A Stubborn Set Algorithm for Optimal Planning

Yusra Alkhazraji[1] and **Martin Wehrle**[2] and **Robert Mattmüller**[1] and **Malte Helmert**[2]

Abstract. We adapt a partial order reduction technique based on *stubborn sets*, originally proposed for detecting dead ends in Petri Nets, to the setting of optimal planning. We demonstrate that stubborn sets can provide significant state space reductions on standard planning benchmarks, outperforming the *expansion core* method.

1 INTRODUCTION

Heuristic search is one of the most successful approaches for domain-independent planning, especially in the case of *optimal* planning [6, 5]. However, recent results show that the potential of conventional heuristic search algorithms for optimal planning is severely limited: even with almost perfect heuristics (which are hardly available in practice), search effort scales exponentially in the size of the planning task in typical planning domains [7]. This motivates the use of additional pruning techniques for optimal heuristic search.

In this context, various pruning techniques based on *partial order reduction* (POR) have been proposed [2, 10, 3]. Partial order reduction has been originally introduced by Valmari [8] for deadend detection in Petri nets. Valmari proposed the notion of *stubborn sets*, restricting exploration to a subset of applicable transitions in every state while maintaining completeness. Stubborn sets exploit that independent transitions do not have to be considered in all possible orderings. However, despite the need for pruning in planning, and despite the fact that partial order reduction techniques have existed in the model checking area for decades, the original stubborn set algorithms have not yet been properly adapted to and evaluated in planning. Recent results [9] show that stubborn sets generalize the *expansion core* method proposed for planning [2], which makes a comparison between these approaches particularly interesting.

We present a pruning algorithm for optimal planning based on strong stubborn sets (SSS), adapting an algorithm proposed by Godefroid for model checking [4]. We show that the algorithm preserves optimality and experimentally demonstrate that it significantly reduces search effort for standard planning benchmarks.

2 PRELIMINARIES

An SAS^+ *planning task* Π is a tuple $\langle V, I, O, G\rangle$ of *state variables* V, an *initial state* I, *operators* O and a *goal* G. A *fact* is a pair $\langle v, d\rangle$ with $v \in V$ and $d \in domain(v)$. A *state* is a set of facts that associates a value with each state variable in V. Operators are defined in terms of a *precondition* $pre(o)$, an *effect* $\mathit{eff}(o)$, and a *cost* $cost(o) \in \mathbb{R}_0^+$. Preconditions, effects, and the goal G are sets of facts. An operator is applicable in state s iff $pre(o) \subseteq s$; the operators applicable in s are denoted by $app(s)$. Applying operator o in s results in state s' which agrees with $\mathit{eff}(o)$ on all state variables mentioned in $\mathit{eff}(o)$ and with s on all other state variables. The objective of optimal planning is to find a cheapest sequence of applicable actions transforming I into a goal state, i. e., a state $s \supseteq G$.

Definition 1 (**dependency**). *Let Π be a planning task with variables V and operators O, and let $o_1, o_2 \in O$.*

1. *Operator o_1* disables *o_2 if there exists a variable $v \in V$ and facts $\langle v, d_1\rangle \in \mathit{eff}(o_1)$ and $\langle v, d_2\rangle \in pre(o_2)$ such that $d_1 \neq d_2$.*
2. *Operators o_1 and o_2* conflict *if there exists a variable $v \in V$ and facts $\langle v, d_1\rangle \in \mathit{eff}(o_1)$ and $\langle v, d_2\rangle \in \mathit{eff}(o_2)$ such that $d_1 \neq d_2$.*
3. *Operators o_1 and o_2 are* dependent *if o_1 disables o_2, or o_2 disables o_1, or o_1 and o_2 conflict. We write $dep(o)$ for the set of operators $o' \in O$ with which o is dependent.*

3 STRONG STUBBORN SETS

Proving correctness of a partial order reduction method is notoriously difficult: several published approaches in planning [1, 2, 11], which follow similar ideas as earlier work in model checking without being entirely equivalent, fail to preserve completeness due to technical errors [9]. To avoid the same pitfall, we stay close to the definition of SSS in model checking [4]. Apart from differences in the formalism used, the main change required is to account for the different objective: in planning, we must find a goal state, rather than a dead-end state (a state s with $app(s) = \emptyset$) as in the original work on SSS.

To define SSS, we need the closely related definitions of disjunctive action landmarks [5] and necessary enabling sets [4]. A *disjunctive action landmark* for a set of facts F in state s is a set of operators L such that every applicable operator sequence that starts in s and ends in $s' \supseteq F$ contains at least one operator $o \in L$. A *necessary enabling set* for operator $o \notin app(s)$ in state s is a disjunctive action landmark for $pre(o)$ in s.

Definition 2 (**strong stubborn set**). *Let Π be a planning task with operators O and goal G, and let s be a state of Π. A* strong stubborn set *(SSS) in s is an operator set $T_s \subseteq O$ such that:*

1. *For each $o \in T_s \cap app(s)$, we have $dep(o) \subseteq T_s$.*
2. *For each $o \in T_s \setminus app(s)$, we have $N_s^o \subseteq T_s$ for some necessary enabling set N_s^o of o in s.*
3. *T_s contains a disjunctive action landmark for G in s.*

The SSS computation algorithm (Alg. 1) starts with a disjunctive action landmark for G (thus satisfying condition 3 of Def. 2) and adds operators to the candidate set until conditions 1 and 2 are satisfied. Hence, Alg. 1 indeed computes a SSS.

Alg. 1 is called by a search algorithm like A* or IDA* before the expansion of each state s. Given the SSS T_s, it is sufficient for the search algorithm to expand s by applying the operators in $T_{app(s)} := T_s \cap app(s)$ instead of the complete set $app(s)$, while preserving completeness and optimality.

[1] University of Freiburg, {alkhazry,mattmuel}@informatik.uni-freiburg.de
[2] University of Basel, {martin.wehrle,malte.helmert}@unibas.ch

Algorithm 1 Strong stubborn set computation for state s

$T_s \leftarrow L_s^G$ for some disjunctive action landmark L_s^G for G in s
repeat
  for all $o \in T_s$ **do**
    if $o \in app(s)$ **then**
      $T_s \leftarrow T_s \cup dep(o)$
    else
      $T_s \leftarrow T_s \cup N_s^o$ for some nec. enabling set N_s^o for o in s
until T_s reaches a fixed-point
return T_s

Theorem 1. *Using only $T_{app(s)}$ instead of $app(s)$ when expanding a state s during A* search preserves completeness and optimality.*

Proof. We show that for all states s from which an optimal plan consisting of $n > 0$ operators exists, $T_{app(s)}$ contains an operator starting such a plan. A simple induction then shows that A* restricting successor generation to $T_{app(s)}$ is optimal.

Let T_s be a SSS as computed by Alg. 1 and $\pi = o_1, \ldots, o_n$ be an optimal plan for s. Since T_s contains a disjunctive action landmark for the goal, π contains an operator from T_s. Let o_k be the operator with smallest index in π that is also contained in T_s, i.e., $o_k \in T_s$ and $\{o_1, \ldots, o_{k-1}\} \cap T_s = \emptyset$. Then:

1. $o_k \in app(s)$: otherwise by the definition of SSS, a necessary enabling set $N_s^{o_k}$ for o_k would have to be contained in T_s, and at least one operator from $N_s^{o_k}$ would have to occur before o_k in π to enable o_k, contradicting that o_k was chosen with smallest index.
2. o_k is independent of $o_1, \ldots, o_{k-1}$: otherwise, using $o_k \in app(s)$ and the definition of SSS, at least one of $o_1, \ldots, o_{k-1}$ would have to be contained in T_s, again contradicting the assumption.

Hence, we can move o_k to the front: $o_k, o_1, \ldots, o_{k-1}, o_{k+1}, \ldots, o_n$ is also a plan for Π. It has the same cost as π and is hence optimal. Thus, we have found an optimal plan of length n started by an operator $o_k \in T_{app(s)}$, completing the proof. □

4 EXPERIMENTS

We investigated the pruning power of SSS by comparing A* guided by the state-of-the-art LM-cut heuristic [5] in three configurations: without POR, with the corrected expansion core method (EC) [2, 9], and with SSS. The results on all IPC optimal planning domains up to IPC 2011 are shown in Tab. 1, with those domains combined where all configurations solve exactly the same instances.

The numbers of node generations are sums over all instances per domain solved by all configurations. We observe that with POR the number of generated nodes *never* increases and often decreases compared to plain A*, and that SSS often has better pruning power than EC. The advantage of SSS is particularly pronounced in LOGISTICS00, PARCPRINTER, SATELLITE, and WOODWORKING.

To assess the impact of the computational overhead of computing SSS on planner performance, we measured coverage and search times (with time limit 30 min and memory limit 2 GB). We observe that the pruning due to SSS translates to significantly higher overall coverage. In many domains, it also results in significantly lower search times. If, however, the number of generated nodes is similar to or the same as with plain A*, then the computational overhead may lead to increased overall runtime (FREECELL, GRIPPER, PARKING, PIPESWORLD, SCANALYZER, and TRUCKS).

Time needed for the precomputation of the interference relation is negligible in most domains from Tab. 1, except for MPRIME and SCANALYZER, where it is about as high as pure search time. Only in a few domains such as FREECELL and PARKING, even excluding precomputation times, plain A* is faster than A* with SSS, although the numbers of generated nodes are the same. This can be attributed to the per-state overhead of computing SSS.

Table 1. Comparison of plain A*, A* with EC, and A* with SSS, all guided by the LM-cut heuristic (nodes +EC, +SSS in % of plain A*).

Domain (problems)	Nodes generated A*	+EC	+SSS	Coverage A*	+EC	+SSS
ELEVATORS-OPT08 (30)	18561161	100%	**55%**	19	19	**22**
ELEVATORS-OPT11 (20)	18006303	100%	**55%**	16	16	**18**
FREECELL (80)	**5543463**	**100%**	**100%**	**15**	11	12
GRIPPER (20)	**10807891**	**100%**	**100%**	**7**	6	**7**
LOGISTICS00 (28)	12855134	100%	**17%**	20	20	**21**
MPRIME (35)	921359	100%	**84%**	**22**	21	**22**
OPENSTACKS-OPT08 (30)	34336295	100%	**52%**	18	16	**20**
OPENSTACKS-OPT11 (20)	34209201	100%	**52%**	13	11	**15**
PARCPRINTER-08 (30)	2461106	100%	**<1%**	18	18	**30**
PARCPRINTER-OPT11 (20)	2460475	100%	**<1%**	13	13	**20**
PARKING-OPT11 (20)	**39354**	**100%**	**100%**	**2**	1	1
PIPESWORLD-NOTK (50)	2798494	100%	**99%**	**17**	16	**17**
PIPESWORLD-TK (50)	585963	100%	**97%**	**9**	8	**9**
PSR-SMALL (50)	1859026	100%	**80%**	**49**	48	**49**
ROVERS (40)	1281967	99%	**87%**	7	7	**8**
SATELLITE (36)	4283651	64%	**5%**	7	7	**10**
SCANALYZER-08 (30)	**7781870**	**100%**	**100%**	**14**	**14**	13
SCANALYZER-OPT11 (20)	**7781742**	**100%**	**100%**	**11**	**11**	10
TRUCKS (30)	**11687203**	**100%**	**100%**	**10**	9	9
WOODWK-OPT08 (30)	7334811	17%	**3%**	17	22	**28**
WOODWK-OPT11 (20)	7334070	17%	**3%**	12	15	**19**
REMAINING DOMAINS (707)	136716998	100%	**94%**	**425**	**425**	**425**
OVERALL (1396)	329647537	96%	**72%**	741	734	**785**

5 CONCLUSION

We have shown that SSS can alleviate the state explosion problem in optimal planning. For the future, we would like to further explore and refine the computation of SSS. In particular, this includes the investigation of the impact of different choices of the necessary enabling sets within the generic framework of SSS.

ACKNOWLEDGEMENTS

This work was supported by the German Research Council (DFG) as part of SFB/TR 14 AVACS and of the KontWiss project.

REFERENCES

[1] Y. Chen, Y. Xu, and G. Yao, 'Stratified planning', in *IJCAI 2009*, pp. 1665–1670, (2009).
[2] Y. Chen and G. Yao, 'Completeness and optimality preserving reduction for planning', in *IJCAI 2009*, pp. 1659–1664, (2009).
[3] A. J. Coles and A. Coles, 'Completeness-preserving pruning for optimal planning', in *ECAI 2010*, pp. 965–966, (2010).
[4] P. Godefroid, *Partial-Order Methods for the Verification of Concurrent Systems — An Approach to the State-Explosion Problem*, 1996.
[5] M. Helmert and C. Domshlak, 'Landmarks, critical paths and abstractions: What's the difference anyway?', in *ICAPS 2009*, pp. 162–169, (2009).
[6] M. Helmert, P. Haslum, and J. Hoffmann, 'Flexible abstraction heuristics for optimal sequential planning', in *ICAPS 2007*, pp. 176–183, (2007).
[7] M. Helmert and G. Röger, 'How good is almost perfect?', in *AAAI 2008*, pp. 944–949, (2008).
[8] A. Valmari, 'Stubborn sets for reduced state space generation', in *APN 1989*, pp. 491–515, (1991).
[9] M. Wehrle and M. Helmert, 'About partial order reduction in planning and computer aided verification', in *ICAPS 2012*, (2012).
[10] J. Wolfe and S. J. Russell, 'Bounded intention planning', in *IJCAI 2011*, pp. 2039–2045, (2011).
[11] Y. Xu, Y. Chen, Q. Lu, and R. Huang, 'Theory and algorithms for partial order based reduction in planning', *CoRR*, **abs/1106.5427**, (2011).