Searching for actual causes: Approximate algorithms with adjustable precision

Samuel Reyd, Ada Diaconescu, Jean-Louis Dessalles
LCTI, IPParis, Palaiseau, France
name.surname@telecom-paris.fr

Abstract

Causality has enhanced the interpretability of machine learning models. However, traditional causality often falls short of producing relevant explanations, those grounded in actual causes. Identifying actual causes remains computationally intractable (NP-complete). We propose two polynomial-time algorithms: a beam-search-based method and an optimized variant (ISI) leveraging causal structure. Experiments on a causal model with varying size show our algorithms (1) identify multiple causes and (2) offer tunable precision-exhaustiveness-runtime tradeoffs, paving the way to using causality for relevant explanation in explainable AI.

1 Introduction

Causality offers several critical advantages for machine learning by enabling models to generalize beyond correlations observed in training data. Notably, causality distinguishes between associations and true cause-effect relationships [34, 36], supports counterfactual reasoning [29], enhances sample efficiency [5], and improves interpretability [28, 35, 25]. Significant research has been conducted to leverage the strengths of causal reasoning and to identify causal structures in data [11, 3].

Causality is central to explainable AI (XAI), but it also contributes to its limitations. As noted by Miller [26], XAI often caters more to researchers than to end users. He emphasizes that effective explanations depend on context and user expectations, rather than general causal relationships. Traditional XAI and causality aim to answer "What factors contributed to this outcome?", i.e., general causation. However, users often seek actual causation, i.e., "What facts caused this outcome?". For example, while the general statement "Smoking causes cancer" provides a broad understanding of a causal relationship, a user usually prefers a specific explanation, such as "Her cancer was caused by her smoking". These distinctions are illustrated in Table 1.

Actual causes are often linked to counterfactual, or but-for, causes: a cause is defined as a fact such that, in a counterfactual world where the cause does not occur, neither does the consequence. Halpern and Pearl [15] proposed the most influential formalism, often referred to as the HP definition. Providing a satisfactory definition of actual causes has been a notable challenge [17, 23, 14, 15, 12, 4].

Identifying actual causation is an NP-complete problem [2, 12]. There exist few, if any, practical solutions for finding exact or approximate HP causes, and, to the best of our knowledge, no practical approach exists to identify the full set of HP causes.

We propose two approximate algorithms for the efficient identification of actual causes. First, **algorithm 1** is a search algorithm, derived from the beam search algorithm [24], that leverages certain aspects of the HP definition to navigate counterfactual situations efficiently and identify HP causes. Second, **algorithm 2** iteratively calls algorithm 1 on sub-instances constructed using the structure of the system, when available, improving on the first algorithm's performance.

Accepted to the NeurIPS 2025 Workshop on CauScien: Uncovering Causality in Science.

Causation: Why did this happen?		
	General Causation	**Actual Causation**
Formulation	What **factors** contributed to this?	What **facts** caused this?
Example	Smoking causes cancer.	Her smoking caused her cancer.
Usage in XAI	Interpretability	Explanations
Formalisation	Bayesian networks Mainly accepted Large literature	HP-causes Still under debate Some literature
Identification	Causal discovery No working solution Huge literature	No unified field No practical solution Little literature

Table 1: General causation versus Actual causation. This paper addresses the red area, i.e., identifying actual causes.

We conduct experiments on a causal model with varying sizes, as reported in the literature [19]. Our experiments demonstrate that our algorithm can approximately identify the set of HP causes. Additionally, we illustrate an actionable tradeoff between precision (i.e., whether the identified causes are correct), exhaustiveness (i.e., whether expected causes are missing), and the runtime of the algorithm. The code for the experiments can be found in the dedicated GitHub repository[1]. This paper is a short version of our full work [32].

These results pave the way to the usage of our method for actual causes identification in general explainability methods, which would help steer XAI towards less "interpretability-driven" approaches and more user-friendly explanations.

2 Background & related work

Structural Causal Models (SCMs) are introduced by Pearl [29] to model counterfactual reasoning. We note an SCM as $\mathcal{M} = (V, U, F, D)$. V is the set of endogenous variables, U of exogenous variables, F of structural assignments, R of domains. Hence, $\forall X \in V : X = F_X(PA_X, U_X) \in D_X$, where PA_X denotes the set of endogenous variables used to compute the value of X, called causal parents. The PA relationship forms a Directed Acyclic Graph (DAG) called a causal graph. Given a context u (i.e., values for U), we can compute the values of V. For instance, we can state that variable $X \in V$ took value x using: $(\mathcal{M}, u) \vDash (X = x)$[2]. Finally, to model counterfactual reasoning, SCMs allow for interventions, i.e., forcing variables $Y \subseteq V$ to take values y'. Interventions can lead to changes in the other variables, e.g., $(\mathcal{M}, u) \vDash [Y \leftarrow y'](X \neq x)$.

This article uses the HP definition of actual cause [12]. A cause is an event such that if it did not occur, neither would the outcome. Additionally, a contingency set W is included, composed of variables that are allowed to keep their actual values in the counterfactual world where the cause does not happen. Formally, $C \subset V$ is the HP cause of $T \in \{0, 1\}$ in $(\mathcal{M}, u)$ if: **(1)** $(M, u) \vDash (C = c)$ and $(M, u) \vDash T$, **(2)** $\exists W \subset V \backslash C$ and $\exists c' \neq c$, s.t. $(M, u) \vDash (W = w)$ and $(M, u) \vDash [C \leftarrow c', W \leftarrow w]\neg T$, and **(3)** No subset of C satisfies the previous conditions.

We choose the HP definition as it is the most popular, and no consensus exists on a better definition. Yet, it fails to match expectations for satisfying causal explanations (such as being contextual or contrastive [26, 27]), and overlooks certain important notions, such as responsibility [6, 12], normality [13, 20, 21], relevance [9, 16], and complex systems related challenges [33].

To the best of our knowledge, only one work specifically focuses on identifying actual causes according to the HP definition [18]. However, they only identify the smallest cause and address solely Boolean systems where the structural assignments are logic formulas. Other methods address actual cause identification but propose an alternative definition from the HP one [1, 30, 8, 31]. Similarly, counterfactual explanations [10] and backtracking counterfactuals [22] tackle a task related to HP-cause identification [7], but fundamental differences in the approach make these methods hard to adapt to identify actual causes.

[1] https://github.com/SamuelReyd/SearchingForCauses

[2] The symbol $\vDash$ denotes causal entailment, expressing that a model and context implies a particular statement

3 Our algorithms

We suppose an SCM $\mathcal{M}$ with discrete domains, a context u and a target predicate $T \in \{0, 1\}$ such that $(\mathcal{M}, u) \models T$. Our algorithms expect the endogenous variables V, the domains D, the instance v^* (i.e., the actual values of V in $(\mathcal{M}, u)$), an oracle function ϕ, and a heuristic function ψ. The oracle evaluates if an intervention "cancels" the consequence (if $\phi(e) = 0$ and $e = ((X_0, x_0), ..., (X_n, x_n))$ then $(\mathcal{M}, u) \models [X_0 \leftarrow x_0, ..., X_n \leftarrow x_n]\neg T$). The heuristic function evaluates how close to "canceling" the consequence an intervention is (similarly, if $\psi(e) \approx 0$ then the intervention is very close to "canceling" the consequence). Our algorithm explores the space of interventions E, where each element is a set of variable-value pairs that characterize an intervention.

3.1 Algorithm 1: base algorithm

To identify the set of actual causes, we explore E and report the elements $e \in E$, such that (i) the intervention "cancels" the consequence, i.e., $\phi(e) = 0$, (ii) the "counterfactual variables" (i.e., variables with intervention values different from the actual ones) are minimal in the sense of inclusion.

We use an approximate algorithm, beam search, that explores only the most promising regions of the search space (which grows exponentially with the number of variables), using heuristic ψ. The algorithm begins by initializing a list of candidate interventions of length 1 (i.e., one variable-value pair). At each step k, the algorithm evaluates the current candidates using the heuristic ψ. It selects the b-best candidates (called the beam) and expands them to generate candidates for the next step. Elements for the next step are obtained by adding one variable-value with a variable missing in the original element. When expanding, we also allow the addition of variables with their actual values, which become part of W (the contingency set). The algorithm continues this process until the elements contain all variables.

We introduce three additions to the standard beam search. (1) We only consider counterfactual values in the initial step to avoid evaluating interventions without effect. (2) We do not expand elements that have $\phi(e) = 1$, as their expansions would be supersets of causes. (3) When we expand the elements from the beam, we pass new elements to the next step only if they are minimal with the already identified causes. The pseudo code and an exemplary run are reported in Annex A.

The algorithm outputs a set of interventions (i.e., variable-value pairs where each variable appears at most once). For each intervention in the output, each variable associated with a counterfactual value is part of the cause C, whereas each variable associated with its actual value is part of a contingency set W. Following the definition, there is no constraint on W (notably, no minimality or size constraint).

Our base beam search algorithm has a temporal complexity of $O(|V|^2 \times |D_{max}| \times b \times N_C \times |C_{max}|)$, with V the set of endogenous variables, D_{max} the size of the larger domain, b the beam size, N_C the number of identified causes, and C_{max} the largest cause. This can be simplified into $O(|V|^3 \times b \times |D_{max}| \times N_C)$. N_C depends on $|V|$ and b in an unclear way, but we can expect polynomial complexity, roughly cubic in the number of variables and linear in the beam size. The size of the domains also contributes to the time complexity of our algorithm.

For the smallest cause identification, we do not iterate through the causes since we stop as soon as one is found, hence the full formula becomes $O(|V|^2 \times |D_{max}| \times b)$. The temporal complexity becomes quadratic in the number of variables. An experimental analysis of the time complexity is shown in Annex C.1.

3.2 Algorithm 2: ISI

We then propose an optimization of our base algorithm when the underlying DAG is known, which we call **Iterative Sub-instance Identification (ISI)**.

This algorithm takes the same inputs as the base beam search, plus the causal graph. It first initializes a one-element queue and an empty memory. The element in the queue is the set of direct causal parents of the target predicate. Until the queue is empty, we pop the first instance $I \subset V$ of the queue and run our base algorithm, identifying causes $\mathcal{C}$ that are subsets of I. We then iterate through the identified causes, expand them, and fill the queue. For each cause $C \in \mathcal{C}$ and the associated contingency W, we list all its subsets $s \in 2^C \backslash \emptyset$. For every s, we create a new instance I'_s made of

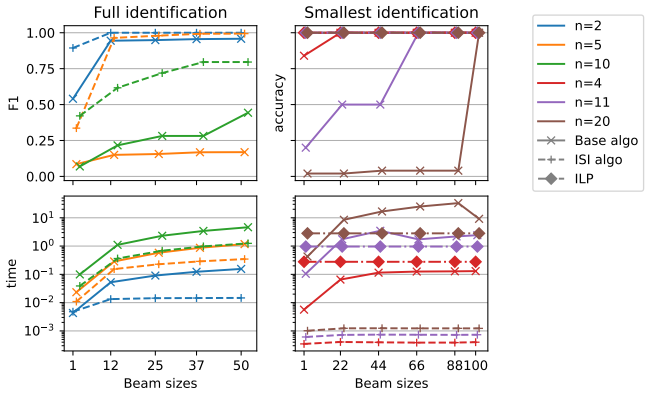

Figure 1: F1 score, accuracy, and runtime (in seconds) for the full and smallest cause identification tasks, comparing our algorithms and ILP. Each line is characterized by the number of attackers (line color) and the algorithm used (line style).

all causal parents of the variables in s and all variables not in s, i.e., $I'_s = \bigcup_{X \in s} PA_X \cup C \backslash s$. To limit redundant computations, when a new instance I'_s is created, we check if it is a subset of any instance saved in memory. If not, we add the pair (I'_s, W) to the queue and memory. The pseudo code and an exemplary run are reported in Annex B.

4 Experiments

We use an SCM [18] that simulates attackers trying to hack a computer system, where F are logical formulas, and the size of V is controlled by the number of attackers. The target T ("Steal Master Key" $SMK = 1$ if the attack succeeds) is a sink of the causal graph and excluded from V (as it's the effect). This SCM is hand-made by the authors of [18] and inspired by their industrial partners. We encourage readers to refer to this work for descriptions of the variables and their logical dependencies. For each experiment, we report results averaged over 50 unique contexts and maintain the same basic heuristic, i.e., minimizing the number of variables with a value of 1, though performance could be further improved with domain-specific heuristics (e.g., prioritizing high-risk attackers).

We first evaluate the "smallest cause identification" task and compare our algorithms with prior approaches [18], which we will refer to as ILP (as it is based on Integer Linear Programming). We run our algorithms and stop when the first cause is found. We use 4, 11, and 20 attackers, and beam sizes of 1, 22, 44, 66, 88, and 100. To evaluate our algorithms, we measure the size of the identified cause and compare it with the output of the ILP approach, which is an exact method. We report an accuracy score that states if the reported cause is the smallest (it has the right size).

We then use our algorithm to identify the full set of causes. We used SCMs with 2, 5, and 10 attackers, and a beam size of 1, 12, 25, 37, and 50. To evaluate our algorithms, we generate the full set of expected causes using the ISI algorithm with an infinite beam size (which can be computationally expensive but returns the exact set of causes). We then compute the recall (% of expected causes that are identified), the precision (% of the identified causes that are correct), and report the F1 score.

5 Results

Results are shown in Figure 1. To identify the smallest cause, with small beam sizes, our algorithm is faster than ILP but often fails to generate the smallest causes. When increasing the beam size, both accuracy and runtime increase. The runtime exceeds ILP before reaching satisfactory accuracies for most model sizes. However, both our algorithm and ILP exhibit an $O(|V|^2)$ complexity (see Annex C.1). The ISI algorithm is faster than ILP and also yields perfect results.

For the full cause identification task, results illustrate an actionable tradeoff between the F1 score and the runtime, controlled by the beam size parameter. The F1 score most often decreases with the number of attackers, while the runtime and F1 score both increase with the beam size. Even if the base algorithm exhibits low scores for the shown beam sizes, we can expect them to reach satisfactory values with enough computation time. The ISI approach demonstrates significantly higher F1 scores and lower runtimes for similar beam sizes and model sizes.

Figure 1 shows the average value between contexts. It shows the tendencies in terms of beam size and the number of attackers. However, some contexts are harder than others, and performance varies greatly between them. Hence, we do not show the standard deviation, which would greatly reduce readability, and we do not refer to an actual variability, as each context would be treated independently in practice. We can note that the distribution of performance amongst contexts tightens as the accuracy increases (see Annex C.2), which suggests that a suitable accuracy can be reached for most contexts with the appropriate beam size.

6 Discussion & Conclusion

Actual causes are crucial to AI explainability. Defining the notion has been challenging for decades. In addition, few to no practical approaches exist to identify actual causes. We propose the first algorithm that can identify the set of actual causes with adjustable precision and exhaustiveness.

Our base algorithm scales with the system size at a polynomial time complexity. It introduces a tradeoff between accuracy and runtime, controlled by a parameter that increases the runtime linearly. We also propose an algorithm that leverages the system structure, when available, to improve accuracy and runtime. We evaluated our algorithms on an SCM from the literature and showed that they identify actual causes with adjustable precision and exhaustiveness.

As our algorithms make few assumptions about the system's nature, they can be used directly on non-Boolean or "black-box" models. For instance, for black-box ML models, ϕ can be estimated via model queries even when the DAG or the structural assignments are unknown. Our algorithms can also be easily adapted to stochastic systems. We tested these adaptations in preliminary experiments, though they are beyond the scope of this paper (see full-size paper [32]). The oracle could be adapted to identify causes with modified definitions.

However, our algorithms still have several limitations. The oracle ϕ requires accurate counterfactual evaluation, which may need structure discovery [37, 11] or domain knowledge. This constitutes a significant limitation when working with cyber-physical systems, but notably less so when working with machine learning models that are often easily actionable. Indeed, our algorithm requires only the result of the interventions, on the opposite of ILP [18] that requires the structural assignment, which are challenging to obtain even for actionable systems. While polynomial, the $O(|V|^3)$ complexity limits the use of large systems without further optimization (e.g., ISI when the DAG is known, which is often the case as the required oracle ϕ frequently relies on the DAG). Additionally, the algorithms lack support for continuous variables, and the variability of our results suggests that certain contexts may require significant beam sizes.

While our algorithms cannot be used without complementary methods to perform explanations, they can contribute to a shift of XAI towards more user-centric explanations and help popularize actual causes in this context, as they will allow researchers to identify such causes in various systems.

References

[1] L. Albantakis, W. Marshall, E. Hoel, and G. Tononi. What Caused What? A Quantitative Account of Actual Causation Using Dynamical Causal Networks. Entropy, 21(5):459, May 2019.

[2] G. Aleksandrowicz, H. Chockler, J. Y. Halpern, and A. Ivrii. The Computational Complexity of Structure-Based Causality. Journal of Artificial Intelligence Research, 58:431–451, 2017.

[3] C. K. Assaad, E. Devijver, and E. Gaussier. Survey and Evaluation of Causal Discovery Methods for Time Series. Journal of Artificial Intelligence Research, 73:767–819, Feb. 2022.

[4] S. Beckers. Causal Sufficiency and Actual Causation. Journal of Philosophical Logic, 50(6):1341–1374, Dec. 2021.

[5] D. C. Castro, I. Walker, and B. Glocker. Causality matters in medical imaging. Nature Communications, 11(1):3673, July 2020.

[6] H. Chockler and J. Y. Halpern. Responsibility and Blame: A Structural-Model Approach. Journal of Artificial Intelligence Research, 22:93–115, Oct. 2004.

[7] Y.-L. Chou, C. Moreira, P. Bruza, C. Ouyang, and J. Jorge. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. Information Fusion, 81:59–83, May 2022.

[8] C. Chuck, S. Vaidyanathan, S. Giguere, A. Zhang, D. Jensen, and S. Niekum. Automated Discovery of Functional Actual Causes in Complex Environments, Apr. 2024. arXiv:2404.10883.

[9] J.-L. Dessalles. Algorithmic Simplicity and Relevance, pages 119–130. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[10] R. Guidotti. Counterfactual explanations and how to find them: Literature review and benchmarking. Data Mining and Knowledge Discovery, 38(5):2770–2824, 2024.

[11] R. Guo, L. Cheng, J. Li, P. R. Hahn, and H. Liu. A Survey of Learning Causality with Data: Problems and Methods. ACM Computing Surveys, 53(4):75:1–75:37, July 2020.

[12] J. Y. Halpern. A modification of the Halpern-Pearl definition of causality. In Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, pages 3022–3033. AAAI Press, 2015.

[13] J. Y. Halpern and C. Hitchcock. Graded Causation and Defaults. The British Journal for the Philosophy of Science, 66(2):413–457, June 2015.

[14] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach: Part i: Causes. In Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, UAI'01, pages 194–202. Morgan Kaufmann Publishers Inc., 2001.

[15] J. Y. Halpern and J. Pearl. Causes and Explanations: A Structural-Model Approach. Part I: Causes. The British Journal for the Philosophy of Science, 56(4):843–887, 2005.

[16] E. Houze, J.-L. Dessalles, A. Diaconescu, and D. Menga. What Should I Notice? Using Algorithmic Information Theory to Evaluate the Memorability of Events in Smart Homes. Entropy, 24(3):346, 2022.

[17] D. Hume. A Treatise of Human Nature. In E. S. Radcliffe, R. McCarty, F. Allhoff, and A. Vaidya, editors, Late Modern Philosophy: Essential Readings with Commentary. Wiley-Blackwell, 2007.

[18] A. Ibrahim and A. Pretschner. From Checking to Inference: Actual Causality Computations as Optimization Problems. In D. V. Hung and O. Sokolsky, editors, Automated Technology for Verification and Analysis, pages 343–359. Springer International Publishing, 2020.

[19] A. Ibrahim, S. Rehwald, and A. Pretschner. Efficient Checking of Actual Causality with SAT Solving. In Engineering Secure and Dependable Software Systems, pages 241–255. IOS Press, 2019.

[20] T. F. Icard, J. F. Kominsky, and J. Knobe. Normality and actual causal strength. Cognition, 161:80–93, Apr. 2017.

[21] D. Kahneman and D. T. Miller. Norm theory: Comparing reality to its alternatives. Psychological Review, 93(2):136–153, 1986.

[22] K.-R. Kladny, J. von Kügelgen, B. Schölkopf, and M. Muehlebach. Deep Backtracking Counterfactuals for Causally Compliant Explanations, Aug. 2024.

[23] D. Lewis. Causation. The Journal of Philosophy, 70(17):556–567, May 1974.

[24] B. T. Lowerre. The harpy speech recognition system. PhD thesis, Carnegie Mellon University, USA, 1976. AAI7619331.

[25] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. Explainable Reinforcement Learning through a Causal Lens. Proceedings of the AAAI Conference on Artificial Intelligence, 34(03):2493–2500, 2020.

[26] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence, 267:1–38, Feb. 2019.

[27] T. Miller. Contrastive explanation: A structural-model approach. The Knowledge Engineering Review, 36:e14, Jan. 2021.

[28] R. Moraffah, M. Karami, R. Guo, A. Raglin, and H. Liu. Causal Interpretability for Machine Learning - Problems, Methods and Evaluation. ACM SIGKDD Explorations Newsletter, 22(1):18–33, May 2020.

[29] J. Pearl. Causality. Cambridge University Press, Cambridge, 2 edition, 2009.

[30] A. Rafieioskouei and B. Bonakdarpour. Efficient Discovery of Actual Causality Using Abstraction Refinement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 43(11):4274–4285, Nov. 2024.

[31] S. Reyd, A. Diaconescu, and J.-L. Dessalles. CIRCE: A Scalable Methodology for Causal Explanations in Cyber-Physical Systems. In 2024 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), pages 81–90, Sept. 2024.

[32] S. Reyd, A. Diaconescu, and J.-L. Dessalles. Searching for actual causes: Approximate algorithms with adjustable precision, 2025.

[33] S. Reyd, A. Diaconescu, J.-L. Dessalles, and L. Esterle. A Roadmap for Causality Research in Complex Adaptive Systems. In 2024 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), pages 35–40, Sept. 2024.

[34] B. Scholkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Toward Causal Representation Learning. Proceedings of the IEEE, 109(5):612–634, May 2021.

[35] P. Schwab and W. Karlen. CXPlain: Causal Explanations for Model Interpretation under Uncertainty. In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

[36] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. On causal and anticausal learning. In Proceedings of the 29th International Coference on International Conference on Machine Learning, ICML'12, pages 459–466. Omnipress, 2012-06-26.

[37] M. J. Vowels, N. C. Camgoz, and R. Bowden. D'ya Like DAGs? A Survey on Structure Learning and Causal Discovery. ACM Computing Surveys, 55(4):82:1–82:36, Nov. 2022.

A Details on algorithm 1

A.1 Pseudo code of the first algorithm

We present the pseudo-code for the beam search algorithm in Algorithm 1. At each step, it expands the current beam using the `expand_beam` function, which is reported in Algorithm 2.

If the beam is null, i.e., this is the first step, we use the `getCfPairs` function, which we do not report. It returns all the possible variable-counterfactual value pairs. Otherwise, the `expandBeam` function iterates through the beam, separates the cause set C and the contingency set W using the `getSets` function. Then, it considers each variable X. We ignore the variables that are already in the element being expanded. We then iterate through D_X, the domain of X, and consider a new element e' by adding the variable-value pair to the current one. If the value is counterfactual and the cause set of e' is minimal, or if the value is actual, we add e' to the set of expanded elements.

Once the beam is expanded, we check for the stop conditions. If no stop condition is met, we evaluate the elements using the oracle and split the beam into `neg`, i.e., elements with $\phi(e) = 0$, and `pos`, i.e., the ones with $\phi(e) = 1$. We then filter elements from `neg` based on their cause set and `Cs`, the set of already identified causes. The minimal ones are added to `Cs`. We then score the elements from `pos` and create the next beam with the top-b ones.

Algorithm 1: Base Algorithm

Input : variables V, instance v^*, domains D, oracle ϕ, heuristic ψ, beam size b, threshold ϵ, early_stop, max_steps
Output: List of all causes

```
Cs ← ∅ ;                                                    // Identified causes
B ← ∅ ;                                                     // Beam
for t ← 1 to max_steps do
    B ← expandBeam(B, V, D, v*, Cs );
    if B = ∅ or (early_stop & Cs ≠ ∅) do
        break;
    neg, pos ← splitEval(B, φ, ϵ);
    C ← filterMinimality(neg, v*);
    Cs ← Cs ∪ C;
    B ← getBest(pos, ψ, b, Cs );
return Cs;
```

A.2 Algorithm 1 on an example

We now present a complete run of our algorithm on an example, which is schematized in Figure A.1. We consider the rock-throwing scenario [12]. In this scenario, Suzy and Billy are throwing rocks at a bottle. Both aim accurately, and Suzy hits the bottle first. The bottle, therefore, shatters. In this scenario, we intuitively consider that the cause of the bottle shattering is Suzzy's throw.

The DAG of this scenario is represented on the left-hand side of the figure. The exogenous variables are st and bt. They set the initial conditions on the variables ST and BT. Both exogenous variables take the value 1 in the context u. The actual values are $ST, BT, SH, \neg BH, BS$. The target predicate is BS. Hence, BS is not included in the causal variables given as input to the algorithm.

We execute our algorithm using `early_stop` with value 0 and no `max_steps`. The heuristic computes the sum of positive variables and must be minimized. The beam B is composed of the top-b elements, i.e., with the lowest score of the heuristic. Elements, shown inside the nodes of the graph, are interventions characterized by a series of variable-value pairs. Since our example uses Boolean variables, we simplify the figure by writing the variable and its value, including or omitting the symbol $\neg$. For instance, the node represented with "$ST, \neg SH$" is associated with the pairs $((ST, 1), (SH, 0))$

Algorithm 2: expandBeam Function

Input : current beam B, variables V, domains D, instance v^*, identified causes Cs
Output: New beam

```
if B = ∅ do
    return getCfPairs(V, v*, D);
nextBeam ← ∅;
for each elt ∈ B do
    C, W ← getSets(elt, v*);
    for each X ∈ V do
        if X ∈ C ∪ W do
            continue;
        nonMinimalCause ← False;
        for each c ∈ Cs do
            if c ⊆ C ∪ {X} do
                nonMinimalCause ← True;
        for each x ∈ D_X do
            if x ≠ v*_X and nonMinimalCause do
                continue;
            newElt ← elt ∪ {(X, x)};
            nextBeam ← nextBeam ∪ {newElt};
return nextBeam;
```

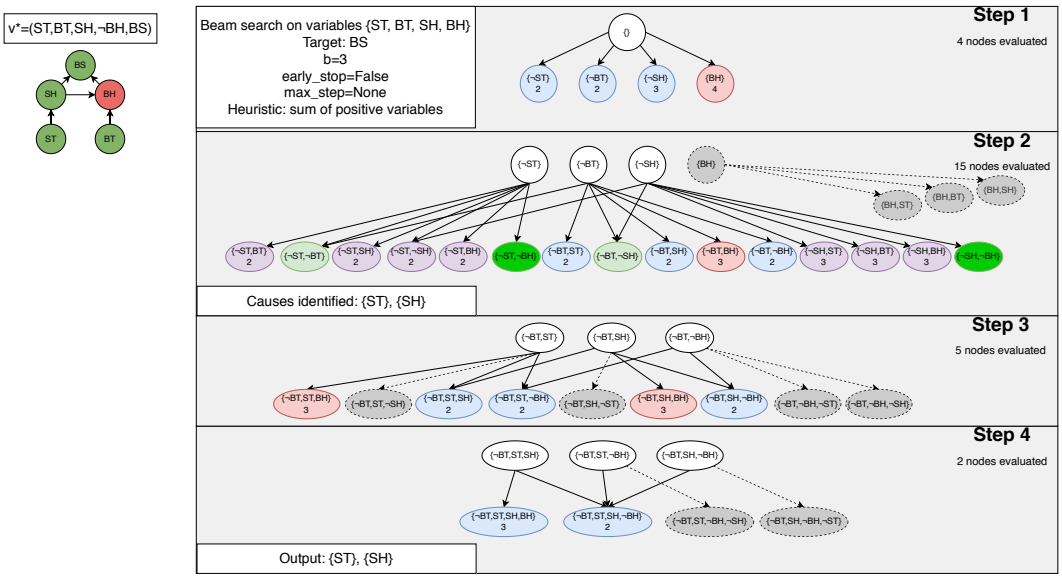

Figure A.1: Illustration of our algorithm on the rock throwing scenario. Nodes in blue constitute the beam, i.e., the nodes selected for expansion. The ones in red are nodes with less optimal heuristic values and are not part of the beam. Green ones are nodes that 'cancel' the consequence, light green is used when the cause is not minimal, and a darker green is used for the identified causes. The ones in purple are nodes that have been evaluated but that are filtered because they are supersets of a cause identified at the same depth. Nodes in gray (with dotted lines) are represented for illustration purposes, but are not considered by the algorithm.

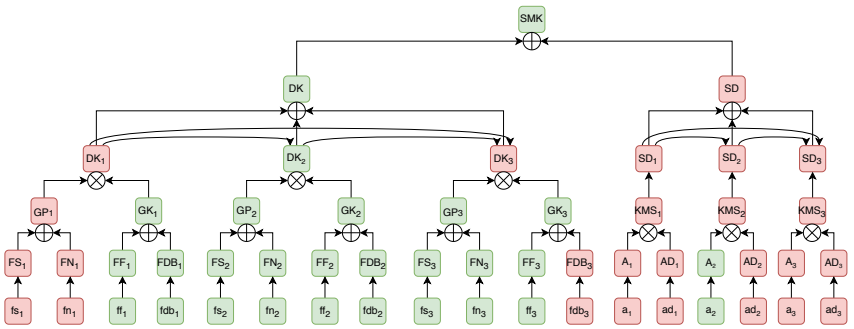

Figure B.1: An instance of the SMK scenario. Green nodes have value 1 and red ones have value 0. Lowercase node labels are exogenous variables.

During the first step, we initialize the root with the empty set. We build the first level with singletons of variables and only counterfactual values. We evaluate these nodes and find none that 'cancel' the consequence. We score them, keep the three best (colored in blue), and discard the other one (in red).

During the second step, we expand the nodes. "BH" expansion is shown with dotted lines to highlight that the algorithm does not build or evaluate these children. At this step, we expand six children per parent, and they all share one child with another parent. We end up with 15 nodes that we all evaluate. We find 4 nodes (colored in green) that 'cancel' the consequence, keep the two minimal ones (in darker green), and discard the non-minimal ones (lighter green). We then discard all nodes that are supersets of the identified causes. We end up with 4 nodes that we score to keep the 3 best.

During step 3, each parent can generate 4 children and share one child with the other parents. Additionally, the children who are supersets of the identified causes are not expanded (here shown in grey with dotted lines). We end up with 5 nodes to score. We keep the 3 best and discard the others.

Finally, in step 4, we have two nodes that are supersets of identified causes (not considered) and 2 that are evaluated. The nodes already use all the variables, so they cannot be expanded further. The algorithm ends here.

The algorithm outputs two causes of size 1, i.e., $\{ST\}$ and $\{SH\}$. The output also includes the contingency set ($\{BH\}$ in both cases). We can also include the counterfactual values as evidence of the cause (here, values 0 for both cases).

B Details on algorithm 2

B.1 Pseudo code for algorithm 2 (ISI)

The pseudo code for the ISI algorithm is reported in Algorithm 3. The queue is composed of pairs, which first contain the set of variables on which beam search will be run, and second contain the reference contingency set that must be included by default when running beam search. In the pseudo code, the full set of inputs for beam search is omitted as they are always identical except for the variables and the reference contingency set. The variables C_{ref} and its elements, which are iterated by variable `elt`, are not causes per se but the interventions identified by beam search. Hence, the function `getSets` is used to obtain the cause set C and the contingency set W. We then iterate through each subset s of the cause set C. For each s, we make a new candidate q for the queue Q composed of the causal parents for the variables of s and the variables in $C \backslash s$. Each of these candidates that is minimal for `memory` is added to Q and to `memory`.

B.2 Algorithm 2 on an example

We show an exemplary run of our algorithms on the instance presented in Figure B.1. It represents all the causal variables of the SMK scenario and schematizes the structural assignments. The "+" denotes logical "or" while the "-" denotes logical "and". For instance, $GK_1 = FF_1 \vee FDB_1$.

Algorithm 3: ISI algorithm

Input : variables V, instance v^*, domains D, oracle ϕ, heuristic ψ, beam size b, threshold ϵ, max_steps, early_stop, DAG G
Output: List of all causes

```
Cs ← ∅;
Q ← ∅;
Q.queue((G.init_variables, ∅));
memory ← ∅;
while Q ≠ ∅ do
    V_temp, W_ref ← Q.dequeue();
    C ← beamSearch(V_temp, W_ref, ...);
    Cs ← Cs + C;
    for each elt of C do
        C, W ← getSets(elt);
        for each s ∈ 2^C do
            q ← ∅;
            for each v ∈ C do
                if v ∉ s do
                    q ← q ∪ {v};
                else
                    q ← q ∪ G.getParents(v);
            if checkInclusion(q,memory) do
                memory ← memory ∪ {q};
                Q.queue((q,W));
return Cs;
```

The horizontal arrows represent precedence. If a user manages to decrypt the key (DK_i) or to steal the decrypted key (SD_i), subsequent attackers are prevented from doing so. For instance, we have $DK_3 = \neg DK_2 \wedge \neg DK_1 \wedge (GP_3 \wedge GK_3)$.

We first describe the steps taken by the base algorithm as a baseline. We use a beam size of 200 and a maximum number of steps of 6. The heuristic is the number of positive variables, which we attempt to minimize. We summarize the steps followed by the algorithm. (1) The first depth of the tree evaluates 35 nodes and finds 1 cause, i.e., $\{DK\}$. All the remaining nodes are expanded. (2) The second depth of the tree evaluates 1,717 nodes. The algorithm finds three causes ($\{DK_2\}$, $\{GK_2\}$, and $\{GP_2\}$, with contingency $\{DK_3\}$ for all three). Removing the supersets of these causes, there are still 1,519 nodes. We extend the top 200. (3) There are 8,550 nodes to evaluate. We identify two causes ($\{FDB_2, FF_2\}$, and $\{FS_2, FN_2\}$, with contingency $\{DK_3\}$ for both) and are left with 8,426 nodes. (4) From the top 200 of the previous nodes, we obtain 8,357 new nodes. From there, we no longer find additional causes and omit reporting the number of remaining nodes after filtering. (5) This step evaluates 9,378 nodes. (6) This step evaluates 8,031.

We now report the steps taken by the ISI algorithm. We report the input and output of the beam search when called, but do not describe any of its internal steps. When running the beam search instances, we choose to have an unlimited beam size and no maximum number of steps. Hence, we run an exact algorithm here. (1) The first step considers the direct causal parents of SMK, i.e., DK and SD. The beam search returns $\{DK\}$. We then add to the queue the set of causal parents of DK, i.e., $\{DK_1, DK_2, DK_3\}$. (2) The second step pops an element from the queue (the only one in this case). We run the beam search with $\{DK_1, DK_2, DK_3\}$. We obtain $\{DK_2\}$ with contingency $\{DK_3\}$. We queue the causal parents of DK_2, i.e., $\{DK_1, GP_2, GK_2\}$, while keeping the contingency set in memory. (3) We run $\{DK_1, GP_2, GK_2\}$ with contingency $\{DK_3\}$. We first obtain $\{GP_2\}$ with contingency $\{DK_3\}$. From this cause, we queue $\{FS_2, FN_2\}$ with contingency $\{DK_3\}$. The second cause is $\{GK_2\}$ with contingency $\{DK_3\}$. From this cause, we queue $\{FF_2, FDB_2\}$ with contingency $\{DK_3\}$. (4) We dequeue $\{FS_2, FN_2\}$ with contingency $\{DK_3\}$ and find one cause: $\{FS_2, FN_2\}$ with contingency $\{DK_3\}$. (5) We dequeue $\{FS_2, FN_2\}$ with contingency $\{DK_3\}$ and find one cause: $\{FF_2, FDB_2\}$ with contingency $\{DK_3\}$.

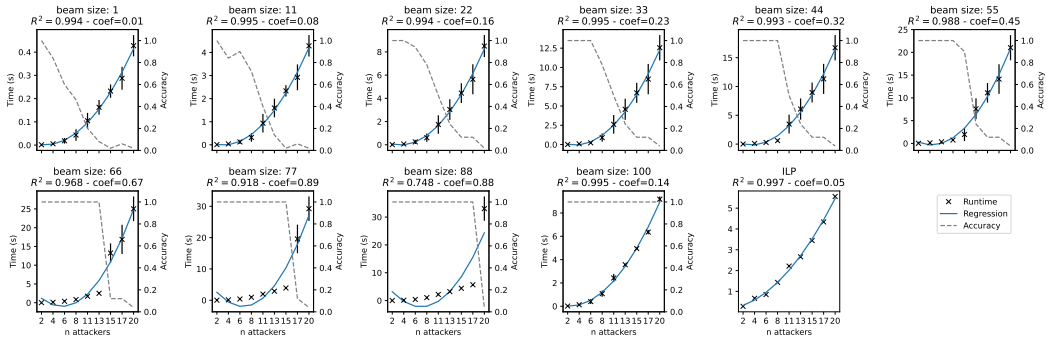

Figure C.1: Time complexity against the number of attackers for smallest causes identification with the base algorithm.

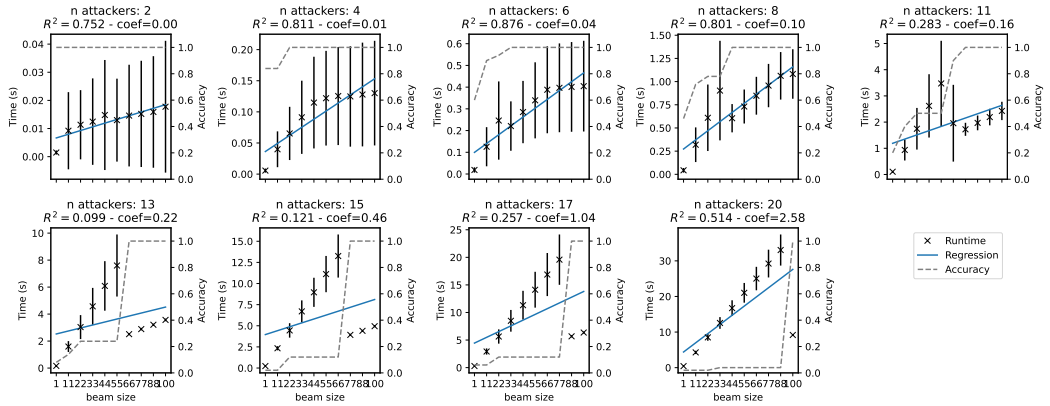

Figure C.2: Time complexity against the beam size for smallest causes identification with the base algorithm.

C Extended experiments

C.1 Empirical time complexity of the smallest cause identification

We made regressions to assess the time complexity of the smallest causes identification task with our base algorithm. Figures C.1 and C.2 show that most of the time, the time complexity is quadratic with the number of attackers and linear with the beam size. For beam sizes below 44, the accuracy slowly drops as the number of attackers varies, and the runtime is quadratic. For a beam size of 100, the accuracy is always 1, and the runtime increases quadratically with the number of attackers.

However, we can see a clear effect of the accuracy on the scalability regime, which corresponds to the impact of N_C, the number of identified causes. For a beam size between 44 and 88, when the number of attackers varies, a clear threshold is observed where the accuracy starts to drop. At this point, the runtime suddenly increases.

When we fix the number of attackers and plot the runtime to the beam size, the effect of the accuracy is even more noticeable. For two attackers, the accuracy is always 1, but the runtime is extremely small (around 0.015s), with a high relative standard deviation (up to 0.04s). Hence, the linear regression is not precise. For all other values of the number of attackers, there is a clear distinction between when the accuracy is one and when it is not. It seems that both regimes are linear, with the one at accuracy 1 being faster.

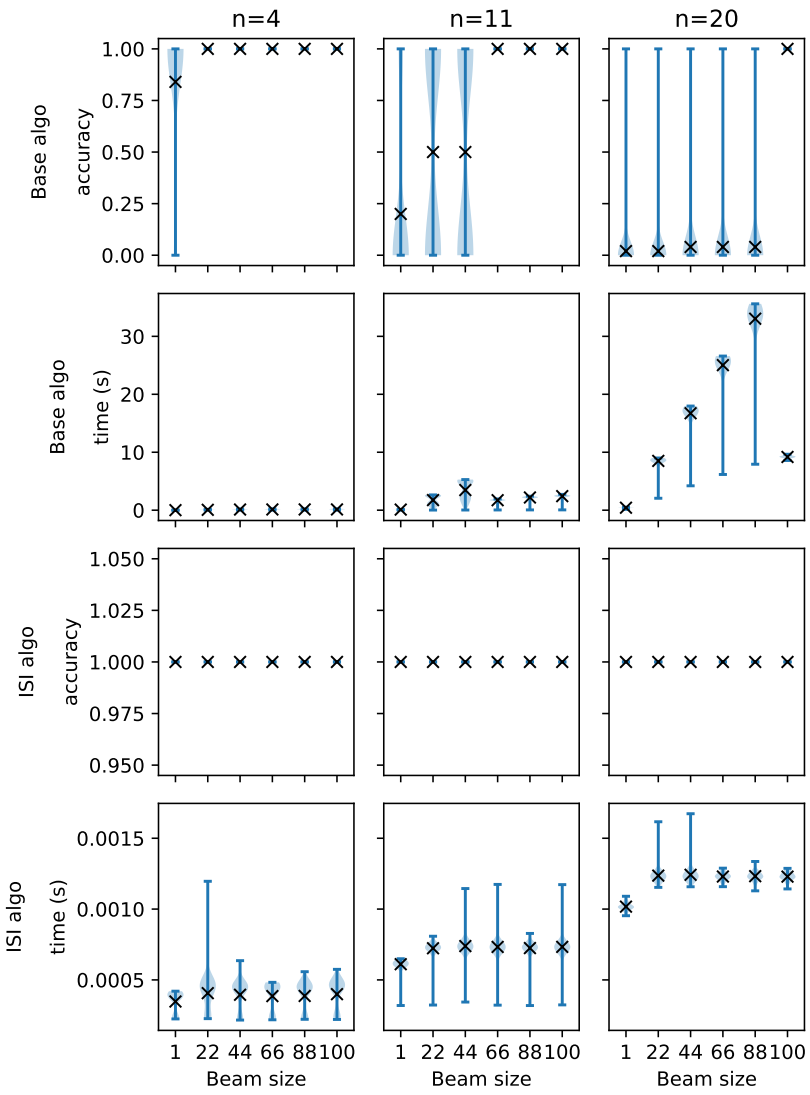

Figure C.3: Distributions of the F1 score and runtime for the smallest cause identification.

C.2 Distributions of the measures

Figures C.3 and C.4 respectively show the distribution of the reported metrics for the smallest and the full cause identification tasks.

F1-scores exhibit extremely variable results depending on the contexts. For both the base and ISI algorithms, the range of the distribution is small compared to the average value only when it is close to 0 or 1. This suggests a drastic impact of the context on the algorithm's performance. However, the lower bound of the distribution often increases with the beam size. Hence, the F1 score increases with the beam size, even with harder contexts.

The runtime distributions are significantly less spread out. For the base algorithm, the range of the distribution is always reasonable compared to the average value. For the ISI algorithm, some runtime distributions exhibit small lower bounds. This corresponds to contexts with smaller causes where the ISI algorithm only explores a small part of the endogenous variables.

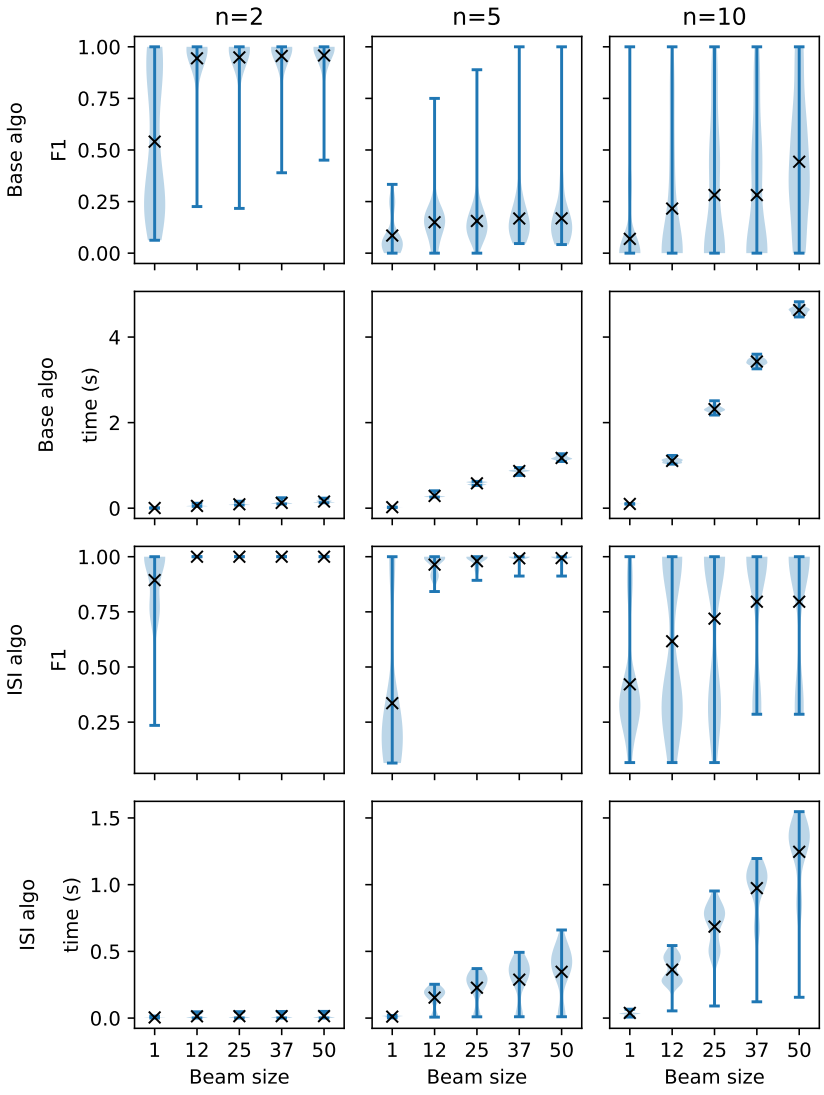

Figure C.4: Distributions of the accuracy score and runtime for the full causes identification.