and Antonín Komenda. Fact-alternating mutex groups for classical planning. *J. Artif. Intell. Res.*, 61:475–521, 2018.

[Gerevini and Schubert, 1998] Alfonso Gerevini and Lenhart K. Schubert. Inferring state constraints for domain-independent planning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI, IAAI)*, pages 905–912, 1998.

[Gerevini and Schubert, 2000] Alfonso Gerevini and Lenhart K. Schubert. Discovering state constraints in DISCOPLAN: some new results. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence (AAAI, IAAI)*, pages 761–767, 2000.

[Haslum and Geffner, 2000] Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS)*, pages 140–149, 2000.

[Haslum, 2009] Patrik Haslum. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 354–357, 2009.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *J. Artif. Intell. Res. (JAIR)*, 26:191–246, 2006.

[Helmert, 2009] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artif. Intell.*, 173(5–6):503–535, 2009.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[Kautz and Selman, 1992] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *Tenth European Conference on Artificial Intelligence (ECAI)*, pages 359–363, 1992.

[Lipovetzky *et al.*, 2016] Nir Lipovetzky, Christian J. Muise, and Hector Geffner. Traps, invariants, and dead-ends. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 211–215, 2016.

[Mukherji and Schubert, 2005] Proshanto Mukherji and Lenhart K. Schubert. Discovering planning invariants as anomalies in state descriptions. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 223–230, 2005.

[Mukherji and Schubert, 2006] Proshanto Mukherji and Lenhart K. Schubert. State-based discovery and verification of propositional planning invariants. In *Proceedings of the 2006 International Conference on Artificial Intelligence (ICAI)*, pages 465–471, 2006.

[Rintanen, 2000] Jussi Rintanen. An iterative algorithm for synthesizing invariants. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence (AAAI, IAAI)*, pages 806–811, 2000.

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

[Sideris and Dimopoulos, 2010] Andreas Sideris and Yannis Dimopoulos. Constraint propagation in propositional planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 153–160, 2010.

[Torralba and Alcázar, 2013] Álvaro Torralba and Vidal Alcázar. Constrained symbolic search: On mutexes, BDD minimization and more. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search (SOCS)*, pages 175–183, 2013.