Metis: Arming Fast Downward with Pruning and Incremental Computation

Yusra Alkhazraji
University of Freiburg, Germany
alkhazry@informatik.uni-freiburg.de

Michael Katz
IBM Haifa Research Labs, Israel
katzm@il.ibm.com

Robert Mattmüller
University of Freiburg, Germany
mattmuel@informatik.uni-freiburg.de

Florian Pommerening
University of Basel, Switzerland
florian.pommerening@unibas.ch

Alexander Shleyfman
Technion, Haifa, Israel
alesh@technion.ac.il

Martin Wehrle
University of Basel, Switzerland
martin.wehrle@unibas.ch

Introduction

Metis is a sequential optimal planner that implements three components on top of the Fast Downward planning system (Helmert 2006). The planner performs an A^* search using the following three major components:

- an admissible incremental LM-cut heuristic (Pommerening and Helmert 2013),
- a symmetry based pruning technique (Domshlak, Katz, and Shleyfman 2012), and
- a partial order reduction based pruning technique based on strong stubborn sets (Wehrle and Helmert 2012).

Each of those techniques was extended to support conditional effects. In addition, Metis features a flexible invocation of partial order reduction based pruning. In what follows, we describe each of these components in detail.

Background

We consider planning tasks $\Pi = \langle V, O, s_0, G, Cost \rangle$ captured by the standard SAS$^+$ formalism (Bäckström and Klein 1991; Bäckström and Nebel 1995) with operator costs, extended by conditional effects. In such a task, V is a set of finite-domain *state variables*, each with domain $\mathcal{D}(v)$. Each complete assignment to V is called a *state*, and $S = \prod_{v \in V} \mathcal{D}(v)$ is the *state space* of Π. The state s_0 is the *initial state* of Π. We sometime refer to a single variable assignment as to *fact*. Furthermore, the goal G is a partial assignment to V, where a state s is a *goal state*, iff $G \subseteq s$[1]. The set O is a finite set of *operators*. Each operator o is given by a pair $\langle \mathsf{pre}, \mathsf{effs} \rangle$. The *precondition* $\mathsf{pre}(o)$ is a partial assignment to V that defines when the operator is applicable. The set $\mathsf{effs}(o)$ is a set of *conditional effects* e, each given by a pair $\langle \mathsf{cond}, \mathsf{eff} \rangle$ of partial assignments to V called *conditions* and *effects*. The condition $\mathsf{cond}(e)$ defines when the conditional effect triggers. For a shorter presentation, we assume that eff assigns a value to exactly one variable. An effect that assigns a value to more variables can be split into multiple effects. Effects that do not assign a value at all can be safely removed. Finally, $Cost : O \to \mathbb{N}_0$ is a real-valued, non-negative *operator cost* function. Applying an applicable operator o in state s results in a state denoted by $s[o]$. The state $s[o]$ is obtained from s by applying all triggered conditional effects of o, setting the value of the state variable to the value in $\mathsf{eff}(e)$. State variables that do not appear in triggered effects receive their values from the state s. By the transition graph $\mathcal{T}_\Pi = \langle S, E \rangle$ of Π we refer to the edge-labeled digraph induced by Π over S: if $o \in O$ is applicable in state s, then $\mathcal{T}_\Pi$ contains an edge $(s, s[o]; o)$ from s to $s[o]$, labeled with o.

[1] We slightly abuse the notation here, treating (partial) assignments as sets of facts.

Heuristic

Metis uses a variant of the admissible LM-cut heuristic (Helmert and Domshlak 2009). In particular, we use the local incremental LM-cut heuristic, $h^{\text{iLM-cut}}_{\text{local}}$ (Pommerening and Helmert 2013) extended to support conditional effects. In the following, we provide a short rehash how the computation of standard LM-cut works, and afterwards discuss the two extensions.

The computation of standard LM-cut is done in *rounds*. Each round discovers a set of operators L such that every plan must contain at least one operator from L. Such a set is called a *disjunctive action landmark* but since we do not use any other kinds of landmarks, we will just use the term *landmark* in the following.

Each round of the LM-cut algorithm does the following steps:

1. Compute the $h^{\max}$ values (Bonet and Geffner 2001) of all variables. If the goal has an infinite $h^{\max}$ value, the task is unsolvable and the heuristic computation stops with a heuristic value of ∞. If the goal has an $h^{\max}$ value of 0, the algorithm stops with the current heuristic value (which is initialized as 0).
2. Define the *justification graph* as the graph $J = (F, E)$ with the set of facts F as nodes and a directed, weighted edge in E for every effect of every operator. The edge for effect e of operator o starts from a precondition of o with maximal $h^{\max}$ value, ends in the single fact in $\mathsf{eff}(e)$ and is labeled with o and weighted with $Cost(o)$. All nodes in the justification graph that have a path to the goal where all edges have weight 0 belong to the *goal zone* $F_g \subseteq F$.

The *cut* C contains all edges that end in F_g and start in a node that can be reached in J without traversing a node in F_g. The set of operators that occur as labels of edges in C is a landmark L of the task.

3. The *cost* of L, $Cost(L)$, is the minimum over the cost of all operators contained in L. This reflects that at least the cost of one operator in L must be used. Reduce the cost of each operator in L by $Cost(L)$ and increase the heuristic value by $Cost(L)$. This induces a cost partitioning and makes the final estimate admissible.
4. Discard L.

After the last round, all operator costs are reset to their original value.

Support for Conditional Effects

The original LM-cut algorithm is only defined for tasks without conditional effects. We extended its definition to handle conditional effects by considering them in the definition of the justification graph in step 2. and conservatively reducing the operator costs in step 3. (Keyder, Hoffmann, and Haslum 2012). Following the naming convention of Röger, Pommerening, and Helmert (2014), we call this heuristic $h^{\text{LM-cut}}_{\text{basic}}$.

Our extended definition of the justification graph handles unconditional effects as before, and includes an edge for every conditional effect. The edge for an effect e of an operator o ends in the single fact in $\mathsf{eff}(e)$ and starts in a fact from $\mathsf{pre}(o) \cup \mathsf{cond}(e)$ with maximal $h^{\max}$ value. It is labeled with o and weighted with $Cost(o)$. The cut C and the landmark L are defined as before.

We call the reduction in cost conservative because the cost of each operator can only be counted once. Once an operator o is part of a cut, the cost of o is reduced, and all effects of o are cheaper. In the presence of conditional effects the optimal relaxed plan can contain operators more than once which our heuristic would not be able to detect. For this reason our heuristic no longer dominates the $h^{\max}$ heuristic (Keyder, Hoffmann, and Haslum 2012). Röger, Pommerening, and Helmert (2014) describe a variant of LM-cut that dominates $h^{\max}$ but this is not implemented in Metis.

The original LM-cut heuristic (without support for conditional effects) $h^{\text{LM-cut}}_{\text{standard}}$ is the same as $h^{\text{LM-cut}}_{\text{basic}}$ on tasks without conditional effects but can be implemented more efficiently because the involved data structures have less memory and time overhead. Metis thus includes both implementations and uses $h^{\text{LM-cut}}_{\text{basic}}$ only on tasks where at least one operator has a conditional effect.

Incremental Computation

A set of operators L is a landmark for state s if every plan from s must use one operator from L. If we apply an operator $o \notin L$ to s, the resulting state s' can only have plans that are suffixes of plans for s. That is, if we add o in front of any plan π for s', we get a plan for s. Since every plan for s must use an operator from L and $o \notin L$, every plan for s' must also use an operator from L and L is also a landmark for s'.

In particular, this means that during the expansion of a state all landmarks that do not mention the applied operator are also landmarks of the successor. If we know landmarks $\mathcal{L}$ of the parent state when we calculate the heuristic value for a newly generated state, we can compute the set of landmarks $\mathcal{L}' = \{L \in \mathcal{L} \mid o \notin L\}$ of all landmarks that do not mention the applied operator o. For each $L \in \mathcal{L}'$ we then reduce the operator costs and increase the heuristic value as defined in step 3. and continue with the regular LM-cut algorithm.

Storing the discovered landmarks for all generated states, can make the heuristic computation much faster but also requires a lot of memory (Pommerening and Helmert 2013). Instead, we use the local incremental computation method $h^{\text{iLM-cut}}_{\text{local}}$ which recomputes the landmarks of a state before it is expanded. This is done with a regular (non-incremental) LM-cut computation that skips step 4. The landmarks are then stored temporarily, used for the incremental heuristic computation of the generated children and are then discarded. With this method the search will do one non-incremental heuristic computation for every expanded state and one incremental computation for every generated state. Since there usually are a lot more generated than expanded states and the incremental computation is faster, the time for the additional non-incremental computation can be amortized.

Symmetry Reduction

The symmetry pruning part of the Metis planner modifies the A^* algorithm to prune symmetrical search nodes. For that, we needed to (a) develop a mechanism identifying symmetrical states, and (b) exploit the information in the search. In what follows, we describe these two in detail.

Symmetries and Conditional Effects

In what follows, we discuss the symmetries of of the state transition graph $\mathcal{T}_\Pi$ of a SAS$^+$ planning task Π that are captured by automorphisms (isomorphisms to itself) of $\mathcal{T}_\Pi$. As the state transition graph $\mathcal{T}_\Pi$ is not (and cannot be) given explicitly, automorphisms of $\mathcal{T}_\Pi$ must be inferred from the description of Π. The specific method that Pochter, Zohar, and Rosenschein (2011) proposed for deducing automorphisms of $\mathcal{T}_\Pi$ exploits automorphisms of a certain graphical structure (colored graph), the *problem description graph (PDG)*, induced by the description of Π. Later, Domshlak, Katz, and Shleyfman (2012) slightly modified the definition, in particular extending it with a support for non-uniform cost operators. Here we extend the definition of Domshlak et al. to support conditional effects.

In the regular SAS$^+$ setting, the PDG has one node for each operator, with incoming edges from variable values in the operator precondition, and outgoing edges to variable values in the effect. The operator nodes are colored according to their costs. When conditional effects come into the picture, an additional node is introduced for each conditional effect. The edges for such nodes are as follows. An incoming edge is added from each variable value in the effect's condition. An outgoing edge is added to the variable value

in the effect. In addition, to preserve the connection between the conditional effect to its operator, an incoming edge is added from the operator node. The color of the conditional effect nodes is the same as the color of the corresponding operator node.

Automorphisms of the PDG define isomorphisms on the states S of the planning task Π, such that if a state s is mapped into s', then s and s' are symmetrical. Given a set of automorphisms of the PDG, Pochter et al. define a procedure, mapping each state to a *canonical* symmetrical state. Obviously, there can be multiple ways to define canonical states. Pochter et al. have chosen a local search procedure, comparing states by their variable values. The procedure terminates with a local minimum. Our implementation adopts their approach with a minor modification to the local search procedure.

Search Algorithm

In order to exploit the information about the problem's symmetries, Domshlak, Katz, and Shleyfman (2012) propose a sound and complete optimal search algorithm (hereafter referred to as DKS), extending A^* as follows. First, DKS extends the duplicate elimination mechanism to consider symmetrical states as duplicates. To do so, DKS requires storing an additional information for each node – the canonical state. Two states are then said to be duplicates, if their canonical state is the same. Unfortunately, using such duplicate elimination comes at a certain cost. When reopening is required, the parent relation, if updated, loses the connectivity property. Thus, once a goal state is reached, it is no longer possible to retrace a path from the goal state by the parent relation. Therefore, DKS introduced a procedure exploiting the symmetry information for reconstructing a plan following the parent relation, without requiring its connectivity.

To overcome the aforementioned requirement of storing an additional state per node for duplicate elimination, we introduce a simple modification of the DKS search algorithm. It is called *orbit search*, and it differs from DKS by storing *only* the canonical state per node. These canonical states define *orbits*, sets of (symmetrical) states that have the same canonical state – thus the name orbit search. Informally, orbit search searches in the space of orbits instead of the space of states. Note that due to the plan reconstruction procedure of DKS, the implementation of orbit search is extremely simple. The syntactical difference between A^* and orbit search is minor, the states are replaced by their canonical representatives when stored in the open and close lists. Our preliminary experiments have shown an advantage of orbit search over DKS, and the less complex implementation makes it especially attractive.

Partial Order Reduction

In addition to symmetry pruning, Metis features a pruning technique based on strong stubborn sets for planning (Wehrle and Helmert 2012), which is a state-based pruning technique based on partial order reduction (POR). In a nutshell, POR attempts to reduce the size of the reachable state space by pruning redundant applications of operator sequences. In the following, we describe strong stubborn sets, and their extension to support operators with conditional effects.

Strong Stubborn Sets

Let Π be a SAS$^+$ planning task without conditional effects. For a state s in Π, a strong stubborn set T_s in s is a set of operators that satisfy the following three requirements: First, T_s contains the operators of a disjunctive action landmark. Second, for all operators o in T_s that are applicable in s, T_s contains all operators that *interfere* with o. Informally, operators o and o' interfere if o falsifies a precondition of o', or vice versa, or o and o' write to a common variable with different values (see below for a definition in presence of operators with conditional effects). Third, for all non-applicable operators o in T_s, T_s contains a *necessary enabling set* for o in T_s. A necessary enabling set N for an inapplicable operator o in s is a set of operators such that every plan π_s from s to a goal state that includes o must include an operator from N before the first occurrence of o in π_s. We compute strong stubborn sets T_s in s with a fixed-point iteration. Generating successors only based on the applicable operators in T_s (instead of all operators applicable in s) preserves completeness and optimality of A^*.

Metis features a rather straight-forward implementation of strong stubborn sets, including some optimizations to reduce the induced computational overhead.

- Previous implementations of strong stubborn sets compute the interference relation between operators in a preprocessing step and cache the result (Alkhazraji et al. 2012; Wehrle et al. 2013). However, in domains with many operators, the precomputation can run out of memory due to the quadratic number of operator pairs.

  Metis computes the relation for operator interferences ("which pairs of operators interfere?") and achievers ("which fact is added by which operator?") on-the-fly and caches the result until a limit of 100 million entries in total is exceeded. In this case, we stop caching, and compute the missing information on-the-fly without storing it.

- In order to avoid unnecessary computational overhead induced by the fixed-point iteration for computing strong stubborn sets, we switch off this computation if we do not get significant pruning. In more detail, if it turns out that the number of node generations is reduced by less than one percent compared to not using POR after at least 1000 node expansions, then the computation of strong stubborn sets is disabled for the rest of the search on this task.

- Necessary enabling sets for o and s are computed by selecting a precondition fact f of o that is unsatisfied in s, and including all operators that set f to true. Metis uses a straight-forward instantiation by greedily selecting the first unsatisfied precondition fact of o. This strategy has been proposed by Alkhazraji et al. (2012), and corresponds to the static *Fast Downward* ordering investigated by Wehrle and Helmert (2014). Analogously, the disjunctive action landmark used to start the fixed-point iteration

to compute strong stubborn sets is obtained by selecting all achievers of an unsatisfied goal fact.

In the following, we describe the extension of the above implementation to deal with conditional effects.

Support for Conditional Effects

Metis treats operators with conditional effects in a conservative (and straight-forward) way based on the following modifications.

- Definition of *operator interference*: Operator $o = \langle \mathsf{pre}, \mathsf{effs} \rangle$ *disables* operator $o' = \langle \mathsf{pre}', \mathsf{effs}' \rangle$ iff there is at least one conditional effect $e = \langle \mathsf{cond}, \mathsf{eff} \rangle \in \mathsf{effs}(o)$ such that eff falsifies a precondition fact in pre' or a fact in the effect condition cond' for a conditional effect $\langle \mathsf{cond}', \mathsf{eff}' \rangle \in \mathsf{effs}(o')$. In other words, the conditions of the conditional effects are handled exactly as preconditions. Operators o and o' have *conflicting effects* iff there is a conditional effect $e = \langle \mathsf{cond}, \mathsf{eff} \rangle \in \mathsf{effs}(o)$ and a conditional effect $e' = \langle \mathsf{cond}', \mathsf{eff}' \rangle \in \mathsf{effs}(o')$ such that eff and eff' modify the same variable with a different value. Operators o and o' *interfere* if o disables o', or vice versa, or o and o' have conflicting effects.
- During the iterative computation of a strong stubborn set T_s in state s, for all operators $o = \langle \mathsf{pre}, \mathsf{effs} \rangle$ applicable in s, we add all operators that interfere with o as for SAS$^+$ planning tasks. In addition, for conditional effects $e = \langle \mathsf{cond}, \mathsf{eff} \rangle \in \mathsf{effs}(o)$ with unsatisfied effect condition cond in s, the set of achievers for an unsatisfied fact of cond is added. Adding such sets is required for preserving the soundness of the algorithm. Intuitively, such sets correspond to necessary enabling sets for non-applicable operators as compiling away conditional effects would result in corresponding non-applicable operators.

Finally, in domains that do not feature conditional effects, Metis additionally performs pruning based on active operators (Wehrle et al. 2013).

Interaction of Components

The partial order reduction technique is orthogonal to the other two techniques. There are no special considerations to consider when combining it with either one or both of them.

However, combining the symmetry based pruning technique with the incremental computation of the heuristic function requires some extra considerations. If a node with state s is expanded in orbit search, the state s' of the successor generated for operator o is not necessarily the result of applying o to s. Orbit search instead uses the canonical representative of $s[o]$ as the state s'. Calculating landmarks for s and re-using those that do not mention o as landmarks for s' thus is no longer valid.

We handle this issue in the following way: during the expansion of state s, we compute the landmarks for s, then generate the actual successor $s[o]$ and incrementally compute its heuristic value. We then look up s', the canonical representative of $s[o]$. Because of the symmetry between the two states, the heuristic value of $s[o]$ can also be used as an admissible estimate for s'. The search algorithm generates the successor node with the state s' but the heuristic value $h^{\text{iLM-cut}}_{\text{local}}(s[o])$. The state $s[o]$ is not stored permanently to save memory.

Acknowledgments

This work was partly supported by the German Research Foundation (DFG) with grant HE 5919/2-1 and by the Swiss National Science Foundation (SNSF) as part of the project "Safe Pruning in Optimal State-Space Search (SPOSSS)".

References

Alkhazraji, Y.; Wehrle, M.; Mattmüller, R.; and Helmert, M. 2012. A stubborn set algorithm for optimal planning. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, 891–892.

Bäckström, C., and Klein, I. 1991. Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence* 7(3):181–197.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Computational Intelligence* 11(4):625–655.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: Whats the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2012. Semi-relaxed plan heuristics. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 128–136.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting problem symmetries in state-based planners. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*.

Pommerening, F., and Helmert, M. 2013. Incremental lmcut. In *Proceedings of the 23nd International Conference on Automated Planning and Scheduling (ICAPS)*, 31–41.

Röger, G.; Pommerening, F.; and Helmert, M. 2014. Optimal planning in the presence of conditional effects: Extending LM-Cut with context splitting. In *ICAPS 2014 Workshop on Heuristics for Domain-Independent Planning*.

Wehrle, M., and Helmert, M. 2012. About partial order reduction in planning and computer aided verification. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 297–305.

Wehrle, M., and Helmert, M. 2014. Efficient stubborn sets: Generalized algorithms and selection strategies. In *Pro-*

ceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS). To appear.

Wehrle, M.; Helmert, M.; Alkhazraji, Y.; and Mattmüller, R. 2013. The relative pruning power of strong stubborn sets and expansion core. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, 251–259.