GRACE-C: Generalized Rate Agnostic Causal Estimation via Constraints

Anonymous authors
Paper under double-blind review

Abstract

Graphical structures estimated by causal learning algorithms from time series data can provide highly misleading causal information if the causal timescale of the generating process fails to match the measurement timescale of the data. Existing algorithms provide limited resources to respond to this challenge, and so researchers must either use models that they know are likely misleading, or else forego causal learning entirely. Existing methods face up-to-four distinct shortfalls, as they might *a*) require that the difference between causal and measurement timescales is known; *b*) only handle very small number of random variables when the timescale difference is unknown; *c*) only apply to pairs of variables (albeit with fewer assumptions about prior knowledge); or *d*) be unable to find a solution given statistical noise in the data. This paper aims to address these challenges. We present an algorithm that combines constraint programming with both theoretical insights into the problem structure and prior information about admissible causal interactions to achieve speed up of multiple orders of magnitude. The resulting system scales to significantly larger sets of random variables (> 100) without knowledge of the timescale difference while maintaining theoretical guarantees. This method is also robust to edge misidentification and can use parametric connection strengths, while optionally finding the optimal among many possible solutions.

1 Introduction

Dynamic causal models play a pivotal role in modeling real-world systems in diverse domains, including economics, education, climatology, and neuroscience. Given a sufficiently accurate causal graph over random variables, one can predict, explain, and potentially control some system; more generally, one can understand it. In practice, however, specifying or learning an accurate causal model of a dynamical system can be challenging for both statistical and theoretical reasons.

One particular challenge arises when data are not measured at the speed of the underlying causal connections. For example, fMRI scanning of the brain measures bloodflow and oxygen level changes in different brain regions, thereby indirectly measuring neural activity (which leads to increased oxygen consumption). fMRI thus provides data about an important dynamical system, but these measures take place (at most) every second while the brain's actual dynamics is known to proceed at a faster rate Oram & Perrett (1992), though we do not know how much faster. In general, when the measurement timescale is significantly slower than the causal timescale (as with fMRI), learning can output importantly incorrect causal information. For instance, if we only measure every other timestep in Figure 1, then the true graph (top left) would differ from the data graph (top right). For example, we might conclude that variable 2 directly influences variable 5, when variable 3 is the actual direct cause. This type of error can lead to inefficient or costly methods of control. More generally, understanding of a system depends on the causal-timescale (i.e., non-undersampled) causal relations, not the measurement-timescale (apparent) relations.

In this paper, we consider the problem of learning the causal structure at the *causal* timescale from data collected at an unknown *measurement* timescale. This challenge has received significant attention in recent years Plis et al. (2015b); Gong et al. (2015); Hyttinen et al. (2017); Plis et al. (2015a), but all current algorithms have significant limitations (see Section 2) that make them unusable for many real-world scientific challenges. Current algorithms show the theoretical possibility of causal learning from undersampled data, but their practical applicability is limited to small graph sizes, sometimes

including only a pair of variables Gong et al. (2015). In contrast, we present a provably correct and complete algorithm that can operate on 100-node graphs and hence be potentially useful in biological and other domains for learning causal timescale structure from undersampled data.

2 Related Work And Notation

A directed dynamic causal model is a generalization of "regular" causal models Pearl et al. (2000); Spirtes et al. (1993): graph $\mathbf{G}$ includes n distinct nodes for random variables $\mathbf{V} = \{V_1, V_2, ..., V_n\}$ at both the current timestep t ($\mathbf{V}^t$), and also each previous timesteps ($\mathbf{V}^{t-k}$) in which there is a direct cause of some V_i^t. We assume that the "true" underlying causal structure is first-order Markov: the independence $\mathbf{V}^t \perp\!\!\!\perp \mathbf{V}^{t-k} \mid \mathbf{V}^{t-1}$ holds for all $k > 1$[1] (i.e. causal sufficiency assumption Spirtes et al. (2000)). $\mathbf{G}$ is thus over $2\mathbf{V}$, and the only permissible edges are $V_i^{t-1} \rightarrow V_j^t$, where possibly $i = j$. The quantitative component of the dynamic causal model is fully specified by parameters for $P(\mathbf{V}^t|\mathbf{V}^{t-1})$. We assume that these conditional probabilities are stationary over time, but the marginal $P(\mathbf{V}^t)$ need not be stationary.

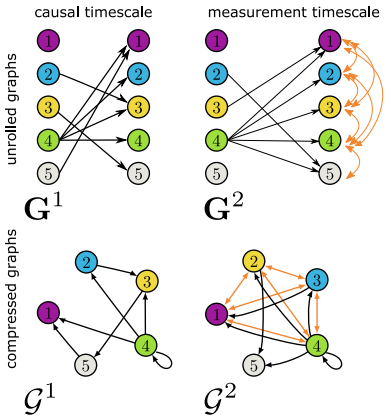

Figure 1: Causal graph $\mathbf{G}^1$ and its undersampled version $\mathbf{G}^2$: unrolled and compressed versions.

We denote the timepoints of the underlying causal structure as $\{t^0, t^1, t^2, ..., t^k, ...\}$. The data are said to be *undersampled at rate u* if measurements occur at $\{t^0, t^u, t^{2u}, ..., t^{ku}, ...\}$. We denote undersample rate with superscripts: the true causal graph (i.e., undersampled at rate 1) is $\mathbf{G}^1$ and that same graph undersampled at rate u is $\mathbf{G}^u$. To determine the implied $\mathbf{G}$ at other timescales, the graph is first "unrolled" by adding instantiations of $\mathbf{G}^1$ at previous and future timesteps, where $\mathbf{V}^{t-2}$ bear the same causal relationships to $\mathbf{V}^{t-1}$ that $\mathbf{V}^{t-1}$ bear to $\mathbf{V}^t$, and so forth. In this unrolled (time-indexed by t) graph, all $\mathbf{V}$ at intermediate timesteps are not measured; this lack of measurement is equivalent to marginalizing out (the variables in) those timesteps to yield $\mathbf{G}^u$. This problem has been parametrically addressed by Gong et al. (2015). Yet, a very interesting approach proposed in the paper was demonstrated only on a 2-variable system. Although an interesting approach, it has not been developed further and made practical.

Various representations have been developed for graphs with latent confounders, including partially-observed ancestral graphs (PAGs) Richardson & Spirtes (2002) and maximal ancestral graphs (MAGs) Zhang (2008). However, these graph-types cannot easily capture the types of latents produced by undersampling Mooij & Claassen (2020). Instead, we use compressed graphs, along with properties that were previously proven for this representation Danks & Plis (2013). A condensed graph includes only $\mathbf{V}$, where temporal information is implicitly encoded in the edges. In particular, a condensed graph version $\mathcal{G}$ of dynamic causal graph $\mathbf{G}$ has $V_i \rightarrow V_j$ in $\mathcal{G}$ iff $V_i^{t-1} \rightarrow V_j^t$ is in $\mathbf{G}$. Undersampling (i.e., marginalizing intermediate timesteps) is a straightforward operation for compressed graphs: (1) $V_i \rightarrow V_j$ in $\mathcal{G}^u$ iff there is a length-u directed path from V_i to V_j in $\mathcal{G}^1$ iff there is a directed path from V_i^{t-u} to V_j^t in $\mathbf{G}^1$; and (2) $V_i \leftrightarrow V_j$ in $\mathcal{G}^u$ iff there exists length-$s < u$ directed paths from V_k to V_i, and to V_j, in $\mathcal{G}^1$ (i.e., V_k is an unobserved common cause in $\mathbf{G}^1$ fewer than u timesteps back). See Appendix for additional proofs. The bottom row of Figure 1 shows compressed graphs for the unrolled ones on the top row; the left shows the causal timescale and the right shows the graphs undersampled at rate 2.

Given this framework, the overall causal learning challenge can now be restated as: given $\mathcal{G}^u$ but not u (or given dataset $\mathbf{D}$ at unknown undersample rate), what is the set of possible $\mathcal{G}^1$? There will often be many possible $\mathcal{G}^1$ for given $\mathcal{G}^u$, and so we use $[\![\mathcal{H}]\!]$ to denote the equivalence class of $\mathcal{G}^1$ that could yield $\mathcal{H}$ (the given causal graph inferred from data $\mathbf{D}$) for some u. That is, $[\![\mathcal{H}]\!] = \{\mathcal{G}^1 : \exists u(\mathcal{G}^u = \mathcal{H})\}$ Various algorithms have been developed to infer $[\![\mathcal{H}]\!]$, each with distinctive shortcomings. There are 2^{n^2} possible $\mathcal{G}^1$, so perhaps unsurprisingly, this problem is NP-complete:

[1]This assumption is relatively weak, as we do not assume that we measure at this "true" causal timescale. The system timescale can be arbitrarily fast to capture all connections.

Theorem 1 (Hyttinen et al. (2017)[Theorem 1]). *Deciding whether a consistent $\mathbf{G}^1$ exists for a given $\mathcal{H}$ is NP-complete, for all undersampling rates $u \geq 2$.*[2]

Mesochronal Structure Learning (MSL) Plis et al. (2015b) showed it is possible to learn $[\![\mathcal{H}]\!]$ in a non-brute force manner if we know u. Every edge in $\mathbf{G}^u$ corresponds to one or more paths of length u in $\mathbf{G}^1$, and so $\mathbf{G}^1$ can be constructed by identifying $u-1$ intermediate nodes for each edge in $\mathbf{G}^u$. MSL searches the state space of possible identifications in a Depth-First Search (DFS) manner. Each identification implies a $\mathbf{G}^1$, and if $\mathbf{G}^u = \mathcal{H}$, then $\mathbf{G}^1 \in [\![\mathcal{H}]\!]$. Otherwise, search continues. MSL backtracks in the DFS whenever some $\mathbf{G}^u$ includes an edge that is absent from $\mathcal{H}$, as the candidate $\mathbf{G}^1$ and all its supergraphs cannot be in $[\![\mathcal{H}]\!]$.

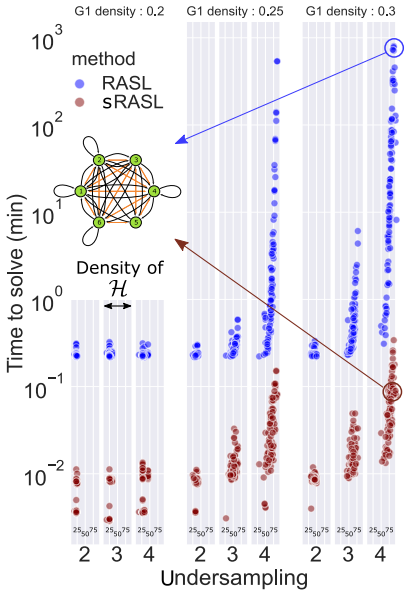

Figure 2: Comparison of sRASL (red) with previous state-of-the-art RASL (blue).

Although Plis et al. (2015b) showed that the concept that causal inference from undersampled data is feasible, MSL is computationally intractable on even moderate-sized graphs. Hyttinen et al. (2017) used the implied constraints to develop an Answer Set Programming (ASP) Simons et al. (2002); Niemelä (1999); Gelfond & Lifschitz; Lifschitz (1988) method that formulated this causal inference challenge as a rule-based constraint satisfaction problem. ASP is a rule-based declarative constraint satisfaction paradigm that is well-suited for representing and solving various NP-hard problems (e.g. Theorem 1). In essence, the algorithm in Hyttinen et al. (2017) takes as input the measured causal graph $\mathcal{H}$, determines the set of implied constraints on $\mathbf{G}^1$, and then uses the general-purpose Answer Set Solver *Clingo* Gebser et al. (2011) to determine the set of possible $\mathbf{G}^1$ significantly faster than MSL. The same idea of using Boolean satisfiability solvers to integrate (in)dependent data constraints has been used for various other causal learning challenges Hyttinen et al. (2013); Triantafillou et al. (2010).

Although the method in Hyttinen et al. (2017) is significantly faster, one must specify the undersampling rate u (or else run the method sequentially for all possible u, thereby losing much of the computational advantage). In contrast, the *Rate-Agnostic (Causal) Structure Learning (RASL)* approach (with three different versions) Plis et al. (2015a) makes no such assumption. These algorithms are similar to MSL, but consider each possible u for some $\mathbf{G}^1$. RASL reduces computational complexity with two additional stopping rules for given $\mathbf{G}^1$: (1) if some $\mathbf{G}^k$ has previously been seen, then further undersampling of $\mathbf{G}^1$ will not produce new graphs; and (2) if $\mathbf{G}^k$ is not an edge-subset of $\mathcal{H}$ for all k, then do not consider any edge-superset of $\mathbf{G}^1$ Plis et al. (2015a). However, despite these improvements, RASL still faces memory and run-time constraints for even moderate numbers of nodes.

One key observation from all of these learning algorithms is the importance of *strongly connected components* (SCCs) Danks & Plis (2013):

Definition 2.1. *An SCC in compressed graph $\mathcal{H}$ is a maximal set of nodes $\boldsymbol{S} \subseteq \boldsymbol{V}$ such that, for every $X, Y \in \boldsymbol{S}$ there is a directed path from X to Y.*

Note that the variables in a compressed graph $\mathcal{H}$ can be fully partitioned based on SCC membership. SCCs can be highly stable, as the node-membership of an SCC will not change as we undersample, as long as the greatest common divisor (gcd) of the set of lengths of all simple loops (directed cycles without repeated nodes) in the SCC is 1:[3]

Theorem 2 (Danks & Plis (2013)[Theorem 3]). *$\boldsymbol{S}$ is an SCC in $\boldsymbol{G}^u$ for all u iff $gcd(\mathcal{L}_{\mathcal{S}}) = 1$ for SCC $\boldsymbol{S} \in \boldsymbol{G}^1$*

[2]Proof provided in Hyttinen et al. (2017). In general, we omit previously published proofs.

[3]The condition easily holds, as it requires only (1) the graph is relatively dense with different loop lengths or (2) any node in the SCC has a self-loop (i.e., is autocorrelated).

In this paper, we develop *sRASL* (for *s*olver-based RASL), a novel algorithm that leverages insights from multiple sources, such as the constraints implied by SCC stability (Theorem 2). We show that sRASL significantly outperforms previous methods. The contributions of this paper are threefold: first, we reformulated the RASL algorithm from a search-based procedure to a constraint satisfaction problem encoded in a declarative language Fahland et al. (2009). Second, this reformulation enables us to add additional constraints based on SCC structure, and thereby gain significant speed-up. Third, we ensure that sRASL provides a straightforward way to find approximate solutions when $\mathcal{H}$ is an unreachable graph (i.e., when $[\![\mathcal{H}]\!] = \emptyset$). These advances collectively provide up to three orders of magnitude improvements in speed, thereby enabling causal inference given undersampling data involving over 100 nodes. As a concreate example of the improvements, Figure 2 compares sRASL (red) with the previously-fastest RASL Plis et al. (2015a) method (blue) on the same graphs. The same input graph $\mathcal{H}$ took RASL nearly 1000 minutes to compute $[\![\mathcal{H}]\!]$, but only 6 seconds for sRASL.

3 sRASL: Optimized ASP-based Causal Discovery

The sRASL algorithm takes as input a (potentially) undersampled graph $\mathcal{H}$, whether learned from data $\mathbf{D}$, expert domain knowledge, a combination of the two, or some other source. sRASL's agnosticism about the source of the input graph enables wider applicability, as we can use whatever information is available Danks & Plis (2019). In the asymptotic (data) limit, the sRASL output is the full $[\![\mathcal{H}]\!]$.

sRASL leverages the fact that connections between SCCs in $\mathcal{H}$ must form a directed acylic graph. More specifically: if $X \rightarrow Y$ with $X \in \mathbf{A}, Y \in \mathbf{B}$ for SCCs $\mathbf{A} \neq \mathbf{B}$, then $C \nleftarrow D$ for all $C \in \mathbf{A}, D \in \mathbf{B}$.[4] Moreover, Theorem 2 provides the (weak) condition under which SCC membership is preserved under undersampling. These two observations imply that structural features potentially provide additional constraints beyond the obvious ones (See Section4.3). In particular, if $\mathcal{H}$ has a roughly modular structure–that is, the SCCs are not too large–then sRASL generates many more constraints than the algorithm of Hyttinen et al. (2017).

Listing 1 shows the `Clingo` (for a brief Introduction on `Clingo` and Answer Set Programming, refer to Appendix C) code of sRASL, which is based on exactly representing the conditioning and marginalization operations (defined in Section 2) in ASP. In the first line, we input the first-order graph-specific specification of $\mathcal{H}$ (e.g., the edge $1 \rightarrow 10$ translates to hdirected$(1, 10)$). Line 2 encodes the second-order structure of $\mathcal{H}$, including the partition of $\mathbf{V}$ into SCCs. These predicates and basic descriptive information are added to the `Clingo` code (lines $3, 4, 5$) in an automated way.[5]

`maxu` on line 3 specifies the maximum undersampling rate, as there is provably such a u where $\mathcal{G}^u = \mathcal{G}^k$ for all $k > u$, if we have the same condition that leads to stable SCC membership:

Theorem 3 (Plis et al. (2015a)[Theorem 3.1])**.** *If $gcd(\mathcal{L}_S) = 1$ for all SCCs $\mathbf{S} \subseteq \mathbf{V}$, then $\mathcal{G}^u = \mathcal{G}^{u+1}$ for all $u > f \leq n_F + \gamma + d + 1$.*

where γ is the transit number[6], d is graph diameter[7] and n_F is the Frobenius number.[8] In practice, the plausible undersampling rate will often be much lower than the theoretical upper bound in Theorem 3. For example, consider fMRI data. The underlying rate of brain activity is generally thought to be ~ 100 milliseconds and fMRI devices measure approximately every two seconds. Hence, $u = 20$ is a plausible upper bound on undersampling in fMRI studies.[9]

Line 6 in Listing 1 stipulates that all edges in $\mathcal{G}^1$ are possible (by default), and so the output will contain any possible model that does not violate the integrity constraints of lines $11 - 16$. Lines 7 and 8 define paths of length L in the graph (i.e., an edge in $\mathcal{G}^L$). As described in Section 2:

[4] If $C \leftarrow D$, then by definition of SCC, there exists $\pi : X \leftarrow \ldots \leftarrow C \leftarrow D \leftarrow \ldots \leftarrow Y$. X, Y are thus mutually reachable so must be in the same SCC, contra $\mathbf{A} \neq \mathbf{B}$.

[5] The code is available at `removed for anonymity`

[6] Transient number is the length of the "longest shortest path" from a node that touches all simple loops of the SCC.

[7] Graph diameter the length of the "longest shortest path" between any two graph nodes.

[8] For set $\mathbf{B}$ of positive integers with $\gcd(\mathbf{B}) = 1$, n_F is the max integer with $n_F \neq \sum_{i=1}^{b} \alpha_i B_i$ for $\alpha_i \geq 0$

[9] Of course, the actual undersample rate could be much lower than 20. Voxels typically contain $8 - 10$ layers of neurons, so the "causal timescale of a voxel" could easily be as high as 1000 ms (i.e., $u = 2$).

```
1  %( * input graph edge specifications here * e.g.: hdirected(1,5) ... )
2  %( * input graph SCC specifications here * e.g.: sccsize(0, 5). scc(1, 0) ...)
3  #const n = 10, maxu = 20
4  node(1..n).
5  1 {u(1..maxu)} 1.
6  {edge1(X,Y)} :- node(X), node(Y).
7  directed(X, Y, 1) :- edge1(X, Y).
8  directed(X, Y, L) :- directed(X, Z, L-1),
9                       edge1(Z, Y), L <= U, u(U).
10 bidirected(X, Y, U) :- directed(Z, X, L), directed(Z, Y, L), node(X;Y;Z), X < Y, L
       < U, u(U).
11 :- directed(X, Y, L), not hdirected(X, Y), node(X;Y), u(L).
12 :- bidirected(X, Y, L), not hbidirected(X, Y), node(X;Y), u(L), X < Y.
13 :- not directed(X, Y, L), hdirected(X, Y), node(X;Y), u(L).
14 :- not bidirected(X, Y, L), hbidirected(X, Y), node(X;Y), u(L), X < Y.
15 % the following is only used when SCC accounting is enabled
16 :- edge1(X, Y), scc(X, K), scc(Y, L), K != L, sccsize(L, Z), Z > 1, not dag(K,L).
```

Listing 1: Clingo code for sRASL

```
    :~ directed(X, Y, L), no_hdirected(X, Y, W), node(X;Y), u(L). [W@1,X,Y]
    :~ bidirected(X, Y, L), no_hbidirected(X, Y, W), node(X;Y), u(L), X < Y.
        [W@1,X,Y]
    :~ not directed(X, Y, L), hdirected(X, Y, W), node(X;Y), u(L). [W@1,X,Y]
    :~ not bidirected(X, Y, L), hbidirected(X, Y, W), node(X;Y), u(L), X < Y.
        [W@1,X,Y]
```

Listing 2: Integrity constraints for turning sRASL algorithm into an optimization problem when they replace lines 11 through 14 in Listing 1

$X \to Y \in \mathcal{G}^u \iff X \overset{u}{\rightsquigarrow} Y \in \mathcal{G}^1$ where $\overset{u}{\rightsquigarrow}$ is a path of length u. Line 10 similarly defines bidirected edges in $\mathcal{G}^L$: $X \leftrightarrow Y \in \mathcal{G}^u \iff \exists Z, l : (X \overset{l}{\leftsquigarrow} Z \overset{l}{\rightsquigarrow} Y \in \mathcal{G}^1)$.

Lines 11 − 14 provide the core constraints, as they ensure that sRASL returns only $\mathcal{G}^1$ for which there exists u such that $\mathcal{G}^u = \mathcal{H}$. Line 16 adds the additional constraints based on impermissibility of cycles between SCCs. That is, if we consider each SCC as a *super-node*, Line 16 ensures that the edges of the directed acyclic graph (DAG) connecting SCCs in $\mathcal{H}$ are not violated in the outputs.

If sRASL initially returns the empty set (i.e., there are no suitable $\mathcal{G}^1$), then it is possible to run sRASL in an optimization mode instead to find optimal (though not perfect) outputs (see Section 4.5 for details). One potential reason for $[\![\mathcal{H}]\!] = \emptyset$ is statistical noise or other errors in estimating or specifying $\mathcal{H}$.[10] In such cases, sRASL finds the set of $\mathbf{G}^1$ that are, for some u, closest to $\mathcal{H}$ by the objective function:

$$\mathcal{G}^{1*}, u^* \in \arg\min \sum_{e \in \mathcal{H}} I[e \notin \mathcal{G}^u] \cdot w(e \in \mathcal{H}) + \sum_{e \notin \mathcal{H}} I[e \in \mathcal{G}^u] \cdot w(e \notin \mathcal{H}), \tag{1}$$

where the indicator function $I(c) = 1$ if the condition holds and zero otherwise. $w(e \in \mathcal{H})$ indicates the importance (i.e., reliability) of edge e; $w(e \notin \mathcal{H})$ indicates the reliability of the absence of an edge. Since $\mathcal{H}$ is an undersampled graph, it consists of directed and bidirected edges. We thus implement both $w(e \in \mathcal{H})$ and $w(e \notin \mathcal{H})$ as two pairs of $n \times n$ matrices, one pair for existence and absence of directed edges, and one pair for bidirected edges. To learn the optimal graph at the true causal timescale, for every $\mathcal{G}^1$ in the solutions set, the corresponding $\mathcal{G}^u$ is compared to the input $\mathcal{H}$ and penalized for the difference according to weights representing the reliability of the measurement timescale estimates.

In order to incorporate Equation 5 in Listing 1, we replace its exact integrity constraints (Lines 11-14) with the optimization formulation Gebser et al. (2011) in Listing 2. In Listing 2 we specify a weight

[10] Note, among all possible graphs that have a combination of both directed (2^{n^2}) and bidirected ($2^{\binom{n}{2}}$) edges only a fraction may be obtained by undersampling a $\mathcal{G}^1$.

for each edge (or lack there of) in $\mathcal{H}$ using `W` and the importance of these weights can be specified for each integrity constraint using the `W@i` syntax with `i` being the importance.

3.1 sRASL Completeness and Correctness

sRASL exhibits significant improvements in computation time, so it is important to show that we do not lose generality or theoretical guarantees. We demonstrate correctness and completeness using the notion of a *direct encoding* of the problem (i.e., the space of solutions is fully characterized, and any non-solution violates a constraint). We first prove (Appendix A) that we have provided a direct encoding:

Theorem 4. *Listing 1 is a direct encoding of the undersampling problem.*

`Clingo` is a complete solver, based on CDNL (Conflict-Driven Nogood Learning) Drescher & Walsh (2011), itself based on CDCL (Conflict-Driven Clause Learning) Marques Silva & Sakallah (1996); Marques-Silva & Sakallah (1999). Hyttinen et al. (2014)[Theorem 2] and Hyttinen et al. (2013)[Section 5.2] show that, if the ASP encoding is the direct encoding of the problem, then ASP will produce the complete set of solutions in the infinite sample space limit. In other words, Theorem 5 implies: since our algorithm yields at least one sound solution, `Clingo` will produce all possible solutions. Therefore, soundness results in completeness. That is, sRASL's success is not due to heuristics or some incomplete or not-everywhere-correct algorithmic step.[11]

4 Results

A major virtue of sRASL is its empirical performance, so we now consider a range of simulations (to ensure known ground truth) to understand this performance in more detail. For these experiments, we used `Clingo` in parallel mode using 10 threads and computing on `AMD EPYC 7551` CPUs. To cope with the multiple repeated calculations and hundreds of graphs we have tested per parameter setting all experiments were run on a `slurm` cluster which submits jobs to one of the 19 machines on the same network. Each of the 19 nodes was equipped with 64 cores and 512 GB of RAM.

4.1 Comparing sRASL vs. RASL

We first compare sRASL with the existing RASL method (Figure 2). We generated 100 6-node SCCs for each density in $[0.2, 0.25, 0.3]$, and then undersampled each graph by $2, 3$, and 4. We used 6-node graphs as RASL struggles to handle larger graphs in reasonable time and space Plis et al. (2015a). Each column of Figure 2 consists of graphs of approximately same density (increasing density from left-to-right), and subcolumns represent different undersample rates (for that density). As Figure 2 shows, sRASL is typically three orders of magnitude faster than RASL, even on relatively small graphs.

4.2 Comparing Graph Size

It is perhaps unsurprising that sRASL runs much faster than RASL, as sRASL uses an ASP solver (which were previously known to yield faster algorithms Hyttinen et al. (2017)). We next wanted to see just how much larger the graphs could be. More generally, we aimed to better understand how sRASL's computational performance scales with the number of nodes for single-SCC graphs. The focus on single SCCs is motivated by the theoretical need to understand the size-speed tradeoff, and also scientific applicability since many real-world systems consist of tightly coupled factors with many feedback loops (i.e., they are a single SCC). We consider multiple-SCC graphs in later subsections.

We generated 50 random single-SCC graphs each of $8, 16$ and 32 nodes, all with average degree of 1.4 outgoing edges per node. We then undersampled each graph by $2, 3$ and 4, and used each individual undersampled graph as input to sRASL. We used a 24-hour timeout (i.e., we stopped an sRASL run if it did not finish in 24 hours). Figure 3 shows the increasing computational costs as both

[11] Simulation testing provides further evidence. We found that sRASL and RASL produced identical outputs for 1000 different input graphs, and RASL is known to be correct and complete Plis et al. (2015a)[Theorem 3.6].

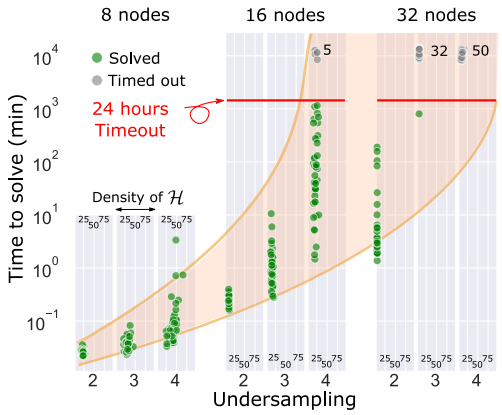

Figure 3: Time behavior of graphs of size 8, 16 and 32. The time out for this experiment indicated by the red line was 24 hours. *Green* dots represent graphs that has been computed within the 24-hours window. *Gray* represent graphs that could not be fully computed within 24−hours window.

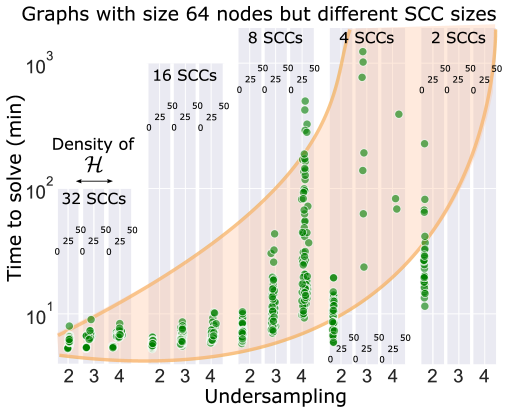

Figure 4: Time behavior of graphs of size 64 with various sub SCC sizes. The time out for this experiment was 24 hours (1440 Minutes).

number of nodes and undersample rate increase. Notably, sRASL was able to learn $[\![\mathcal{H}]\!]$ for 32-node single-SCC graphs, though it reached timeout for all $\mathcal{H}$ at $u = 4$ 32-node graphs. That is, for low u, sRASL scales to much larger single-SCC graphs than RASL.

4.3 Comparing SCC Size

The other major innovation of sRASL is incorporation of constraints derived from the SCC structure. We thus investigated the performance of sRASL on large, structured, multiple-SCC graphs. Many real-world systems exhibit some degree of modularity, where there are dense or feedback connections within a module or subsystem, and relatively sparser connections between modules or subsystems. In theory, sRASL should perform well on these kinds of structures since it incorporates SCC-based constraints. Please refer to Appendix B for an ablation study on effect of using additional constraints for SCC structures.

We tested the value of SCC-based constraints using graphs with 64 nodes that differed in their SCC structure. Specifically, we randomly generated 50 graphs each of: 32 size-2 SCCs; 16 size-4 SCCs; 8 size-8 SCCs; 4 size-16 SCCs; or 2 size-32 SCCs. We then undersampled each graph by $u = 2, 3$, or 4, and ran sRASL (again with a 24-hour timeout).

Figure 4 shows the computation time for these graphs, with increasing SCC size (and decreasing number of SCCs) from left to right. The first key observation is that sRASL successfully found $[\![\mathcal{H}]\!]$ for 64-node graphs, at least when there was some internal structure. Second, and relatedly, we observe a wide range of computation times for these graphs, even though all had the same number of nodes (64). We clearly see the impact of SCC structure, as sRASL was dramatically faster when there were many small SCCs, rather than a few large SCCs. The results in Figure 3 might seem to suggest an "upper bound" around 30 nodes for sRASL. But the results in Figure 4 make it clear that any potential "upper bound" is primarily on the number of nodes *in the SCCs*, rather than the total number of nodes in the graph.

4.4 Comparing Graph Size With Constant SCC Size

The previous results suggest that sRASL might be able to solve much larger graphs, as long as the SCCs are not overly large. More generally, the previous simulations showed that sRASL's computational cost scales (at least) exponentially in the *size* of the SCC, but did not reveal how it scales in the *number* of SCCs.

We again generated 50 different graphs for each of several settings. We considered SCCs with 7, 8, and 10 nodes, and varied the number of SCCs within the graph (again for $u = 2, 3$, and 4). Figure 5

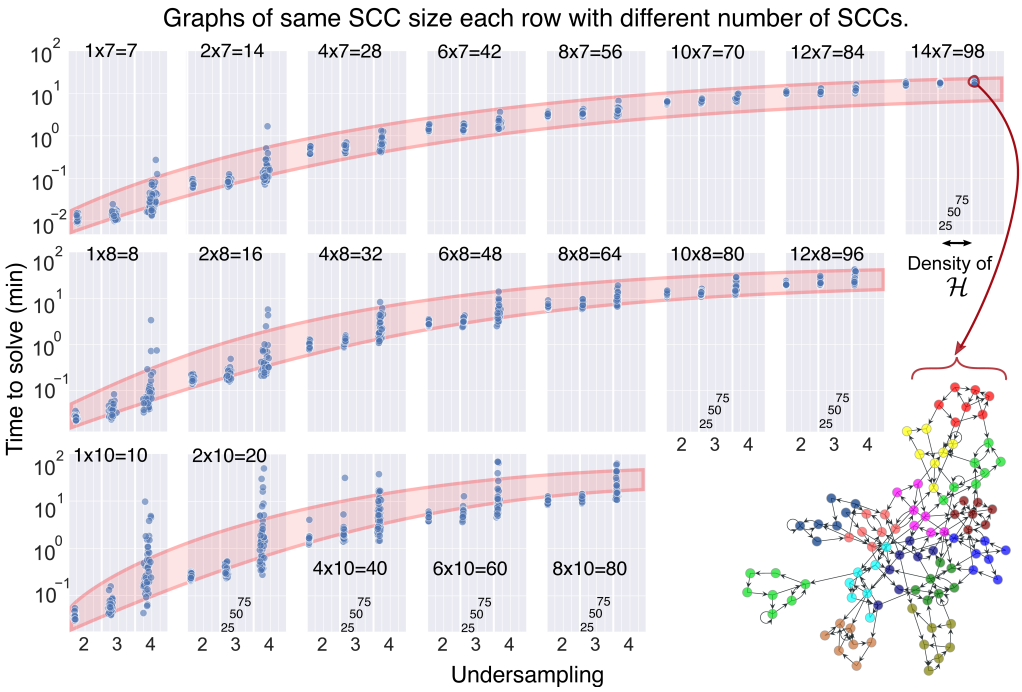

Figure 5: Time behaviour of graphs with the same SCCs sizes but with multiple number of SCCs. Top row graphs of SCC size 7 with $1, 2, ..., 14$ number of SCCs. Middle row graphs of SCC size 8. Bottom row graphs of SCC size 10.Bottom right corner is an example of a structured graph with 98 nodes structured as 14 SCCs of size 7. Each color represents one Strongly Connected Component.

shows the computational cost of sRASL, where each row includes graphs with SCCs of the same size, but the number of SCCs increasing from left-to-right. The critical observation here is that the time complexity grows approximately *linearly*, rather than exponentially (or worse). For example, the graph shown in Figure 5 has 98 nodes, but sRASL successfully computes $[\![\mathcal{H}]\!]$ in approximately 20 minutes. (Recall that RASL took 17 hours to compute a graph with only 6 nodes.)

This simulation demonstrates that sRASL is usable on relatively large graphs, as long as there is appropriate internal structure. One might worry, though, whether real-world systems do not have the right structure. If we consider fMRI (brain) data, Sanchez-Romero et al. (2019) recently aggregated a number of simulations of realistic causal graphs for brain processes studied with fMRI, and the largest SCC in these widely-accepted models has only seven nodes. Moreover, typical brain parcellations contain $50 - 100$ regions (= nodes), and sRASL can easily handle graphs with 100 nodes if the SCC size is in the $8 - 10$ range.

The results in this subsection suggest that we could potentially find $[\![\mathcal{H}]\!]$ for each larger graphs, as long as they were composed of reasonably-sized SCCs. However, we found that the `Clingo` language and solver seems to be limited in the number of atoms that it can handle. In our simulations, graphs of size 100 seem to be the limit for `Clingo` to handle all the predicates. An open question is whether sRASL can be optimized to produce fewer predicates (or `Clingo` improved to handle more atoms).

4.5 Optimization

Finally, we explored the optimization capability of `Clingo`. Recall that sometimes $[\![\mathcal{H}]\!] = \emptyset$ due to statistical errors or other noise in learning $\mathcal{H}$. `Clingo` can solve an optimization problem based on user-specified weights and priorities, and output a single solution with minimum cost function (along with u for this solution). In particular, we can use `Clingo` to find $\mathcal{G}^1$ whose $\mathcal{G}^u$ (for some u) are closest (relative to the edge weights) to $\mathcal{H}$.[12]

[12] If $[\![\mathcal{H}]\!] \neq \emptyset$, then this optimization will return a graph from $[\![\mathcal{H}]\!]$,

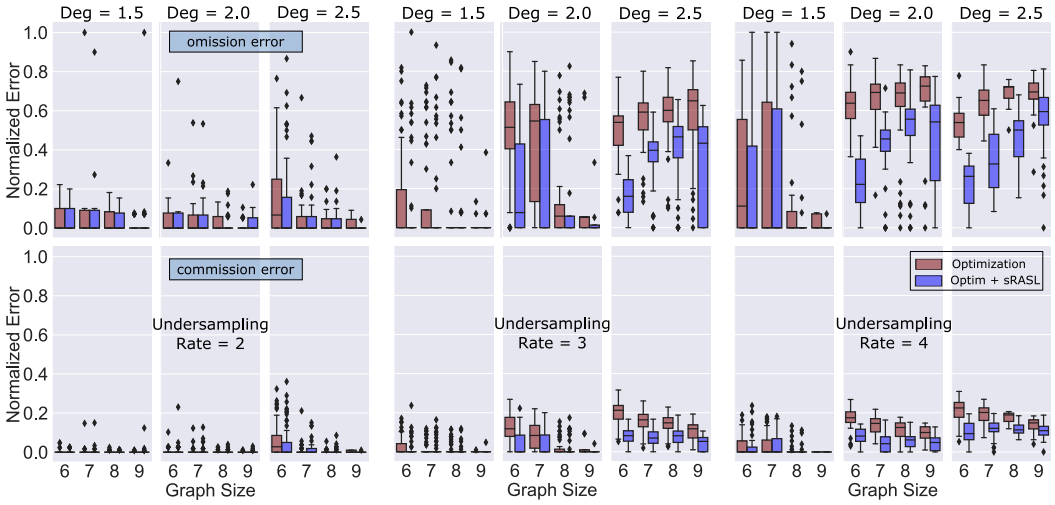

Figure 6: The omission (top) and commission (bottom) error of different graph sizes and undersampling of two, three and four from left to right.

In this simulation, we first randomly generate $\mathcal{G}^1$ and undersample it to a random u to get $\mathcal{G}^u = \mathcal{H}$ such that $[\![\mathcal{H}]\!] \neq \emptyset$. We then assign weights to the edges of $\mathcal{H}$ and randomly break one edge from it. We then run sRASL on this "broken" $\mathcal{H}$ to learn a suitable $\mathcal{G}^1$. Red bars in Figure 6 show the edge omission and commission errors for this approach. We see that, except for high undersamplings, the optimization capability of `Clingo` can be used to frequently retrieve the true $\mathcal{G}^1$; that is, this version of sRASL is robust to small errors in $\mathcal{H}$ in many settings.

A more complex approach to finding suitable solutions is to first run the optimization method to identify a solution $\mathbf{G}^1_{opt}$ and undersample rate u_{opt}. We can then undersample this solution $\mathbf{G}^1_{opt}$ by u_{opt} to get $\mathbf{G}^u_{opt}$. We then use sRASL to obtain $[\![\mathbf{G}^u_{opt}]\!]$ (i.e., the full equivalence class of the undersampled graph that is "nearest" to $\mathcal{H}$). We then compute the error based on the minimum error among all $\mathbf{G}^1 \in [\![\mathbf{G}^u_{opt}]\!]$; that is, we ask whether the true graph was actually found. This approach is motivated by the intended use of sRASL by domain scientists, where the final decision on which graph in the equivalence class better suits the question is made by the scientist using the algorithm. Blue bars in Figure 6 show that this more complex method provides improved performance compared to regular optimization.

5 Conclusion and Discussion

Real-world scientific problems frequently involve measurement processes that operate at a different timescale than the causal structure of the system under study. As causal learning and analysis methods are increasingly used to address societal and policy challenges, it is increasingly critical that we use methods that reveal usable information (while also being clear when we *cannot* infer some information). Obviously, like any method, sRASL could yield information that is misused, but the aim here is to provide another useful tool in the scientists' policy-makers' toolboxes. If measurements occur at a slower rate than the causal influences, then causal discovery from those undersampled data can yield highly misleading outputs. Multiple methods have been developed to infer aspects of the underlying causal structure from the undersampled data/graph. However, the assumptions or computational complexities of those algorithms make them unusable for most real-world challenges. In this paper, we have developed and tested sRASL, a novel algorithm that is less subject to those same limitations. More specifically, sRASL provides all consistent solutions (without knowledge of exact undersampling rate) for large (100-node) graphs in a usable amount of time. sRASL also shows reasonable robustness to statistical error in the estimated graph by finding the closest consistent solution. Future research will focus on application of sRASL to actual neuroimaging data, and extensions to situations with multiple measurement modalities.

References

David Danks and Sergey Plis. Learning causal structure from undersampled time series. In *NIPS Workshop on Causality*, volume 1, pp. 1–10, 2013.

David Danks and Sergey Plis. Amalgamating evidence of dynamics. *Synthese*, 196(8):3213–3230, 2019.

Christian Drescher and Toby Walsh. Conflict-driven constraint answer set solving with lazy nogood generation. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of understandability. In *Enterprise, Business-Process and Information Systems Modeling*, pp. 353–366. Springer, 2009.

Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The Potsdam answer set solving collection. *Ai Communications*, 24(2): 107–124, 2011.

M Gelfond and V Lifschitz. The stable model semantics for logic programming. ICSLP, 1988.

Mingming Gong, Kun Zhang, Bernhard Schoelkopf, Dacheng Tao, and Philipp Geiger. Discovering temporal causal relations from subsampled data. In *International Conference on Machine Learning*, pp. 1898–1906. PMLR, 2015.

Antti Hyttinen, Patrik O Hoyer, Frederick Eberhardt, and Matti Jarvisalo. Discovering cyclic causal models with latent variables: A general SAT-based procedure. *arXiv preprint arXiv:1309.6836*, 2013.

Antti Hyttinen, Frederick Eberhardt, and Matti Järvisalo. Constraint-based Causal Discovery: Conflict Resolution with Answer Set Programming. In *UAI*, pp. 340–349, 2014.

Antti Hyttinen, Sergey Plis, Matti Järvisalo, Frederick Eberhardt, and David Danks. A constraint optimization approach to causal discovery from subsampled time series data. *International Journal of Approximate Reasoning*, 90:208–225, 2017.

V Lifschitz. The stable model semantics for logic programming, 1988.

Joao P Marques-Silva and Karem A Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

J.P. Marques Silva and K.A. Sakallah. GRASP-A new search algorithm for satisfiability. In *Proceedings of International Conference on Computer Aided Design*, pp. 220–227, 1996. doi: 10.1109/ICCAD.1996.569607.

Joris M Mooij and Tom Claassen. Constraint-based causal discovery using partial ancestral graphs in the presence of cycles. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1159–1168. PMLR, 2020.

Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of mathematics and Artificial Intelligence*, 25(3):241–273, 1999.

MW Oram and DI Perrett. Time course of neural responses discriminating different views of the face and head. *Journal of neurophysiology*, 68(1):70–84, 1992.

Judea Pearl et al. Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19: 2, 2000.

Sergey Plis, David Danks, Cynthia Freeman, and Vince Calhoun. Rate-agnostic (causal) structure learning. In *Advances in neural information processing systems*, pp. 3303–3311, 2015a.

Sergey Plis, David Danks, and Jianyu Yang. Mesochronal structure learning. In *Uncertainty in artificial intelligence: proceedings of the... conference. Conference on Uncertainty in Artificial Intelligence*, volume 31. NIH Public Access, 2015b.

Thomas Richardson and Peter Spirtes. Ancestral graph Markov models. *The Annals of Statistics*, 30 (4):962–1030, 2002.

Ruben Sanchez-Romero, Joseph D Ramsey, Kun Zhang, Madelyn RK Glymour, Biwei Huang, and Clark Glymour. Estimating feedforward and feedback effective connections from fMRI time series: Assessments of statistical methods. *Network Neuroscience*, 3(2):274–306, 2019.

Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer New York, 1993. doi: 10.1007/978-1-4612-2748-9. URL https://doi.org/10.1007/978-1-4612-2748-9.

Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, Prediction, and Search*. MIT press, 2000.

Sofia Triantafillou, Ioannis Tsamardinos, and Ioannis Tollis. Learning causal structure from overlapping variable sets. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 860–867. JMLR Workshop and Conference Proceedings, 2010.

Jiji Zhang. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research*, 9: 1437–1474, 2008.

A Appendix

We start with proving some results used in conversion of the DBN structures to their compressed graph representations.

Lemma 1. *For all u, G_u contains no directed edges between variables at the same time step.*

Proof. $u = 1$ holds by assumption for G_1. For $u > 1$, every directed edge corresponds to a directed path of length u in G_1. Since all directed edges in G_1 are from $t-1$ to t (or more generally, from $t-k$ to $t-(k+1)$), every directed path in G_1 is from an earlier time step to the current one. Hence, no directed edge in G_u can be from V_i^t to V_j^t. □

Lemma 2. *If the Markov order of G_1 is 1, then the Markov order of all G_u is also 1 (relative to measurement at rate u).*

Proof. The Markov order of a dynamic causal graph is the smallest m such that $\mathbf{V}^t$ is independent of $\mathbf{V}^{t-r}$ given $\mathbf{V}^{t-1}, \ldots, \mathbf{V}^{t-m}$ for all $r > m$. If the Markov order of G_1 is 1, then all paths from $\mathbf{V}^{t-r}$ to $\mathbf{V}^t$ must be blocked by $\mathbf{V}^{t-1}$ for $r > 1$. Since graphical structure is replicated across timesteps, it follows that all paths from $\mathbf{V}^{t-r}$ to $\mathbf{V}^t$ must be blocked by $\mathbf{V}^{t-u}$ for $r > u$. Therefore, the Markov order of G_u is u, which corresponds to Markov order 1 for measurements at rate u. □

The following theorem demonstrates correctness of our ASP algorithm.

Theorem 5. *Listing 1 is a direct encoding of the undersampling problem.*

Proof. We will prove this by contradiction. Let us call the undersampled input graph to the algorithm $\mathcal{H}$, considering that is the undersampled version of a graph $\mathbf{G}^1_{true}$ at rate u_{true}. By definition, every directed edge in $\mathcal{H}$ corresponds to a path of length u_{true} in $\mathbf{G}^1_{true}$. Similarly, every bidirected edge in $\mathcal{H}$ corresponds to an unobserved common cause fewer than u_{true} timesteps back(refer to Section 2 for exact definition). Line $7-11$ in Listing 1 considers all such $\mathbf{G}^1$s without exclusion. Let us call the set all the pairs of graphs and corresponding undersampling rates u described by Listing 1 $\mathbf{S}$.

Let us assume there is a pair $\mathbf{G}^1_a$ and u_a that is in $\mathbf{S}$ but if we undersample $\mathbf{G}^1_a$ by u_a, let us call it $\mathbf{G}^u_a$, will not be the same as $\mathcal{H}$. If $\mathbf{G}^u_a$ has an extra directed(bidirected) edge, this will contradict with line 12(13) of Listing 1. Similarly, if $\mathcal{H}$ has a directed(bidirected) edge that in not present in $\mathbf{G}^u_a$, it will contradict with line 14(16). Therefore, Listing 1 is a direct encoding of the undersampling problem. □

B The Effects of Accounting for SCCs In sRASL

In this section, we show the results of additional experiments on the effects of accounting for strongly connected components (SCCs) when the graph has a modular structure (i.e., consists of several interconnected strongly connected components). For this experiment, we generated 50 random graphs sized 8 to 15 with multiple SCCs as described in Table1. Then on the same set of graphs, we ran sRASL once with using our additional constraints for SCC structures and once without accounting for the modular structure. We limited the computational resources available to each run to 24 hours time cutoff with a RAM limit of 50 GB. The results presented in Figure8 show that using additional constraints to account for SCC structure dramatically reduces the time and memory needed to compute equivalent classes for undersampled graphs. Furthermore, the difference between time and memory requirements to solve for these graphs with and without constraints for SCCs increases for larger graphs as the computational requirements for the latter grow at a much faster pace. This result allows us to handle much larger graphs as shown in Figure 5 of the main paper.

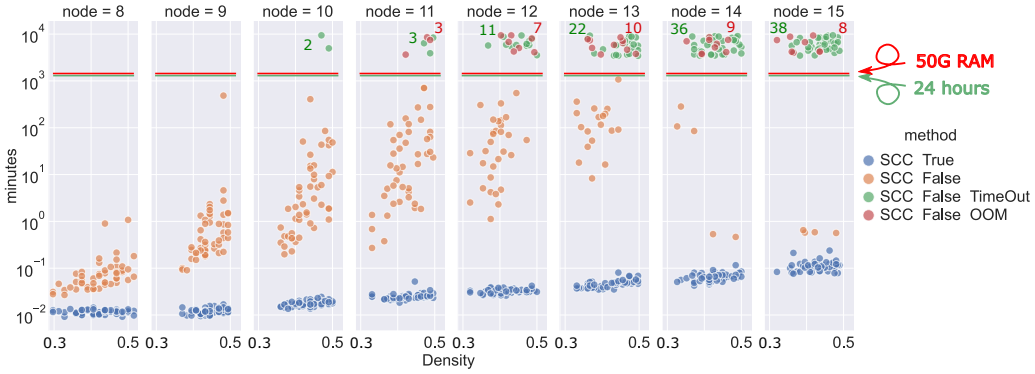

Figure 7: Time behavior of the same set of graphs when solved with and without accounting for additional constraints accounting for the SCC structure. While sRASL most of the 15-node graphs in a 24 hours period without the SCC constraints due to either timeout or Out Of Memory error(OOM), the longest it takes to solve a 15-node graph with SCC constraints is 14 seconds. None of the graphs failed to compute the complete equivalence class within the time and memory allocated when solved accounting for the SCC structure.

Table 1: Number of SCCs and nodes per SCC of the graphs in the benchmark dataset

Num Nodes	8	9	10	11	12	13	14	15
Num SCCs	2	3	3	3	3	3	3	3
SCC Sizes	4,4	3,3,3	3,3,4	3,4,4	4,4,4	4,4,5	4,5,5	5,5,5

C Brief Introduction on `clingo` and Answer Set Programming (ASP)

`clingo` Gebser et al. (2011) combines a grounder `gringo` and a solver `clasp`. `clingo` is a declarative programming system based on logic programs and their answer sets, used to accelerate solutions of computationally involved combinatorial problems. The grounder converts all parts of a `clingo` program to "atoms," (grounds the statements) and the solver finds "stable models." In ASP, the answer set is a model in which all the atoms are derived from the program and each "answer" is a stable model where all the atoms are simultaneously true.

A general `clingo` program includes three main sections, which we show below using our algorithm as an example:

1. **Facts:** these are the known elements of the problem. For example, the input to Listing 1 is a graph for which we know the edges. A directed edge from node 1 to node 5 is in $\mathcal{H}$ translates to *`hdirected(1,5)`* (line 1) or if node 1 is part of the SCC number 2, we state this fact in `clingo` by *`scc(1,2)`* (line 2).

Figure 8: A knowledge of a definite presence of an edge in $\mathcal{G}^1$ between, for example, nodes 3 and 4, i.e. $V_3^t \rightarrow V_4^{t+1}$, can be easily encoded by adding ‘ `edge1(3,4).` ‘ to Listing 1. In this experiment, we have added knowledge about a pair of arbitrary selected edges of $\mathcal{G}^1$ to the problem specification (orange dots) and compared the run time with the ASP specification that does not include this additional information about the solution (blue dots). The time out for the new computation was set to 1 hours and the examples were all the same as the ones already shown in Figure 1. The speed up with the additional constraints is clearly visible on the plots.

2. **Rules:** much like an `if-else` statement, a rule in `clingo` consists of a body and a head, formatted as *`head :- body.`* If all the literals in the body are true, then the head must also be true. Rules can include variables (starting with capital letters), and they are used to derive new facts after grounding. For example:

```
directed(X, Y, 1) :- edge1(X, Y).
```
(2)

means that for any instantiations of the variables X and Y, if we have an edge from X to Y, there is a directed path from X to Y of length 1. Before this line, if the model contained the fact *`edge1(2,3)`*, this line would generate a new fact: *`directed(2,3,1)`*.

Another type of rule is the “choice rule” that describes all the possible ways to choose which atoms are included in the model. For example, in line 5 of Listing 1 we used a choice rule to state that the undersampling rate *`u`* can be anything from 1 to *`maxu`*. The cardinality constraint:

```
{u(1..20)}.
```
(3)

will generate 2^{20} different models (they will not all actually be generated if they conflict with other predicate in each model, or else it would not be possible). In each of these 2^{20} models, one subset of all possible atoms generated with this choice rule exists (ϕ, `{u(1)}`, `{u(1), u(2)}`, ...). An example of an unconstrained choice rule is line 6 in Listing 1, where we want to generate one model for each possible way edges can be present in a graph between two nodes X and Y. We can also limit the choice rule. In our problem, only one undersampling rate is present at each solution. We limit the cardinality constraint to have only one member in each model:

```
1 {u(1..20)} 1.
```
(4)

the 1 on the left is the minimum instantiations of this atom in the model and the 1 on the right is the maximum. Therefore, we only generate $\binom{20}{1} = 20$ models with this rule, namely one for each undersampling rate. Having several choice rules will multiply the number of generated models by each choice rule.

3. **Integrity Constraints:** if choice rules are to generate new models, integrity constraints are there to remove the wrong models from the answers set. More specifically, an integrity constraint is of the form:

```
:- L0, L1, ...  .
```
(5)

where literals $L_0, L_1, ...$ cannot be simultaneously positive. For example, in line 16 of Listing1, we have:

```
:- edge1(X, Y), scc(X, K), scc(Y, L), K != L,
        sccsize(L, Z), Z > 1, not dag(K,L).
```
(6)

for cases where the graph consists of several SCCs that are connected using a DAG. If the SCCs are connected by a cyclic directed graph, then the whole graph will become one big Strongly Connected Component. Integrity constraint 6 states that if there is not a directed edge from a node in SCC K to a node in SCC L as part of the initial DAG, there cannot be such *`edge1(X, Y)`* from node X to node Y, if node X is in SCC K and node Y is in SCC L.