Sequencing Operator Counts with State-Space Search

Wesley L. Kaizer; André G. Pereira; Marcus Ritt
Federal University of Rio Grande do Sul, Brazil

ABSTRACT

A search algorithm with an admissible heuristic function is the most common approach to optimally solve classical planning tasks. Recently [1] introduced the solver *OpSeq* using Logic-Based Benders Decomposition to solve planning tasks optimally. In this approach, the master problem is an integer program derived from the *operator-counting* framework that generates operator counts, i.e., an assignment of integer counts for each task operator. Then, the *operator counts sequencing subproblem* verifies if a plan satisfying these operator counts exists, or generates a necessary violated constraint to strengthen the master problem. In *OpSeq* the subproblem is solved by a SAT solver.

In this paper we show that operator counts sequencing can be better solved by state-space search. We introduce *OpSearch*, an A*-based algorithm to solve the operator counts sequencing subproblem: it either finds an optimal plan, or uses the frontier of the search to derive a violated constraint. We show that using a standard search framework has two advantages: i) search scales better than a SAT-based approach for solving the operator counts sequencing, ii) explicit information in the search frontier can be used to derive stronger constraints.

We present results on the IPC-2011 benchmarks showing that this approach solves more planning tasks, using less memory. On tasks solved by both methods *OpSearch* usually requires to solve fewer operator counts sequencing problems than *OpSeq*, evidencing the stronger constraints generated by *OpSearch*.

INTRODUCTION

[1] introduced a novel approach for cost-optimal planning, recognizing that the primal solution of the operator-counting linear program contains useful information that can be understood as a possible incomplete and unordered plan. This approach interprets the operator-counting framework beyond its primary use as a heuristic function and decomposes the process of finding solutions to a planning task into two independent but related subproblems, in a similar way to Logic-Based Benders Decomposition [2]. There is a master problem and a combinatorial subproblem used to explain the infeasibility of a solution of the master problem. The master problem is modeled as an integer program, corresponding to an operator-counting heuristic. The subproblem is modeled as a *propositional satisfiability* (SAT) problem encoding the planning task and the operator counts obtained from the solution of the master. A SAT solver is then used to sequence the operator counts, i.e., to check if a plan with these counts exists. If there is no plan with the given operator counts, the SAT solver returns a violated constraint for the master problem.

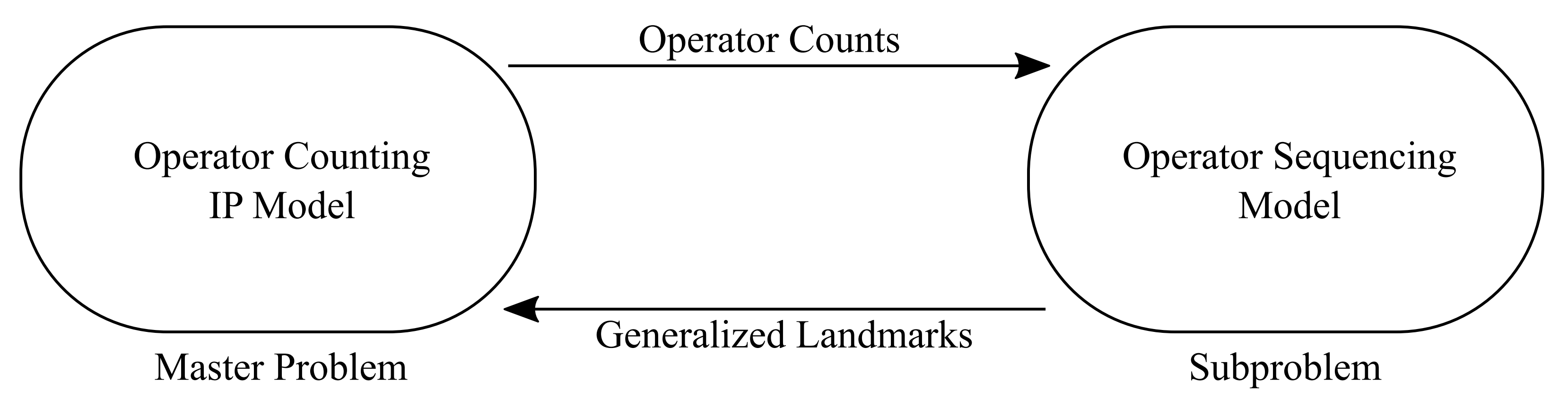
In this paper, we solve the operator counts sequencing subproblem using heuristic search instead of a SAT-based formulation. This new approach is based on an A* search that employs information unavailable to SAT solvers, such as the f-value of search nodes and the explicit structure of the search graph. We present a novel strategy to construct a violated constraint during the expansion of the search graph by considering the frontier of the search. We show that this strategy generates an admissible operator-counting constraint. We show experimentally that the resulting algorithm *OpSearch* has better coverage and less computational requirements than a SAT-based approach and can generate smaller and more informative explanations of infeasibility, as shown by the total number of solved subproblems required to solve planning tasks. We believe this approach is relevant because it opens new research directions towards specialized operator counts sequencing methods based on well-known classical planning technology.

PLANNING USING LOGIC-BASED BENDERS DECOMPOSITION

Usually in classical planning, only the objective function value of the operator-counting heuristic guides the search. However, variables Y_o in the primal solution of IP$_C$ contains useful information. This suggests a novel approach to solve planning tasks optimally.

[1] propose a Logic-Based Benders Decomposition that decomposes the process of solving planning tasks into two related problems: a master problem that solves IP$_C$ – a relaxation of the original planning task, which generates operator counts $\mathcal{C}_s$, and a subproblem that tries to sequence $\mathcal{C}_s$, constructing a violated constraint on failure.

The main idea consists of incrementally strengthening the master problem relaxation with some learned knowledge about the infeasibility of its current solution. These constraints should be as informative as possible to decrease the number of total iterations between master and the subproblem. The process stops when the Branch and Cut algorithm (BC) from master proves the optimality of the current incumbent plan. This decomposition establishes an interface between operator-counting heuristics and operator counts sequencing procedures.



PROPOSED APPROACH: OPSEARCH

We propose a solver *OpSearch*, which uses the A* search algorithm to solve the operator counts sequencing subproblem. Given an initial operator counts $\mathcal{C}_{s_0}$, it returns a plan π if $\mathcal{C}_{s_0}$ is sequencable, or a violated condition as a generalized landmark constraint L, otherwise. The presence of potentially useful information in the search graph, such as f-values, motivates its use as base for an alternative algorithm. This approach could generate smaller and more informed constraints and eliminating irrelevant parts of constraints can significantly decrease solving time of an integer program.

Our approach follows the main idea of planning using Logic-Based Benders Decomposition. We initiate the process using a BC to solve the IP$_C$. If BC finds an integer solution it calls *OpSearch* and we try to sequence the corresponding operator counts. If BC finds a relaxed solution we obtain a valid operator counts by rounding up the primal solution values to the nearest integers, and sequencing only if its cardinality and objective value are within 20% of the linear count. If the operator counts provided is sequencable *OpSearch* informs the BC that a new solution has been found. This process continues until BC proves that one of the found plans is optimal.

CONSTRAINT GENERATION STRATEGY

We now explore the situation when $v_o \notin vars(s) \land c(o) > 0$ and $s(v_o) = 0$ to derive some violated condition on the current operator counts. This condition is modeled as a generalized landmark constraint with bounds literals for operator o and can be interpreted as follows: if we had one more instance of o, we could further expand a state, that could possibly reach a goal state with optimal cost. Additionally, we can use other information available during A* to strengthen the generated constraints, such as the f-value of state s, since it is an estimate of the plan cost through s.

Next we present the strategy to generate violated constraints from non-sequencable operator counts. It incrementally generates bounds literals during A* search to compose the final learned generalized landmark constraint L, that includes at most one bounds literal for each operator. The strategy returns bounds for operators that currently have count 0 but might generate new states with an f-value at most $f_{\max}$, the objective value of the relaxation of the node in the BC tree that called the sequencing subproblem. State s denotes a state expanded by A* and s' is a generated one.

$$L = \{\, [Y_o \geq \mathcal{C}_{s_0}(o) + 1] \mid \exists s \xrightarrow{o} s' : f(s') \leq f_{\max} \land \\ ((v_o \notin vars(s) \land c(o) > 0) \lor s(v_o) = 0)\}$$

RESULTS

The following tables show results for *OpSeq* and *OpSearch*. Best results are highlighted. Column C presents the coverage; S the total number of sequencing calls; $\bar{T}_t$ the mean total solving time in seconds; $\overline{M}$ the mean memory usage in MB; and $\overline{u}$ the mean percentage of operators included in the generated constraints.

	OpSeq	*OpSearch*
C	63	**73**
S	121202	**99437**
$\bar{T}_t$	1783	**1720**
$\bar{M}$	865	**367**
$\bar{u}$	20	**6**

	OpSeq	h^{blind}	h^{LMCut}	h^{OC}	h^*
S	29106	25059	13304	7215	**3214**
$\bar{T}_t$	37	**10**	11	39	13
$\bar{M}$	95	82	82	**81**	234
$\bar{u}$	18	18	11	10	**8**
C	169	191	195	200	**241**

These results show that *OpSearch* is better than *OpSeq* since it solves more tasks, solves less subproblems, uses less memory and generate smaller constraints. Also, we see that, as a more informed heuristic is used by *OpSearch*, the number of subproblems solved, the memory usage and the size of the generated constraints decrease and the number of solved tasks increases.

CONCLUSIONS

In this work we introduced *OpSearch*, a technique inspired by Logic-Based Benders Decomposition that uses an A*-based algorithm to solve the problem of sequencing operator counts. As main results we showed that heuristic search is able to sequence operator counts or to generate admissible constraints in the form of generalized landmarks, and that it can perform better than *OpSeq*, a SAT-based approach to sequencing, solving fewer subproblems and presenting a higher coverage. We also presented results indicating that an approach based on A* can scale better than *OpSeq* in terms of overall memory usage.

REFERENCES

[1] Toby O Davies et al. “Sequencing operator counts”. In: *International Conference on Automated Planning and Scheduling*. 2015, pp. 61–69.

[2] John N Hooker and Greger Ottosson. “Logic-based Benders decomposition”. In: *Mathematical Programming* 96.1 (2003), pp. 33–60.