Artificial Intelligence

16. Causal Graphs

How to Capture the Problem *Structure*

Prof Sara Bernardini
bernardini@diag.uniroma1.it
www.sara-bernardini.com

SAPIENZA UNIVERSITÀ DI ROMA

Autumn Term

Agenda

1. Introduction
2. Causal Graphs
3. Domain Transition Graphs
4. Example Results
5. Conclusion

Motivation

What is general problem solving all about?

→ Writing a program that is not specialized to a particular problem.

How can such a program be effective?

→ By self-adapting to any particular problem it is given as input.

What does the program require to be able to self-adapt?

→ Understand and exploit the structure of the problem.

But then, what should we – as researchers – do, first of all?

→ Figure out what the "problem structure" is.

→ Causal graphs capture the structure of the planning task input, in terms of the direct **dependencies between state variables**. This can be exploited for many different purposes.

Example Uses of Causal Graphs

- Identifying a sub-class of planning tasks where generating a partial delete relaxation heuristic is tractable.
- Avoiding redundant work in the search for a pattern collection when generating a pattern database heuristic.
- Search space surface analysis. Identifying a sub-class of planning tasks where h^+ provably has no local minima [Hoffmann (2011)].
- Complexity analysis: [Domshlak and Dinitz (2001); Brafman and Domshlak (2003); Katz and Domshlak (2008); Giménez and Jonsson (2009); Chen and Giménez (2010); Katz and Keyder (2012)].
- Designing and generating (yet more) heuristic functions: [Helmert (2004); Katz and Domshlak (2010); Domshlak *et al.* (2015)].
- System design: Guaranteeing desired behavior [Williams and Nayak (1997)].
- Factorized planning: Problem decomposition [Knoblock (1994); Brafman and Domshlak (2013); Gnad and Hoffmann (2015)].

Our Agenda for This Chapter

2. **Causal Graphs:** For explicitly capturing the "internal structure" of a planning task, causal graphs are by far the most prominent notion in the planning literature.
3. **Domain Transition Graphs:** These are simple graphs describing the behavior of individual state variables; they are often considered in connection with causal graphs.
4. **Example Results:** We show some examples of causal graph based analyses, which are easy to explain.

Causal Graphs

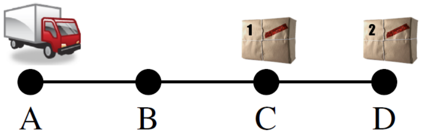
Definition (Causal Graph). *Let $\Pi = (V, A, c, I, G)$ be an FDR planning task. The causal graph of Π is the directed graph $CG(\Pi)$ with vertices V and an arc (u, v) if $u \neq v$ and there exists an action $a \in A$ so that either*

1. *$pre_a(u)$ and $eff_a(v)$ are both defined; or*
2. *$eff_a(u)$ and $eff_a(v)$ are both defined.*

Causal graphs capture variable dependencies:

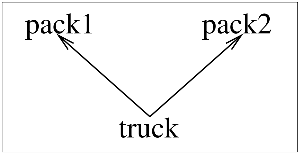
- Arc (1) (u, v): we may have to change u to be able to change v.
- Arc (2) (u, v): changing u may, as a side effect, change v as well.
  $\rightarrow$ Note that we also get the arc (v, u) in this situation, constituting a cycle between u and v.

Example: “Logistics”

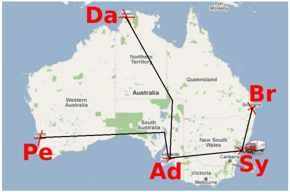


- State variables V: $truck : \{A, B, C, D\}$; $pack1, pack2 : \{A, B, C, D, T\}$.
- Initial state I: $truck = A$, $pack1 = C$, $pack2 = D$.
- Goal G: $truck = A$, $pack1 = D$.
- Actions A (unit costs): $drive(x, y)$, $load(p, x)$, $unload(p, x)$. E.g.: $load(pack1, x)$ precondition $truck = x, pack1 = x$, effect $pack1 = T$.

Causal graph?

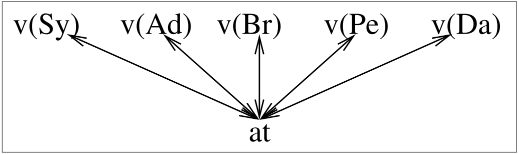


Example: "TSP"



- Variables V: $at : \{Sy, Ad, Br, Pe, Da\}$; $v(x) : \{T, F\}$ for $x \in \{Sy, Ad, Br, Pe, Da\}$.
- Initial state I: $at = Sy, v(Sy) = T, v(x) = F$ for $x \neq Sy$.
- Goal G: $at = Sy, v(x) = T$ for all x.
- Actions A: $drive(x, y)$ where x, y have a road; pre $at = x$, eff $at = y, v(y)$.
- Cost function c: $Sy \leftrightarrow Br : 1$, $Sy \leftrightarrow Ad : 1.5$, $Ad \leftrightarrow Pe : 3.5$, $Ad \leftrightarrow Da : 4$.

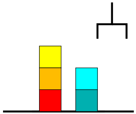
Causal graph?



Causal Graphs Cycles: Class (2) Effect-Effect

Abstract example: If $V = \{u, v\}$ and $A = \{a\}$ with $\mathit{eff}_a = \{u = d, v = e\}$, the causal graph has arcs (u, v) and (v, u).

Less abstract example: Blocksworld.

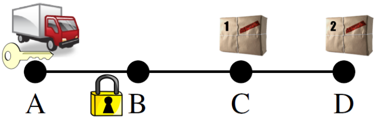


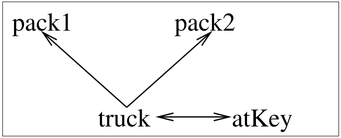
- n blocks, 1 hand.
- A single action either takes a block with the hand or puts a block we're holding onto some other block/the table.
- For example, say $\mathit{pickup}(x, y)$ has precondition $\mathit{atx} = y$, $\mathit{clearx} = \mathit{true}$, $\mathit{handEmpty} = \mathit{true}$; and effect $\mathit{atx} = \mathit{hand}$, $\mathit{cleary} = \mathit{true}$, $\mathit{handEmpty} = \mathit{false}$.

$\rightarrow$ So there are class (2) cycles in the Blocksworld, for example between variables of the form "atx" and "cleary".

$\rightarrow$ Class (2) (effect-effect) causal graph cycles occur whenever an action has more than one effect. Their absence is equivalent to "unary" actions, each affecting only a single variable.

Causal Graphs Cycles: Class (1) Precondition-Effect





- State variables V: $truck$, $pack1$, $pack2$ as before; $atKey : \{A, B, C, D, T\}$.
- Initial state I: $truck = A$, $pack1 = C$, $pack2 = D$, $atKey = A$.
- Goal G: $truck = A$, $pack1 = D$.
- Actions A: As before; and $takeKey(x)$ with pre $truck = x, atKey = x$, effect $atKey = T$; and $drive(A, B)$ has additional pre $atKey = T$.

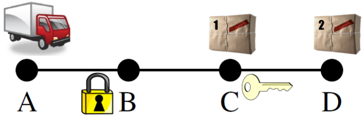
$\rightarrow$ Are there class (1) cycles in this example? Yes, between variables $truck$ and $atKey$.

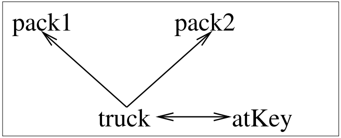
$\rightarrow$ Class (1) (precondition-effect) causal graph cycles occur when there are "cyclic support dependencies", where moving variable x requires a precondition on y which (transitively) requires a precondition on x.

Where Causal Graphs Fail

→ Does $CG(\Pi)$ depend on either of I or G? No, $CG(\Pi)$ remains the same whichever I and G we choose.

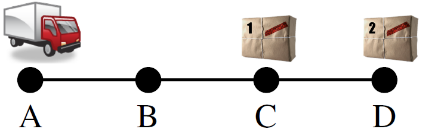
→ This is a main weakness of causal graphs! They capture only the structure of the variables and actions, and can by design not account for the influence of different initial states and goals.





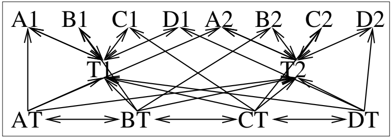
- State variables V: $truck$, $pack1$, $pack2$ as before; $atKey : \{A, B, C, D, T\}$.
- Initial state I: $truck = A$, $pack1 = C$, $pack2 = D$, $atKey = C$.
- Goal G: $truck = A$, $pack1 = D$.
- Actions A: As before; and $takeKey(x)$ with pre $truck = x$, $atKey = x$, effect $atKey = T$; and $drive(A, B)$ has additional pre $atKey = T$.

Why not in STRIPS?



- Facts P: $truck(x)\ x \in \{A, B, C, D\}$; $pack1(x), pack2(x)\ x \in \{A, B, C, D, T\}$.
- Initial state I: $\{truck(A),\ pack1(C),\ pack2(D)\}$.
- goal G: $\{truck(A),\ pack1(D)\}$.
- Actions A (unit costs): $drive(x, y)$, $load(p, x)$, $unload(p, x)$. E.g.: $load(pack1, x)$ pre $truck(x), pack1(x)$, add $pack1(T)$, del $pack1(x)$.

Causal graph?



→ Reminder **Chapter 14**: "Causal graphs have a much clearer structure for FDR (e.g., acyclic vs. cyclic)."

Domain Transition Graphs

Definition (Domain Transition Graph). *Let $\Pi = (V, A, c, I, G)$ be an FDR planning task, and let $v \in V$. The domain transition graph (DTG) of v is the labeled directed graph $DTG(v, \Pi)$ with vertices D_v and an arc (d, d') labeled with action $a \in A$ whenever either (i) $pre_a(v) = d$ and $eff_a(v) = d'$, or (ii) $pre_a(v)$ is not defined and $eff_a(v) = d'$.*
We refer to (d, d') as a value transition of v. We write $d \xrightarrow{a}_\varphi d'$ where $\varphi = pre_a \setminus \{v = d\}$ is the (outside) condition. Where not relevant, we omit "a" and/or "φ".

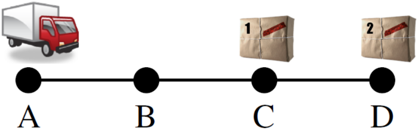
→ DTG captures "where a variable can go and how".

→ Attention: "value transition $d \xrightarrow{a}_\varphi d'$" $\neq$ "state transition $s \rightarrow s'$". (Value transition focuses on v, state transition encompasses all variables.)

Definition (Invertible Value Transition). *Let $\Pi = (V, A, c, I, G)$ be an FDR planning task, let $v \in V$, and let $d \rightarrow_\varphi d'$ be a value transition of v. We say that $d \rightarrow_\varphi d'$ is invertible if there exists a value transition $d' \rightarrow_{\varphi'} d$ where $\varphi' \subseteq \varphi$.*

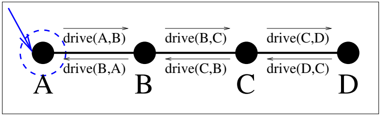
→ DTG captures whether "we can go back".

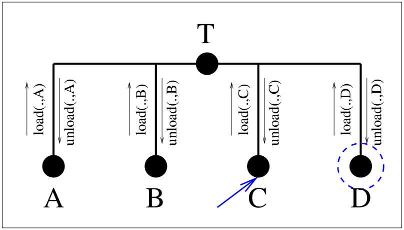
Example: "Logistics"



- State variables V: $truck : \{A, B, C, D\}$; $pack1, pack2 : \{A, B, C, D, T\}$.
- Actions A: $drive(x, y)$, $load(p, x)$, $unload(p, x)$. (Unit costs.)
- Initial state I: $truck = A$, $pack1 = C$, $pack2 = D$.
- goal G: $truck = A$, $pack1 = D$.

DTGs?

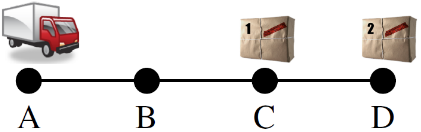




→ Are these value transitions invertible? Yes.

→ Example of non-invertible? One-way street, e.g. no $drive(B, A)$.

Why not in STRIPS?

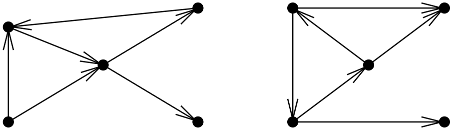


- Facts P: $\mathit{truck}(x)\; x \in \{A, B, C, D\}$; $\mathit{pack1}(x), \mathit{pack2}(x)\; x \in \{A, B, C, D, T\}$.
- Actions A: $\mathit{drive}(x, y)$, $\mathit{load}(p, x)$, $\mathit{unload}(p, x)$. (Unit costs.)
- Initial state I: $\{\mathit{truck}(A), \mathit{pack1}(C), \mathit{pack2}(D)\}$.
- goal G: $\{\mathit{truck}(A), \mathit{pack1}(D)\}$.

DTGs? This'll be "true $\leftrightarrow$ false" for each of the 14 variables ...

$\rightarrow$ DTGs capture the "travel routes" of individual variables. For domain size 2, these routes hardly capture any interesting structure.

Task Decomposition: Unconnected Sub-Tasks



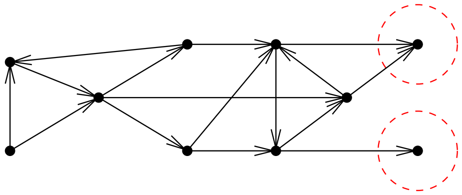
→ Unconnected parts of the task can be solved separately:

Lemma. *Let $\Pi = (V, A, c, I, G)$ be an FDR planning task, and let V_1, V_2 be a partition of V such that $CG(\Pi)$ contains no arc between the two sets. Let Π_i, for $i \in \{1, 2\}$, be identical to Π except that we use variables V_i, restrict I and G to V_i, and remove all actions a where either pre_a or eff_a is defined on a variable outside V_i. Then, for any pair $\vec{a}_1$ and $\vec{a}_2$ of (optimal) plans for Π_1 and Π_2, $\vec{a}_1 \circ \vec{a}_2$ is an (optimal) plan for Π.*

Proof Intuition: Since $CG(\Pi)$ contains no arc between V_1 and V_2, every action either touches only variables from V_1, or touches only variables from V_2.

Hence any plan for Π can be separated into independent sub-sequences touching V_1 respectively V_2, corresponding to plans for Π_1 respectively Π_2.

Task Simplification: Non-Goal Leaf Variables



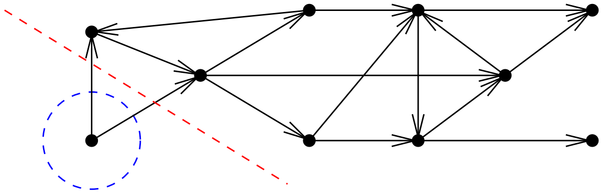
Question!

How can we simplify Π if there is a leaf vertex v on which $G(v)$ is undefined?

$\rightarrow$ Since v is a leaf in $CG(\Pi)$, the actions that do affect v affect no other variables, and the actions that do not affect v do not have preconditions on v. So v is a "client": it moves only for its own purpose.

But if v has no own goal, then it has no "purpose". Thus, denoting by Π' the modified task where v has been removed, any (optimal) plan for Π' is an (optimal) plan for Π.

Task Simplification: Invertible Root Variables



→ Root variables with invertible & connected DTGs can be handled separately:

1. Remove v from Π to obtain Π'; find plan $\vec{a}$ for Π'.
2. Extend $\vec{a}$ with move sequence for v that achieves all preconditions on v as needed, then moves to $v's$ own goal (if any) at the end.

→ Intuition: v is a "servant". Thanks to its connected and invertible DTG, it can always go wherever it is needed.

→ Does this hold for optimal planning? No. The optimal plan for Π' ignores the cost of moving v so may incur unnecessarily high costs on v.

Task Simplification: Invertible Root Variables, ctd.

Lemma. *Let $\Pi = (V, A, c, I, G)$ be an FDR planning task, and let $v \in V$ be a root vertex in $CG(\Pi)$ such that $DTG(v, \Pi)$ is connected and all value transitions of v are invertible. Let Π' be identical to Π except that we remove v, restrict I and G to $V \setminus \{v\}$, remove any assignment to v from all action preconditions, and remove all actions a where $\mathit{eff}_a(v)$ is defined. Then, from any plan $\vec{a}$ for Π', a plan for Π can be obtained in time polynomial in $|\Pi|$ and $|\vec{a}|$.*

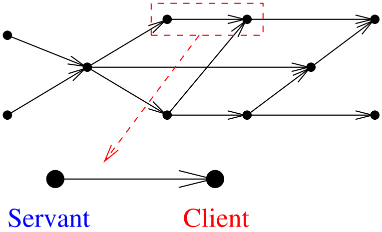
Proof Intuition: Since v is a root in $CG(\Pi)$, the actions that affect v affect no other variables, and have no preconditions on any variables other than v. In other words, "moving v has no side effects, and does not need any outside help".

Since $DTG(v, \Pi)$ is connected and all value transitions of v are invertible, $DTG(v, \Pi)$ is strongly connected i.e. from any value d of v we can reach any other value d' of v. Hence "v can always move to any value desired".

These two things together imply that, given a plan $\vec{a}$ for Π', we can insert a suitable move sequence for v into $\vec{a}$ to obtain a plan for Π.

Complexity: Acyclic + Invertible

Servants + clients, now in full:

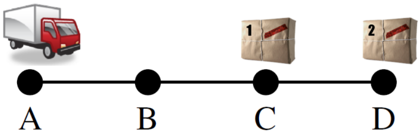


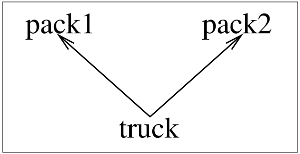
→ A plan can be constructed in polynomial time:

1. Order the variables topologically $v_1, \dots, v_n$ from "servants" to "clients".
2. Iteratively apply step 1 on slide 22 to $v_1, \dots, v_n$ in this order. Then Π' is empty, and the empty plan $\vec{a} := \langle\rangle$ solves it.
3. Iteratively apply step 2 on slide 22 to $v_n, \dots, v_1$ in this order.

→ Intuition: Iteratively deal with clients, then insert needed moves for servants.

The Plan Construction in "Logistics"





- Initial state I: $truck = A$, $pack1 = C$, $pack2 = D$.
- goal G: $truck = A$, $pack1 = D$, $pack2 = A$.

$\rightarrow$ Topological order: $v_1 = truck$, $v_2 = pack2$, $v_3 = pack1$.

$\rightarrow$ Targets for $pack1$: D [Goal].

$\rightarrow$ Path for $pack1$: $C \xrightarrow{load(pack1,C)} T \xrightarrow{unload(pack1,D)} D$.

$\rightarrow$ Targets for $pack2$: A [Goal].

$\rightarrow$ Path for $pack2$: $D \xrightarrow{load(pack2,D)} T \xrightarrow{unload(pack2,A)} A$.

$\rightarrow$ Targets for $truck$: C [$pack1$], D [$pack1$, $pack2$], A [Goal, $pack2$].

$\rightarrow$ Path for $truck$: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow C \rightarrow B \rightarrow A$.

Complexity: Acyclic + Invertible, ctd.

Theorem. *Restrict the input to FDR tasks* $\Pi = (V, A, c, I, G)$ *such that* $CG(\Pi)$ *is acyclic and, for all* $v \in V$*, all value transitions of* v *are invertible. Then PlanEx can be decided in polynomial time.*

$\rightarrow$ Note: We do *not* require the DTGs to be connected here. If they were connected, Π would be solvable and there would be no PlanEx to decide. Also, Π can be solvable even for non-connected DTGs:

Proof intuition [for reference]: If every $v \in V$ can reach all target values – those requested in preconditions by its clients – in $DTG(v, \Pi)$, then, due to invertibility, these target values can be provided whenever they are requested.

If there exists $v \in V$ that can *not* reach all target values in $DTG(v, \Pi)$, then the plan cannot be constructed.

So PlanEx is equivalent to the question whether there exists an arrangement where all target values are reachable in all $DTG(v, \Pi)$. This is equivalent to the existence of a delete-relaxed plan (**Chapter 20**), because we can read off reachable target values from a delete-relaxed plan, and vice versa.

Summary

- For general problem solving to be effective, it is essential to automatically detect and exploit problem structure.
- Causal graphs are the most prominent means to capture problem structure in planning; they are typically considered along with domain transition graphs.
- Causal graphs can be used for a variety of purposes, including task decomposition/simplification and complexity analysis.
- Simple decomposition/simplification methods are to split up unconnected components, remove invertible root variables, remove non-goal leaf variables.
- One tractable class is the special case where the causal graph is acyclic and all value transitions are invertible.

Reading

- *The Fast Downward Planning System* [Helmert (2006)].

  Available at:

  https://www.jair.org/index.php/jair/article/view/10457

  Content: This is the initial paper on the Fast Downward planning system, which in the meantime has grown into the main implementation basis for heuristic search planning. I suggest it for this chapter because it very clearly compares causal graphs for STRIPS vs. those for FDR (FDR is called "SAS+" and "multi-valued planning" in there), and to my knowledge it's the first paper introducing DTGs. The part of the paper up to Section 5.2 (first 25 pages) is directly relevant to the present chapter; the remainder of the paper is not, but is definitely useful background knowledge for heuristic search planning, and thus for this course as a whole.

References I

Ronen Brafman and Carmel Domshlak. Structure and complexity in planning with unary operators. *Journal of Artificial Intelligence Research*, 18:315–349, 2003.

Ronen Brafman and Carmel Domshlak. On the complexity of planning for agent teams and its implications for single agent planning. *Artificial Intelligence*, 198:52–71, 2013.

Hubie Chen and Omer Giménez. Causal graphs and structurally restricted planning. *Journal of Computer and System Sciences*, 76(7):579–592, 2010.

Carmel Domshlak and Yefim Dinitz. Multi-agent offline coordination: Structure and complexity. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 34–43. Springer-Verlag, 2001.

Carmel Domshlak, Jörg Hoffmann, and Michael Katz. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114, 2015.

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*, 34:675–706, 2009.

References II

Daniel Gnad and Jörg Hoffmann. Beating LM-cut with h^{max} (sometimes): Fork-decoupled state space search. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, pages 88–96. AAAI Press, 2015.

Malte Helmert. A planning heuristic based on causal graph analysis. In Sven Koenig, Shlomo Zilberstein, and Jana Koehler, editors, *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, pages 161–170, Whistler, Canada, 2004. AAAI Press.

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

Jörg Hoffmann. Analyzing search topology without running any search: On the connection between causal graphs and h^+. *Journal of Artificial Intelligence Research*, 41:155–229, 2011.

Michael Katz and Carmel Domshlak. New islands of tractability of cost-optimal planning. *Journal of Artificial Intelligence Research*, 32:203–288, 2008.

References III

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

Michael Katz and Emil Keyder. Structural patterns beyond forks: Extending the complexity boundaries of classical planning. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*, pages 1779–1785, Toronto, ON, Canada, July 2012. AAAI Press.

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

Brian C. Williams and P. P. Nayak. A reactive planner for a model-based executive. In M. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 1178–1185, Nagoya, Japan, August 1997. Morgan Kaufmann.