Tractable Cost-optimal Planning over Restricted Polytree Causal Graphs

Meysam Aghighi, Peter Jonsson and Simon Ståhlberg
Department of Computer and Information Science, Linköping University, Sweden
{meysam.aghighi, peter.jonsson, simon.stahlberg} at liu.se

Abstract

Causal graphs are widely used to analyze the complexity of planning problems. Many tractable classes have been identified with their aid and state-of-the-art heuristics have been derived by exploiting such classes. In particular, Katz and Keyder have studied causal graphs that are hourglasses (which is a generalization of forks and inverted-forks) and shown that the corresponding cost-optimal planning problem is tractable under certain restrictions. We continue this work by studying polytrees (which is a generalization of hourglasses) under similar restrictions. We prove tractability of cost-optimal planning by providing an algorithm based on a novel notion of variable isomorphism. Our algorithm also sheds light on the k-consistency procedure for identifying unsolvable planning instances. We speculate that this may, at least partially, explain why merge-and-shrink heuristics have been successful for recognizing unsolvable instances.

Introduction

d \ k	2	fix. const	free
2	COPG is in P	COPG is in P*	PG is NP-h.**
fix. const	**COPG is in P**	**COPG is in P**	PG is NP-h.
free	PG is NP-h.***	PG is NP-h.	PG is NP-h.

Table 1: *The known results using the restrictions below. COPG stands for Cost-Optimal Plan Generation, and PG stands for Plan Generation. The shaded cells are results that can be derived from the neighboring cells, and the results in bold are the new results presented in this paper.*
: [6]. **: [1]. *: [2].*

- The paper presents a polynomial-time algorithm
- Let k and d be fixed constants, then the algorithm generates an optimal plan for SAS$^+$ planning instances whose:
  - Variables' domain size $\leq k$
  - Causal graph is a polytree
  - Causal graph has a diameter $\leq d$
- The classes induced by these restrictions are a generalization of some previous discovered tractable classes

Defoliation of Polytrees

- Identifying *isomorphic variables* is the keystone of our algorithm
- Whether two variables are isomorphic is decided by analyzing the structure of the domain transition graph and their dependencies to other variables
- Figure 1 demonstrates two variables that are isomorphic
- For variables that only has leaves there is a constant upper bound on the number of non-isomorphic leaves
- Two isomorphic variables can be combined and replaced by a new variable
- Combine isomorphic leaves to achieve the upper bound, Figure 2 combines the isomorphic variables in Figure 1
- We have a constant number of leaves! Compute the product and replace the variable and its leaves with it
- The cost of an optimal solution does not change by combining isomorphic variables or by replacing variables with the product

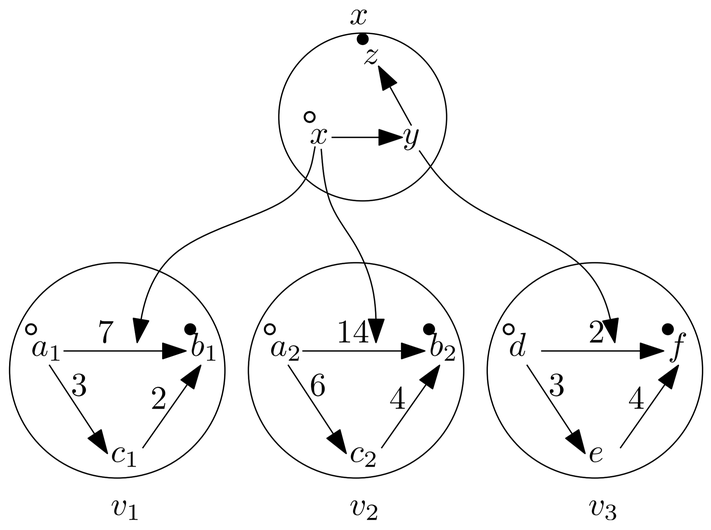

Figure 1: *The causal graph of an instance. The graph inside a vertex is the domain transition graph of the variable. The small circles denotes the initial values, and the dots denotes the goal values. Actions in the domain transition graph denotes actions, and arrows to actions denotes prevailconditions. The variable x is the critical vertex, v_1 and v_2 are isomorphic but v_3 is not isomorphic to any other variable.*

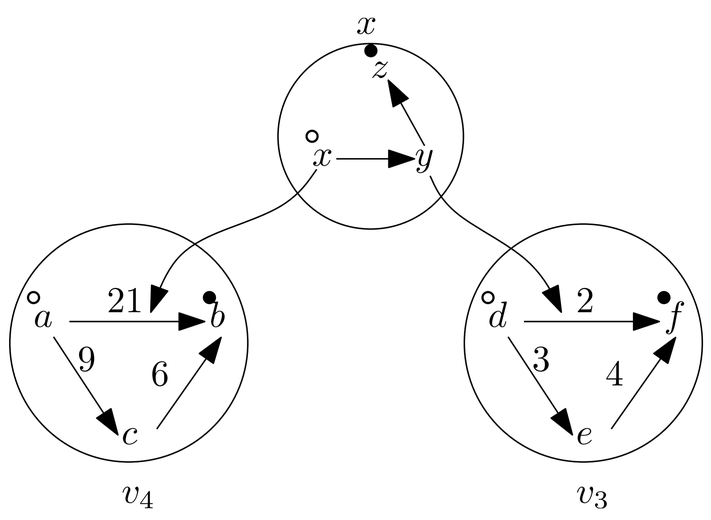

Figure 2: *The result of combining v_1 and v_2 in Figure 1. Note that the costs actions for the new variable v_4 is simply the sum of the corresponding actions for v_1 and v_2. Furthermore, the domain transition graph has the same shortest paths from and to the corresponding values for v_1 and v_2.*

The Algorithm

function POLYTREE($\Pi = (V, A, I, G, c)$)
  if $|V| \leq 2$ **then**
    return PRODUCT(Π, V)
  end if
  $x =$ CRITICALVERTEX(Π)
  $(V_o, \Pi_1) =$ COMBINEISOVARS(Π, OUT(x))
  $\mathcal{R}_x =$ RELEVANTACTIONS(Π_1, x)
  Replace A with $(A \setminus A_x) \cup \mathcal{R}_x$
  $(V_i, \Pi_2) =$ COMBINEISOVARS(Π_1, IN(x))
  $\Pi_3 =$ PRODUCT(Π_2, $\{x\} \cup V_o \cup V_i$)
  return POLYTREE(Π_3)
end function

Figure 3: *The algorithm devised in this paper, which outputs a new instance with a single variable of constant size.*

- Figure 3 is the algorithm in all its glory
- A critical vertex is a variable that is connected to at most one non-leaf in the causal graph
- We have not discussed relevant actions, which is a set of actions relevant for an optimal solution
- The computed relevant actions is bounded by a constant, and it necessary step to get an indegree bounded by a constant
- A variable can only be a critical vertex one, which means that the recursive depth is at most $|V|$. The steps in the algorithm can be performed in polynomial time thanks for the restrictions. Hence, the time complexity for the algorithm is polynomial
- The output of the algorithm is a new instance with a single variable, and we can simply use Dijkstra to get the cost of an optimal plan
- Use prefix search to generate an optimal plan

Discussion

- Polynomial-time algorithm for instances whose
  - Variables' domain size is bounded by a fixed constant
  - Causal graph is a polytree
  - Causal graph has a diameter bounded by a fixed constant
- The algorithm is based on identifying isomorpic variables, combining them and replacing variables with its product
- By also restricting the indegree on the causal graph the algorithm runs in at most time $O(\|\Pi\|^4)$, which is a lot better (Π is the instance and $\|\Pi\|$ is the size of it)

Inspirational Previous Work

[1] Carmel Domshlak and Yefim Dinitz. Multi-agent off-line coordination: Structure and complexity. In *Proceedings of 6th European Conference on Planning (ECP)*, pages 34–43, 2001.

[2] Omer Giménez and Anders Jonsson. The complexity of planning problems with simple causal graphs. *Journal of Artificial Intelligence Research (JAIR)*, pages 319–351, 2008.

[3] Omer Giménez and Anders Jonsson. Planning over chain causal graphs for variables with domains of size 5 is NP-hard. *Journal of Artificial Intelligence Research (JAIR)*, pages 675–706, 2009.

[4] Omer Giménez and Anders Jonsson. The influence of k-dependence on the complexity of planning. *Artificial Intelligence*, 177-179:25–45, 2012.

[5] Michael Katz and Carmel Domshlak. Structural-pattern databases. In *19th International Conference on Automated Planning & Scheduling (ICAPS)*, pages 186–193, 2009.

[6] Michael Katz and Emil Keyder. Structural patterns beyond forks: Extending the complexity boundaries of classical planning. In *Proceedings of 26th AAAI Conference on Artificial Intelligence*, pages 1779–1785, 2012.