Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

What's in It for My BDD? On Causal Graphs and Variable Orders in Planning

Peter Kissmann and **Jörg Hoffmann**
Saarland University
Saarbrücken, Germany
{kissmann, hoffmann}@cs.uni-saarland.de

Abstract

One decisive factor for the success of symbolic search using BDDs is whether or not the variable ordering is good. A general intuition is that smaller BDDs result if inter-dependent variables are close together. The most common means to capture variable dependencies in planning are causal graphs, and consequently previous work defined variable orders based on these. Starting from the observation that the two concepts of "dependency" are actually quite different, we introduce a framework for assessing the strength of variable ordering heuristics in sub-classes of planning. It turns out that causal graph based variable orders may be exponentially worse than optimal even for very simple planning tasks. Experiments with a broad range of such variable ordering variants indicate that they are mediocre at best.

Introduction

The variable ordering is a decisive factor for BDD-based planning. Roughly speaking, BDDs are small if "dependent" variables are scheduled close to each other. The planning literature contains a widely used notion to capture variable dependencies – causal graphs (Knoblock 1994; Domshlak and Dinitz 2001) – so the straightforward approach is to plug that concept into a BDD variable ordering heuristic. Indeed, that is the approach of Gamer (Kissmann and Edelkamp 2011), the state of the art planner of this kind.

"So, what is the problem?" we hear the reader asking. Our reply is that the use of the word "dependency" in the above deserves a second inspection. In the causal graph, it means that the variables appear in at least one common action, entailing that we cannot, in general, change the value of one variable without also changing the other. BDDs, on the other hand, represent Boolean functions φ. If many of the possible assignments to a subset P of variables directly entail the value of φ, independently of the assignment to all the other variables, then the variables P should be close to each other. In planning, φ will represent layers of states with equal distance from the initial state (forward search) or the goal (backward search). So the concept of "dependence" here is one of being able to quickly determine whether or not a state is a member of such a layer. *What, if anything, does this have to do with causal graph dependencies?*

We do not wish to claim that we close this question conclusively, but we contribute a number of insights suggesting that the overall answer is "not much". We introduce a simple formal framework for assessing the strength of variable ordering heuristics in sub-classes of planning. Applying this to causal graph based orders, it turns out that these may be exponentially worse than optimal even for very simple planning tasks. (For space restrictions, we only outline our proofs. Full proofs are available in a technical report (Kissmann and Hoffmann 2013).) We complement these findings by a large experiment with many variants of known variable ordering heuristics, showing that these work better than random orders, but *much* worse than off-the-shelf BDD reordering techniques.

Background

To minimize encoding size, it is essential for BDD-based planning to use a finite-domain variable representation. We therefore locate our investigation in that framework. A **finite-domain representation** (FDR) planning task is a tuple $\Pi = \langle V, A, I, G\rangle$, where V are the state variables each of which is associated with its finite domain $\mathcal{D}(v)$, A is a finite set of actions a each of which is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V, the initial state I is a complete assignment to V, and the goal is a partial assignment to V. To save space, we do not specify the (well-known) semantics of this construction. By $\mathcal{V}(pa)$, for a partial assignment pa, we denote the variables $v \in V$ where $pa(v)$ is defined.

Binary decision diagrams (BDDs) represent Boolean functions φ. A BDD β is a directed acyclic graph with one root and two terminal vertices, the 0-sink and the 1-sink. Each internal vertex corresponds to a binary variable p and has two successors, one taken if p is true and one taken if p is false. For any assignment to all variables p, the sink reached is the value of the function φ represented by β.

We consider BDD-based planning as implemented in Gamer. The finite-domain variables V of the FDR task are encoded by replacing each $v \in V$ with a binary **counter** $\gamma(v)$ using $log_2|\mathcal{D}(v)|$ bits. Search is forward and/or backward breadth-first. Each **layer** L of states during search – a subset of states with identical distance to the initial state (forward search) or the goal (backward search) – is then represented by a BDD for its characteristic function.

The BDDs are ordered, i.e., the ordering of the binary

variables on any path through β is fixed. The size of the BDD may vary exponentially as a function of this ordering, so it is crucial in practice to come up with good orderings. BDD packages come with dynamic reordering algorithms, but in planning their runtime overhead typically outweighs the benefit (Kissmann and Edelkamp 2011). Hence practical systems employ **variable ordering schemes**. We define these here as functions Ω mapping any planning task Π to a *non-empty set* $\Omega(\Pi)$ of **variable orderings**, i.e., orderings of the planning task's finite-domain variables V. We use sets, rather than unique $\Omega(\Pi)$, to capture the ambiguity inherent in the ordering schemes we are interested in here. $\Omega(\Pi)$ is computed in a pre-process, and one order $\langle v_1, \ldots, v_n \rangle = o \in \Omega(\Pi)$ is chosen arbitrarily (i.e., we do not consider that latter step here). To obtain the actual BDD binary variable order, we then simply replace each finite-domain variable v_i in o with its binary counter $\gamma(v_i)$. In other words, the BDD treats the counters $\gamma(v)$ like inseparable fixed blocks. (Since the counter bits are not represented at the level of the planning task Π, it would be impossible for Ω anyhow to make informed choices in such separations.)

Given this, for any layer L and ordering o of the planning task's finite-domain variables, the ordered BDD is unique. We denote its size (number of vertices) by $\mathsf{BDDSize}(o, L)$. By $\mathsf{BDDSize}^*(L) := \min_o \mathsf{BDDSize}(o, L)$, we denote the size of the BDD for an **optimal variable ordering**; finding such an ordering is computationally hard (Bryant 1986).

The state of the art ordering scheme is based on the **causal graph** CG_Π of the planning task. CG_Π is a directed graph with vertices V, and an arc (v, v') iff $v \neq v'$ and there exists an action $a \in A$ such that $(v, v') \in \mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a)) \times \mathcal{V}(\mathsf{eff}(a))$. Gamer's scheme, denoted Ω^{ga}, maps Π to the set of orderings $o = \langle v_1, \ldots, v_n \rangle$ that minimize the expression $\sum_{(v_i, v_j) \in CG_\Pi} (i - j)^2$. The underlying intuition is that adjacent variables are dependent, and should be scheduled close to each other. In practice, Gamer approximates Ω^{ga} by a limited amount of local search in the space of orderings.

Apart from Ω^{ga}, we also consider the scheme Ω^{cg}, which is defined only if CG_Π is acyclic, and in that case maps Π to the set of topological orderings of the vertices in CG_Π. We consider this to be of theoretical interest since it is the straightforward way to "trust the causal graph completely".

What's in a Causal Graph: Theory

As discussed, it is doubtful whether the concept of "dependency" in the causal graph has any real relation with the concept of "dependency" relevant to BDD size. We now frame this doubt in terms of a classification of the guarantees offered, or rather, the guarantees *not* offered, by Ω^{ga} and Ω^{cg} in restricted classes of planning tasks.

Definition 1 (Classification of Ordering Schemes). *Let $\mathcal{F} = \{\Pi_n\}$ be an infinite family of* FDR *planning tasks parameterized by n, where the size of Π_n is bounded by a polynomial in n. Let $d \in \{forward, backward\}$ be a search direction. A variable ordering scheme Ω is:*

(i) **perfect** *in $\mathcal{F}$ for d if for all $\Pi_n \in \mathcal{F}$, all d-layers L in Π_n, and all $o \in \Omega(\Pi_n)$, we have $\mathsf{BDDSize}(o, L) = \mathsf{BDDSize}^*(L)$.*

(ii) **safe** *in $\mathcal{F}$ for d if there exists a polynomial p s.t. for all $\Pi_n \in \mathcal{F}$, all d-layers L in Π_n, and all $o \in \Omega(\Pi_n)$, we have $\mathsf{BDDSize}(o, L) \leq p(\mathsf{BDDSize}^*(L))$.*

(iii) **viable** *in $\mathcal{F}$ for d if there exists a polynomial p s.t. for all $\Pi_n \in \mathcal{F}$ and all d-layers L in Π_n, there exists $o \in \Omega(\Pi_n)$ with $\mathsf{BDDSize}(o, L) \leq p(\mathsf{BDDSize}^*(L))$.*

Perfect Ω guarantees to deliver perfect orderings, safe Ω guarantees polynomial overhead, viable Ω always delivers at least one good ordering but runs the risk of super-polynomial overhead. If Ω is not viable, then all its orderings are super-polynomially bad in some task and layer.

We extend our classification to arbitrary sub-classes $\mathcal{C}$ of FDR by the worst case over all families $\mathcal{F}$ contained in $\mathcal{C}$: *if $\mathcal{C}$ contains at least one $\mathcal{F}$ where Ω is not perfect, then Ω is said to be not perfect in $\mathcal{C}$, and so forth.*

As we are interested in variable orderings derived from the causal graph, it is natural to consider sub-classes of FDR characterized by their causal graphs. For a set of directed graphs $\mathcal{G}$, by $\mathsf{FDR}(\mathcal{G})$ we denote the class of FDR planning tasks whose causal graphs are elements of $\mathcal{G}$. We investigate widely considered causal graph special cases, namely chains $\mathcal{G}^{\text{chain}}$, forks $\mathcal{G}^{\text{fork}}$, inverted forks $\mathcal{G}^{\text{ifork}}$, and DAGs $\mathcal{G}^{\text{dag}}$ (directed acyclic graphs). As simple limiting cases, we consider causal graphs $\mathcal{G}^{\emptyset}$ without any arcs, and unrestricted causal graphs $\mathcal{G}^{\forall}$. Figure 1 illustrates.

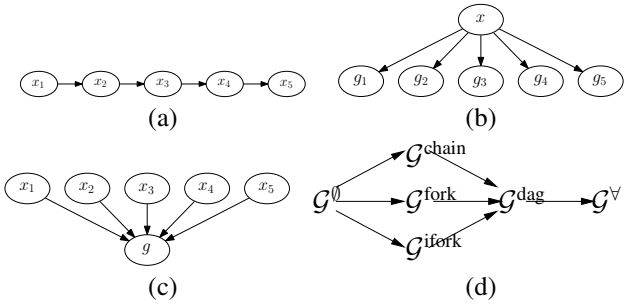

Figure 1: Causal graph special cases (a) chains, (b) forks, (c) inverted forks, and (d) their relation (arrows mean $\subseteq$).

Bad cases are inherited in the hierarchy of Figure 1 (d): if $\mathcal{G} \subseteq \mathcal{G}'$, then for any ordering scheme the classification within $\mathsf{FDR}(\mathcal{G}')$ is at least as bad as that in $\mathsf{FDR}(\mathcal{G})$.[1] We start our investigation with empty causal graphs:

Theorem 1. *For both search directions, any ordering scheme is safe in $\mathsf{FDR}(\mathcal{G}^{\emptyset})$. Ω^{ga} and Ω^{cg} are not perfect.*

If the causal graph has no arcs, then all variables move independently. So any forward/backward layer with distance d contains exactly the states in which the sum of individual distances (from a variable's initial value/to a variable's goal value) equals d. For any binary counter $\gamma(v)$ in the BDD, the number of vertices needed is bounded by the number of possible individual-distance sums of the variables preceding v (intuitively, that's all we need to remember, to correctly evaluate the characteristic function). Thus BDD size is polyno-

[1] An interesting side remark is that, given a task Π, we can always create a task Π' with arbitrarily complex causal graph without affecting the classification of Ω^{cg} and Ω^{ga}: we add a separate part to Π, with a complex causal graph, but with no influence on the layers L, and increasing BDD size constantly under Ω^{cg} and Ω^{ga}.

mially bounded for any variable ordering. It is easy to find examples where different orderings result in BDDs of different sizes. Ω^{ga} and Ω^{cg} each return the set of all orders. Putting these facts together, the claim follows.

Note that Theorem 1 is a "good case", not for the schemes Ω^{ga} and Ω^{cg} – these are devoid of information – but for the ability of causal graphs to entail *anything* for BDD orderings. Empty causal graphs entail that all orderings are safe. That connection doesn't carry any further than this trivial case, though: in all other sub-classes considered, the space of BDD orderings contains exponentially bad ones.

The news regarding the informedness of Ω^{ga} and Ω^{cg} is almost universally very bad, with a little bit of hope only for chain causal graphs. Let us give you the bad news first:

Theorem 2. *For both search directions, Ω^{ga} and Ω^{cg} are not safe in* $\mathsf{FDR}(\mathcal{G}^{\text{ifork}})$.

Our negative results employ Boolean functions in **quadratic form**. These have the variables $\{x_1, y_1, \ldots, x_n, y_n\}$, and take the form $(x_1 \mathsf{op}^{\text{low}} y_1) \mathsf{op}^{\text{hi}} \ldots \mathsf{op}^{\text{hi}} (x_n \mathsf{op}^{\text{low}} y_n)$, where either $\mathsf{op}^{\text{hi}} \in \{\vee, \oplus\}$ and $\mathsf{op}^{\text{low}} = \wedge$, or vice versa. We denote these functions by $Q(\mathsf{op}^{\text{hi}}, \mathsf{op}^{\text{low}})$. For each of these functions, the ordering $\langle x_1, y_1, \ldots, x_n, y_n \rangle$ yields a BDD whose size is polynomial in n, while the ordering $\langle x_1, \ldots, x_n, y_1, \ldots, y_n \rangle$ yields an exponential-size BDD. (Wegener (2000) proves this for $Q(\vee, \wedge)$; similar arguments apply to the other quadratic forms.)

To prove Theorem 2, consider now the function $Q(\vee, \wedge) = \bigvee_{i=1}^{n} (x_i \wedge y_i)$. We design an FDR task Π_n. All variables are Boolean. We use $\{x_1, y_1, \ldots, x_n, y_n\}$ plus a new variable g that the goal requires to be true. There are n actions achieving g, each of which requires x_i and y_i to be true as the precondition. Then the first backward layer is characterized by $\neg g \wedge \bigvee_{i=1}^{n} (x_i \wedge y_i)$, and it is easy to see that $\Omega^{\text{ga}}(\Pi_n)$ and $\Omega^{\text{cg}}(\Pi_n)$ each contain some orders that are exponentially bad. Clearly, $\Pi_n \in \mathsf{FDR}(\mathcal{G}^{\text{ifork}})$, which proves the claim of Theorem 2 for backward search.

For forward search, we consider the same function $Q(\vee, \wedge)$, and construct Π_n, which has the same variables $\{g, x_1, y_1, \ldots, x_n, y_n\}$ but where the domains of $\{x_1, y_1, \ldots, x_n, y_n\}$ are ternary: unknown, true, false. All x_i and y_i are initially unknown, and can be set to either true or false. There are n actions achieving g, exactly as above. Then the states with initial state distance $2n + 1$ are exactly those that satisfy $g \wedge Q(\vee, \wedge)$. It is not difficult to verify that this shows the claim as before.

Theorem 3. *For both search directions, Ω^{ga} and Ω^{cg} are not safe in* $\mathsf{FDR}(\mathcal{G}^{\text{fork}})$.

For both search directions, we use the same function $Q(\wedge, \oplus) = \bigwedge_{i=1}^{n} (x_i \oplus y_i)$, and the same Π_n with Boolean variables $\{x_1, y_1, \ldots, x_n, y_n\}$ plus a new variable z with domain $\{d_1, dx_1, dy_1, d_2, dx_2, dy_2, \ldots, d_n, dx_n, dy_n, d_{n+1}\}$. The actions are such that, for $1 \leq i \leq n$, z can move from d_i to either dx_i or dy_i, and from each of these to d_{i+1}. An action preconditioned on dx_i achieves x_i; same for dy_i and y_i. Initially $z = d_1$ and all x_i, y_i are false; the goal requires that $z = d_{n+1}$ and all x_i, y_i are true. The states with initial state distance $3n$ are exactly those where $z = d_{n+1}$ and $Q(\wedge, \oplus)$ is true, and the states with goal state distance $3n$ are exactly those where $z = d_1$ and $Q(\wedge, \oplus)$ is true. Since neither $\Omega^{\text{ga}}(\Pi_n)$ nor $\Omega^{\text{cg}}(\Pi_n)$ constrain the ordering of the variables $\{x_1, y_1, \ldots, x_n, y_n\}$, the claim follows as before.

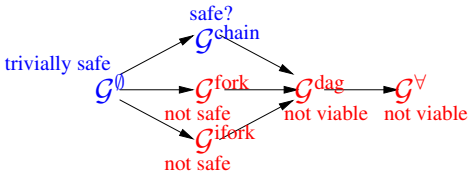

Figure 2: Overview of our classification results. These hold for each of Ω^{ga} and Ω^{cg}, and for each search direction.

For each of $\mathsf{FDR}(\mathcal{G}^{\text{ifork}})$ and $\mathsf{FDR}(\mathcal{G}^{\text{fork}})$, it is an open question whether $\Omega^{\text{ga}}(\Pi_n)$ and $\Omega^{\text{cg}}(\Pi_n)$ are viable. For DAG causal graphs, that question is closed:

Theorem 4. *For both search directions, Ω^{ga} and Ω^{cg} are not viable in* $\mathsf{FDR}(\mathcal{G}^{\text{dag}})$.

Corollary 1. *Ω^{ga} is not viable in* $\mathsf{FDR}(\mathcal{G}^{\forall})$.

The proof modifies the constructions underlying Theorem 2 to include additional causal graph arcs, forcing Ω^{ga} and Ω^{cg} to yield bad orderings grouping $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ into separate blocks. We arrange the root variables of the inverted forks as $x_1, \ldots, x_n, y_1, \ldots, y_n$, and make the actions setting a variable dependent on its left-hand side neighbor. That is easy to do without interfering with the required properties. This proves the claim of Theorem 4. Corollary 1 follows immediately (recall that Ω^{cg} is defined only for acyclic causal graphs).

We close our investigation with the only somewhat positive case, chain causal graphs:

Theorem 5. *For both search directions, Ω^{ga} and Ω^{cg} are not perfect in* $\mathsf{FDR}(\mathcal{G}^{\text{chain}})$. *There exists an ordering scheme that is not viable.*

Cases where Ω^{ga} and Ω^{cg} aren't perfect are inherited from $\mathsf{FDR}(\mathcal{G}^{\emptyset})$: we can enforce causal graph arcs that are irrelevant to the initial state distance or the goal distance. To obtain a non-viable ordering scheme in forward and backward search, we employ the quadratic form functions $Q(\wedge, \vee)$ and $Q(\vee, \wedge)$, respectively. In the chain causal graphs, x_i and y_i are neighbors, rendering Ω^{ga} and Ω^{cg} safe, whereas a non-viable ordering separates $x_1, \ldots, x_n$ from $y_1, \ldots, y_n$.

The two planning task families just described constitute our only truly positive result: there, the ordering information in the causal graph keeps us from making exponentially bad mistakes. That positive message would be much stronger if it pertained to the entire planning sub-class $\mathsf{FDR}(\mathcal{G}^{\text{chain}})$, i.e., if Ω^{ga} and Ω^{cg} were safe for all families of tasks with chain causal graphs. It remains an open question whether this is true; we conjecture that it is.

Figure 2 overviews our results. The evidence speaks against a strong connection between causal graph dependencies, and dependencies as relevant for BDD size. Note: The DAG causal graph underlying Theorem 4 has a very simple form combining a chain with an inverted fork, and Theorem 2 relies on planning tasks that fall into a known tractable class for *optimal* planning (Katz and Domshlak 2010).

What's in a Causal Graph: Practice

Poor performance in the worst case does not entail poor performance in practice. To get a picture of where causal-graph based variable ordering schemes stand, we run 11 variants thereof, and compare them to practical "good"/"bad" delimiters. As the "bad" delimiter, we use random orderings. As the "good" delimiter, we use the off-the-shelf dynamic reordering algorithm of Gamer's BDD package CUDD, which is based on sifting (Rudell 1993). For better comparability with our ordering schemes, we restrict the algorithm to not separate the $\gamma(v)$ blocks. (As previously indicated, the algorithm consumes too much runtime to be cost-effective; here, we give it ample runtime, considering only BDD size.)

We use the IPC'11 benchmarks, and use Gamer as the base implementation for all planners. We run Gamer's original ordering scheme, denoted *Gamer*, that approximates Ω^{ga}. We run 5 other schemes based directly on the causal graph: *GamerPre* which is like *Gamer* but on an enriched causal graph also featuring arcs between pairs of precondition variables; *WGamer* and *WGamerPre* which are like *Gamer* and *GamerPre* but with arcs weighted by the number of relevant actions; Fast Downward's (Helmert 2006) level heuristic, denoted *CGLevel*, which approximates Ω^{cg}; and *CGSons*, another approximation of Ω^{cg}, that always selects a variable v all of whose parents have already been selected, or at least one of whose parents has already been selected, or an arbitrary variable if no such v exists. Further, we run 5 ordering schemes we adopted from the model checking literature, based on a structure called the *abstract syntax tree* (basically listing all actions and variables they touch). We do not have the space to describe these schemes; using their first authors' names for reference, we call them *Butler* (Butler et al. 1991), *Chung* (Chung, Hajj, and Patel 1993), *Malik* (Malik et al. 1988), *Maisonneuve* (Maisonneuve 2009), and *Minato* (Minato, Ishiura, and Yajima 1990).

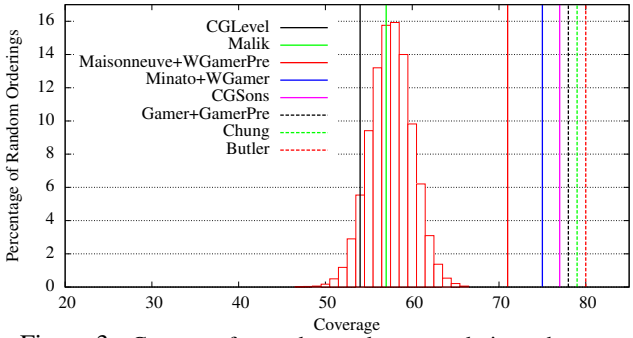

Figure 3: Coverage for random orders vs. ordering schemes.

First we compared the schemes against "random orderings", where each of these corresponds to one run of all IPC'11 benchmarks, using a random variable ordering for each instance. We performed 5000 of these runs; the time-out is one minute to make this feasible. Figure 3 shows coverage, i.e., number of found solutions, on the x axis, and the fraction of random orderings having that coverage on the y axis. The coverage achieved by each of our 11 ordering schemes is shown as vertical lines (scheme names in the figure are ordered top-to-bottom from worst to best coverage).

Malik and *CGLevel* lie in respectively below (!) the middle of the Gaussian distribution, so are quite bad indeed. Matters are not as bad for the other 9 ordering schemes, which are close together.[2] Compared to a best-of over the random orders, all the ordering schemes appear rather humble. Let x be the number of instances solved by a scheme but not by any random order, and y the number not solved by a scheme but solved by some random order. Then $x < y$ for all but 2 cases (where $x - y = 1$ and $x = y$, respectively), and the average over x is 3.00 while that over y is 9.73.

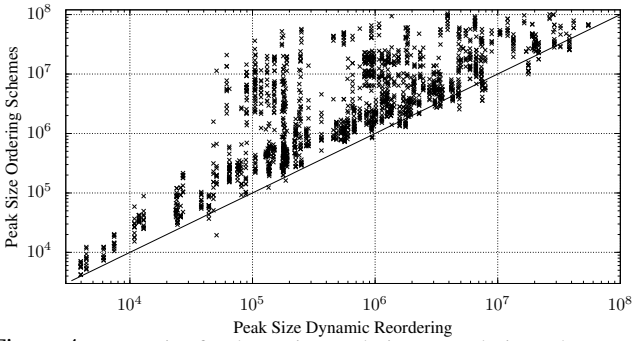

Figure 4: BDD size for dynamic reordering vs. ordering schemes.

Figure 4 contains one data point for every pair (I, Ω) of IPC'11 benchmark instance I and ordering scheme Ω that were solved by both (a) Gamer using dynamic reordering starting from an arbitrary variable order (the one returned by Gamer's grounding process), and (b) Gamer using ordering scheme Ω (without dynamic reordering). The time-out is 6 hours for (a), and 30 minutes for (b). Data point (X, Y) is the size of the largest BDD constructed for I by ((a),(b)). The time-out is larger for dynamic reordering because such reordering is not runtime effective: The question we are asking here is merely which of the two methods yields smaller BDDs. Figure 4 shows that dynamic reordering is universally much better at this.

Conclusion

The proposed theoretical framework suggests that causal graphs are not a good source of information for ordering BDD variables. Even though one may not, in general, *expect* the ordering heuristic to not be exponentially bad in the worst case, some of our results are quite striking. For inverted fork causal graphs, in particular, there are exponentially bad orderings in planning tasks so restricted as to be tractable for domain-independent optimal planning. Our empirical results corroborate this view, Fast Downward's level heuristic being worse than random, and all ordering schemes lagging far behind off-the-shelf reordering.

We do not wish to claim that the present results provide a conclusive answer to the question we started out with. Rather, we view this research as a first step towards a systematic investigation of BDD variable orderings in planning, which we hope will inspire other researchers as well.

[2] With a 30 minute time-out, we obtain a similar picture, *Malik* and *CGLevel* lagging behind while all others are close.

Acknowledgments

We thank the anonymous reviewers, whose comments helped to improve the paper.

References

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* 35(8):677–691.

Butler, K. M.; Ross, D. E.; Kapur, R.; and Mercer, M. R. 1991. Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams. In *Proceedings of the 28th Conference on Design Automation (DAC-91)*, 417–420. San Francisco, CA, USA: ACM.

Chung, P.-Y.; Hajj, I. N.; and Patel, J. H. 1993. Efficient variable ordering heuristics for shared ROBDD. In *Proceedings of the 1993 IEEE International Symposium on Circuits and Systems (ISCAS-93)*, 1690–1693. Chicago, IL, USA: IEEE.

Domshlak, C., and Dinitz, Y. 2001. Multi-agent offline coordination: Structure and complexity. In Cesta, A., and Borrajo, D., eds., *Recent Advances in AI Planning. 6th European Conference on Planning (ECP-01)*, Lecture Notes in Artificial Intelligence, 34–43. Toledo, Spain: Springer-Verlag.

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

Katz, M., and Domshlak, C. 2010. Implicit abstraction heuristics. *Journal of Artificial Intelligence Research* 39:51–126.

Kissmann, P., and Edelkamp, S. 2011. Improving cost-optimal domain-independent symbolic planning. In Burgard, W., and Roth, D., eds., *Proceedings of the 25th National Conference of the American Association for Artificial Intelligence (AAAI-11)*, 992–997. San Francisco, CA, USA: AAAI Press.

Kissmann, P., and Hoffmann, J. 2013. Whats in it for my BDD? On causal graphs and variable orders in planning. Technical Report A 01/2013, Saarland University, Saarbrücken, Germany.

Knoblock, C. 1994. Automatically generating abstractions for planning. *Artificial Intelligence* 68(2):243–302.

Maisonneuve, V. 2009. Automatic heuristic-based generation of MTBDD variable orderings for PRISM models. Internship report, Oxford University Computing Laboratory.

Malik, S.; Wang, A.; Brayton, R.; and Sangiovanni-Vincentelli, A. 1988. Logic verification using binary decision diagrams in a logic synthesis environment. In *Proceedings of the 1988 International Conference on Computer-Aided Design (ICCAD-98)*, 6–9. IEEE Computer Society Press.

Minato, S.; Ishiura, N.; and Yajima, S. 1990. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. In *Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC-90)*, 52–57. Orlando, FL, USA: IEEE Computer Society Press.

Rudell, R. 1993. Dynamic variable ordering for ordered binary decision diagrams. In Lightner, M. R., and Jess, J. A. G., eds., *Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design (ICCAD-93)*, 42–47. Santa Clara, CA, USA: IEEE Computer Society.

Wegener, I. 2000. *Branching Programs and Binary Decision Diagrams*. SIAM.