CALDERA: Finding all significant de Bruijn subgraphs for bacterial GWAS

Hector Roux de Bézieux* Leandro Lima† Fanny Perraudeau* Arnaud Mary‡
Sandrine Dudoit§ Laurent Jacob‡

November 5, 2021

Abstract

Genome wide association studies (GWAS), aiming to find genetic variants associated with a trait, have widely been used on bacteria to identify genetic determinants of drug resistance or hypervirulence. Recent bacterial GWAS methods usually rely on k-mers, whose presence in a genome can denote variants ranging from single nucleotide polymorphisms to mobile genetic elements. Since many bacterial species include genes that are not shared among all strains, this approach avoids the reliance on a common reference genome. However, the same gene can exist in slightly different versions across different strains, leading to diluted effects when trying to detect its association to a phenotype through k-mer based GWAS. Here we propose to overcome this by testing covariates built from closed connected subgraphs of the De Bruijn graph defined over genomic k-mers. These covariates are able to capture polymorphic genes as a single entity, improving k-mer based GWAS in terms of power and interpretability. As the number of subgraphs is exponential in the number of nodes in the DBG, a method naively testing all possible subgraphs would result in very low statistical power due to multiple testing corrections, and the mere exploration of these subgraphs would quickly become computationally intractable. The concept of testable hypothesis has successfully been used to address both problems in similar contexts. We leverage this concept to test all closed connected subgraphs by proposing a novel enumeration scheme for these objects which fully exploits the pruning opportunity offered by testability, resulting in drastic improvements in computational efficiency. We illustrate this on both real and simulated datasets and also demonstrate how considering subgraphs leads to a more powerful and interpretable method. Our method integrates with existing visual tools to facilitate interpretation. We also provide an implementation of our method, as well as code to reproduce all results at `https://github.com/HectorRDB/Caldera_Recomb`.

*Pendulum Therapeutics, Inc. CA, USA
†European Bioinformatics Institute, Cambridge, UK
‡Univ. Lyon, Université Lyon 1, CNRS, Laboratoire de Biométrie et Biologie Évolutive UMR 5558, Lyon, France
§Department of Statistics and Division of Biostatistics, University of California, Berkeley, CA, USA

1 Introduction

Genome-wide association studies (GWAS) look for genetic variants whose presence or absence is associated with a trait of interest, such as the risk for a person to develop a disease, or the yield for a crop. They were originally used on human genomes using single nucleotide polymorphisms (SNPs) as genetic variants [Visscher et al., 2017]. While SNPs do capture most of the genetic variation in genomes that are *similar enough*, they can miss essential variants in other situations. For example, some bacterial species are known to have large accessory genomes, *i.e.*, sets of genes that are not present in every strain in the species. In spite of their name, some of these accessory genes play a central role for some traits of interest, such as antibiotic resistance. In *P. aeruginosa*, for instance, they account for 70% of known genetic determinants of resistance to amikacin [Jaillard et al., 2017]. In this context, k-mers—defined as all words of length k found in the genomes—have emerged as a popular alternative to SNPs to describe genetic diversity [Sheppard et al., 2013, Earle et al., 2016]. More specifically, bacterial GWAS often test the association between the trait of interest and the presence/absence of k-mers. A broad variety of genetic variants–ranging from SNPs to mobile genetic elements or translocations–cause the mutated strains to contain one or several specific k-mers. These GWAS are therefore able to capture any of these variants without requiring their prior identification or even definition. On the other hand, k-mer-based GWAS suffer from two important limitations. First, interpreting their result is notoriously tedious: any given k-mer can belong to several regions of the same genome, and conversely a gene causing the trait of interest can contain a large number of specific k-mers. Second, because a resistance-causing gene often exists in slightly different version, its k-mers are only present in a fraction of the resistant strains. As a consequence, these k-mers are less strongly associated with resistance than the gene itself.

Jaillard et al. [2018] proposed `DBGWAS` to help interpret the result of k-mer based GWAS using the De Bruijn graph (DBG [de Bruijn, 1946, Pevzner et al., 2001]), which connects overlapping k-mers. Several significant k-mers arising from a single polymorphic gene typically aggregate into a somewhat linear subgraph of the DBG (Figure 1), making their interpretation easier. However, `DBGWAS` still tests the individual k-mers of this subgraph separately, at the risk of missing causal genes whose presence is too diluted across different versions and therefore different k-mers.

Here we propose to test the association between the phenotype and a single covariate capturing the presence of any version of a gene—or any other potential genetic determinant. Concretely, this covariate indicates the presence of any k-mer among those represented in a connected subgraph of the DBG. More specifically, we choose only closed connected subgraphs (CCSs). A CCS is a connected subgraph such that adding any neighbor does not affect the created covariate. Non-closed subgraphs are represented by the same covariate as their closure, and are therefore redundant.

As any such subgraph may represent a causal variant that exists in several version in the dataset, we take an agnostic approach and test the association between the phenotype and one covariate for each connected subgraphs of the DBG. By contrast, `DBGWAS` relies on one covariate for each node of the DBG. This new approach has two potential issues: (1) the number of CCSs grows exponentially with the number of nodes in the DBG, making the task computationally intractable, and (2) adjusting for multiple testing over this very large number of tests leaves little to no power to detect associations. Our method addresses these two issues by using the concept of testability introduced by Tarone [1990]. Tarone's procedure controls the family-wise error rate (FWER) while disregarding a large number of *non-testable* hypotheses in its multiple testing correction. Intuitively, a covariate representing the presence of any k-mer among a growing set that corresponds to larger and larger CCS quickly becomes true for all samples. It thus cannot possibly be associated to any phenotype and can therefore be discarded without being tested or counted towards multiple testing correction. Testability provides a well-grounded and quantitative version of this intuition. Furthermore, since adding nodes to a connected subgraph can only increase the number of present k-mers in the corresponding covariate, we can develop a method that rapidly prunes non-testable CCSs, thereby solving the computational problem.

Testability has been used in similar situations, but most existing procedures are restricted to complete [Terada et al., 2013, Minato et al., 2014] or linear graphs [Llinares-López et al., 2015, 2017]. Sese et al. [2014] described an algorithm to test all CCSs: their algorithm combined the testability-based procedure `LAMP` of Terada et al. [2013] with `COIN` [Sese et al., 2010], an enumeration method for CCSs. While no experiment was provided in Sese et al. [2014], we found that a version of this algorithm using an improved version of `LAMP` [Minato et al., 2014, Llinares-López et al., 2015] could find all significant CCSs in graphs with up to

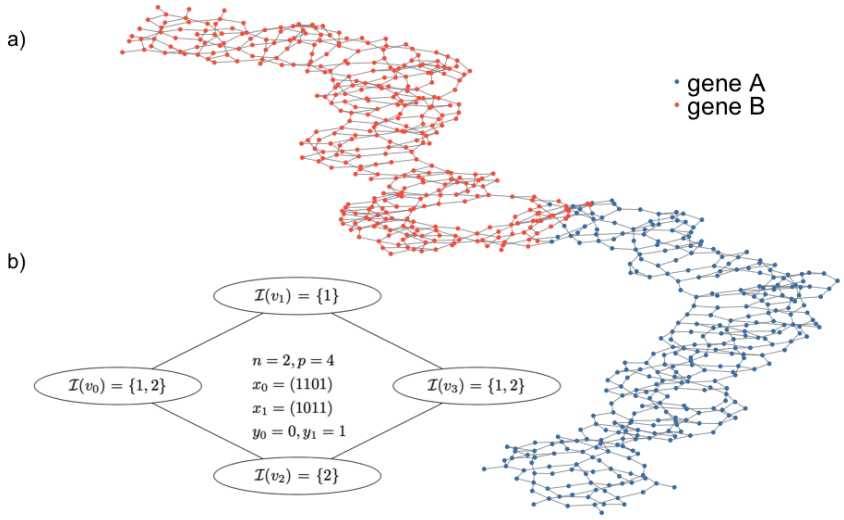


Figure 1: *Example of de Bruijn Graphs.* **a.** A general example with two genes, each with some variability, resulting in a mostly linear sequence only at the coarse level. More details in the result section. **b.** A simpler setting with 2 samples and 4 nodes. We have three CCSs: $\{v_1\}$, $\{v_2\}$ and $\{v_0, v_1, v_2, v_3\}$.

20,000 nodes in less than a day in only the most favorable settings. However the DBG built for typical bacterial GWAS involve millions of nodes, so a more scalable method is necessary to make CCSs testing amenable.

Our contributions are the following: We introduce a novel, provably complete and non-redundant enumeration scheme for CCSs called `CALDERA`. We also improve an existing pruning criterion for the Cochran-Mantel-Haenszel test. We show that combining these contributions with Tarone's testability-based procedure makes it possible to find all significant CCSs in a large graph, making it suited to bacterial GWAS. We provide the first implementation of a procedure finding all significant CCSs, along with a user-friendly visualization tool derived from `DBGWAS`. Finally, we demonstrate the advantages of `CALDERA` over competing methods on both simulated and real examples in term of computational speed, statistical power and biological interpretation.

Notation and goal for `CALDERA` We consider a set of n samples, $(x_i, y_i, c_i)_{i=1}^n$, where $x_i \in \{0,1\}^p$ are p binary covariates describing sample i, $y_i \in \{0,1\}$ denotes a binary phenotype, and $c_i \in \{1, \dots, J\}$ assigns sample i to one population among J. We denote n_1 and n_2 the number of samples such that $y_i = 0$ and 1 respectively. Furthermore, we consider an undirected unweighted connected graph $\mathcal{G} = (\mathcal{V}, E)$, where $\mathcal{V} = \{v_1, \dots, v_p\}$ and each vertex $v_j \in \mathcal{V}$ is associated with one of the p binary covariates represented in x. We denote by $\mathcal{I}(v_j) = \{i : x_i^j = 1\}$. For $i \in [1:n]$, we note $\mathbb{V}_i = \{v \in \mathcal{V} : i \in \mathcal{I}(v)\}$. For any connected subgraph $\mathcal{S} = (\mathcal{V}', E')$, such that $\mathcal{V}' \subseteq \mathcal{V}$ and $E' \subseteq E$, we let $\mathcal{I}(\mathcal{S}) = \bigcap_{v \in S} \mathcal{I}(v)$. Of note, this framework addresses both disjunctions and conjunctions, as the latter can simply be obtained by replacing each x_i by its complement. We now properly define the notion of closed connected subgraph and the closure operation (proof in Supplementary S-1.1).

Definition 1. *A connected subgraph $\mathcal{S}$ is closed if and only if there exists no edge $(v_1, v_2) \in E$ such that $v_1 \in \mathcal{S}$, $v_2 \notin \mathcal{S}$, and $\mathcal{I}(\mathcal{S} \bigcap \{v_2\}) = \mathcal{I}(\mathcal{S})$. We denote by $\mathcal{C}$ the set of all closed connected subgraphs of $\mathcal{G}$.*

Lemma 1. *For any connected subgraph $\mathcal{S}$ of $\mathcal{G}$, there exists a unique subgraph $\mathcal{S}' \in \mathcal{C}$ such that $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$ and $\mathcal{S} \subseteq \mathcal{S}'$, which we note $cl(\mathcal{S})$.*

Assuming that $(x_i, y_i, c_i)_{i=1}^n$ are n i.i.d. realizations of random variables $\mathbf{X}, \mathbf{Y}$, and $\mathbf{C}$, our objective is to test null hypotheses of the form $H_0^{\mathcal{S}}(\mathbf{X}, \mathbf{Y}, \mathbf{C}) : (\mathcal{I}(S) \perp \mathbf{Y}) | \mathbf{C}$ for all $\mathcal{S} \in \mathcal{C}$, while controlling the family-wise error rate (FWER, i.e., the chance of at least one Type I error or false positive) at level α. Translated in the

Variable	$i \in \mathcal{I}(\mathcal{S})$	$i \notin \mathcal{I}(\mathcal{S})$	**Rows totals**
$\mathbf{y_i = 1}$	$a_{\mathcal{S},j}$	$n_{1,j} - a_{\mathcal{S},j}$	$n_{1,j}$
$\mathbf{y_i = 0}$	$x_{\mathcal{S},j} - a_{\mathcal{S},j}$	$n_{2,j} - x_{\mathcal{S},j} + a_{\mathcal{S},j}$	$n_{2,j}$
Cols Totals	$x_{\mathcal{S},j}$	$n_j - x_{\mathcal{S},j}$	n_j

Table 1: Association table in community j for subgraph $\mathcal{S}$, used for the CMH test.

context of GWAS, we want to test the association between the pattern $\mathcal{I}(\mathcal{S})$ of each closed connected subgraph $\mathcal{S}$ with the phenotype $\mathbf{Y}$, while controlling for the population structure $\mathbf{C}$. We denote $H_0(\mathcal{S}) = H_0^{\mathcal{S}}(\mathbf{X}, \mathbf{Y}, \mathbf{C})$ in the remainder of this manuscript, as $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{C}$ are common for all elements of $\mathcal{C}$.

2 Background on significant subgraph detection using testability

Here, we describe the important concept of minimal attainable p-value proposed by Tarone [1990], and how it can be used to (i) retain more power that the Bonferroni procedure while still controlling the FWER and (ii) test more rapidly a large set of hypotheses. Both improvements come from the possibility to discard a large proportion of hypotheses without explicitly testing them and will be exploited in Section 3 to propose our procedure testing all CCSs in $\mathcal{C}$.

Minimal p-values are a property of discrete tests. For example, Fisher's exact test [Fisher, 1922] relies on a 2×2 contingency table, whose margins would describe in our case the number of sensitive and resistant bacteria and the number of bacteria whose genome contains or not a genetic variant. Given the margins of this table, only a finite number of cell count values are possible and Fisher's test can only lead to a finite number of values, the smallest of which is strictly positive (Fig S1). Importantly, this minimal attainable p-value $p^\star$ is entirely determined by the margins of the contingency table: given these margins, $p^\star$ is the minimum over a finite number of possible partitions, and is independent from the actual observed cell counts. Intuitively, strongly imbalanced margins (*e.g.*, variants that are present in a very large proportion of samples) cannot possibly lead to small p-values, no matter how the table is filled (*i.e.*, how the few samples that do not have the variant are distributed among resistant and sensitive phenotypes).

2.1 Using minimal attainable p-values for a tighter FWER control

The family-wise error rate is the probability to incorrectly reject at least one null hypothesis. When testing N of them and rejecting those whose p-value p_i is smaller than a threshold δ, $\text{FWER}(\delta) = \mathbb{P}\left(\bigvee_{i=1}^{N}(p_i \leq \delta)\right)$, where $\mathbb{P}$ is taken over the N null distributions $\left(H_0^i\right)_{i=1}^N$. The Bonferroni correction [Bonferroni, 1936] is a common procedure to control the FWER at a level α. It is motivated by a simple union bound: as $\text{FWER}(\delta)$ is upper-bounded by $\sum_{i=1}^{N} \mathbb{P}_{H_0^i}(p_i \leq \delta)$ and since by definition $\mathbb{P}_{H_0^i}(p_i \leq \delta) \leq \delta$, controlling each individual tests at level $\delta = \frac{\alpha}{N}$ makes the FWER upper-bounded by α. Tarone [1990] sharpens this bound, by using the fact that $p_i^\star > \frac{\alpha}{N}$ for some hypotheses. Since by definition $p_i \geq p_i^*$, the corresponding term $\mathbb{P}\left(p_i \leq \frac{\alpha}{N}\right)$ is exactly 0. Therefore, the FWER is actually controlled at level $\frac{m\alpha}{N} \leq \alpha$ where m is the number of testable hypotheses, for which $p_i^\star \leq \frac{\alpha}{N}$. This suggests that using a larger threshold δ than the Bonferroni $\frac{\alpha}{N}$ could still control the FWER at level α—while rejecting more hypotheses and therefore increasing power. Choosing the largest such δ is not a trivial task, as increasing the threshold also decreases the number of non-testable hypotheses. Let $m(k)$ be the number of testable hypotheses at level $\frac{\alpha}{k}$, *i.e.* such that $p^\star < \frac{\alpha}{k}$. In the worst case, $m(k) = N$ and we recover the Bonferroni procedure. By contrast, we define k_0 as the smallest k such that $m(k) \leq k$. The largest threshold guaranteeing $\text{FWER}(\delta) \leq \alpha$ is $\delta = \frac{\alpha}{k_0}$.

2.2 Using minimal attainable p-values to efficiently explore $\mathcal{C}$

Provided that enough CCSs have sufficiently large $p^\star$, Tarone's procedure could therefore address the loss of power incurred when exploring $\mathcal{C}$. However, naively finding k_0 requires to compute the minimal p-values for all $|\mathcal{C}|$ CCSs and iterate through these minimal p-values to adjust the threshold, leaving the computational problem unsolved. A more efficient strategies has been introduced to compute k_0 [Llinares-López et al., 2015, Minato et al., 2014]: starting from $k = 1$ a set $\mathcal{R}$ of *testable* hypotheses, *i.e.*, of elements with $p^\star < \frac{\alpha}{k}$ is grown. When $|\mathcal{R}|$ becomes larger than k, k is incremented to $|\mathcal{R}|$. All hypotheses that are not testable

anymore under the new threshold—*i.e.*, such that $\frac{\alpha}{|\mathcal{R}|} \leq p^\star < \frac{\alpha}{k}$—are removed from $|\mathcal{R}|$, and the exploration continues until the point where all testable hypotheses are in $\mathcal{R}$ and $k = k_0$. This strategy finds k_0 in a single enumeration of all tests, but still requires to compute all minimal p-values, which would not be feasible in our case. However, this search algorithm is also well suited to pruning strategies—a fact already used in [Llinares-López et al., 2015, Minato et al., 2014]. Let $p^\star(\mathcal{S})$ be the minimal p-value associated with $H_0(\mathcal{S})$ for a CCS $\mathcal{S}$. Assuming that for some pairs of subgraphs $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{S}_1 \subseteq \mathcal{S}_2 \Rightarrow p^\star(\mathcal{S}_1) \leq p^\star(\mathcal{S}_2)$, we can stop exploring all subgraphs including $\mathcal{S}_1$ as soon as $\mathcal{S}_1$ itself is found non-testable. This monotonicity property is verified when using Fisher's exact test to test $H_0(\mathcal{S})$: provided that $|\mathcal{I}(\mathcal{S})| \geq \max(n_1, n_2)$, $p^\star$ is strictly increasing in $|\mathcal{I}|$, and adding nodes to $\mathcal{S}$ can only increase $|\mathcal{I}|$ (Figure S2). Our main contribution, presented in Section 3 will be an efficient exploration algorithm for $\mathcal{S}$, which is well suited to pruning.

2.3 Controlling for a categorical covariate: the CMH test

When testing for associations, controlling for confounders is essential to avoid spurious discoveries. This is particularly important in bacterial GWAS, where strong population structures can lead to large sets of clade-specific variants to be found associated with a phenotype. The Cochran-Mantel-Haenszel (CMH) test can be used to test associations of two binary variables while controlling for a third categorical variable. It relies on J two-by-two association tables such as the one in Table 1, with $j \in \{1, \ldots, J\}$, $a_{\mathcal{S},j} = \left|\{i : y_i = 1, i \in \mathcal{I}(\mathcal{S}), c_i = j\}\right|$, $x_{\mathcal{S},j} = \left|\{i : i \in \mathcal{I}(\mathcal{S}), c_i = j\}\right|$ and $n_{1,j} = \left|\{i : y_i = 1, c_i = j\}\right|$.

Like Fisher's exact test, the CMH test is done conditional on all margins $(x_{\mathcal{S},j}, n_{1,j}, n_{2,j})_{j=1}^{J}$. Papaxanthos et al. [2016] furthermore demonstrated that its minimal p-value could be computed in $O(J)$ (Supplementary S-1.5) using the margins. However, the minimal p-value of the CMH test does not verify the monotonicity property $\mathcal{S}_1 \subseteq \mathcal{S}_2 \Rightarrow p^\star(\mathcal{S}_1) \leq p^\star(\mathcal{S}_2)$ which is required to prune while exploring $\mathcal{C}$. Papaxanthos et al. [2016] introduced the envelope, a lower bound on $p^\star(\mathcal{S})$, which verifies the monotonicity property. It can also be computed in $O(J \log(J))$ for all $\mathcal{S}$ such that, for all categories j, $x_{\mathcal{S},j} \geq \max(n_{1,j}, n_{2,j})$. This allows for a valid pruning strategy. The condition on $x_{\mathcal{S},j}$ is the CMH analogous of the $|\mathcal{I}(\mathcal{S})| \geq \max(n_1, n_2)$ condition of Fisher's test, and can decrease the number of prunable subgraphs as it must be verified for all J groups.

3 Speeding up the detection of all significant CCSs with `CALDERA`

We are now ready to present our contributions for scalable detection of significants elements in $\mathcal{C}$: an efficient exploration algorithm and an improved envelope for the CMH test, allowing for more pruning in the presence of imbalanced populations.

3.1 Critical properties for a fast, Tarone-aware enumeration of $\mathcal{C}$

We exploit several factors to provide a fast exploration of $\mathcal{C}$. First, we ensure that it is non-redundant, *i.e.*, that each element of $\mathcal{C}$ is enumerated exactly once, by defining a tree whose nodes are the elements of $\mathcal{C}$ and propose an algorithm to traverse this tree. Second, the tree is directly built over $\mathcal{C}$, as opposed to the set of connected subgraphs. The latter option, as proposed in [Sese et al., 2010] is more straightforward to define and to explore and still induces a tree over $\mathcal{C}$, but yields a much larger object and results in a more expensive traversal. Third, we avoid maintaining subgraph connectivity such as a block-cut tree [Westbrook and Tarjan, 1992]. Such a mechanism is efficient to build a tree over connected subgraphs but is costly to compute. Finally, in order to exploit the pruning opportunity offered by the testing procedure, the exploration should be such that all $\mathcal{S}'$ explored from a given $\mathcal{S}$ verify $\mathcal{S}' \supset \mathcal{S}$.

Haraguchi et al. [2019], Okuno et al. [2017] define a tree on $\mathcal{C}$, but the root of the tree corresponds to the entire graph $\mathcal{G}$: the inclusion relationship along edges of the tree is the opposite to the one we need, making their exploration unsuited to our problem. The `COIN`/`COPINE` algorithm described in Seki and Sese [2008], Sese et al. [2010] builds a tree over the set of connected subgraphs, which induces a tree over $\mathcal{C}$ but has two drawbacks. First, it maintains an itemtable to enforce a tree structure by avoiding the enumeration of the same element twice. This itemtable has an important memory footprint, and only guarantees a tree structure when exploring in depth first. Secondly, the enumeration of connected subgraphs requires maintaining a list of articulation points along each explored branch, a costly operation.

Algorithm 1 Children of $\mathcal{S}_p$

Input parent CCS $\mathcal{S}_p$, current CCS $\mathcal{S}$, largest index i, itemtable $\mathcal{T}$

procedure CHILDREN($\mathcal{S}_p$, $\mathcal{S}$, i, $\mathcal{T}$)
  children$\leftarrow \emptyset$
  for k, G in enumerate(`EqGroups`($\mathcal{S}$)) **do**
    $\mathcal{S}' \leftarrow cl(\mathcal{S} \bigcup \{G[0]\})$ — $\mathcal{S}'$ is a candidate child
    if $(\mathcal{S}_p, \mathcal{S}')$ verify (1-3) **then** — $\mathcal{S}'$ is a child
      if i is NULL **then** — Exploring from the direct neighbors of $\mathcal{S}_p$
        Add $\mathcal{S}'$ to children
        Add `Children`$(\mathcal{S}_p, \mathcal{S}', i_{\mathcal{S}'}, \mathcal{T} = \emptyset)$ to children
      else — Exploring from the neighbors of another child
        if $i_{\mathcal{S}'} = i$ and $\{\mathcal{I} \in \mathcal{T} : \mathcal{I} \subset \mathcal{I}(\mathcal{S}')\} = \emptyset$ **then** — Check that $\mathcal{S}'$ was not enumerated earlier
          $\mathcal{T}' = \mathcal{T} \bigcup \{\mathcal{I}_1(\mathcal{S}), \ldots, \mathcal{I}_{k-1}(\mathcal{S})\}$
          Add $\mathcal{S}'$ to children
          Add `Children`$(\mathcal{S}_p, \mathcal{S}', i_{\mathcal{S}'}, \mathcal{T}')$ to children
        end if
      end if
    end if
  end for
  return children
end procedure

3.2 Defining and exploring the tree over $\mathcal{C}$

In order to build a tree over $\mathcal{C}$ rooted on the empty CCS, we use a reverse search, introduced in Avis and Fukuda [1993]. Reverse search relies on a reduction operation, which takes one element of the set to be enumerated, and returns a unique, strictly smaller element of the same set. This operation necessarily defines a tree over the elements of the set, by ensuring a unique path between any element and the empty one—the root of the tree. This reduction operation defines the unique parent of every element in the tree. In order to traverse the tree from the root, one needs to inverse the reduction operation, *i.e.* in our setting, given a CCS $\mathcal{S}$ to recover all CCSs that lead to $\mathcal{S}$ by reduction. Here we introduce a reduction operation over $\mathcal{C}$, as well as its inversion. We consider the parent operation $\mathcal{P}$ given by Definition 2 for any element of $\mathcal{C}$, and show that it defines a valid reduction as introduced above. All proofs are presented in the appendix.

Definition 2. *For a subgraph $\mathcal{S} \in \mathcal{C}$, we denote $\mathcal{J}(\mathcal{S}) = \bigcap_{v \in \mathcal{S}} \mathcal{I}(v)$.*

- *If $\mathcal{I}(\mathcal{S}) = \mathcal{J}(\mathcal{S})$, then the parent of $\mathcal{S}$ is $\emptyset$, i.e., $\mathcal{P}(\mathcal{S}) = \emptyset$.*
- *Else we note $i_{\mathcal{S}} = \max_i\{i : i \in \mathcal{I}(\mathcal{S}) \setminus \mathcal{J}(\mathcal{S})\}$. The parent $\mathcal{P}(\mathcal{S})$ of $\mathcal{S}$ is the connected subgraph of $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}$ that contains $\max_v\{v : v \in \mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}\}$.*

Note that we arbitrarily assign a number to each node to be able to define the max.

Lemma 2. *The function $\mathcal{P}$ defines a valid reduction over $\mathcal{C}$.*

Note that we have $\mathcal{S} \supset \mathcal{P}(\mathcal{S})$ for all $\mathcal{S}$ so this structure allows pruning. Lemma 3 then provides necessary and sufficient conditions for $\mathcal{S}' \in \mathcal{C}$ to be a child of $\mathcal{S} \in \mathcal{C}$. The third condition involves the set of neighbouring nodes of $\mathcal{S}$, $Ne(\mathcal{S}) = \{v \in \mathcal{G} \setminus \mathcal{S} : \exists v_1 \in \mathcal{S}, (v, v_1) \in E\}$.

Lemma 3. *For $\mathcal{S}, \mathcal{S}' \in \mathcal{C}$ such that $\mathcal{S} \subset \mathcal{S}' \neq \emptyset$, we have: $\mathcal{S} = \mathcal{P}(\mathcal{S}')$ if and only if the three following conditions are verified:*

(C1) $i_{\mathcal{S}'} \notin \mathcal{I}(\mathcal{S})$

(C2) $\max_{v'}\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} = \max_v\{v : v \in \mathcal{S}\}$

(C3) $\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}} : v' \in Ne(\mathcal{S})\} = \emptyset$, *or written differently,* $(\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}) \bigcap Ne(\mathcal{S}) = \emptyset$.

Using (C1–3) in Lemma 3 to check whether $\mathcal{S} = \mathcal{P}(\mathcal{S}')$ for any $\mathcal{S}'$ does not require to identify the connected components of $\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}$, even though the reduction $\mathcal{P}$ itself does rely on these connected components. This property of the inverse reduction is critical for the scalability of `CALDERA`: repeatedly identifying or maintaining these components would be very costly. It is because the reduction operation $\mathcal{P}$ does not maintain full connectivity: it only retains one of the connected components obtained by removing a subset of its nodes. Doing so comes at a price. Finding all children of $\mathcal{S}$ is not straightforward, as we must identify and reconnect all the connected components involved—Lemma 3 only provides a way to check if a candidate $\mathcal{S}'$ is a child of $\mathcal{S}$.

More precisely, reducing any CCS $\mathcal{S}'$ to its parent $\mathcal{S}$ involves the removal of a subset $\mathbb{V}_{i_{\mathcal{S}'}}$ of its nodes, breaking $\mathcal{S}'$ into several connected components—the one containing the largest vertex being retained as the unique parent. For this reason, the reverse search formalized in Algorithm 1 cannot just search for children of $\mathcal{S}$ among all closures obtained after adding one of its neighbors $Ne(\mathcal{S})$ (Lines 6-7): larger CCSs may also lead to $\mathcal{S}$ by reduction if they involve other connected components that are not in its direct neighborhood. Once a child $\mathcal{S}'$ has been identified, we must therefore recursively search for other candidates among the closures obtained after adding one of its neighbors $Ne(\mathcal{S}')$ (Line 8). This procedure is necessary to reconnect all children that include $\mathcal{S}'$ but would leave it as a separated connected component after removing nodes $\mathbb{V}_{i_{\mathcal{S}'}}$. However, Lemma 3 used in Line 5 guarantees only that actual children of $\mathcal{S}$ are retained, it does not guarantee uniqueness. A redundant exploration would lose the benefit of building a tree over $\mathcal{C}$ to explore it efficiently. We therefore need an itemtable $\mathcal{T}$ that keeps track of visited patterns $\mathcal{I}$: if a candidate child $\mathcal{S}''$ has a pattern $\mathcal{I}(\mathcal{S}'')$ that includes the pattern of an already enumerated child from the neighborhood of the same $\mathcal{S}'$, we know that $\mathcal{S}''$—and any child that could be obtained from it—has already been visited and the algorithm stops exploring from $\mathcal{S}''$. In practice, we do not need to store the full table $\mathcal{T}$ in order to verify the second condition of Algorithm 1, Line 12. We rely on a concept from [Uno et al., 2004] and further described in Supplementary S-2 to reduce memory footprint.

By Theorem 1, Algorithm 1 solves the problem of inverting the reduction, and therefore of building a tree structure on $\mathcal{C}$. Of note, Algorithm 1 effectively explores equivalence groups of neighbours yielding the same pattern. Formally, an equivalence group $G_k(\mathcal{S}) \subset Ne(\mathcal{S})$ verifies: $v_1, v_2 \in G_k(\mathcal{S}) \implies \mathcal{I}(\mathcal{S} \bigcup \{v_1\}) = \mathcal{I}(\mathcal{S} \bigcup \{v_2\})$. We name $\mathcal{I}_k(\mathcal{S})$ the pattern of the equivalence group $G_k(\mathcal{S})$.

Theorem 1. *For any $\mathcal{S} \in \mathcal{C}$, Algorithm 1 applied on $(\mathcal{S}, \mathcal{S}, NULL, \emptyset)$ returns the set $\{\mathcal{S}' \in \mathcal{C} : \mathcal{S} = \mathcal{P}(\mathcal{S}')\}$.*

3.3 A breadth-first-search enumeration

We argue that exploring any tree structure on $\mathcal{C}$ in breadth first will often allow for more pruning than in depth first. At any level, even if the CCSs visited along a branch do increase k and therefore lower the testability threshold, all the other CCSs of the level will need to be visited regardless of their testability. By contrast, the increase of k gained by visiting all CCSs of the same level in the tree will lower the threshold α/k for all CCSs at the next level, making more branches prunable. We demonstrate this in section 4 and provide more intuitive examples in the appendix, (Supplementary S-5.1 and S-5.2). A search in breadth is also easily parallelized since the computation of the minimal p-value, the envelope and the children of every CCS of a given level can be done in parallel, before increasing k and updating $\mathcal{R}$. By contrast, a parallelized search in depth-first must share and regularly update k and $\mathcal{R}$, which negates the advantages of parallelization.

Algorithm 2 explores $\mathcal{C}$ through a BFS traversal of the tree defined by the reduction $\mathcal{P}$, exploiting Algorithm 1 (L.15) to invert the reduction and using this exploration to apply the Tarone testing procedure described in Section 2.2 (L7-12, 14), before finally testing the testable CCSs (L21-25). However, BFS is more memory intensive than DFS (see results). To be able to have a better trade-off between speed and memory, we also implemented a hybrid exploration scheme that is used in practice in which each stage of the tree is explored by batch in a BFS manner, and a DFS is performed over each batch.

3.4 Pruning more CCSs when controlling for an imbalanced categorical covariate

The envelope $\tilde{p}^\star(\mathcal{S}) = \min_{x' \geq x_\mathcal{S}} p^\star(\mathcal{S})$ introduced in Papaxanthos et al. [2016] verifies the monotonicity for any subgraph $\mathcal{S}$ because $\mathcal{S}' \supseteq \mathcal{S} \Rightarrow x_{\mathcal{S}'} \geq x_\mathcal{S}$. However, the $O(J \log J)$ algorithm to compute this envelope only applies to the so called *potentially prunable* subgraphs which are such that $x_{\mathcal{S},j} \geq \max(n_{1,j}, n_{2,j})$ for

all subgroups $j = 1, \dots, J$ defined by the categorical covariate adjusted for by the CMH test. Pruning can therefore not be done from subgraphs for which at least one of the J groups has few occurrences of the corresponding covariate. This limitation arises in Lemma 2 of Papaxanthos et al. [2016], which characterizes the argmin of the envelope of a subgraph $\mathcal{S}$. Lemma 4 lifts this restriction:

Lemma 4. *For any connected subgraph $\mathcal{S}$, the envelope $\tilde{p}^\star$ is attained for an optimum $x^*_{\mathcal{S}'}$ such that $x^*_{\mathcal{S}',j} \in \{\max(x_{\mathcal{S},j}, n_{1,j}), \max(x_{\mathcal{S},j}, n_{2,j}), n_j\}$.*

Algorithm 2 List significant closed connected subgraphs

procedure LIST_SIG_CLOSED_SUBGRAPHS($\mathcal{G}, \alpha$)
  $Q \leftarrow Children(\emptyset, \emptyset$,NULL$, \emptyset)$
  $\mathcal{R} \leftarrow \emptyset$
  $k \leftarrow 1$
  while $Q \neq \emptyset$ **do**
    $\mathcal{S} \leftarrow$ `Dequeue`(Q)
    if $p^\star(\mathcal{S}) \leq \alpha/k$ **then**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{S}\}$
    end if
    if $\left|\mathcal{R}\right| > k$ **then**
      $k \leftarrow k + 1$
      $\mathcal{R} \leftarrow \{\mathcal{S} \in \mathcal{R} : p^\star(\mathcal{S}) \leq \alpha/k\}$
    end if
    if $\tilde{p}^\star(\mathcal{S}) \leq \alpha/k$ **then**
      for $\mathcal{S}' \in$ `Children`$(\mathcal{S}, \mathcal{S}$,NULL$, \emptyset)$ **do**
        `Enqueue`$(\mathcal{S}', Q)$
      end for
    end if
  end while
  Solutions$\leftarrow \emptyset$
  for $\mathcal{S} \in \mathcal{R}$ **do**
    if $p(\mathcal{S}) \leq \alpha/k$ **then**
      Add $\mathcal{S}$ to Solutions
    end if
  end for
  return Solutions
end procedure

The proof is provided in Supplementary Material S-1.5. Lemma 4 exploits a cruder bound for groups that are not in the increasing regime of the minimal p-value. It recovers the Lemma 2 of Papaxanthos et al. [2016] for potentially prunable subgraphs, while offering an additional pruning opportunity for the other ones. If a subgraph was not potentially prunable only because it was missing the $x_{\mathcal{S},j} \geq \max(n_{1,j}, n_{2,j})$ condition for one small group j, it may still be actually prunable since small groups of samples only affect the CMH test statistic marginally. On the other hand if the condition is not verified for a large group or several small ones, the resulting envelope will be very lose and will not allow for pruning in practice. We provide some intuition in the appendix (Figure S3).

4 Experiments

We demonstrate the superiority of `CALDERA` in terms of computational speed, statistical power and biological interpretation. To do so, we rely on both simulated and real datasets.

4.1 Datasets and settings

To test the speed of the methods, we generate datasets with n samples represented by $p \in [100 : 20,000]$ covariates, and a graph connecting these covariates. We vary both the proportion *prop* of samples that are resistant, i.e. have a phenotype of 1, and the number of samples. We also perform exploration when changing the value of α, which impacts pruning. This generates 4 scenarios to compare the runtimes of the methods, named **Speed 1** to **Speed 4**. More details on implementations and parameters can be found in section S-6. We can also add a binary confounding variable, with $n = 100$ and $p = 3,000$, where we vary the ratio between the size of those two populations, n_2/n_1. This is useful to test the speed gains provided by the new lower bound provided in `CALDERA`. This scenario is called **Imbalance**

To test the power of the different methods, we rely on a simulation where the ground truth is known, named **Exploration**. We generate a dataset with $n = 100$ samples, 50 of each phenotype, where two genes A and B are present. Gene A is present for all samples while gene B is only present for resistant samples. We introduce heterogeneity such that the DBG of the two genes is only linear at a coarse level (Fig.1b). More details for the setting of those simulations are provided in S-7.

We also rely on two real datasets. The first, which we name **Pseudomonas**, consists of the $n = 280$ *Pseudomonas Aeruginosa* genomes along with their resistance phenotype to amikacin, used in `DBGWAS` [Jaillard et al., 2018]. The bacteria are partitioned based on k-mean into two distinct groups. The compacted DBG is constructed using the k-mers with $k = 31$ (default) using `DBGWAS`, leading to a graph with over 2.3 million

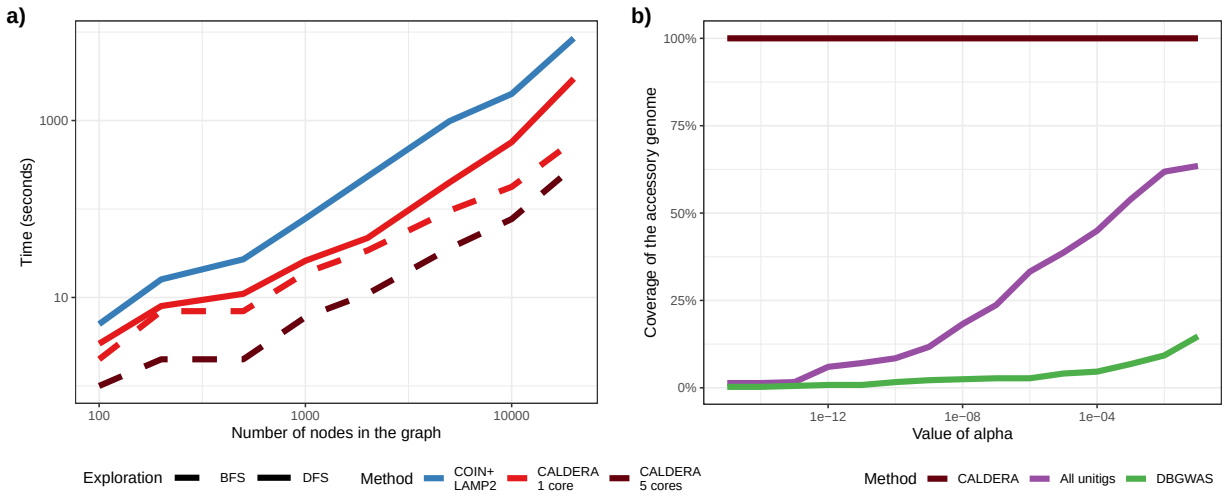


Figure 2: *Results of CALDERA.* **a.** Run times for `CALDERA` and `COIN+LAMP` on graphs with various values of p. In this setting, $n = 100$. **b.** Proportion of all unitigs associated with the resistant phenotype that are found to be significant by `CALDERA`, the `LAMP2` procedure on all unitigs and `DBGWAS`, as the value of α changes.

nodes and average degree ~ 2.7. The second, named **Akkermansia**, consists of the Akkermansia muciniphila genomes collected in Karcher et al. [2021]. We use host information as covariates: we want to identify genetic sequences that are associated with a body mass index (BMI) over 30. The DBG constructed over those $n = 401$ strains has 1.3 million nodes with an average degree of ~ 2.7. On these two real datasets, we rely on heuristics to choose the level α at which the FWER is controlled and the number of stages explored in the BFS search—a full exploration being too memory intensive. The level is fixed at the lowest value at which 10 CCSs at stage 1 of the BFS (*i.e*, unitig closures) are found significant. The stage is chosen by stopping when the number of unitigs covered by a significant CCS reaches a plateau—suggesting that further exploration would not bring much novelty.

4.2 Speed gains of `CALDERA`

`COIN` [Sese et al., 2014] was to our knowledge the only described algorithm to identify significant CCSs, combining the enumeration method of `COPINE` with the `LAMP` algorithm. Minato et al. [2014] presented a provably superior version of `LAMP`, which we denote `LAMP2`. Since no implementation was provided in Sese et al. [2014], we implemented as a baseline `COIN+LAMP2`. Since `CALDERA` and `COIN+LAMP2` both rely on the same statistical procedures (the identification of testable hypotheses with Fisher's test), the set of significant CCSs found is the same regardless of the method. In addition to `COIN+LAMP2`, we benchmark 3 versions of `CALDERA`. The first one, closest to `COIN+LAMP2`, is the DFS implementation. The second one is the BFS implementation, where we modify the enumeration order of the elements of $\mathcal{C}$ to promote pruning. The last is a parallelized BFS implementation, using 5 cores.

Benefit of `CALDERA`'s exploration scheme In Figure 2a, representing the results of **Speed 2**, we see that the ranking in speed is uniform over all value of p, with `COIN+LAMP2` being the slowest, followed by the DFS and BFS implementation, and finally the parallelized version of `CALDERA`. For $p = 20,000$, `COIN+LAMP2` runs in 2h20 while the parallelized version of `CALDERA` takes 5 minutes. The ranking is the same for **Speed 1**, **Speed 3** and **Speed 4** (see Supplementary S-6). For example, for **Speed 1** and $p = 20,000$, `COIN+LAMP2` times out (two day threshold) before finishing while the parallelized version of `CALDERA` runs in 6 hours. Over all parameter values, the average ratio of runtime for `COIN+LAMP2` over `CALDERA BFS` with 5 cores is 76 and we tested `CALDERA` on values of $p = 100,000$ in 14h. More details on memory usage can be found in section S-6.

On the larger **Pseudomonas** dataset, even `CALDERA` is unable to explore the entire $\mathcal{C}$ with the heuristic level $\alpha = 10^{-6}$, but we observe that the unitigs covered by the significant CCSs reached a plateau after the

first 6 stages of the BFS. CALDERA took 3h20 on 4 cores, using 200Gb of RAM to complete these stages using batches of size 200,000, leading to $k_0 = 4,671,265$ potentially testable CCSs and only 39 significant ones. For comparison, after running for 24h, COIN+LAMP2 was exploring the tree structure with a running $k_0 = 10^5$. We provide a more general analysis of the computational cost of CALDERA against the number of BFS stages in Supplementary S-9 and recommend using a similar analysis and stop after a few stages in cases where a full exploration is no feasible.

Benefit of CALDERA's lower-bound on runtime for imbalanced population For extreme ratios—below 0.02—the new lower bound allows much more pruning and enumerates an order of magnitude fewer elements of $\mathcal{C}$. Up to a ratio of 0.1, the new lower bound leads to a decrease of at least 10% in the number of explored subgraphs (see Fig S7).

4.3 Power gains of CALDERA

As mentioned above, COIN+LAMP2 and all versions of CALDERA rely on the same statistical procedures and therefore find an identical list of significant CCSs for a given level of α. However, we can compare the power of CALDERA with two other alternatives. DBGWAS tests individual unitigs for association with a phenotype, using a mixed-model. We also use the LAMP2 procedure when testing All Unitigs separately using Fisher's test—like CALDERA.

We run all three methods on the dataset **Exploration** and measure how many of the 367 unitigs of gene B are called significant, when controlling the FWER at a varying level α. For CALDERA, a significant unitig is one that is contained in a significant CCS. Even when controlling the FWER at very low levels ($\alpha = 10^{-16}$), CALDERA correctly recovers the entirety of the resistant gene. On the other hand, the other two methods fail to ever recover the entire gene, even at $\alpha = 0.1$. This clearly show the enhanced power of CALDERA: because of variations along the genome, the association of any individual unitig with the phenotype is weak, while a covariate that jointly represents all 367 unitigs of the resistant gene is very strongly associated with that phenotype.

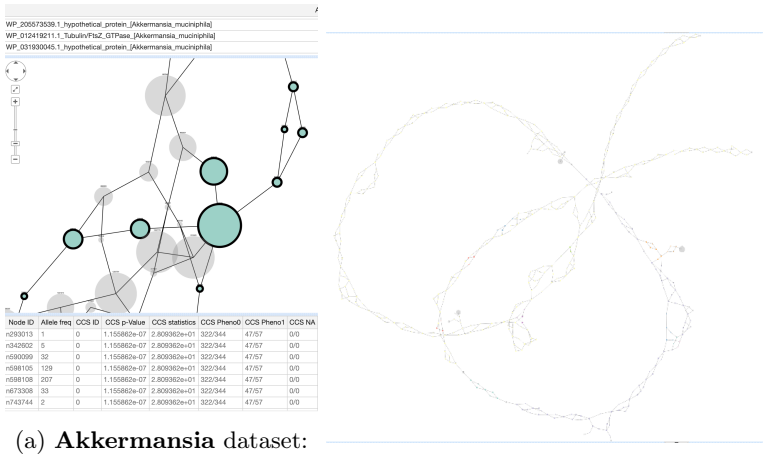
We also apply those three methods to the **Pseudomonas** dataset. While there is no ground truth, this dataset contains two confirmed genetic variants linked to resistance to amikacin: a SNP on the aac(6') gene, represented by one unitig, and the pHS87b plasmid, represented by 476 unitigs. This allows us to see how the methods handle those different scales. At the default $\alpha = 10^{-6}$, CALDERA find the aac(6') mutation as one CCS, and finds significant CCSs that covers 96% of the plasmid. Those two components represent 59% of all significant unitigs. In contrast, All Unitigs and DBGWAS do recover the mutation but only 34% and 0% of the plasmid respectively. Even at $\alpha = 0.1$, All Unitigs and DBGWAS only recover respectively 72% and 8% of the plasmid. Moreover, while it is not possible to compute a false negative rate on real dataset, we can see that, at this level, the two known sequences—the plasmid and the aac(6') mutation—only represent 6% and 17% of all significant unitigs.

4.4 Simplified biological interpretation

Biological interpretation in DBGWAS or CALDERA happens at the component levels: significant unitigs or CCSs separated by only a a few non-significant unitigs are displayed as one component. Unitigs can also be annotated using various databases, to enhance interpretation. Components are ranked in order of decreasing p-values, choosing the smallest p-value among all unitigs/CCSs. As such, both the number of components and their rankings will impact the ease of interpretation.

Figure 3 gives two examples. On panel **a)**, we see the results of running CALDERA on the **Akkermansia** dataset. Only one CCS is significant, while DBGWAS returns no significant unitig. All the unitigs in the significant CCS (colored in green) are annotated, using the *RefSeq* database of all known Akkermansia muciniphila proteins as a reference [Tatusova et al., 2016], and map to a common gene. This gene is not well annotated (hypothetical protein) but partially map to a Tubulin / FtsZ_GTPase Akkermansia muciniphila protein.

On panel **b)**, we see the plasmid, returned as the first component for CALDERA. Visually, we can see clearly a broad circular graph, with local genetic variations. On the **Pseudomonas** dataset, CALDERA returns 8 components: the first is the entire plasmid, returned as one component. The second is the aac(6') mutation.



(a) **Akkermansia** dataset: Tubulin / FtsZ_GTPase gene

(b) **Pseudomonas** dataset: plasmid

Figure 3: Screenshot from the output of `CALDERA`. We select the first component, that is the one which contains the most significant CCS. Unitigs belonging to the same CCS are colored in the same way. If two significant CCSs are less than two unitigs apart, they are represented in the same component.

`DBGWAS` always ranks the aac(6') mutation first but never returns the plasmid as one component, even when controlling the FWER at a level of 0.1 (3 components, the first one ranked fourth). Moreover, at this level, `DBWGAS` returns 77 components, making the interpretation much harder.

5 Discussion

This article presented `CALDERA`, an algorithm to enumerate all significant closed connected subgraphs. `CALDERA` easily scales to large datasets, relying on an efficient structure on $\mathcal{C}$ and an exploration scheme that leverages the pruning opportunity offered by discrete statistics. This increased computational speed allows to deploy this method to De Bruijn graph-based bacterial GWAS, which we demonstrate on two real examples. Moreover, we show that considering the CCSs, as done by `CALDERA`, leads to increased power and facilitates interpretation, compared to previous methods to performed statistical tests at the node level. `CALDERA` can better detect low signal caused by variability in genetic elements. It also returns larger and more coherent outputs that are easier to interpret.

We extensively discussed how `CALDERA` performs on bacterial GWAS. However, `CALDERA` can also be used for other tests of association involving a graph structure. We provide in S-8 an example: we look at the association between SNPs on *A. thaliana* genomes and a "date to flowering" phenotype. In that setting, the graph is a regulatory network on the genes and the objective is to identify subnetworks whose disruption by at least one mutation is associated with the phenotype.

In settings where the node is a more natural object than the CCS, discrete testing can still be used to take advantage of Tarone [1990]'s procedure and increase power. However, pruning will no longer be possible, unless some other order can be established between nodes that preserve the order of minimal p-values.

For now, `CALDERA` does not scale to datasets with hundreds of millions of nodes that are possible in metagenome-wide association studies. Future work that focuses on incorporating pre-processing schemes before `CALDERA` would be needed to compact the graph to both reduce its size and facilitate pruning by increasing the average $|\mathcal{I}(v_j)|$.

Acknowledgment

LJ is funded by the Agence Nationale de la Recherche ANR-17-CE23-0011-01 (FAST-BIG). A part of this work was performed using the computing facilities of the CC LBBE/PRABI.

Conflicts of Interest

Hector Roux de Bezieux and Fanny Perraudeau own stocks in Pendulum Therapeutics

References

P M Visscher, N R Wray, Q Zhang, P Sklar, M I McCarthy, M A Brown, and J Yang. 10 years of GWAS discovery: Biology, function, and translation. *Am J Hum Genet*, 101(1):5–22, July 2017. doi: 10.1016/j.ajhg.2017.06.005. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5501872/`.

Magali Jaillard, Alex van Belkum, Kyle C Cady, David P. Creely, Dee Shortridge, Bernadette Blanc, Elena Magda Barbu, William Michael Dunne, Gilles Zambardi, Mark Charles Enright, Nathalie Mugnier, Christophe Le Priol, Stéphane Schicklin, Ghislaine Guigon, and Jean-Baptiste Veyrieras. Correlation between phenotypic antibiotic susceptibility and the resistome in pseudomonas aeruginosa. *International journal of antimicrobial agents*, 50 2:210–218, 2017.

Samuel K. Sheppard, Xavier Didelot, Guillaume Meric, Alicia Torralbo, Keith A. Jolley, David J. Kelly, Stephen D. Bentley, Martin C. J. Maiden, Julian Parkhill, and Daniel Falush. Genome-wide association study identifies vitamin b5 biosynthesis as a host specificity factor in campylobacter. *Proceedings of the National Academy of Sciences*, 110(29):11923–11927, 2013. ISSN 0027-8424. doi: 10.1073/pnas.1305559110. URL `https://www.pnas.org/content/110/29/11923`.

Sarah G. Earle, Chieh-Hsi Wu, Jane Charlesworth, Nicole Stoesser, N. Claire Gordon, Timothy M. Walker, Chris C. A. Spencer, Zamin Iqbal, David A. Clifton, Katie L. Hopkins, Neil Woodford, E. Grace Smith, Nazir Ismail, Martin J. Llewelyn, Tim E. Peto, Derrick W. Crook, Gil McVean, A. Sarah Walker, and Daniel J. Wilson. Identifying lineage effects when controlling for population structure improves power in bacterial association studies. *Nature Microbiology*, 1(5):16041, Apr 2016. ISSN 2058-5276. doi: 10.1038/nmicrobiol.2016.41. URL `https://doi.org/10.1038/nmicrobiol.2016.41`.

Magali Jaillard, Leandro Lima, Maud Tournoud, Pierre Mahé, Alex van Belkum, Vincent Lacroix, and Laurent Jacob. A fast and agnostic method for bacterial genome-wide association studies: Bridging the gap between k-mers and genetic events. *PLoS genetics*, 14(11):e1007758, 2018. ISSN 1553-7404. doi: 10.1371/journal.pgen.1007758. URL `http://www.ncbi.nlm.nih.gov/pubmed/30419019`.

N.G. de Bruijn. A combinatorial problem. *Proceedings of the Section of Sciences of the Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*, 49(7):758–764, 1946. ISSN 0370-0348.

Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001. ISSN 0027-8424. doi: 10.1073/pnas.171285098. URL `https://www.pnas.org/content/98/17/9748`.

R. E. Tarone. A Modified Bonferroni Method for Discrete Data. *Biometrics*, 46(2):515, jun 1990. ISSN 0006341X. doi: 10.2307/2531456.

Aika Terada, Mariko Okada-Hatakeyama, Koji Tsuda, and Jun Sese. Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences of the United States of America*, 110(32): 12996–13001, aug 2013. doi: 10.1073/pnas.90.1.203.

Shin Ichi Minato, Takeaki Uno, Koji Tsuda, Aika Terada, and Jun Sese. A fast method of statistical assessment for combinatorial hypotheses based on frequent itemset enumeration. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8725 LNAI, pages 422–436. Springer Verlag, 2014. ISBN 9783662448502. doi: 10.1007/978-3-662-44851-9_27.

Felipe Llinares-López, Dominik G. Grimm, Dean A. Bodenham, Udo Gieraths, Mahito Sugiyama, Beth Rowan, and Karsten Borgwardt. Genome-wide detection of intervals of genetic heterogeneity associated with complex traits. *Bioinformatics*, 31(12):i240–i249, 2015. ISSN 14602059. doi: 10.1093/bioinformatics/btv263.

Felipe Llinares-López, Laetitia Papaxanthos, Dean Bodenham, Damian Roqueiro, and Karsten Borgwardt. Genome-wide genetic heterogeneity discovery with categorical covariates. *Bioinformatics*, 33(12):1820–1828, 2017. ISSN 14602059. doi: 10.1093/bioinformatics/btx071.

Jun Sese, Aika Terada, Yuki Saito, and Koji Tsuda. Statistically significant subgraphs for genome-wide association study. *SDM*, 47:1–7, 2014.

Jun Sese, Mio Seki, and Mutsumi Fukuzaki. Mining networks with shared items. In *International Conference on Information and Knowledge Management, Proceedings*, pages 1681–1684, New York, New York, USA, 2010. ACM Press. ISBN 9781450300995. doi: 10.1145/1871437.1871703. URL `http://portal.acm.org/citation.cfm?doid=1871437.1871703`.

Ronald A Fisher. On the Interpretation of χ 2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922. ISSN 09528385. doi: 10.2307/2340521.

CE Bonferroni. Teoria Statistica Delle Classi e Calcolo Delle Probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936. doi: 10.4135/9781412961288.n455.

Laetitia Papaxanthos, Felipe Llinares-Lopez, Dean Bodenham, and Karsten Borgwardt. Finding significant combinations of features in the presence of categorical covariates. *NEURIPS*, pages 2271–2279, 2016. ISSN 10495258. doi: 10.1007/s00464-011-2087-1.

Jeffery Westbrook and Robert E. Tarjan. Maintaining bridge-connected and biconnected components on-line. *Algorithmica*, 7(1-6):433–464, dec 1992. ISSN 01784617. doi: 10.1007/BF01758773.

Kazuya Haraguchi, Yusuke Momoi, Aleksandar Shurbevski, and Hiroshi Nagamochi. COOMA: A components overlaid mining algorithm for enumerating connected subgraphs with common itemsets. *Journal of Graph Algorithms and Applications*, 23(2):434–458, 2019. ISSN 15261719. doi: 10.7155/jgaa.00497. URL `http://jgaa.info/vol`.

Shingo Okuno, Tasuku Hiraishi, Hiroshi Nakashima, Masahiro Yasugi, and Sese Jun. Parallelization of extracting connected subgraphs with common itemsets in distributed memory environments. *Journal of Information Processing*, 25(3):256–267, 2017. ISSN 18826652. doi: 10.2197/ipsjjip.25.256.

Mio Seki and Jun Sese. Identification of active biological networks and common expression conditions. In *8th IEEE International Conference on BioInformatics and BioEngineering, BIBE 2008*, 2008. ISBN 9781424428458. doi: 10.1109/BIBE.2008.4696746.

David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1993.

T Uno, M Kiyomi, and H Arimura. Efficient mining algorithms for frequent/closed/maximal itemsets. In *IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2004.

Nicolai Karcher, Eleonora Nigro, Michal Punčochář, Aitor Blanco-Míguez, Matteo Ciciani, Paolo Manghi, Moreno Zolfo, Fabio Cumbo, Serena Manara, Davide Golzato, Anna Cereseto, Manimozhiyan Arumugam, Thi Phuong Nam Bui, Hanne L P Tytgat, Mireia Valles-Colomer, Willem M de Vos, and Nicola Segata. Genomic diversity and ecology of human-associated Akkermansia species in the gut microbiome revealed by extensive metagenomic assembly. *Genome Biology*, 22(1):209, 2021. ISSN 1474-760X. doi: 10.1186/s13059-021-02427-7. URL `https://doi.org/10.1186/s13059-021-02427-7`.

Tatiana Tatusova, Michael DiCuccio, Azat Badretdin, Vyacheslav Chetvernin, Eric P Nawrocki, Leonid Zaslavsky, Alexandre Lomsadze, Kim D Pruitt, Mark Borodovsky, and James Ostell. NCBI prokaryotic genome annotation pipeline. *Nucleic acids research*, 44(14):6614–6624, aug 2016. ISSN 1362-4962 (Electronic). doi: 10.1093/nar/gkw569.

R3: Announcing the next generation of Amazon EC2 Memory-optimized instances, 2020. URL `https://aws.amazon.com/about-aws/whats-new/2014/04/10/r3-announcing-the-next-generation-of-amazon-ec2-memory-optimized-instances/`.

Dominik G. Grimm, Damian Roqueiro, Patrice A. Salomé, Stefan Kleeberger, Bastian Greshake, Wangsheng Zhu, Chang Liu, Christoph Lippert, Oliver Stegle, Bernhard Schölkopf, Detlef Weigel, and Karsten M. Borgwardt. easygwas: A cloud-based platform for comparing the results of genome-wide association

studies. *The Plant Cell*, 29(1):5–19, 2017. ISSN 1040-4651. doi: 10.1105/tpc.16.00551. URL `http://www.plantcell.org/content/29/1/5`.

M Kanehisa and S Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 28(1):27–30, January 2000. URL `http://www.ncbi.nlm.nih.gov/pubmed/10592173`.

D Tenenbaum. KEGGREST: Client-side REST access to KEGG, 2020.

Laurent Jacob, Pierre Neuvial, and Sandrine Dudoit. More power via graph-structured tests for differential expression of gene networks. *Ann. Appl. Stat.*, 6(2):561–600, 06 2012. doi: 10.1214/11-AOAS528. URL `https://doi.org/10.1214/11-AOAS528`.

P. Cingolani, A. Platts, M. Coon, T. Nguyen, L. Wang, S.J. Land, X. Lu, and D.M. Ruden. A program for annotating and predicting the effects of single nucleotide polymorphisms, snpeff: Snps in the genome of drosophila melanogaster strain w1118; iso-2; iso-3. *Fly*, 6(2):80–92, 2012.

Holger Hesse and Rainer Hoefgen. Molecular aspects of methionine biosynthesis. *Trends in Plant Science*, 8 (6):259–262, jun 2003. ISSN 13601385. doi: 10.1016/S1360-1385(03)00107-9.

Christina Pfaff, Hans F. Ehrnsberger, María Flores-Tornero, Brian B. Sørensen, Thomas Schubert, Gernot Längst, Joachim Griesenbeck, Stefanie Sprunck, Marion Grasser, and Klaus D. Grasser. ALY RNA-Binding Proteins Are Required for Nucleocytosolic mRNA Transport and Modulate Plant Growth and Development. *Plant physiology*, 177(1):226–240, may 2018. ISSN 15322548. doi: 10.1104/pp.18.00173.

S-1 Proofs

S-1.1 Lemma 1: correctness of the closure

Lemma 1 provides that the operator cl is well defined on connected subgraphs.

Proof of Lemma 1 First let's show that there exists $\mathcal{S}' \in \mathcal{C}$ such that $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$ and $\mathcal{S} \subseteq \mathcal{S}'$. Let $\mathcal{S}'$ be a (inclusionwise) maximal connected subgraph containing $\mathcal{S}$ and such that $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$. By maximality of $\mathcal{S}'$, for every edge $(v_1, v_2) \in E$ with $v_1 \in \mathcal{S}'$ and $v_2 \notin \mathcal{S}'$, we have $\mathcal{I}(\mathcal{S}' \cup \{v_2\}) \neq \mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}')$, thus $\mathcal{S}' \in \mathcal{C}$.

Now let's show that such a subgraph is unique. Assume that there exits two different subgraphs $\mathcal{S}_1$ and $\mathcal{S}_2$ in $\mathcal{C}$ such that $\mathcal{S} \subseteq \mathcal{S}_1$ and $\mathcal{S} \subseteq S_2$ with $\mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}_1) = \mathcal{I}(\mathcal{S}_2)$. Since $\mathcal{S}_1 \neq \mathcal{S}_2$, at least one of the subgraphs $\mathcal{S}_1 \setminus \mathcal{S}_2$ and $\mathcal{S}_2 \setminus \mathcal{S}_1$ is not empty. Assume without loss of generality that $\mathcal{S}_1 \setminus \mathcal{S}_2 \neq \emptyset$. Since $\mathcal{S}_1$ is connected and since $\mathcal{S}_1 \cap \mathcal{S}_2 \supseteq \mathcal{S} \neq \emptyset$, there is at least one edge (u, v) with $u \in \mathcal{S}_1 \cap \mathcal{S}_2$ and $v \in \mathcal{S}_1 \setminus \mathcal{S}_2$. This leads to a contradiction since the edge (u, v) is such that $u \in \mathcal{S}_2$, $v \notin \mathcal{S}_2$ and $\mathcal{I}(\mathcal{S}_2 \cup v) = \mathcal{I}(\mathcal{S}) = \mathcal{I}(\mathcal{S}_2)$, which is in contradiction with $\mathcal{S}_2 \in \mathcal{C}$.

S-1.2 Lemma 2: $\mathcal{P}$ is a valid reduction

Case if $\mathcal{I}(\mathcal{S}) = \mathcal{J}(\mathcal{S})$: Then, either $\mathcal{S} = \emptyset$ which has trivially no parent by this reduction. Or all nodes of $\mathcal{S}$ contain exactly the same pattern. For any $v \in \mathcal{S}$, $\mathcal{S} = cl(v)$. $\mathcal{S}$ is a root of our exploration. Its parent is $\emptyset \subseteq \mathcal{S}$. Note that, to avoid enumerating those roots more than once, we only start from $v_{\max} = \max \mathcal{S}$.

Case if $i_{\mathcal{S}}$ is defined: Then, $i_{\mathcal{S}} \in \mathcal{I}(\mathcal{S})$ so $\mathcal{S} \bigcap \mathbb{V}_{i_{\mathcal{S}}} \neq \emptyset$ and $i_{\mathcal{S}} \notin \mathcal{I}(\mathcal{S})$ so $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}} \neq \emptyset$. Therefore, there is at least one connected component in $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}$. Moreover, any connected component of $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}$ is included but not equal to $\mathcal{S}$. From [Haraguchi et al., 2019], Lemma 1, we know that, if $\mathcal{S} \in \mathcal{C}$, any connected component of $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}$ is also in $\mathcal{C}$. So any connected component of $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}}$ can be defined as a parent of $\mathcal{S}$. To identify a unique parent, we select $\mathcal{S}_p$, the one with the highest node number, $\mathcal{S}_p$. Since $\mathcal{S} \setminus \mathbb{V}_{i_{\mathcal{S}}} \neq \emptyset$ and $\mathcal{S}_p \subset \left(\mathcal{S} \bigcap \mathbb{V}_{i_{\mathcal{S}}}\right)$, then $\mathcal{S}_p \subsetneq \mathcal{S}$: it is a strictly smaller subgraph by inclusion. This proves that reduction defines a unique parent and proves that the reduction is valid.

S-1.3 Lemma 3: conditions (C1-3) are necessary and sufficient for $\mathcal{S} = \mathcal{P}(\mathcal{S}')$

We first prove the following lemma:

Lemma 5. *For two subgraphs $\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{C}$, if $\mathcal{S}_1 \subset \mathcal{S}_2$, then $cl(\mathcal{S}_1) \subset cl(\mathcal{S}_2)$.*

Proof: Since $\mathcal{S}_1 \subset cl(\mathcal{S}_1)$ and $\mathcal{S}_1 \subset \mathcal{S}_2 \subset cl(\mathcal{S}_2)$, we then known that $cl(\mathcal{S}_1) \bigcap cl(\mathcal{S}_2) \neq \emptyset$. Let' now assume that $cl(\mathcal{S}_1) \nsubseteq cl(\mathcal{S}_2)$. Since $cl(\mathcal{S}_1)$ is a connected subgraph, there exists $v \in cl(\mathcal{S}_1) \bigcap Ne(cl(\mathcal{S}_2))$. $v \in cl(\mathcal{S}_1)$ so $\mathcal{I}(v) \subset \mathcal{I}(cl(\mathcal{S}_1)) = \mathcal{I}(\mathcal{S}_1)$. But $\mathcal{S}_1 \subset \mathcal{S}_2$ so $\mathcal{I}(\mathcal{S}_1)] \subset \mathcal{I}(\mathcal{S}_2)$. Therefore, $\mathcal{I}(v) \subset \mathcal{I}(cl(\mathcal{S}_2))$ and $v \in Ne(cl(\mathcal{S}_2))$. This contradicts the definition of the closure. So we prove the lemma.

S-1.3.1 Proof that for any $\mathcal{S}'$, $(\mathcal{S} = \mathcal{P}(\mathcal{S}'), \mathcal{S}')$ verify $(C1-3)$

$\mathcal{S} \subset \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}$ so $i_{\mathcal{S}'} \notin \mathcal{I}(\mathcal{S})$. This proves (1). $\max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} \in \mathcal{S}$ by construction of the parent so $\max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} \leq \max \mathcal{S}$. Moreover, $\mathcal{S} \subset \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}$ so $\max \mathcal{S} \leq \max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\}$. So $\max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\} = \max \mathcal{S}$, this proves (2).

Suppose (3) is false. Then, there exists $v \in Ne(\mathcal{S}) \bigcap (\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}})$. $\mathcal{S} \bigcup \{v\} \supset \mathcal{S}' = cl(\mathcal{S})'$ so $\mathcal{S}_2 = cl(\mathcal{S} \bigcup \{v\}) \subset \mathcal{S}'$, $\max \mathcal{S}_2 = \max\{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\}$ since $\mathcal{S} \subset \mathcal{S}_2$ and $\mathcal{S}_2 \bigcap \mathbb{V}_{i_{\mathcal{S}'}} = \emptyset$ so $\mathcal{S}_2 \subset \mathcal{P}(\mathcal{S}')$. But $\mathcal{S}2 \supsetneq \mathcal{S} = \mathcal{P}(\mathcal{S})$. This is not possible. So (3) is true.

This proves the implication in the first sense.

S-1.3.2 Proof that for any $(\mathcal{S}, \mathcal{S}')$ that verify $(C1-3)$, $\mathcal{S} = \mathcal{P}(\mathcal{S}')$

We consider two closed connected subgraph $\mathcal{S}, \mathcal{S}' \in \mathcal{C}$ that verify (1-3). We want to prove that $\mathcal{P}(\mathcal{S}') = \mathcal{S}$. Point (1) insures that $\mathcal{S} \subseteq (\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}})$. Since $\mathcal{S} \in \mathcal{C}$ and contains the maximal node (from (2)), this ensures that $\mathcal{S} \subseteq \mathcal{P}(\mathcal{S}')$.

Suppose $\mathcal{S} \subsetneq \mathcal{P}(\mathcal{S}')$. Then, $\mathcal{P}(\mathcal{S}') \setminus \mathcal{S} \neq \emptyset$. In particular, since $\mathcal{S}$ and $\mathcal{P}(\mathcal{S}')$ are both connected subgraphs, there exists $v' \in \big(\mathcal{P}(\mathcal{S}') \setminus \mathcal{S}\big) \bigcap Ne(\mathcal{S})$. Since this neighbour is in $\mathcal{P}(\mathcal{S}')$, it is also in $\mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}}\}$. That is impossible from (3). So $\mathcal{S} = \mathcal{P}(\mathcal{S}')$. (Note that point (3) includes the fact that $i_{\mathcal{S}'} \in \mathcal{I}(v)$).

This proves the converse implication.

S-1.4 Theorem 1: Algorithm 1 correctly inverts the reduction

We consider a subgraph $\mathcal{S}' \in \mathcal{S}$ and its parent $\mathcal{S} = \mathcal{P}(\mathcal{S}')$.

We first show two lemmas

Lemma 6. *For two subgraphs $\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{C}$, if $\mathcal{S}_1 \subset \mathcal{S}_2$, then $i_{\mathcal{S}_1} \leq i_{\mathcal{S}_2}$.*

Proof:

$$\mathcal{S}_1 \subset \mathcal{S}_2 \implies \mathcal{I}(\mathcal{S}_1) \subset \mathcal{I}(\mathcal{S}_2) \quad \text{and} \tag{1}$$

$$\mathcal{S}_1 \subset \mathcal{S}_2 \implies \mathcal{J}(\mathcal{S}_2) \subset \mathcal{J}(\mathcal{S}_1) \tag{2}$$

$$(1) \text{ and } (2) \implies (\mathcal{I}(\mathcal{S}_1) \setminus \mathcal{J}(\mathcal{S}_1)) \subset (\mathcal{I}(\mathcal{S}_2) \setminus \mathcal{J}(\mathcal{S}_2)) \tag{3}$$

$$\implies i_{\mathcal{S}_1} \leq i_{\mathcal{S}_2} \tag{4}$$

Lemma 7. *For a subgraph $\mathcal{S}' \in \mathcal{C}$ such that $\mathcal{S} = \mathcal{P}(\mathcal{S}') \neq \emptyset$, any subgraph $\mathcal{S}_2 \in \mathcal{C}$ that verifies:*

- $\mathcal{S} \subsetneq \mathcal{S}_2$
- $\mathcal{S}_2 \subset \mathcal{S}'$

is a child of $\mathcal{S}$, that is $\mathcal{P}(\mathcal{S}_2) = \mathcal{S}$

Proof: We know that $\mathcal{S} \subsetneq \mathcal{S}_2$ so $Ne(\mathcal{S}) \bigcap \mathcal{S}_2 \neq \emptyset$. Since $\mathcal{S}_2 \subset \mathcal{S}'$, $Ne(\mathcal{S}) \bigcap \mathcal{S}_2 \subset \mathcal{S}'$ so, from (3) for $\mathcal{S}, \mathcal{S}'$, we have $Ne(\mathcal{S}) \bigcap \mathcal{S}_2 \subset \mathbb{V}_{i_{\mathcal{S}'}}$. So $i_{\mathcal{S}'} \in \mathcal{I}(\mathcal{S}_2)$. $i_{\mathcal{S}'} \notin \mathcal{I}(\mathcal{S})$ so $i_{\mathcal{S}'} \notin \mathcal{J}(\mathcal{S}_2)$. Therefore, $i_{\mathcal{S}'} \leq i_{\mathcal{S}_2}$. But since $\mathcal{S}_2 \subset \mathcal{S}'$, $i_{\mathcal{S}'} \geq i_{\mathcal{S}_2}$. So $i_{\mathcal{S}'} = i_{\mathcal{S}_2}$. Then, we know that $\mathcal{S}, \mathcal{S}_2$ verifies (1). Since $\mathcal{S} \subsetneq \mathcal{S}_2$, we also have (2). Finally $\{v' \in \mathcal{S}_2 \setminus \mathbb{V}_{i_{\mathcal{S}_2}} : v' \in Ne(\mathcal{S})\} = \{v' \in \mathcal{S}_2 \setminus \mathbb{V}_{i_{\mathcal{S}'}} : v' \in Ne(\mathcal{S})\} \subset \{v' \in \mathcal{S}' \setminus \mathbb{V}_{i_{\mathcal{S}'}} : v' \in Ne(\mathcal{S})\} = \emptyset$. This proves (3). Since we have (1-3), we know that $\mathcal{P}(\mathcal{S}_2) = \mathcal{S}$.

Main proof: Now let us prove the main result: Let' consider a subgraph $\mathcal{S}$ and assume that we cannot generate $\mathcal{S}'$ that verifies $\mathcal{P}(\mathcal{S}') = \mathcal{S}$ with the procedure from algorithm 1.

We take $v \in Ne(\mathcal{S}) \bigcap \mathbb{V}_{i'_{\mathcal{S}}}$. $\mathcal{S}_d = cl(\mathcal{S} \bigcup \{v\})$ verifies $\mathcal{S}_d \in \mathcal{C}$, $\mathcal{S} \subseteq \mathcal{S}_d$ and $\mathcal{S}_d \subset \mathcal{S}'$ (from Lemma 5). So, from lemma 7, we know that $\mathcal{P}(\mathcal{S}_d) = \mathcal{S}$ so $(\mathcal{S}, \mathcal{S}_d)$ verifies (1-3). Therefore, we generate $\mathcal{S}_d$ through lines 4-8 of algorithm 1. (Note that we can call $\mathcal{S}_d$ a *direct* child of $\mathcal{S}$, and the other subgraphs generated via lines 11-15 as its siblings).

We know that we generate through algorithm 1 from $\mathcal{S}_d$ a set of subgraphs $\mathcal{S}_s \subset \mathcal{S}'$. Let's then consider the largest $\mathcal{S}'' \subsetneq \mathcal{S}'$ generated with the algorithm 1, that is the one with the largest number of nodes. We note $\mathcal{T}$ the itemtable that accompanies the creation of $\mathcal{S}''$. Note that we know that $i_{\mathcal{S}''} = i_{\mathcal{S}'}$ since $i_{\mathcal{S}''} \leq i_{\mathcal{S}'}$ (lemma 6) and $i_{\mathcal{S}''} \geq i_{\mathcal{S}_d} = i_{\mathcal{S}'}$.

By assumption, $\mathcal{S}'' \subsetneq \mathcal{S}'$. Therefore, there exists a neighbour $v \in Ne(\mathcal{S}'') \bigcap \mathcal{S}'$ since $\mathcal{S}'$ and $\mathcal{S}''$ are connected subgraphs. Moreover, $\mathcal{S}_2 = cl(\mathcal{S}'' \bigcup \{v\}))$ is a child of $\mathcal{S}$ by Lemma 7 so ($\mathcal{S}$, verify (1-3) by Lemma 2.

Case 1: $\{\mathcal{I} \in \mathcal{T} : \mathcal{I} \subset \mathcal{I}(\mathcal{S})2)\} = \emptyset$**:** Since $\mathcal{S}_2 \subset \mathcal{S}'$, $i_{\mathcal{S}_2} \leq i_{\mathcal{S}'}$. However, since $\mathcal{S}'' \subset \mathcal{S}_2$, $i_{\mathcal{S}_2} \geq i_{\mathcal{S}''} = i_{\mathcal{S}'}$. So $i_{\mathcal{S}_2} = i_{\mathcal{S}'}$. Moreover, we already know that $(\mathcal{S}, \mathcal{S}_2)$ verify (1-3). We therefore try $\mathcal{S}_2$ following Line 5, we checked all the conditions (Line 11-12). That means we can created $\mathcal{S}_2 \supsetneq \mathcal{S}''$ which contradicts our assumption that $\mathcal{S}''$ is the largest closed subgraph strictly included in $\mathcal{S}'$ that could be generated.

Case 2: $\{\mathcal{I} \in \mathcal{T} : \mathcal{I} \subset \mathcal{I}(\mathcal{S})2)\} \neq \emptyset$**:** We note $v_1, \ldots, v_l$ the sequence that created $\mathcal{S}''$ from $\mathcal{S}_d$ through successive additions and closures (following Line 5). At one point in that process, we added a pattern to $\mathcal{T}$ that is now contained in $\mathcal{I}(cl(\mathcal{S}'' \bigcup \{v\}))$, let's say when adding v_k. Note that it cannot be the same pattern otherwise $\mathcal{S}''$ would not be closed. This pattern was linked to another equivalence group (Line 13).. If we consider v' a node from that group, we will then construct a subgraph using the sequence $v_1, \ldots, v_{k-1}, v', v_k, \ldots, ...v_l$. Note that since at each new addition, the constructed graph is included in $\mathcal{S}_2$, it's also included in $\mathcal{S}'$. Moreover, each one contains $\mathcal{S}$ and a node from $\mathbb{V}_{i_{\mathcal{S}'}}$ by construction (since it contains $\mathcal{S}_d$). So, using Lemma 2, we know that those additions verifies the conditions of Lines 11 and 12: they are valid additions according to our algorithm. This way, we can create a subgraph that contains $\mathcal{S}''$ and v' with our procedure. This contradicts our assumption that $\mathcal{S}''$ is the largest closed subgraph strictly included in $\mathcal{S}'$ that could be generated.

This proves that all $\mathcal{S}'$ will be generated from $\mathcal{S}$ and therefore that we have properly inverted the reduction

S-1.5 Proof of Lemma 4

The next two lemmas are directly taken from [Papaxanthos et al., 2016].

Minimal p-value of the CMH test

Lemma 8 ([Papaxanthos et al., 2016])**.** *The minimal p-value of the CMH test can be computed in* $O(J)$.

Proof The p-value associated with the CHM test, conditioning on the margins of all the tables, is:

$$\begin{aligned} p_{CMH}(\mathcal{S}, \mathbf{Y}, \mathbf{C}) =& 1 - \mathbf{F}_{\chi^2_1}\Big(\frac{\big(\sum_{j=1}^{J} a_{\mathcal{S},j} - \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\big)^2}{\sum_{j=1}^{J} \frac{n_{1,j}}{n_j - 1}\frac{n_{2,j}}{n_j} x_{\mathcal{S},j}(1 - \frac{x_{\mathcal{S},j}}{n_j})}\Big) \\ =& 1 - \mathbf{F}_{\chi^2_1}\Big(\frac{\big(a_{\mathcal{S}} - \sum_{j=1}^{J} \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\big)^2}{\sum_{j=1}^{J} \frac{n_{1,j}}{n_j - 1}\frac{n_{2,j}}{n_j} x_{\mathcal{S},j}(1 - \frac{x_{\mathcal{S},j}}{n_j})}\Big) \\ =& 1 - \mathbf{F}_{\chi^2_1}\Big(T_{\mathcal{S}}(a_{\mathcal{S}}, x_{\mathcal{S}})\Big) \end{aligned}$$

Since $\mathbf{F}_{\chi^2_1}$ is monotonically increasing, the minimal p-value is obtained for the smallest value $T_{\mathcal{S}}(a_{\mathcal{S}}, x_{\mathcal{S}}$. This is a function of $a_{\mathcal{S}}$ that is quadratic with a positive definite hessian[Papaxanthos et al., 2016] so the function is maximal for $\min a_{\mathcal{S}}$ or $\max a_{\mathcal{S}}$. We have that $a_{\mathcal{S},j,\min} = 0$ if $x_{\mathcal{S},j} \leq n_{2,j}$ and $a_{\mathcal{S},j,\min} = x_{\mathcal{S},j} - n_{2,j}$; and $a_{\mathcal{S},j,\max} = x_{\mathcal{S},j}$ for $x_{\mathcal{S},j} \leq n_{1,j}$ and $a_{\mathcal{S},j,\max} = n_{1,j}$ otherwise. So, $T^{\max}_{\mathcal{S}}(x_{\mathcal{S}}) = \max(T^l_{\mathcal{S}}, T^r_{\mathcal{S}})$ where

$$\begin{aligned} T^l_{\mathcal{S}} =& \frac{\big(\sum_{j=1}^{J} a_{\mathcal{S},j,\min} - \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\big)^2}{\sum_{j=1}^{J} \frac{n_{1,j}}{n_j - 1}\frac{n_{2,j}}{n_j} x_{\mathcal{S},j}(1 - \frac{x_{\mathcal{S},j}}{n_j})} \\ T^r_{\mathcal{S}} =& \frac{\big(\sum_{j=1}^{J} a_{\mathcal{S},j,\max} - \frac{x_{\mathcal{S},j} n_{1,j}}{n_j}\big)^2}{\sum_{j=1}^{J} \frac{n_{1,j}}{n_j - 1}\frac{n_{2,j}}{n_j} x_{\mathcal{S},j}(1 - \frac{x_{\mathcal{S},j}}{n_j})} \end{aligned}$$

Computing the envelope

Definition 3. *For each* $\mathcal{S} \in \mathcal{C}$*, the envelope of* $\mathcal{S}$ *is defined as* $\tilde{p}^\star(\mathcal{S}) \equiv \min_{\mathcal{S}'' : x_{\mathcal{S}''} \geq x_{\mathcal{S}}} p^*(\mathcal{S}'')$.

Lemma 9 ([Papaxanthos et al., 2016])**.** *If a subgraph is prunable, i.e* $\tilde{p}^\star(\mathcal{S}) > \alpha/k$*, then any subgraph* $\mathcal{S}' \supset \mathcal{S}$ *is also prunable*

Proof :

$$\begin{aligned}\mathcal{S}' \supset \mathcal{S} &\implies \{\mathcal{S}'' : x_{\mathcal{S}''} \geq x_{\mathcal{S}}\} \supset \{\mathcal{S}'' : x_{\mathcal{S}''} \geq x'_{\mathcal{S}}\} \\ &\implies \min_{\mathcal{S}'':x_{\mathcal{S}''}\geq x_{\mathcal{S}}} p^*(\mathcal{S}'') \leq \min_{\mathcal{S}'':x_{\mathcal{S}''}\geq x'_{\mathcal{S}}} p^*(\mathcal{S}'') \\ \tilde{p}^\star(\mathcal{S}) > \alpha/k \text{ and above} &\implies \tilde{p}^\star(\mathcal{S}') > \alpha/k\end{aligned}$$

Proof of Lemma 4 We consider the function

$$T_l(x'_{\mathcal{S}}) = \frac{\left(a_{\mathcal{S}',\min} - \sum_{j=1}^J \frac{x_{\mathcal{S}',j}n_{1,j}}{n_j}\right)^2}{\sum_{j=1}^J \frac{n_{1,j}}{n_j-1}\frac{n_{2,j}}{n_j}x_{\mathcal{S}',j}(1-\frac{x_{\mathcal{S}',j}}{n_j})}$$

with $x_{\mathcal{S},j} \leq x_{\mathcal{S}',j} \leq n_j$ and we will look at the partial derivatives. We add a few notations:

$$\begin{aligned}\mu =& \frac{n_{1,j}}{n_j-1}\frac{n_{2,j}}{n_j} \\ K =& \sum_{j=1}^J \frac{n_{1,j}}{n_j-1}\frac{n_{2,j}}{n_j}x_{\mathcal{S}',j}(1-\frac{x_{\mathcal{S}',j}}{n_j}) \\ K_{-i} =& \sum_{j=1,j\neq i}^J \frac{n_{1,j}}{n_j-1}\frac{n_{2,j}}{n_j}x_{\mathcal{S}',j}(1-\frac{x_{\mathcal{S}',j}}{n_j}) \\ S =& \sum_{j=1}^J \frac{x_{\mathcal{S}',j}n_{1,j}}{n_j} - a_{\mathcal{S},j,\min} \\ S_{-i} =& \sum_{j=1,j\neq i}^J \frac{x_{\mathcal{S}',j}n_{1,j}}{n_j} - a_{\mathcal{S},j,\min} \\ a'_{\mathcal{S},\min} =& \frac{\partial a_{\mathcal{S},\min}}{\partial x_{\mathcal{S}',i}}\end{aligned}$$

We then have

$$\begin{aligned}\frac{\partial T_l(x_{\mathcal{S}'})}{\partial x_{\mathcal{S}',i}} =& N_l(x_{\mathcal{S}'}) \times D_l(x_{\mathcal{S}'}) && \text{where} \\ N_l(x_{\mathcal{S}'}) =& 2(\frac{n_{1,i}}{n_i} - a'_{\mathcal{S},\min})K - (1-2\frac{x_{\mathcal{S}',i}}{n_i})\mu S && \text{and} \\ D_l(x_{\mathcal{S}'}) =& \frac{S}{K^2}\end{aligned}$$

For all j, $\frac{x_{\mathcal{S}',j}n_{1,j}}{n_j} - a_{\mathcal{S},j,\min} \geq \frac{x_{\mathcal{S}',j}n_{1,j}}{n_j} - x_{\mathcal{S}',j} + n_{2,j} = n_{2,j}(1-\frac{x_{\mathcal{S}',j}}{n_j}) \geq 0$ so $D_l(x_{\mathcal{S}'}) \geq 0$. We only need look at $N_l(x_{\mathcal{S}'})$ to find maxima. We can also note that consequently, $S \geq 0$ and $S_{-i} \geq 0$.

$$\begin{aligned}N_l(x_{\mathcal{S}'}) =& 2(\frac{n_{1,i}}{n_j} - a'_{\mathcal{S},\min})K - (1-2\frac{x_{\mathcal{S}',i}}{n_j})\mu S \\ =& 2\frac{n_{1,i}}{n_i}K_{-i} + 2\frac{n_{1,i}}{n_i}\mu x_{\mathcal{S}',i}(1-\frac{x_{\mathcal{S}',i}}{n_i}) - 2a'_{\mathcal{S},\min}K_{-i} - 2a'_{\mathcal{S},\min}\mu x_{\mathcal{S}',i}(1-\frac{x_{\mathcal{S}',i}}{n_i}) - \\ & \mu S_{-i} - \mu(\frac{x_{\mathcal{S}',i}n_{1,i}}{n_i} - a_{\mathcal{S},i,\min}) + 2\frac{x_{\mathcal{S}',i}}{n_i}\mu S_{-i} + 2\frac{x_{\mathcal{S}',i}}{n_i}\mu(\frac{x_{\mathcal{S}',i}xn_{1,i}}{n_i} - a_{\mathcal{S},i,\min}) \\ =& \left[2\frac{n_{1,i}}{n_i}K_{-i} - 2a'_{\mathcal{S},\min}K_{-i} - \mu S_{-i}\right] + x_{\mathcal{S}',i}\left[2\mu\frac{n_{1,i}}{n_i} - 2a'_{\mathcal{S},\min}\mu - \mu\frac{n_{1,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\right] + \\ & - 2\frac{n_{1,i}}{n_i^2}\mu x_{\mathcal{S}',i}^2 + 2\frac{n_{1,i}}{n_i^2}\mu x_{\mathcal{S}',i}^2 + 2a'_{\mathcal{S},\min}\mu\frac{x_{\mathcal{S}',i}^2}{n_i} - 2a_{\mathcal{S},i,\min}\mu\frac{x_{\mathcal{S}',i}}{n_i} + \mu a_{\mathcal{S},\min}\end{aligned}$$

We then have two cases:

$x_{\mathcal{S}',i} \leq n_{2,i}$**:** Then, $a_{\mathcal{S},i,\min} = 0$ and $a'_{\mathcal{S},\min} = 0$. So

$$N_l(x_{\mathcal{S}'}) = \big[2\frac{n_{1,i}}{n_i}K_{-i} - \mu S_{-i}\big] + x_{\mathcal{S}',i}\big[\mu\frac{n_{1,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\big]$$

It is an affine function with a positive slope. Moreover, at the boundary at $n_{2,i}$, the function is positive. So the function is maximal at $n_{2,i}$.

$x_{\mathcal{S}',i} \geq n_{2,i}$**:** Then, $a_{\mathcal{S},i,\min} = x_{\mathcal{S}',i} - n_{2,i}$ and $a'_{\mathcal{S},\min} = 1$. So

$$\begin{aligned} N_l(x_{\mathcal{S}'}) =& \big[2\frac{n_{1,i}}{n_i}K_{-i} - 2K_{-i} - \mu S_{-i} - \mu n_{2,i}\big] + x_{\mathcal{S}',i}\big[\mu\frac{n_{1,i}}{n_i} - 2\mu + \mu + 2\mu\frac{n_{1,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\big] + \\ & 2\mu\frac{x^2_{\mathcal{S}',i}}{n_i} - 2\mu\frac{x^2_{\mathcal{S}',i}}{n_i} \\ =& \big[2\frac{n_{1,i}}{n_i}K_{-i} - 2K_{-i} - \mu S_{-i}\big] + x_{\mathcal{S}',i}\big[\mu\frac{n_{2,i}}{n_i} + 2\frac{\mu}{n_i}S_{-i}\big] \end{aligned}$$

So $N_l(x_{\mathcal{S}'})$ is an affine by-piece function of $x_{\mathcal{S}',i}$, whose slope $A_l(x_{\mathcal{S}',-i}) \geq 0$. So, the only possible maxima are at the boundary, where $x_{\mathcal{S}',i} = n_{2,i}$ or $x_{\mathcal{S}',i} = n_i$. Since this is true for all values of $x_{\mathcal{S}',-i}$, we know that we can only achieve a maximum for T_l at the boundaries. So the only two possible maxima are $n_{2,i}$ and n_i. Note that, in the case where $x_{\mathcal{S}',i} \geq n_{2,i}$, then the possible maxima becomes $x_{\mathcal{S}',i}$ and n_i so in general, the two possible maxima for $T_l(x'_{\mathcal{S}})$ are $\max\{n_{2,i}, x_{\mathcal{S}',i}\}$ and n_i.

The same proof holds for T_r (given the symmetry of the expressions), where the maxima is in $\{\max(x_{\mathcal{S},i}, n_{1,i}), n_i\}$. This proves the lemma.We have reduced the space of possibilities from $O(m^J)$ to $O(2^J)$ (with m the geometric mean of $x_{\mathcal{S}'}$).

We now need to show how to compute this value in $O(J\log(J))$. For this, we rely on the following theorem.

Lemma 10 ([Papaxanthos et al., 2016])**.** *Let $\mathcal{S}$ be a potentially testable subgraph and define $\beta^l_{\mathcal{S}',1} = \frac{n_{1,j}x_{\mathcal{S}',j}}{n_j^2}$ and $\beta^l_{\mathcal{S}',1} = \frac{n_{2,j}x_{\mathcal{S}',j}}{n_j^2}$, for $j \in \{1,\dots,J\}$ Let π_l and π_r be permutations of $\{1,\dots,J\}$ such that $\beta^l_{\mathcal{S}',\pi_l(1)} \leq \dots \leq \beta^l_{\mathcal{S}',\pi_l(J)}$ and $\beta^r_{\mathcal{S}',\pi_r(1)} \leq \dots \leq \beta^r_{\mathcal{S}',\pi_r(J)}$, respectively.*
Then, there exist $\kappa \in \{1,\dots,J\}$ such that the optimum $x^*_{\mathcal{S}'}$ satisfies either

- *$x^*_{\mathcal{S}',\pi_l(j)} = x_{\mathcal{S},\pi_l(j)}$ for $j \leq \kappa$ and $x^*_{\mathcal{S}',\pi_l(j)} = n_j$ otherwise*

or

- *$x^*_{\mathcal{S}',\pi_r(j)} = x_{\mathcal{S},\pi_r(j)}$ for $j \leq \kappa$ and $x^*_{\mathcal{S}',\pi_r(j)} = n_j$ otherwise*

Proof of lemma 10 We will do the proof for $T_l(x'_{\mathcal{S}})$. The proof for $T_r(x'_{\mathcal{S}})$ is identical, up to notation.

We can note that since the optimal is at least equal to $n_{2,j}$, the value of $a_{\mathcal{S},j,\min}$ is $x_{\mathcal{S}',j} - n_{2,j}$. We note $\beta_j = \frac{n_{1,j}x^*_{\mathcal{S}',j}}{n_j^2}$. We also note $l(\beta_j) = n_{1,j}n_j(1 - \frac{n_j\beta_j}{n_{1,j}}) = n_{2,j}(1 - \frac{x^*_{\mathcal{S}',j}}{n_j})$. With those notations, we can write

$$T_l(x'_{\mathcal{S}}) = \frac{\sum_{j=1}^J l(\beta_j)}{\sum_{j=1}^J \beta_j l(\beta_j)}$$

It is straightforward to see that, if $x^*_{\mathcal{S}',j} = n_j$, then $l(\beta_j) = 0$. We are then exactly in the setting of Papaxanthos et al. [2016] and we refer the reader to the proof in the supplementary, p.3-6.

This shows that we can compute the envelope in $J\log(J)$

S-2 Efficient implementation of CALDERA

We consider a subgraph $\mathcal{S}'$ created following Algorithm 1, in the second case (Lines 11-15). We therefore have $\mathcal{S}, \mathcal{S}_p$ such that: $\mathcal{P}(\mathcal{S}') = \mathcal{P}(\mathcal{S}) = \mathcal{S}_p$ and $i_{\mathcal{S}'} = i_{\mathcal{S}}$. After creating $\mathcal{S}'$, we explore its children, with an itemtable $\mathcal{T}$. All elements of $\texttt{Children}(\mathcal{S}', \mathcal{S}_p, i_{\mathcal{S}'}, \mathcal{T})$ will have a pattern which includes $\mathcal{I}(\mathcal{S}')$. Moreover, by definition of the equivalence groups, we already know that $\{\mathcal{I} \in \mathcal{T} : \mathcal{I} \subset \mathcal{I}(\mathcal{S}')\} = \emptyset$. Therefore, when constructing $\mathcal{S}'' \in \texttt{Children}(\mathcal{S}', \mathcal{S}_p, i_{\mathcal{S}'}, \mathcal{T})$, only the elements in $\mathcal{I}(\mathcal{S}'') \setminus \mathcal{I}(\mathcal{S}')$ need to be considered.

We store $\mathcal{T}$ as a matrix of binary patterns. Therefore, some columns can be deleted without loss of information: in Line 13 of algorithm 1, we only keep the columns that are not in $\mathcal{I}(\mathcal{S})$. As the `Children` function is called recursively, the itemtable $\mathcal{T}$ will grow in the number of patterns saved (*i.e* number of rows) but the memory footprint of each pattern will be smaller (*i.e* fewer columns).

S-3 Background on the minimal p-value

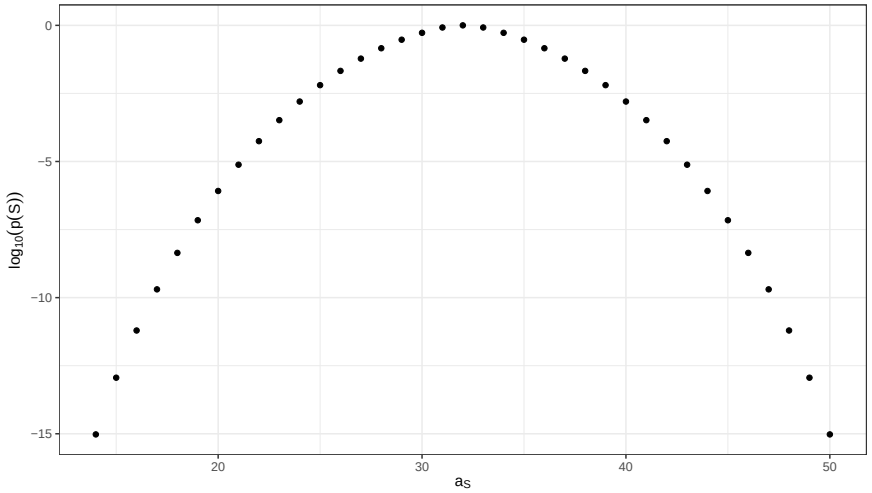


Figure S1: *Finite numbers of possible p-values (log scale) for a fixed value of* $n1 = 50$ *and* $x_{\mathcal{S}} = 64$. *Using the notation from table 1, with* $J = 1$, $n_1 = 50$, $n = 100$ *and* $x_{\mathcal{S}} = 64$, *the p-value of the* χ^2 *test is computed for all possible values of* $a_{\mathcal{S}}$. *Since there are only a finite number of possible* $a_{\mathcal{S}}$ *values, there are a finite number of possible p-values, and therefore a smallest one. This minimal p-value can be computed from* $x_{\mathcal{S}}$, $n1$ *and* n *alone and is* $\sim 10^{-15}$

S-4 Benefits of the new envelope

To demonstrate the situations where the new envelope is beneficial, and where it is not, we consider a situation where $n = 280, J = 2$. Then, we look at two cases: $n_1 = n_2 = 140$ and $n_1 = 13 \times n_2 = 260$. In both settings, we compute the envelope as defined in Papaxanthos et al. [2016] and in our case. Then, for $\alpha = 10^{-8}$ (a value that we used in practice) and for all possible values of $\{x_{\mathcal{S},1}, x_{\mathcal{S},2}\}$, we consider whether we would prune (for $k = 1$) using the definition of the envelope from Papaxanthos et al. [2016] or the extended new bound defined in this paper. The new bound nearly doubles the space of prunable subgraphs when there is a clear imbalance, as evidenced in Fig S3b, while it has no effect when the two populations are perfectly balanced, as in Fig S3a.

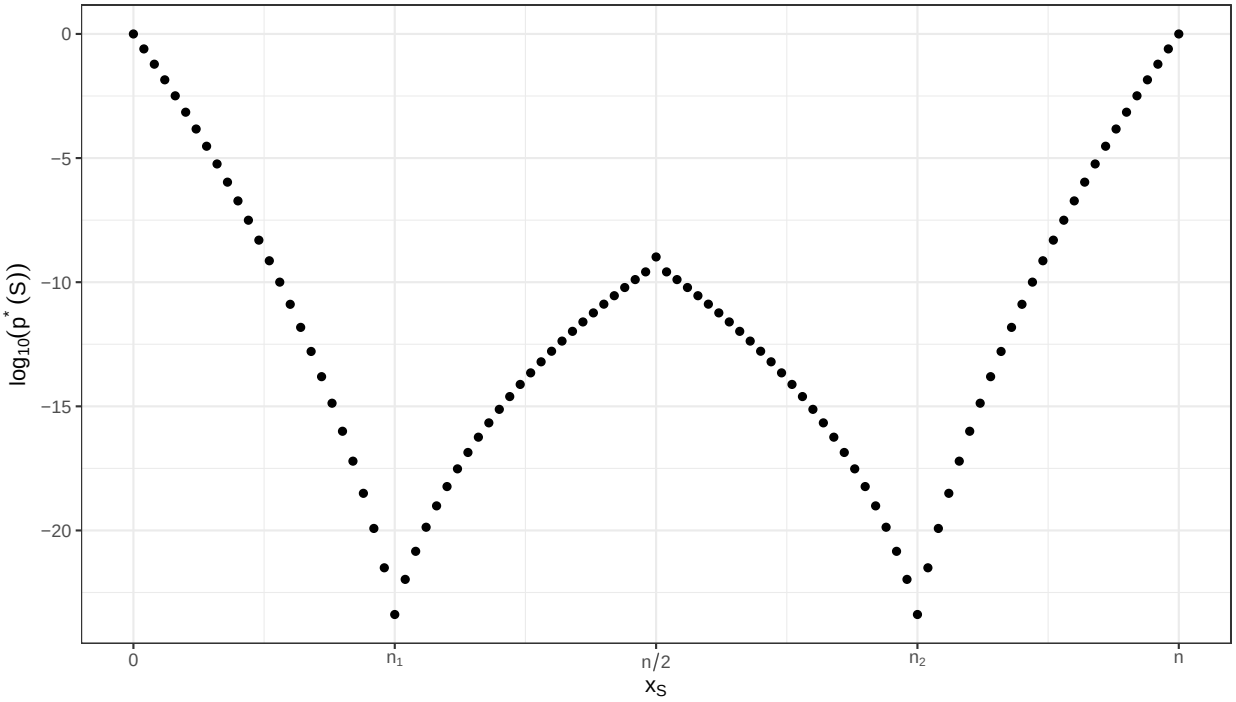


Figure S2: Inspired from Llinares-López et al. [2015] *Minimum p-value as a function of* $x_{\mathcal{S}}$ *for fixed values of* $n_1 = 25$ *and* $n = 100$. *Using the notation from table 1, with* $J = 1$, $n_1 = 25$, $n = 100$, *the minimal p-value* $p^*(\mathcal{S})$ *of the* χ^2 *test is computed for all possible values of* $x_{\mathcal{S}}$. *For* $x_{\mathcal{S}} \geq \max(n_1, n_2)$, *the minimal p-value is strictly increasing. If we reach that stage, we can prune the graph and stop the exploration in that direction. Indeed, if* $\mathcal{S}' \supseteq \mathcal{S}$ *then* $x_{\mathcal{S}'} \geq x_{\mathcal{S}}$. *So if* $p^*(\mathcal{S}) > \frac{\alpha}{k}$, *we know that* $p^*(\mathcal{S}') > \frac{\alpha}{k}$ *without computing it.*

S-5 Benefit of breadth-first search

S-5.1 A simplified scenario

We consider a very simple graph with $p = 3$ nodes, $J = 1$ population and $n = 12$ samples. The graph is displayed in Fig S4a. Using the reduction from `CALDERA`, we generate a tree structure on $\mathcal{C}$, displayed in Fig S4b.

Then we can explore this structure in depth-first or breadth-first, while pruning using $\alpha = 1$. The order resulting from an exploration in depth-first can be found in Table S1 and the order fom the exploration in breadth-first can be found in Table S2. In this simple setting, exploring in breadth only visits 4 subgraphs while exploring in depth visits 7. This is because the BFS enumerates testable subgraphs more quickly, thereby increasing k and lowering the threshold, which means that the branch starting at $\{v_1\}$ is pruned earlier in the exploration.

Subgraph explored	Number of subgraphs explored	Value of the threshold	Testable subgraphs
$\{v_1\}$	1	.15	$\{\{v_1\}\}$
$\{v_1, v_2\}$	2	.15/2	$\{\{v_1\}, \{v_1, v_2\}\}$
$\{v_1, v_2, v_3\}$	3	.15/2	$\{\{v_1\}, \{v_1, v_2\}\}$
$\{v_1, v_3\}$	4	.15/3	$\emptyset$
$\{v_2\}$	5	.15/3	$\{\{v_2\}\}$
$\{v_2, v_3\}$	6	.15/3	$\{\{v_2\}, \{v_2, v_3\}\}$
$\{v_3\}$	7	.15/3	$\{\{v_2\}, \{v_2, v_3\}, \{v_3\}\}$

Table S1: Order of exploration of the elements of $\mathcal{C}$ while exploring depth-first

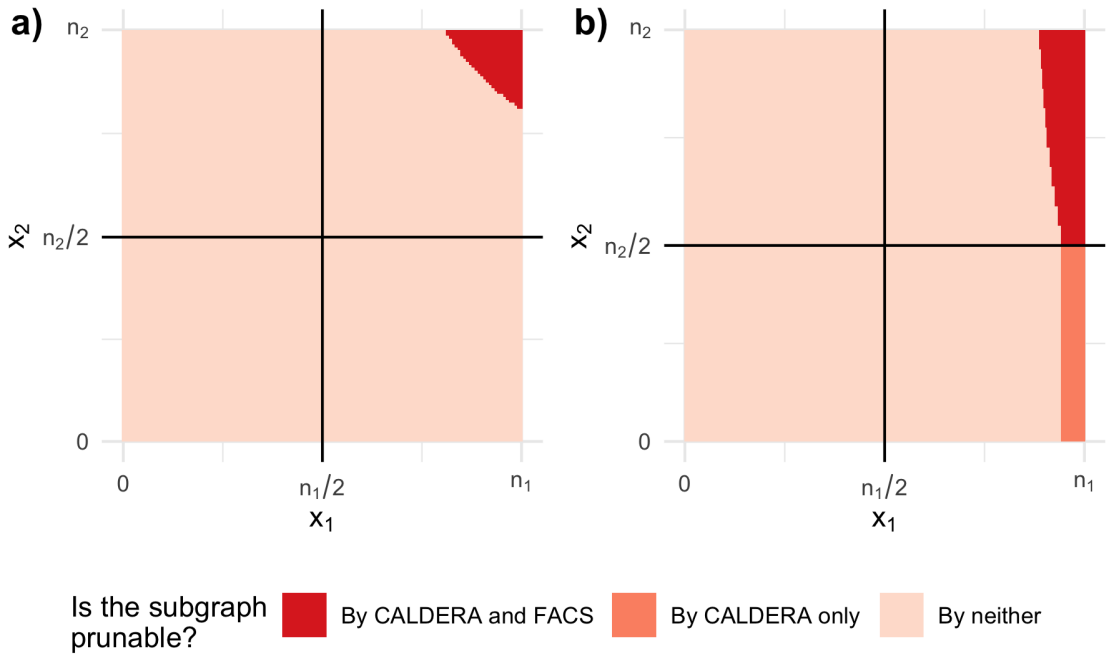


Figure S3: We consider the space of all possible patterns for $n = 280, J = 2$ and two cases: a) $n_1 = n_2 = 140$ and b) $n_1 = 260 = 13 \times n_2$. The phenotypes are well balanced in each population and $alpha = 10^{-8}$. The extended lower bound increases the number of prunable subgraphs when the populations are imbalanced.

Subgraph explored	Number of subgraphs explored	Value of the threshold	Testable subgraphs
$\{v_1\}$	1	.15	$\{\{v_1\}\}$
$\{v_2\}$	2	.15/2	$\{\{v_1\}, \{v_2\}\}$
$\{v_3\}$	3	.15/3	$\{\{v_2\}, \{v_3\}\}$
$\{v_2, v_3\}$	4	.15/3	$\{\{v_2\}, \{v_2, v_3\}, \{v_3\}\}$

Table S2: Order of exploration of the elements of $\mathcal{C}$ while exploring breadth-first

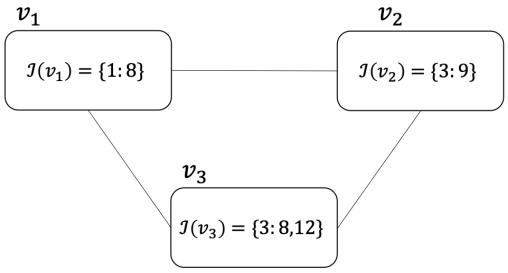
S-5.2 A more general setting to understand why BFS is more efficient than DFS

We consider a very simple graph model where, for $v \in \mathcal{V}$ and $i \in \{1, \dots, n\}$, $i \in \mathcal{I}(v) \sim \text{Binom}(\text{prop})$ and the patterns are independent across nodes . We have no population structure, which means that we consider Fisher's exact test. For a given level α, we want to compute $f(\alpha, \text{prop}) = \mathbb{P}(p^\star(\{v\}) > \alpha, \forall v \in \mathcal{V})$, that is the probability that no subgraph is testable at the first stage of our tree on $\mathcal{C}$.

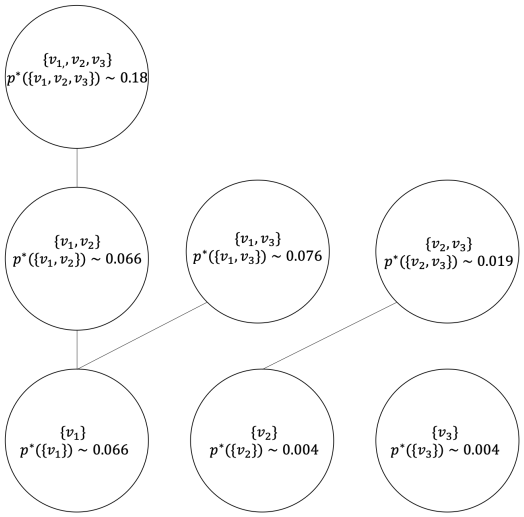
Since we consider Fisher's exact test, there is a bijection between $p^\star(\{v\})$ and $x_{\{v\}}$ so $p^\star(\{v\}) > \alpha \implies x_{\{v\}} \geq \sigma(\alpha)$. Moreover, $x_{\{v\}} \sim \mathcal{B}(\text{prop}, n)$, so $f(\alpha, \text{prop}) = 1 - \left(\mathbf{F}_{\mathcal{B}(\text{prop},n)}(\sigma_\alpha)\right)^p$ with $\mathbf{F}_{\mathcal{B}(\text{prop},n)}$ the cumulative distribution function of the binomial (prop, n). Since the nodes are independent, the distribution of $x_{\mathcal{S}}$ at any stage of the tree can be computed by recursion. We furthermore assume that the graph structure is such that the number of closed subgraphs is $s \times p$ at stage s.

In Fig S4c, we display the probability that any subgraph is 1-testable or prunable at stage s, for $s \in \{1, 2, 3\}$, $p = 100$ and $\alpha = 10^{-4}$.

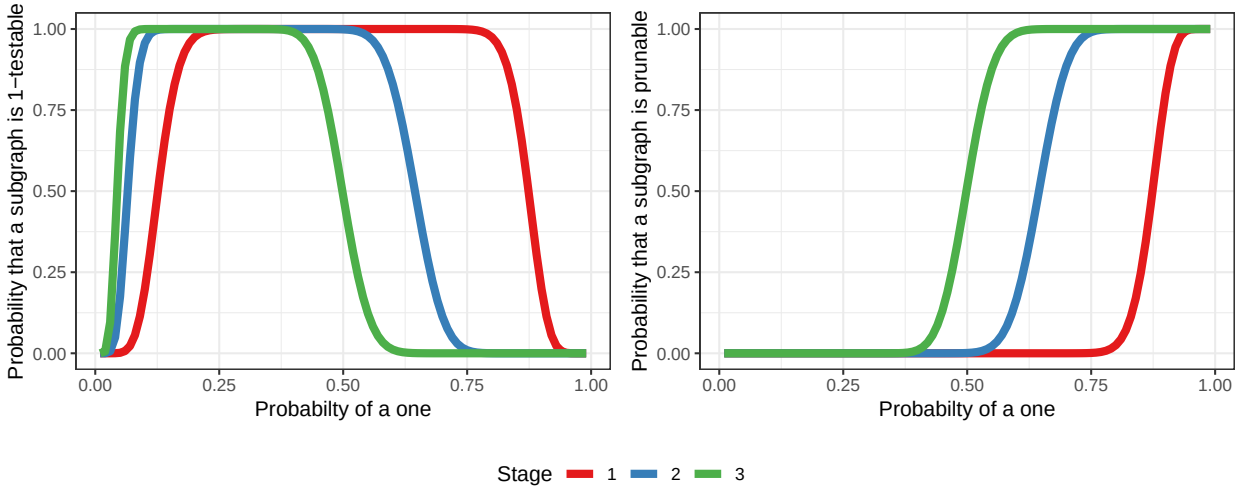
For most of the range of values, there is at least one testable subgraph in the first stage. So, by exploring in a BFS manner, we start the second stage with a much lower threshold (i.e., a much higher value of k) which leads to more pruning. For very low values of `prop`, there might be no testable subgraphs at the first



(a) A simple graph example with $p = 3$, $J = 1$ and $N = 12$ which we explore using `CALDERA` with pruning, and $\alpha = .15$



(b) Order on elements of $\mathcal{C}$ from the graph in a), according to the reduction of definition 2



(c) $n = 100, p = 100, \alpha = 10^{-4}$. For a simple model described in S-5.2 where $i \in \mathcal{I}(v) \sim \text{Binom}(\text{prop})$, we plot the probability that any subgraph is 1-testable or prunable as a function of prop

Figure S4: Simple examples where the search in breadth-first is much more efficient that depth-first

stage but there will be at the second stage, which still justifies an exploration in depth. Note that for large p, we can see that there is no testable subgraph at the stages 2 and 3. That is because all such subgraphs have a pattern that is too large. While there may be not testable subgraphs, there are many prunable ones. In that case, an exploration in breadth-first or depth-first would be identical.

This example simplifies two aspects which have opposite effects. The first is that, in practice, the probability of $i \in \mathcal{I}(v)$ is of course not uniform across the graph. It is a distribution with much heavier tails which means that, even if the average number of 1 might be small, it is still quite likely that at least one subgraph is testable. The second is that the patterns of neighbouring nodes are correlated. As such, the patterns cannot increase by as much between stages, which limits both the increase in testable pattern discovery, and the pruning.

S-6 Speed Simulations

S-6.1 General simulation settings

For given values of n (number of samples) and p (number of nodes), we first generate n samples with phenotype $y_i \in \{0, 1\}$ such that $\mathbf{P}(y_i = 0) = prop$ (user defined parameter). Then, we generate p nodes. 10% of the nodes will be associated with the phenotype. For each node in the remaining 90%, we randomly generate 3 edges between this node and another in the 90%. The average degree is therefore 6. For those nodes v_j, the associated pattern $\mathcal{I}(v_j)$ is a random vector such that $\mathbf{P}(i \in \mathcal{I}(v_j)) = 0.5$.

Then, we generate the remaining 10% of the nodes associated with the phenotype. We first generate associated patterns $\mathcal{I}_{sig}$ such that $\mathbf{P}(i \in \mathcal{I}_{sig} | y_i = 1) = 0.95$ and $\mathbf{P}(i \in \mathcal{I}_{sig} | y_i = 0) = 0.05$. Then, those patterns are split into 10 significant nodes sig_j such that $\mathbf{P}(i \in \mathcal{I}(\text{sig}_j) | i \in \mathcal{I}_{sig}) = 0.9$ and $\mathcal{I}(\bigcup_{j \in [1 \ldots 10]}) = \mathcal{I}_{sig}$.

S-6.2 Parameters for various scenarios

We increase p until `COIN+LAMP2` times out (sometimes we went a little further to continue investigation the behaviors).

Scenario	1	2	3	4
n	100	50	100	100
$prop$	.5	.5	.2	.2
α	.05	.05	.05	10^{-4}
timeout (days)	2	1	1	1
max value of p	2×10^4	2×10^4	2×10^3	5×10^4

Table S3: Parameter values for the simulations

S-6.3 Results on all scenarios

All computations were run on a r3.4xlarge AWS machine with 16 vCPUs (8 physical ones) and 122GiB[AWS, 2020]. We stop every method once it runs for more than timeout. We also stopped running the entire scenario once we have reached timeout for `COIN+LAMP2` (expect for scenario 3 where we continued further to study the behavior of the various modes of `CALDERA`.

S-6.4 Memory requirements

We also launched scenario 2 while monitoring memory usage for `COIN+LAMP2`, `CALDERA BFS` and `CALDERA DFS`. `CALDERA DFS` uses 1/3 of the peak memory of `CALDERA BFS`. This is expected since the tree structure that is explored scales in p in breadth but in $n << p$ in depth. Memory-wise, `CALDERA BFS` is on par with `COIN+LAMP2`, which relies on a DFS search. This shows that the use of local itemset tables offers memory gains that are enough to offset the exploration in breath, while providing large speed gains. This also suggests that hybrid explorations might be even better at navigating the memory-speed trade-off.

S-6.5 Imbalance

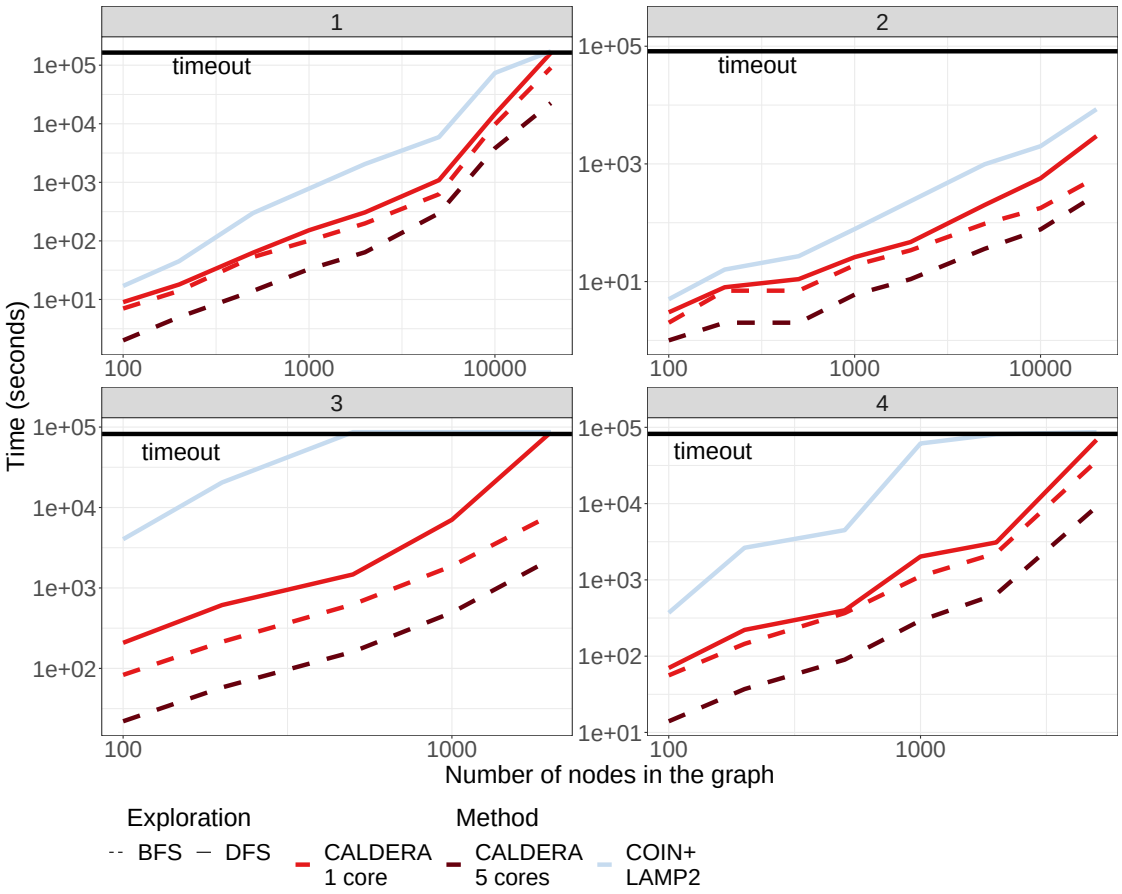


Figure S5: Runtimes for `CALDERA` and `COIN+LAMP` on graphs with various values of covariates p and various values of the simulation parameters.

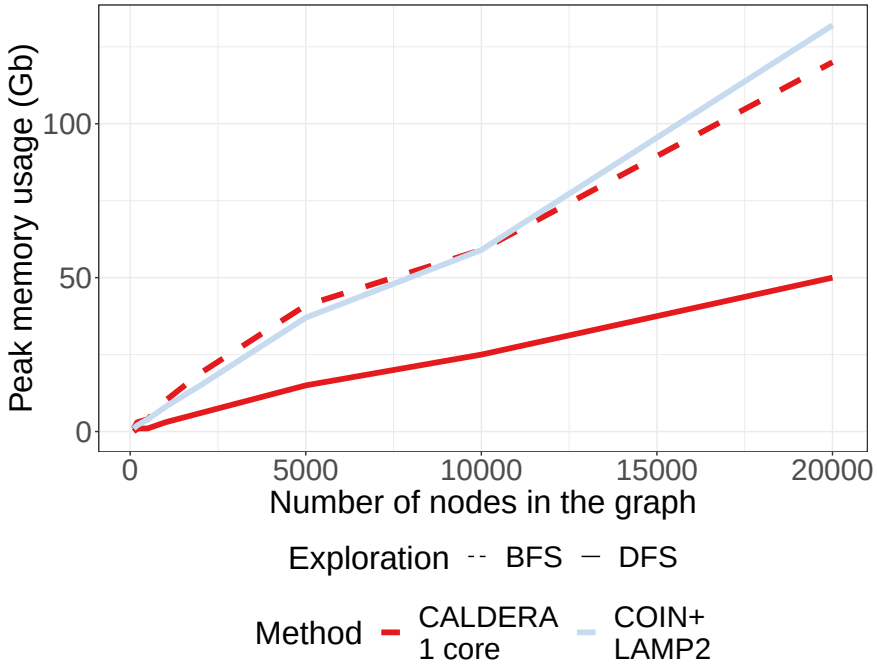


Figure S6: Peak memory usage for `CALDERA` and `COIN+LAMP` on graphs with various values of covariates p.

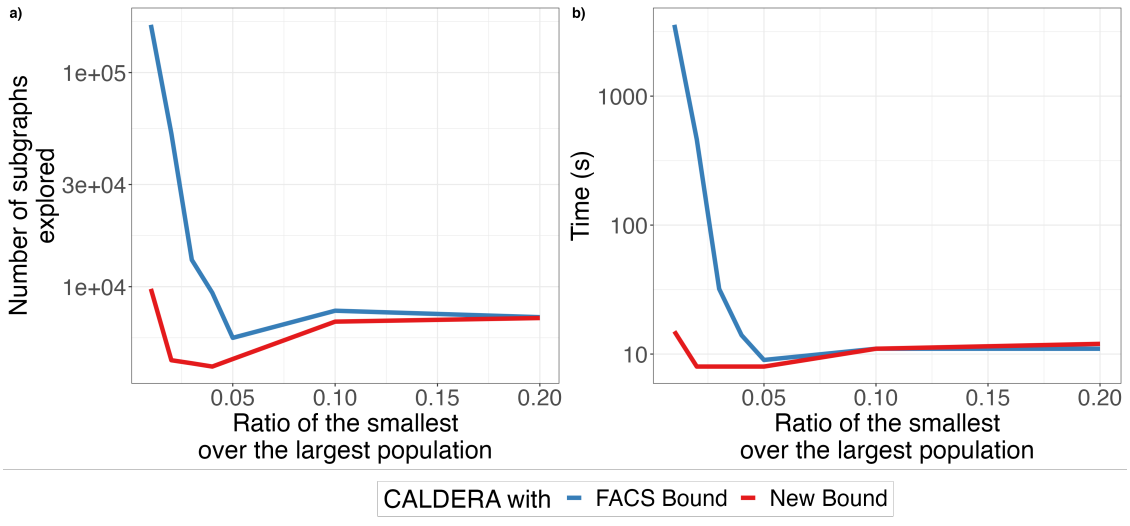


Figure S7: Impact of the new envelope bound on the number of explored subgraphs **a)** and subsequently on runtime **b)** when there is clear imbalance between the size of the two populations.

S-7 Power Simulations

S-7.1 Data generation

We first generate two sequences of nucleotides of 1000 bps, gene A and gene B. Then, for each gene, we generate 10 versions. For each version, we start from the original copy and introduce mutations in the following manners: at each base, there is a 1% chance of a point mutation (with each mutation equally likely), a 1% chance of a deletion, a 1% chance of a insertion after the base (with each insertion being equally likely) and therefore a 97% chance that nothing happens. We then generate 50 sensitive samples by randomly selecting with replacement a version of gene B: this is their entire genome. We also generate 50 resistant samples by randomly selecting with replacement a version of gene A and a version of gene B and collating them together. We therefore obtain 100 genetic sequences, 50 of each phenotype.

S-7.2 Testing all methods

We build a De Bruijn Graph of 704 unitigs using the first step of `DBGWAS`. Then, we run `DBGWAS` with the parameter *-SFF q1.0* to avoid filtering any unitig and then keep those whose p-values are smaller than $\alpha/704$. We also tests the pattern of each unitig for association with the vector of phenotypes. We use the `LAMP2` procedure to remove non-testable hypotheses, at levels of α. Finally, we run `CALDERA` with options *–Lmax=2000 -C 1*, and we either specify the maximum number of stages (among $[1, 2, 3, 5, 10, 15, 20]$) or we run all stages. We look at the list of significant CCSs and keep all their unitigs as significant.

S-7.3 Results

We compute two metrics: the proportion of unitigs called significant divided by all the unitigs that originated from gene A, named **coverage**; and the proportion of unitigs called significant divided by all the unitigs that originated from gene B, named **False Positive Rate**. We can plot those values for all three methods, and over all ranges of the parameter *–stage* for `CALDERA`

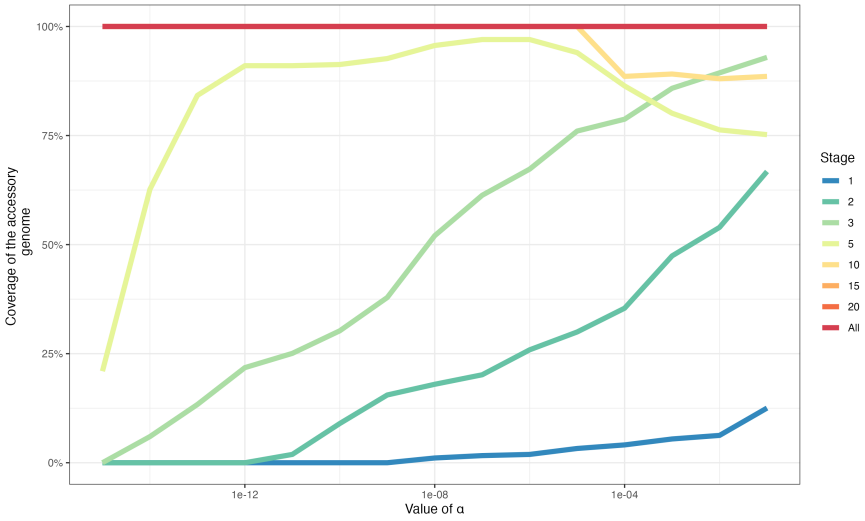


Figure S8: Proportion of all unitigs associated with the resistant phenotype that are found to be significant by `CALDERA` when varying the maximum number of stages explored.

S-8 Network-guided GWAS on *A. thaliana* genomes

We obtained over 6 millions SNPs and a "date to flowering" phenotype for $n = 936$ *A. thaliana* genomes from easyGWAS [Grimm et al., 2017]. We also obtained 137 *A. thaliana* metabolic pathways from KEGG [Kanehisa

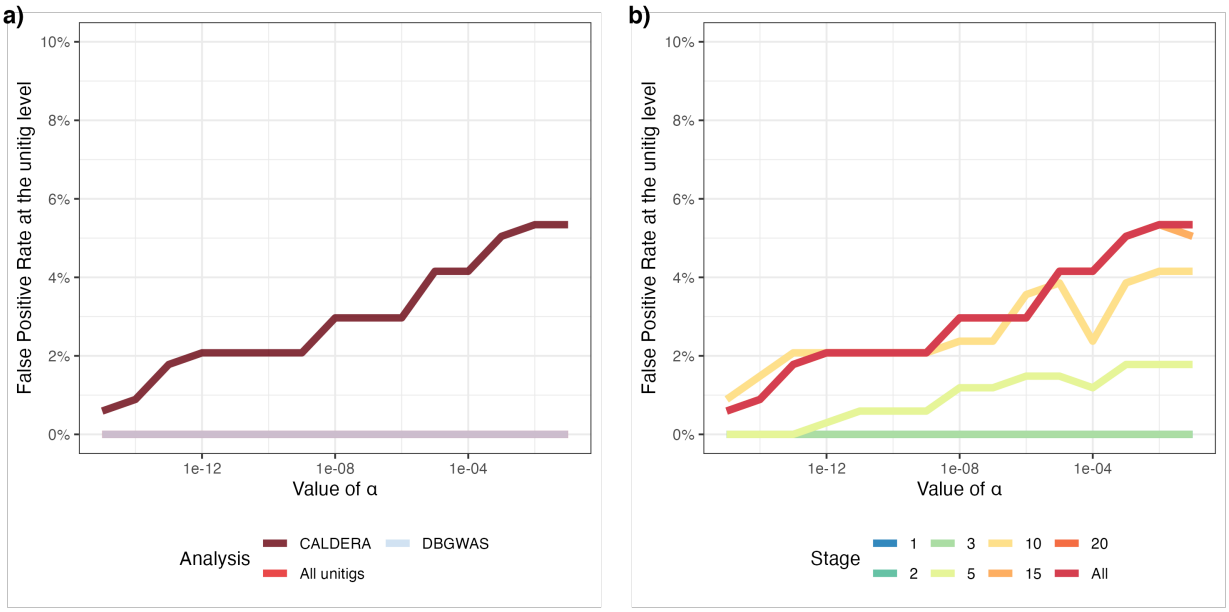


Figure S9: Proportion of all unitigs not associated with the resistant phenotype that are found to be significant as the value of α changes **a)** by `CALDERA`, the `LAMP2` procedure on all unitigs and `DBGWAS`, and **b)** by `CALDERA` when varying the maximum number of stages explored.

and Goto, 2000] using the `KEGGrest` [Tenenbaum, 2020] and `DEGraph` [Jacob et al., 2012] R packages. The union of the pathways involved $p = 3150$ genes and the average degree of the resulting graph is ~ 21.2. We mapped each SNP to the closest gene using `snpEff` [Cingolani et al., 2012] and defined each gene to be mutated in a sample if it contained at least one mutation mapping to the gene. Runtime was under a minute using 8 cores. $k_0 = 70$ and 10 subgraphs are found to be significant. In particular, the first and second most significant subgraphs involve pathways `ATH00260`:*Glycine, serine and threonine metabolism* and `03013`: *RNA transport*, which were previously linked to flower development [Hesse and Hoefgen, 2003, Pfaff et al., 2018].

S-9 Behavior of `CALDERA` against BFS stages

Figure S10 shows the number of unitigs belonging to at least one significant CCS for increasing BFS stages on the **Pseudomonas** dataset with $\alpha = 10^{-6}$. This number increases sharply at stage 4, where the large CCS containing the plasmid are found, then increases more slowly and reaches a plateau after stage 6.

Figure S11 shows the computational ressources used by `CALDERA` against the BFS stage. Memory usage is exponential in the BFS stage, which is consistent with the number k_0 of explored CCS (not shown), but as seen above, this does not reflect the final coverage of the compacted DBG by significant CCS. The elapsed time increases linearly, with some variation that can be explained by external factors impacting the computation nodes. We do not report the results for stage 8 on the same plot as they were done with a smaller batch size of $100,000$ and are therefore not comparable: the corresponding run used 947Gb of RAM and took 44h30.

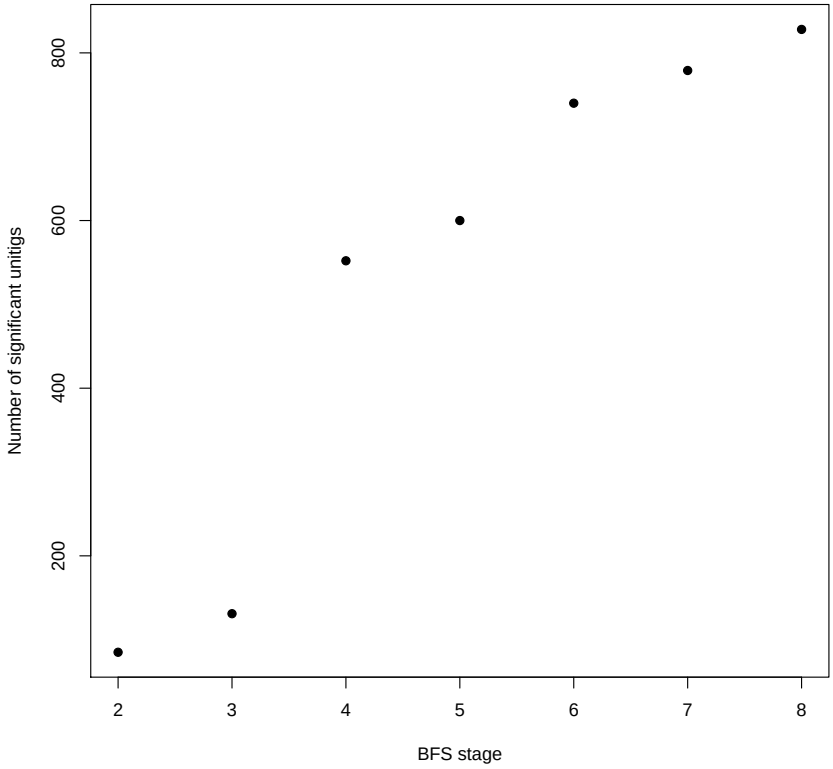


Figure S10: Number of unitigs belonging to at least one significant CCS for increasing BFS stages on the **Pseudomonas** dataset with $\alpha = 10^{-6}$.

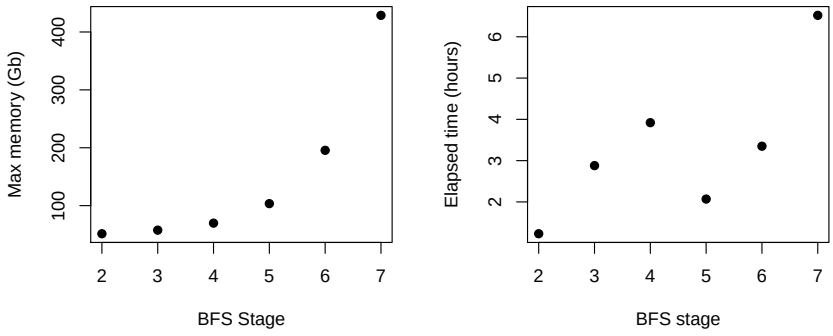


Figure S11: Computational ressources used for increasing BFS stages on the **Pseudomonas** dataset with $\alpha = 10^{-6}$, using a batch size of $200,000$ and four cores. Left panel: peak memory used during computation in Gb. Right panel: elapsed time in hours.