Sequencing Operator Counts

Toby O. Davies, Adrian R. Pearce, Peter Stuckey
National ICT Australia and
The University of Melbourne
Melbourne, Australia
first.last@nicta.com.au

Nir Lipovetzky
Computing & Information Systems
The University of Melbourne
Melbourne, Australia
nir.lipovetzky@unimelb.edu.au

Abstract

Operator-counting is a recently developed framework for analysing and integrating many state-of-the-art heuristics for planning using Linear Programming. In cost-optimal planning only the objective value of these heuristics is traditionally used to guide the search. However the primal solution, i.e. the operator counts, contains useful information. We exploit this information using a SAT-based approach which given an operator-count, either finds a valid plan; or generates a generalized landmark constraint violated by that count. We show that these generalized landmarks can be used to encode the perfect heuristic, h^*, as a Mixed Integer Program. Our most interesting experimental result is that finding or refuting a sequence for an operator-count is most often empirically efficient, enabling a novel and promising approach to planning based on Logic-Based Benders Decomposition (LBBD). **This paper originally appeared at ICAPS 2015** and is reproduced with the permission of the Association for Artificial Intelligence ([Davies *et al.*, 2015] Copyright 2016 AAAI, all rights reserved.)

1 Introduction

We investigate the problem of sequencing operator counts obtained from an operator counting heuristic. The algorithm will find a feasible sequence, if it exists, or obtain an explanation why there is no plan that uses only the operators counted. We refer to these explanations as generalised disjunctive action landmarks.

Disjunctive action landmarks are a core feature of many admissible heuristics [Helmert and Domshlak, 2009; Bonet and Helmert, 2010; Haslum *et al.*, 2012; Imai and Fukunaga, 2014]. Admissible heuristics based on these landmarks count the occurrence of any operator at most once. Most are dominated by the optimal delete relaxation h^+ [Helmert and Domshlak, 2009].

We generalize this notion of disjunctive action landmarks to count operators multiple times, and show that admissible heuristics using generalized landmarks are capable of defining the perfect heuristic h^*. As disjunctive action landmarks are the only kind of landmark we consider in this paper, we will refer to them simply as "landmarks".

We present a complete, incremental algorithm for generating generalized landmarks, prove that generalized landmarks can encode h^*, and experimentally verify that this algorithm computes h^*. Our approach can be used both as an incremental lower bound function and as an optimal planner, much like h^{++} [Haslum, 2012], as our approach does not terminate until it finds a proof that it has computed h^*, i.e. finds a plan with optimal cost.

We explain this approach to planning in terms of Logic-Based Benders Decomposition (LBBD). LBBD partitions an optimization problem in terms of a Mixed Integer Programming master problem, and one or more combinatorial subproblems used to explain flaws in the master problem.

This approach to planning is particularly promising for two reasons. Firstly, it introduces a principled interaction between operator-counting heuristics and SAT. This interaction can be applied to any explanation-based combinatorial search approach including SAT Modulo Theories (SMT) [Nieuwenhuis *et al.*, 2006] and constraint programming using Lazy Clause Generation (LCG) [Ohrimenko *et al.*, 2009]. Constraints or theories capable of generating clausal explanations can be added to the SAT model we present, potentially allowing direct integration of cost-optimal planning with SMT and state-of-the-art scheduling approaches based on LCG. Planning Modulo Theories problems [Gregory *et al.*, 2012] could therefore potentially be tackled using the extensive range of *existing* theories and constraints *already implemented* by the SMT and constraint programming communities.

Secondly, this approach decomposes the planning problem into problems for which there exist well-suited optimisation technologies: Mixed Integer Programming handling the linear counting constraints; and Conflict-Directed Clause Learning for the problem of operator sequencing given operator counts. This allows planning to take advantage of the ever improving performance of both of these widespread and industrially applied technologies.

2 Background

***SAS*$^+$ planning** A *SAS*$^+$ planning task is a tuple $\langle V, O, s_0, s_*, c\rangle$ where V is a set of finite domain state variables, O is a set of operators, s_0 is a full assignment of each variable to one of its values representing the initial state, and

s_* is a partial assignment of some subset of V representing the goal states. Finally c is a function $O \to \mathbb{N}_0^+$ that assigns a non-negative cost to each operator.

Each variable $X \in V$ has a domain $D(X)$, we sometimes abuse notation and write $X{=}x \in V$ which should be read $X \in V \wedge x \in D(X)$. Each operator o has a set of preconditions $pre(o)$ which is a partial assignment representing the preconditions of that operator, and a set of postconditions $post(o)$ which is a partial assignment representing the effects of the operator. Producers, $prod(X{=}x) = \{o \mid X{=}x' \in pre(o) \wedge X{=}x \in post(o) \wedge x' \neq x\}$ are the operators which cause $X{=}x$ to become true. Note that for simplicity, we do not distinguish between preconditions and prevail conditions in this paper.

A state s in the search space is a full assignment of every variable to a value. State s is said to satisfy a partial assignment F if all assignments in F are also in s, i.e. $X{=}x \in F \Rightarrow X{=}x \in s$. A state is said to be a goal state if it satisfies the partial assignment s_*.

An operator $o \in O$ is applicable in s if s satisfies the partial assignment $pre(o)$. If o is applicable in state s, applying o yields a new state s' which is the same as s except that all assignments $X{=}x \in post(o)$ replace any assignment to X.

A plan π is a sequence of operators $o_1, \cdots o_n$ such that o_1 is applicable in s_0, each subsequent operator is applicable in the state resulting from applying the previous operators in sequence, and the final state satisfies s_*. An optimal plan has the minimum sum of operator costs of all plans, a *SAS*$^+$ planning task may have many optimal plans.

Mixed Integer Programming A Mixed Integer Program (MIP) is a representation of a combinatorial optimisation problem in terms of linear constraints over some finite set of integer and continuous variables. Finding a solution to a MIP is an NP-complete problem, however its linear relaxation, (which replaces all integer variables with continuous ones) can be optimised in polynomial time.

In recent years many admissible planning heuristics have been proposed that use linear programs [van den Briel *et al.*, 2007; Coles *et al.*, 2008; Bonet, 2013; Pommerening *et al.*, 2014; Bonet and van den Briel, 2014].

Of particular interest is the family of operator-counting heuristics. Operator-counting uses a linear programming framework with a common set of variables Y_o representing the count of occurrences of each operator o in some relaxed representation of a plan. One or more component heuristics can be encoded as linear constraints on these variables such that the combined operator-counting heuristic dominates each of the component heuristics, often strictly [Pommerening *et al.*, 2014].

Operator Counts The solution to an operator-counting heuristic assigns a count to each operator o whenever the MIP is optimized. To distinguish the count assigned to each operator by a solution to an operator-counting heuristic from the variable Y_o, we refer to a solution to the heuristic as an operator-count $\mathcal{C}$.

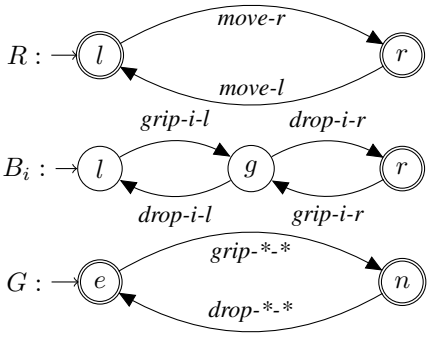

Figure 1: Domain Transition Graphs in gripper. The variable R represents the location of the robot (in the left or right room) ; B_i represents the location of ball i, (in the left or right room, or in the gripper); G represents the state of the gripper (empty or non-empty). For each automaton, the initial state is marked by an incoming arrow, and the states consistent with the goal s_* are double circled.

An operator-count $\mathcal{C}$ is said to be a projection of a plan π if for each distinct operator o, there exist exactly $\mathcal{C}(o)$ copies of that operator in π. We say an operator count is *perfect* if it is the projection of an optimal plan.

Incremental lower bounding Incremental lower bounding is a general technique for obtaining high-quality lower bounds, which can be useful in proving the quality of an existing plan. Incremental lower bounding was most prominently used in planning by Haslum (2012), however the technique is used throughout the various optimisation communities, referred to as "dual" techniques, reflecting the dual (lower) bound obtained from a linear program. Haslum (2012) describe a distinctive property of incremental lower bounding techniques as "informed by flaws in the current [optimum]".

3 Generalized Landmarks

To express some of the combinatorial aspects of planning problems it is useful to separate variables representing the number of times an operator o occurs, Y_o, from variables representing if an operator occurs at least k times, $[Y_o \geq k]$[1], which we refer to as bounds literals.

Definition 1. *A generalized landmark constraint is a linear inequality of the form:*

$$\sum_{i \in L} [Y_{o_i} \geq k_i] \geq 1$$

for some $L \subseteq O$.

We call these generalized landmarks because any traditional disjunctive action landmark can be encoded as a generalized landmark by setting all $k_i = 1$.

Consider an instance of the simplified gripper domain shown in Figure 1 with 2 balls. The goal is to move both

[1]Iverson brackets denote binary variables of the form $[P]$ that take the value 1 iff the condition P holds.

balls from the left room to the right, using a robot with a single "gripper" which can hold only one ball at a time. The robot starts in the left room, and can only pick up and drop balls in the room it is currently occupying.

An operator count obtained by some admissible heuristic on this domain counts the following.

$$\mathcal{C}(o) = 1 \quad \textbf{if } o \in \{\textit{move-l}, \textit{move-r}, \textit{grip-1-l}, \textit{grip-2-l}, \textit{drop-1-l}, \textit{drop-2-l}\}$$
$$\mathcal{C}(o) = 0 \quad \textbf{otherwise}$$

Note that there is only one occurrence of the *move-r* operator, however all feasible plans must contain two of this operator. We can add the constraint $[Y_{\textit{move-r}} \geq 2] \geq 1$ to explain this requirement. With the addition of this constraint the MIP returns the optimal operator count for this instance.

To enforce the correct behaviour of bounds literals we need to add the following *domain constraints*[2] to our model:

$$[Y_o \geq k] \leq [Y_o \geq k-1] \qquad \text{if } k > 0 \qquad (1)$$
$$Y_o \geq \sum_{i=1}^{\infty} [Y_o \geq i] \qquad (2)$$
$$Y_o \leq M[Y_o \geq k] + k - 1 \qquad (3)$$

Where M is a sufficiently large number such that no feasible plan could contain more than M of any individual operator. In practice this number need only be as large as the longest plan the solver could feasibly solve. Constraint 1 ensures that a bound can't hold unless the next smallest bound also holds; 2 ensures that if k bounds literals are set, then at least k operators must occur; and finally 3 ensures that if k or more operators occur, the bounds literal $[Y_o \geq k]$ must be set.

Theorem 1. *For any solvable SAS$^+$ planning problem having strictly positive action costs, there exists a set of generalized landmark constraints (with the domain constraints for all the bounds literals involved) such that solving a MIP with these constraints will compute $h^*(s_0)$.*

Proof Sketch. An optimal operator count $\mathcal{C}$ (which may initially be empty) can be obtained by solving the MIP. If $\mathcal{C}$ does not represent the projection of a plan, then the generalized landmark constraint:

$$\sum_{o \in O} [Y_o \geq \mathcal{C}(o) + 1] \geq 1$$

can be added.

This constraint can be read "at least one operator must be applied at least one more time". This is clearly violated by $\mathcal{C}$, and can only possibly invalidate subsets of $\mathcal{C}$. If any strict subset of $\mathcal{C}$ were feasible, $\mathcal{C}$ would not be optimal. Consequently this new constraint changes the optimum solution iff $\mathcal{C}$ is not perfect.

□

[2] *Domain constraints* reflect the fact that Y_o variables are finite domain variables, and the bounds literals we use are closely related to bounds literals used in lazy clause generation [Ohrimenko *et al.*, 2009], where the same term is used.

If we were to omit bounds literals for some operators from the landmark, it would invalidate many more operator counts. This is similar to traditional landmarks which are stronger when they contain a small subset of operators. To obtain smaller, more focused landmarks we turn to the conflict analysis built into modern "Conflict-Directed Clause Learning" SAT solvers.

4 SAT Encoding for Operator Sequencing

Assumptions are a feature of most SAT solvers' incremental interfaces. These allow the user to temporarily assert unit clauses. Importantly, if the resulting formula including these unit clauses is not satisfiable, the final conflict in the unsatisfiability proof can always be re-written in terms of a subset of the assumptions. This conflict clause represents a necessary (though not in general sufficient) property required of any model. In our SAT encoding assumptions will be used to ensure that only the operators selected by the operator counting heuristic are actually used.

Our SAT model (ommited for brevity), follows a standard layer-based encoding with variables for each operator and each fact in each layer. Our contribution adds a constraint for each operator o, ensuring that the total number of occurances of that operator do not exceed $\mathcal{C}(o)$, these constraints are asserted using assumptions on bounds literals $[Y_o \geq \mathcal{C}(o)]$. The goal is asserted in the final layer L, by assuming the literal $\neg[\Sigma\mathcal{C}(o) \geq L + 1]$, which implies the goal must be achieved by layer L.

The conflict clause will thus contain the goal assumption, plus some subset of the assumed bounds literals. Specifically it will be of the form:

$$[\Sigma\mathcal{C}(o) \geq L+1] \vee [Y_{o_1} \geq \mathcal{C}(o_1)+1] \vee \cdots \vee [Y_{o_n} \geq \mathcal{C}(o_n)+1]$$

This clause must be a necessary condition on all plans of length L or less. Since it is also satisfied by any operator count having more than L actions in total, it is also a necessary condition on all plans. This translates to a generalized landmark cut by replacing $\vee$ with $+$ and appending ≥ 1. The only complication is the $\neg[\Sigma\mathcal{C}(o) \geq L + 1]$ literal, which we tackle by adding an artificial operator T with zero cost (representing the total operator count) to the MIP, constrained such that:

$$Y_T = \sum_{o \in O} Y_o$$

Using this new operator, we can replace the total operator count literal $[\Sigma\mathcal{C}(o) \geq L + 1]$ with the bounds literal for the artificial operator T:

$$[\Sigma\mathcal{C}(o) \geq L + 1] \equiv [Y_T \geq L + 1]$$

It should be noted that the SAT formula we use only ensures that no more operators occur than were chosen in the operator count. Thus it can sequence any subset of an operator count, allowing it to be used with approximate solutions while guaranteeing that the same proof of admissibility as in Theorem 1 applies.

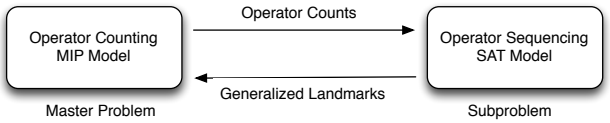

Figure 2: A Logic-Based Benders Decomposition Approach to Optimal Planning

5 Planning using Logic-Based Benders

Logic-Based Benders Decomposition (LBBD) [Hooker and Ottosson, 2003] is an approach to decomposing combinatorial search problems into a master MIP and one or more combinatorial subproblems. The master and subproblems share some variables such that the subproblem becomes easier to solve or prove infeasible once those variables it shares with the master are fixed. Importantly, LBBD allows for mixing of different optimisation technologies which may be better suited to the master problem and subproblems.

The master problem represents a relaxation of the original problem, and the subproblem checks for and *explains* flaws in that relaxation. Explanations in this context are constraints on the variables in the master problem. By incrementally adding these explanations, the master problem incrementally approaches the true solution.

First the master MIP is solved, and the optimal values of the shared variables are taken from the master, and this optimal assignment is assumed within the subproblem, which is then solved. If the subproblem is satisfiable then the optimal solution to the original problem has been found. [3]

If the subproblem is not satisfiable, some violated necessary condition on the shared variables is detected, and a constraint (the Benders cut) is added to the master problem. The process is then iterated until the master problem's relaxation becomes satisfiable.

In our case, the master MIP is any operator-counting heuristic, and the operator counts are shared variables.

Canonical planning (where each operator can be applied at most once) is NP-complete [Vidal and Geffner, 2006], meaningfully easier than the full planning problem. Many domains in planning are canonically plannable, that is there exists a plan containing only one instance of each operator. Our subproblem of sequencing the operators is pseudo-polynomially reducible to a canonical planning problem, by replacing each operator o with $\mathcal{C}(o)$ copies of itself, since $\mathcal{C}(o)$ is usually small in optimal plans the subproblem is often much easier than the full problem in practice.

6 Results

Figure 3 shows that 99% of all the sequence calls take less than 1 second, although there is a significant long-tailed distribution: 0.01% of sequence calls took over 5 minutes.

In our full paper, we compare opseq with *hpp* and *SymBA*-2* as an incremental lower-bounding approach. Our results

[3] In general, where the subproblem requires optimisation this is not true, but we omit this case for simplicity as it does not apply to our decomposition. See Hooker and Ottosson (2003).

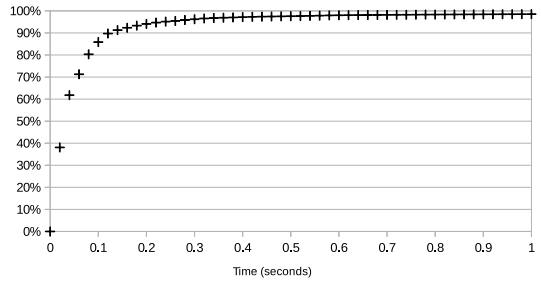

Figure 3: Cumulative Frequency of Sequence Times

show that *SymBA*-2* is extremely effective, beating both dual techniques in all three metrics in all but 5 domains, although in 4 of these 5 domains, *OpSeq* earns the best dual quality score, and in 3 domains even beats *SymBA** in coverage. *OpSeq* also beats the previous state-of-the-art in incremental lower bounding in 9 of the 11 domains investigated.

7 Conclusions and Further Work

We have defined a simple generalization of landmarks which allows the encoding of admissible heuristics upper-bounded only by h^*. We also introduce a SAT-based, complete algorithm for generating a generalized landmark violated by a given operator count which is usually very fast. We experimentally confirmed that h^* can be computed using only this algorithm, and demonstrated that it outperforms the previous state-of-the-art in incremental lower bounding: h^{++}.

There are other more conventional applications for generalized landmarks as well, such as pre-processing to generate an initial set of generalized landmarks which can then be used in an analogue of Incremental LM-Cut [Pommerening and Helmert, 2013]. We expect this to provide improved heuristic guidance near the root of the search where it is most valuable. While we use a complete algorithm to generate landmarks, there is an obvious fast but incomplete algorithm obtained by simply terminating early when long-tailed behaviour is observed.

We believe this approach is interesting and promising because it allows a principled interaction between state-of-the-art heuristics and explanation-based combinatorial search approaches including SAT, SMT and LCG. Any constraint capable of explaining its inferences can be added to the SAT subproblem, potentially allowing direct integration of cost-optimal planning with SMT and state-of-the-art scheduling approaches based on constraint programming with lazy clause generation. This means that, by extending the approach we present, we should be able to solve similar problems to Planning Modulo Theories [Gregory *et al.*, 2012] by taking advantage of the extensive range of *existing* theories and constraints *already implemented* by the SMT and constraint programming communities.

8 Acknowledgements

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research

Council through the ICT Centre of Excellence Program.

References

[Bonet and Helmert, 2010] Blai Bonet and Malte Helmert. Strengthening landmark heuristics via hitting sets. In *European Conference on Artificial Intelligence*, pages 329–334. IOS press, 2010.

[Bonet and van den Briel, 2014] Blai Bonet and Menkes van den Briel. Flow-based heuristics for optimal planning: Landmarks and merges. In *International Conference on Automated Planning and Scheduling*, pages 47–55. AAAI Press, 2014.

[Bonet, 2013] Blai Bonet. An admissible heuristic for SAS$^+$ planning obtained from the state equation. In Francesca Rossi, editor, *International Joint Conference on Artificial Intelligence*, pages 2268–2274. AAAI Press/IJCAI, 2013.

[Coles *et al.*, 2008] A. I. Coles, M. Fox, D. Long, and A. J. Smith. A hybrid relaxed planning graph-LP heuristic for numeric planning domains. In *International Conference on Automated Planning and Scheduling*, pages 52–59. AAAI Press, September 2008.

[Davies *et al.*, 2015] Toby Davies, Adrian R. Pearce, Peter J. Stuckey, and Nir Lipovetzky. Sequencing operator counts. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2015.

[Gregory *et al.*, 2012] Peter Gregory, Derek Long, Maria Fox, and J Christopher Beck. Planning modulo theories: Extending the planning paradigm. In *International Conference on Automated Planning and Scheduling*, pages 65–73. AAAI Press, 2012.

[Haslum *et al.*, 2012] Patrik Haslum, John Slaney, and Sylvie Thiébaux. Minimal landmarks for optimal delete-free planning. In *International Conference on Automated Planning and Scheduling*, pages 353–357. AAAI Press, 2012.

[Haslum, 2012] Patrik Haslum. Incremental lower bounds for additive cost planning problems. In *International Conference on Automated Planning and Scheduling*, pages 74–82. AAAI Press, 2012.

[Helmert and Domshlak, 2009] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *International Conference on Automated Planning and Scheduling*, pages 162–169. AAAI Press, 2009.

[Hooker and Ottosson, 2003] John N Hooker and Greger Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[Imai and Fukunaga, 2014] Tatsuya Imai and Alex Fukunaga. A practical, integer-linear programming model for the delete-relaxation in cost-optimal planning. In *European Conference on Artificial Intelligence*, volume 263, pages 459–464. IOS Press, 2014.

[Nieuwenhuis *et al.*, 2006] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, 2006.

[Ohrimenko *et al.*, 2009] O. Ohrimenko, P.J. Stuckey, and M. Codish. Propagation via lazy clause generation. *Constraints*, 14(3):357–391, 2009.

[Pommerening and Helmert, 2013] Florian Pommerening and Malte Helmert. Incremental LMs-cut. In *International Conference on Automated Planning and Scheduling*, pages 162–170. AAAI Press, 2013.

[Pommerening *et al.*, 2014] Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. LP-based heuristics for cost-optimal planning. In *International Conference on Automated Planning and Scheduling*, pages 226–234. AAAI Press, 2014.

[van den Briel *et al.*, 2007] Menkes van den Briel, J. Benton, Subbarao Kambhampati, and Thomas Vossen. An LP-based heuristic for optimal planning. In Christian Bessière, editor, *Principles and Practice of Constraint Programming*, volume 4741 of *Lecture Notes in Computer Science*, pages 651–665. Springer, 2007.

[Vidal and Geffner, 2006] Vincent Vidal and Héctor Geffner. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3):298–335, 2006.