BMC Bioinformatics

RESEARCH

Open Access

CausNet: generational orderings based search for optimal Bayesian networks via dynamic programming with parent set constraints

Nand Sharma* and Joshua Millstein

*Correspondence:
nandsh11@gmail.com

Division of Biostatistics, Department of Population and Public Health Sciences, University of Southern California, Los Angeles, USA

Abstract

Background: Finding a globally optimal Bayesian Network using exhaustive search is a problem with super-exponential complexity, which severely restricts the number of variables that can feasibly be included. We implement a dynamic programming based algorithm with built-in dimensionality reduction and parent set identification. This reduces the search space substantially and can be applied to large-dimensional data. We use what we call 'generational orderings' based search for optimal networks, which is a novel way to efficiently search the space of possible networks given the possible parent sets. The algorithm supports both continuous and categorical data, as well as continuous, binary and survival outcomes.

Results: We demonstrate the efficacy of our algorithm on both synthetic and real data. In simulations, our algorithm performs better than three state-of-art algorithms that are currently used extensively. We then apply it to an Ovarian Cancer gene expression dataset with 513 genes and a survival outcome. Our algorithm is able to find an optimal network describing the disease pathway consisting of 6 genes leading to the outcome node in just 3.4 min on a personal computer with a 2.3 GHz Intel Core i9 processor with 16 GB RAM.

Conclusions: Our generational orderings based search for optimal networks is both an efficient and highly scalable approach for finding optimal Bayesian Networks and can be applied to 1000 s of variables. Using specifiable parameters—correlation, FDR cutoffs, and in-degree—one can increase or decrease the number of nodes and density of the networks. Availability of two scoring option—BIC and Bge—and implementation for survival outcomes and mixed data types makes our algorithm very suitable for many types of high dimensional data in a variety of fields.

Keywords: Optimal Bayesian network, Dynamic programming, Generational orderings

Introduction

Optimal Bayesian network (BN) structure discovery is a method of learning Bayesian networks from data that has applications in wide variety of areas including epidemiology (see e.g. [1–4]). Disease pathways found using directed BN edges leading to a phenotype outcome can improve understanding, diagnosis and treatment of a disease. The main challenge in finding an optimal BN lies in the super-exponential complexity of the search [5]. Dynamic programming can reduce the complexity to exponential [6, 7], but still the number of features/nodes feasibly explored remains very small—usually no more than 30—and only by restricting the maximum number of parents for each node [6, 7]. To alleviate the challenge of high dimensionality, we implement a dynamic programming algorithm with parent set constraints. We use what we call 'generational orderings' based search for optimal networks, which is a novel way to efficiently search the space of possible networks given the possible parent sets.

Current algorithms typically do not accommodate both continuous and categorical nodes, and we were not able to find any that accommodate a survival outcome. We implement support for both continuous and categorical data, as well as continuous, binary and survival outcomes. This is especially useful for disease modeling where mixed data and survival outcomes are common. We also provide options for two common scoring functions, and allow for multiple best networks to be returned if there are ties.

Our main novel contribution in addition to providing software is the revision of the Silander algorithm 3 [6] to incorporate possible parent sets, and use of 'generational orderings' for a much more efficient way to explore the search space as compared to the original approach, which is based on lexicographical ordering. The proposed approach covers the entire constrained search space without searching through networks that don't conform to the parent set constraints.

Background

In this section, we briefly review Bayesian networks and the Bayesian network structure discovery problem (for more background on these topics see, for example, [8, 9]).

A Bayesian network (BN) is a probabilistic graphical model that consists of a labeled directed acyclic graph (DAG) in which the vertices $V = \{v_1, \ldots, v_p\}$ correspond to random variables and the edges represent conditional dependence of one random variable on another. Each vertex v_i is labeled with a conditional probability distribution $P(v_i|parents(v_i))$ that specifies the dependence of the variable v_i on its set of parents $parents(v_i)$ in the DAG. A BN can also be viewed as a factorized representation of the joint probability distribution over the random variables and as an encoding of conditional dependence and independence assumptions.

A Bayesian network G can be described as a vector $G = (G_1, \ldots, G_p)$ of parent sets: G_i is the subset of V from which there are directed edges to v_i. Any G that is a DAG corresponds to an ordering of the nodes, given by the ordered set $\{v_{\sigma_i}\}, i \in \{1, 2, \ldots, p\}$, where σ is a permutation of $[p]$ - the ordered set of first p natural numbers, with $\sigma(i) = \sigma_i$. A BN is said to consistent with an ordering $\{v_{\sigma_i}\}$ if parents of v_{σ_i} are a subset of $\{v_{\sigma_j}\}$, if $j < i$, i.e. $G_{\sigma_i} \subseteq \{v_{\sigma_j}\}, j < i$.

One of the main methods for BN structure learning from data uses a scoring function that assigns a real value to the quality of G given the data. For finding a best network structure, we maximize this score over the space of possible networks. Note that we can have multiple best networks with the same score. Scoring functions balance goodness of fit to the data with a penalty term for model complexity. Some commonly used scoring functions are BIC/MDL [10], BDeu [11], and BGe [12, 13]. We use BIC (Bayesian information criterion) and BGe (Bayesian Gaussian equivalent) scoring functions as two options for using Causnet. BIC is a log-likelihood (LL) score where the overfitting is avoided by using a penalty term for the number of parameters in the model, specifically $p\ln(n)$, where n is the sample size. The BGe score is the posterior probability of the model hypothesis that the true distribution of the set of variables is faithful to the DAG model, meaning that it satisfies all the conditional independencies encoded by the DAG, and is proportional to the marginal likelihood and the graphical prior [12, 13].

CausNet

CausNet uses the dynamic programming (DP) approach to finding a best Bayesian network structure for a given dataset. The idea of using dynamic programming for exact Bayesian network structure discovery was first proposed by Koivisto and Sood [14, 15]. Recent work using dynamic programming, includes that by Silander and Myllymäki in [6] and by Singh and Andrew in [7].

We closely follow the algorithm proposed by Silander and Myllymäki (SM algorithm henceforth) and make heuristic modifications focusing on finding sparse networks and disease pathways. This is achieved by dimensionality reduction with parent set identification, and by restricting the search to the space of 'generational' orderings rather than lexicographical orderings as in the original SM algorithm.

Finding a best Bayesian network structure is NP-hard [5]. The number of possible structures for n variables is $\mathcal{O}\left(n!2^{\binom{n}{2}}\right)$ [16], making exhaustive search impractical. So the dynamic programming algorithms for optimal BNs are feasible only for a small numbers of features, usually less than 30 [6, 7]. These approaches are optimal in the sense that they are guaranteed to find a network with the best score. The number of variables can be increased somewhat by using small in-degree (maximum number of parents for any node), in which case the approach is optimal conditional on the constraint. However, bounding the in-degree by a constant k does not help much, the lower bound for the number of possible graphs is still $n!2^{kn\log n}$ (for large enough n) [14]. We introduce parent set identification and 'generational' orderings based search to reduce the search space and thus scale up the SM algorithm to a substantially larger numbers of variables.

The SM algorithm uses the key fact about DAGs that every DAG has at least one sink, which is a node with no outgoing edges. The problem of finding a best Bayesian network given the data $\mathcal{D}$ starts with finding a best sink for the whole set of nodes. That node is removed and the process is then repeated recursively for the remaining set of nodes, which makes it a dynamic programming algorithm. The result is an ordering of the nodes $\{v_{\sigma_i}\}, i \in \{1, 2, .., p\}$, from which the DAG can be recovered. Denoting the best sink by s, and the score of a best network with nodes V by $bestscore(V)$, and the best score of s with parents in U by $bestScore(s, U)$, the recursion is given by the following relation:

$$bestScore(V) = bestscore(V \setminus \{s\}) + bestScore(s, V \setminus \{s\}), \tag{1}$$

where $V \setminus \{s\}$ denotes the set difference between the variable set V and the sink s.

To implement the above recursion, the idea of a local score for a node v_i with parents $parents(v_i)$ is used, which we get using a scoring function. The requirement for a score function $score(G)$ for a network G is that it should be decomposable, meaning that the total score of the network is the sum of scores for each node in the network, and the score of a node depends only on the node and its parents. Formally,

$$score(G) = \sum_{i=1}^{p} localscore(v_i, G_i). \tag{2}$$

where the local scoring function $localscore(x, y)$ gives the score of x with parents y in the network G. In a given set of possible parents pp_i for node v_i, we find the best set of parents bps_i which give the best local score for v_i, so that

$$bestScore(v_i, pp_i) = \max_{g \subseteq pp_i} localscore(v_i, g), \tag{3}$$

$$bps_i(pp_i) = \operatorname*{argmax}_{g \subseteq pp_i} localscore(v_i, g). \tag{4}$$

Now the best sink s can be found by Eq. 5, and the best score for a best network in V can be found by Eq. 6.

$$bestSink(V) = \operatorname*{argmax}_{s \in V} bestscore(V \setminus \{s\}) + bestScore(s, V \setminus \{s\}). \tag{5}$$

$$bestscore(V) = \max_{s \in V} bestscore(V \setminus \{s\}) + bestScore(s, V \setminus \{s\}). \tag{6}$$

In Fig. 1, the subset lattice shows all the paths that need to be searched to find a best network. Observe that each edge in the lattice encodes a sink, so that each path also encodes an ordering on the 4 variables, e.g. the rightmost path encodes the reverse-ordering $\{1, 2, 3, 4\}$. There are a total of 4! paths/orderings to be searched. Now suppose we knew the best score corresponding to each edge in all the paths, meaning the best score for the sink corresponding to that edge with the best parents from the subset at

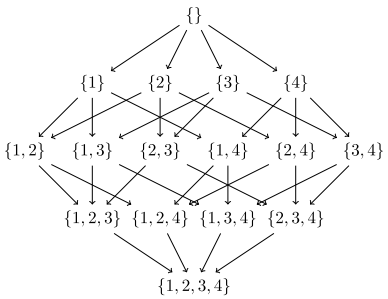

Fig. 1 Subset lattice on a network with four nodes $\{1, 2, 3, 4\}$

the source of that edge. Then naive depth/width first search would compare the sum of scores along all paths to get the best network. In the SM approach, we proceed from the top of the lattice. Ignoring the empty set, we start with finding the best sink for each singleton in the first row which trivially is the singleton itself. Next, we find the best sink for the subsets of cardinality 2 in the second row using the edge best scores. And we continue all the way down. Suppose we get the best sinks sublattice as in Fig. 2, then the best network is given by the only fully connected path, and corresponds to the reverse-ordering $\{4, 1, 2, 3\}$.

Now, using the SM algorithm, finding the best Bayesian network structure, also the basic CausNet approach without restricting the search space, has the following five steps:

1. Calculate the local scores for all $p2^{p-1}$ different (variable, variable set)-pairs.
2. Using the local scores, find best parents for all $p2^{p-1}$ (variable, possible parent set)-pairs.
3. Find the best sink for all 2^p variable sets.
4. Using the results from Step 3, find a best ordering of the variables.
5. Find a best network using results computed in Steps 2 and 4.

The extensions to this base version of CausNet—possible parent sets identification, phenotype driven search, and search space based on 'Generational orderings' reduce the search space.

Possible parent sets identification

The possible parent set pp_i for each node V_i, such that $pp_i \subseteq U_i \subseteq V \setminus \{V_i\}$, is determined in a preliminary step using marginal association. For this, we test for pairwise association between variables using Pearson's product moment correlation test. Either a specifiable p value α of the test is used as False Discovery Rate (FDR) cut-off to identify associations between pairs or we can use correlation value cutoffs. The choice of a target FDR level can also be made in a post hoc fashion and can be determined by such factors as strength of evidence for observed associations, investigators tolerance for complexity versus need for interpretable results, and cost of follow-up

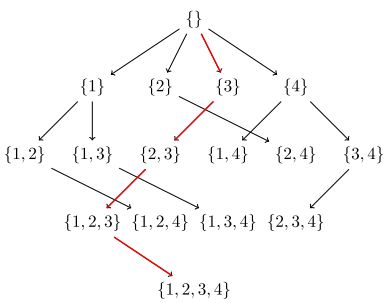

Fig. 2 An example Best sinks sublattice of four nodes $\{1, 2, 3, 4\}$. The arrows encode the best sink for each subset. The red arrows indicate the best network given by the only fully connected path, and corresponds to the reverse-ordering $\{4, 1, 2, 3\}$

studies [17]. Decreasing the FDR or increasing the correlation cutoff reduces the number of nodes to be considered, thus leading to sparser networks.

This gives a possible parent set pp_i for each node V_i. If the response is a survival outcome, we evaluate associations using Cox proportional hazards regression, independently for each feature. Thus, we make the practical approximation that if there is no detectable marginal association, then the feature is unlikely to have a causal input that is meaningful or at least detectable in the data. Therefore, the feature without evidence of marginal association need not be included as a possible parent.

Phenotype driven search

In biomedical applications, one is often interested in a small set of predictors that affect a phenotype of interest. Diffusion-based prioritization of genes as risk predictors leading to an outcome has been shown to identify a subnetwork of interest [18, 19]. A one-hop or k-hop approach is shown to find such networks [18]. We apply a similar approach in our algorithm with a 'phenotype driven search'. In this approach, we consider only two or three levels of associations (similar to 1-hop and 2-hop respectively in [18]) starting with the outcome variable, i.e. we identify the parents, 'grandparents' and 'great-grandparents' of the phenotype outcome.

Identifying variables with evidence of association as defined by the threshold gives us the "feasible set" (*feasSet*) of nodes. The original data is then reduced to the *feasSetData*, which includes only the *feasSet* variables. This is implemented as in Algorithm 1. After Algorithm 1, the dimension of data is reduced to $\bar{p}$, $\bar{p} < p$, where $\bar{p}$ is the number of nodes in the *feasSet*.

Algorithm 1 Find pp, Compute feasSet and feasSetData

Input : Data, α, phenotypeBased, pp
1: **if** pp Not NULL **then**
2: pp = pp
3: **else**
4: **if** phenotypeBased =True **then**
5: find pp for the phenotype output
6: find pp for the phenotype's pp
7: **else**
8: find pp for all variables
9: **end if**
10: **end if**
11: find possible offsprings (po) for all variables
12: find the feasible set of variables $feasSet$
13: get the reduced dimensional data $feasSetData$
Output : pp, po, feasSet and feasSetData

Dynamic programming on the space of 'Generational orderings' of nodes

After the possible parent sets are identified, the next step is to compute local scores for *feasSet* nodes as shown in Algorithm 2. Computing local scores has computational complexity $\mathcal{O}\left(\bar{p}2^{\bar{p}-1}\right)$ if there was no possible parent sets identified for nodes, but reduces to $\mathcal{O}\left(\bar{p}r^{d}\right)$, where r is the maximum cardinality of possible parent sets of all nodes and d is the in-degree. Because of bounded indegree, the step at line 3 in this algorithm is truncated at cardinality indegree.

Algorithm 2 Compute local scores for feasSet nodes

Input : feasSetData, pp

1: **for** v_i in $feasSet$ **do**
2: find all parent subsets $\{pps_j\}$ of possible parents set pp_i of the node v_i
3: compute local score for node v_i with parents pps_j
4: **end for**

Output : pps, ppss

Algorithm 3 Compute best scores and best parents for feasSet nodes in all parent subsets

Input : feasSetData, pps, ppss, indegree

1: **for** v_i in $feasSet$ **do**
2: **for** p_{ij} in pps_i **do**
3: find all subsets $\{ppsSub_{ijk}\}$ upto cardinality indegree
4: Compute best scores $bpss_ij$ and best parents bps_{ij} of v_i for parent subset pps_{ij} from among $\{ppsSub_{ijk}\}$ using local scores in $ppss$
5: **end for**
6: **end for**

Output : pps, ppss, bps, bpss

The next step is to compute the best scores and best parents for feasSet nodes in all possible parent subsets as shown in Algorithm 3. This uses local scores already calculated in Algorithm 2. This has computational complexity $\mathcal{O}\left(\bar{p}r2^{r-1}\right)$, where r is the maximum cardinality of possible parent sets of all nodes.

Then we compute best sinks for all possible subsets of feasSet nodes restricted by the parent set constraints. Now compared with the SM algorithm that explores all orderings, we explore only the space of what we call *'generational orderings'* to find the best BNs. Recall from the Background section above that a DAG corresponds to an ordering of the nodes $\{v_{\sigma_i}\}$ with the constraints that parents of a node v_{σ_i} in the ordering are a subset of $\{v_{\sigma_j}\}$, with $j < i$, i.e. each node can have parents only from the set of nodes above itself in the ordering. To include the parent set constraints, we search only the orderings that are consistent with the possible parent sets for each node. The resulting subset of orderings is the space of *'generational orderings'*.

Definition 3.1 A generational ordering is an ordering such that each variable in the ordering has at least one parent from the set of possible parents in the set of variables preceding it in the ordering, consistent with the possible parent sets.

Starting with $\bar{p}$ networks of single nodes, we add one offspring at a time, and iterate over all nodes in the set to find a best sink in a subset of cardinality increasing from 1 to $\bar{p}$, as in algorithm 4. Observe that while finding best sink for a subset of cardinalty k at level k, we already have the best networks of cardinalty $k-1$ at level $k-1$, which is the key feature of the DP approach. Once we have these best sinks, we compute the reverse ordering (possibly multiple orderings) of $\bar{p}$ nodes and compute best network as shown in algorithm 5.

Algorithm 4 Compute best sinks for feasSet subsets

Input : feasSetData, po, pps, ppss, bps

1: Start with $\bar{p}$ networks of single nodes, where $\bar{p}$ is the number of nodes in $feasSet$, with their NULL scores
2: Add one offspring at a time, and iterate over all nodes in the set to find a best sink using information about best score/parent set from algorithm 3 ; end when number of nodes in the iteration is $\bar{p}$
3: **return** list of all possible $2^{\bar{p}}$ subsets of $feasSet$ with best sink and best network score for that subset of nodes

Output : bsinks

Algorithm 5 Find the best networks

Input : bsinks

1: Find the reverse ordering (possibly multiple orderings) of $\bar{p}$ nodes using the list from algorithm 4
2: Compute best network(s) using the reverse ordering(s) of $\bar{p}$ nodes and using the best parent set for each node in a possible parent set found in algorithm 3

Output : bestNetwork(s)

Theorem 3.1 *Without parent set and in-degree restrictions, the Causnet -the generational ordering based DP algorithm—explores all the* $\mathcal{O}\left(p!2^{\binom{p}{2}}\right)$ *network structures for p nodes.*

Proof

Without parent set restrictions, every node is s possible parent of every other node. In Fig. 1, let $k, 0 \leq k \leq p$ *be the cardinality of subsets in the subset lattice for p nodes. Let each row in the subset lattice be the kth level in the lattice. Now adding a new element in Algorithm 4 corresponds to an edge between a subset of cardinality k and* $k-1$*, which considers the added element as a sink in the subset of cardinality k. Number of edges to a subset of cardinality k from subsets of cardinality* $k-1$ *is given by* $\binom{k}{k-1}$*. The number of possible parent combinations for a sink in subset of cardinality k, without in-degree restrictions, is given by* 2^{k-1}*. In Algorithm 3, we explore all these possible parent sets to find the best parents for each sink s in each subset at each level k. The Algorithm 4 uses this information to get the best sink(possibly multiple) for each subset at level k. The total number of networks thus searched by the Causnet algorithm is given by -*

$$\prod_{k=1}^{p}\left(\frac{k}{k-1}\right)2^{k-1} = \binom{1}{0}\binom{2}{1}\cdots\binom{p-1}{p-2}\binom{p}{p-1}2^0 2^1 \ldots 2^{p-1} = p!2^{\binom{p}{2}}$$

The number of network structures is $\mathcal{O}\left(p!2^{\binom{p}{2}}\right)$ because there are many repeated structures in this combinatorial computation; e.g. there are $p!$ structures with all p nodes disconnected. □

Now suppose the possible parent sets for the four nodes $\{1,2,3,4\}$are as follows: $pp_1 = \{2,4\}$, $pp_2 = \{3,1\}$, $pp_3 = \{2\}$,$pp_4 = \{1\}$. Factoring in these possible parent sets, the subset lattices corresponding to those in Figs. 1 and 2 reduce to those in Fig. 3. Here we are showing the lattices with parent set constraints, so that some arrows are omitted e.g. in the top lattice, there is no arrow from $\{4\}$ to $\{3,4\}$ because node 3 does not have

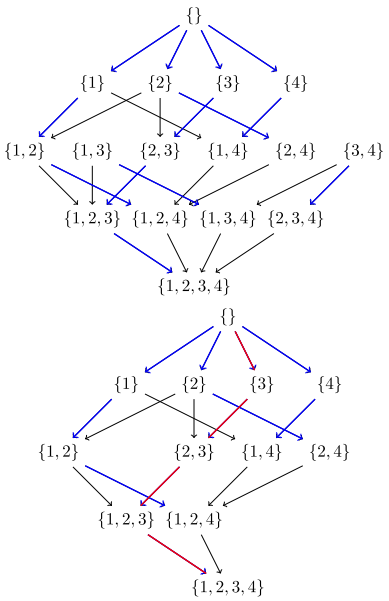

Fig. 3 Subset lattices with parent set constraints. The top subset lattice is the lattice with parent set restrictions. The bottom lattice, obtained by Causnet, retains only the subsets that are in the complete generational orderings. The red arrows which at base are blue as well, represent the best network

node 4 as a possible parent in our example. The bottom lattice retains only the subsets that are in the complete path from the top of the lattice to its bottom, and discards the remaining paths. These full paths represent what we call *complete generational orderings*. This is how generational ordering of Causnet ensures maximum connectivity among the reduced set of $\hat{p}$ nodes in the *feasSet*. The red arrows which at base are blue as well, represent the best network.

Definition 3.2 A generational ordering is a complete generational ordering if it has all the variables in the *feasSet* in the ordering.

Example 3.1

In the top subset lattice in Fig. 3, the two paths from subset $\{3, 4\}$ *downwards can be seen as two generational orderings missing a generational order relation between nodes 3 and 4. Let's denote this ordering as* $\{2, 1\}$*, which is not a complete generational ordering.*

Lemma 3.1 *With parent set restrictions, the Causnet—the generational ordering based DP algorithm—searches the whole space of complete generational orderings.*

Proof

To see this, suppose not. Then there is a complete generational ordering that is not searched. But the algorithm 4 adds a variable at level k that is a possible offspring of the preceding subset at level $k-1$ *at each step starting with the empty set. So, this missed ordering must have a variable at level k in the ordering that has no possible parent in the set of variables before it at level* $k-1$*. That makes it an incomplete generational ordering, which is a contradiction.* □

Theorem 3.2 *With parent set restrictions, the Causnet algorithm explores all the network structures consistent with possible parent sets for p nodes.*

Proof

Combining lemma 3.1, and theorem 3.1, it's straightforward to see that Causnet discards only the networks that are either inconsistent with parent set restrictions or in an incomplete generational ordering. In each complete generational ordering, it searches all $2^{\hat{k}}$ *possible parent combinations for each sink at level k, where* $\hat{k} \leq k-1$ *because of parent set restrictions.* □

Simulations

We compare CausNet with three other methods that have been widely used for optimal Bayesian network identification to infer disease pathways from multiscale genomics data. The first method is Bartlett and Cussens' GOBNILP [20], an integer learning based method that's considered state-of-art exact method for finding optimal Bayesian network. The other two methods are BNlearn's Hill Climbing (HC) and Max-min Hill Climbing (MMHC) [21, 22], which are both widely used approximate methods, see e.g. [23, 24]. Hill-Climbing (HC) is a score-based algorithm that uses greedy search on the space of the directed graphs [25]. Max-Min Hill-Climbing (MMHC) is a hybrid algorithm [26] that first learns the undirected skeleton of a graph using a constraint-based algorithm called Max-Min Parents and Children (MMPC); this is followed by the application of a score-based search to orient the edges.

We simulated Bayesian networks by generating an N x p data matrix of continuous Gaussian data. The dependencies are simulated using linear regression with the option to control effect sizes. Some number of the p nodes were designated as sources (p_1), some intermediate (p_2), and some sinks (p_3), the remainder (p_0) being completely independent. The actual DAGs of the $p_1 + p_2 + p_3$ nodes vary across replicates. The requirement for being a DAG is implemented using the idea of ordering of vertices. We pick a random ordering of a randomly chosen subset of p vertices. Then enforcing each vertex to have parents only from the set of vertices above itself in the ordering guarantees a DAG.

The False Discovery Rate (FDR) and Hamming Distance are used as the metrics to compare the methods. With *FP* defined as the number of false positives and *TP* defined as the number of true positives, FDR is defined as:

$$FDR = \frac{FP}{FP + TP}.$$

Controlling for the false discovery rate (FDR) is a way to identify as many significant features (edges in case of BNs) as possible while incurring a relatively low proportion of false positives. This is especially useful metric for high dimensional data and for network analysis ([27]).

The Hamming distance between two labeled graphs G_1 and G_2 is given by $|\{((e \in E(G_1)\&(e \notin E(G_2)))or((e \notin E(G_1)\&e \in E(G_2)))\}|$, where $E(G_i)$ is the edge set of graph G_i. Simply put, this is the number of addition/deletion operations required to turn the edge set of G_1 into that of G_2. The Hamming distance is a measure of

structural similarity, and forms a metric on the space of graphs (simple or directed), and gives a good measure of goodness of a predicted graph ([28, 29]). In the context of predicted and the truth graph, with *FP* defined as the number of false positives and *FN* as the number of false negatives, Hamming Distance is defined as:

$$HammingDistance = FP + FN.$$

As the first set of simulations using parent set identification, we ran simulations using multiple replicates of networks, the first with $p = 10, 20, 40, 50, 60, 100$, and $N = 500, 1000, 2000$. Figure 4 shows the plot of average FDR for different values of p and N, and their linear trend with BIC scoring for CausNet. For using CausNet, We have used FDR cutoff of 0.3 and an in-degree of 2 for all the experiments.

We can see that for lower values of p, Gobnilp has the lowest values of FDR with CausNet the second best, but CausNet performs the best for higher values of p. The results for the BGe scoring for CausNet are similar qualitatively. In the Tables 1 and 2, we show the average *FDR* across the 9 combinations of N and p, both with and without taking directionality into account. The results for CausNet with the choice of scoring function—either BGE or BIC—are given. The results show that our method performs very well compared with the methods considered.

For the number of variables up to 40 (Table 1), on the metric of FDR, CausNet performs second best after Gobnilp for directed graphs and the best for undirected graphs. This is true for both scoring methods—BIC and BGE. For the number of variables between 50 and 100 (Table 2), our method performs the best for both directed and undirected graphs, using either of the two scoring methods, BIC and BGE.

Figure 5 shows the plot of average Hamming Distance for different values of p and N, and their linear trend with BIC scoring for CausNet. We can see that for lower values of p, Gobnilp has the lowest values of Hamming Distance with CausNet the second best for most part, but CausNet performs the best for higher values of p. The results for the BGe scoring for CausNet are the same qualitatively.

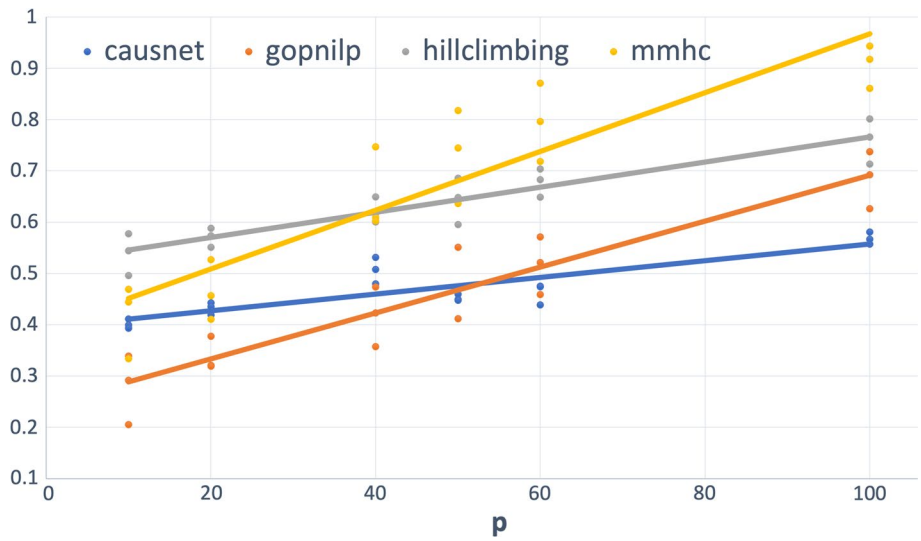

Fig. 4 Average FDR—upto 100 variables. $p = 10, 20, 40, 50, 60, 80, 100$ and $N = 500, 1000, 2000$

Table 1 FDR ($p = 10, 20, 40, N = 500, 1000, 2000$)

Method	FDR(undirected)	FDR(directed)
CausNet (BIC)	0.229	0.412
CausNet (BGE)	**0.223**	0.404
Gobnilp	0.312	**0.345**
BN-HC	0.466	0.577
BN-MMHC	0.368	0.511

Table 2 FDR ($p = 50, 60, 100, N = 500, 1000, 2000$)

Method	FDR(undirected)	FDR(directed)
CausNet (BIC)	0.359	0.494
CausNet (BGE)	**0.328**	**0.466**
Gobnilp	0.540	0.560
BN-HC	0.635	0.694
BN-MMHC	0.787	0.812

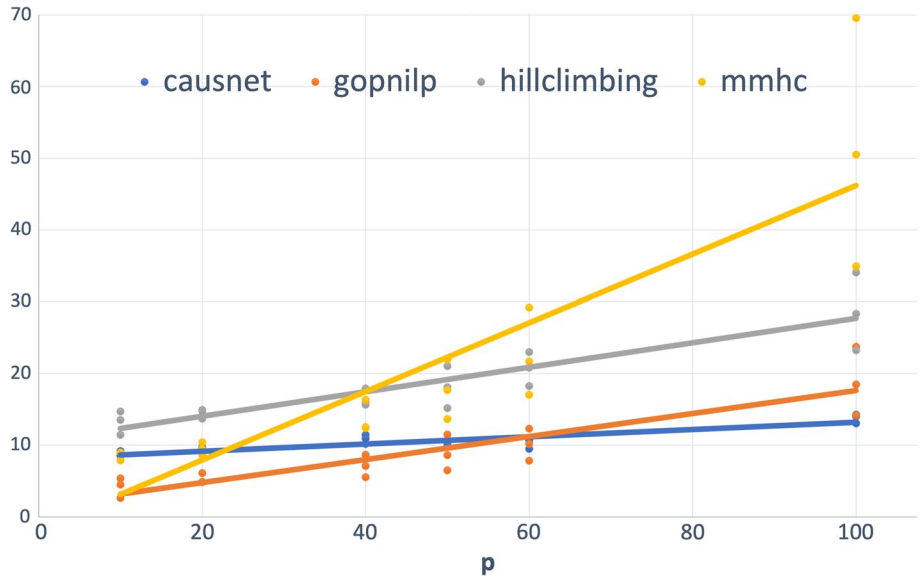

Fig. 5 Average Hamming Distance—upto 100 variables. $p = 10, 20, 40, 50, 60, 80, 100$ and $N = 500, 1000, 2000$

Phenotype based search

For phenotype based parent set identification, we find 3 levels of possible parents of the outcome variable. For these simulations, we use $p = 10, 20, 40, 50, 60, 80, 100$ and $N = 500, 1000, 2000$. The results are shown in Figs. 6 and 7.

The FDR for phenotype-driven parent sets is the best for number of variables greater than 10; Gobnilp is the second best. In terms of Hamming distance too, Causnet again is the best for variables more than 20; Gobnilp and MMHC have lower

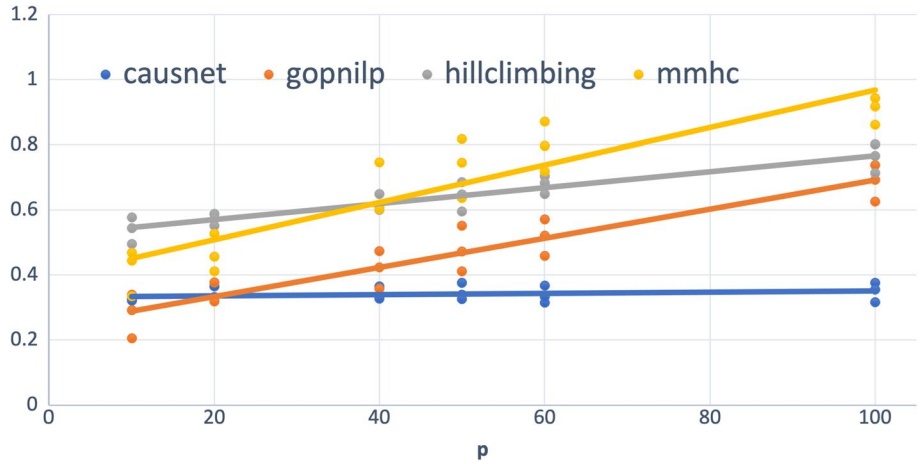

Fig. 6 Average FDR - Phenotype based search. $p = 10, 20, 40, 50, 60, 80, 100$ and $N = 500, 1000, 2000$

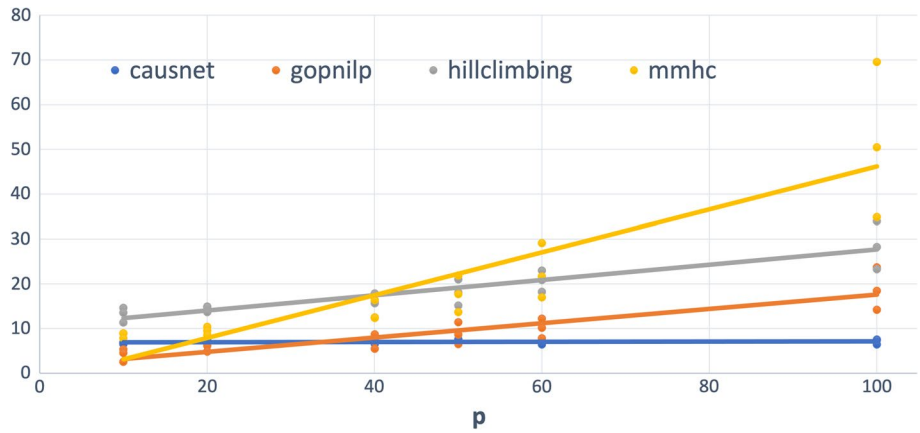

Fig. 7 Average Hamming Distance - Phenotype based search. $p = 10, 20, 40, 50, 60, 80, 100$ and $N = 500, 1000, 2000$

Hamming distance$p$ than that of Causnet for variables less than 20, but rise quickly for variables more than 20; overall, Gobnilp is again the second best.

Number of variables up to 1000

For these simulations, we use $p = 200, 500, 1000$, and $N = 500, 1000, 2000$. For these simulations, we don't compare with the other three algorithms as they either can not handle such high number of variables or take many orders of magnitude longer. The results are shown in Figs. 8 and 9. Observe that both the FDR and Hamming Distance values are better than what we had with other methods when the number of variables was less than 100.

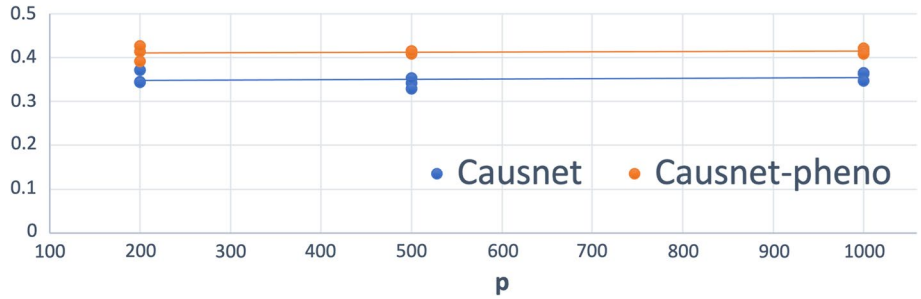

Fig. 8 Average FDR—upto 1000 variables. $p = 200, 500, 1000$ and $N = 500, 1000, 2000$

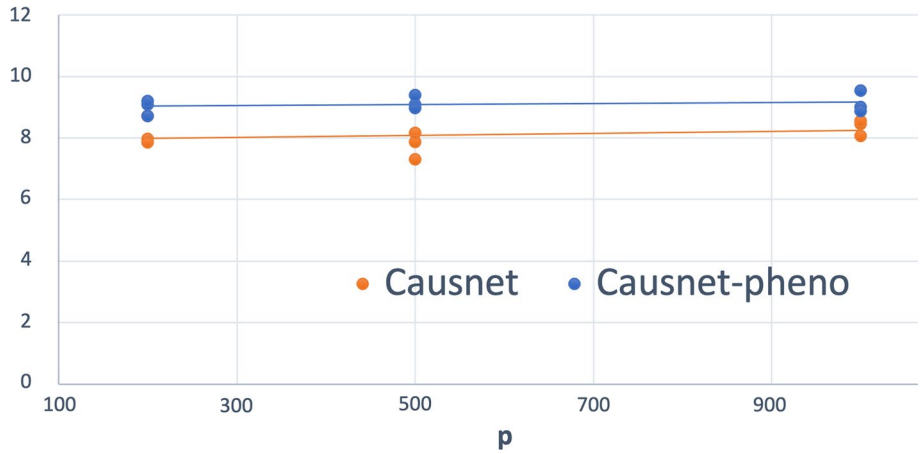

Fig. 9 Average Hamming Distance—upto 1000 variables. $p = 200, 500, 1000$ and $N = 500, 1000, 2000$

Table 3 Average Runtime in seconds

	$p \leq 100$	$100 < p \leq 1000$
CausNet	0.025	277.87
Gobnilp	99.8	*
HC	0.088	68.17
MMHC	0.288	18763.493

* Terminates without output

Runtime

Having confirmed the performance of CausNet as superior to other algorithms, especially for the number of variables greater than 40, we compare the runtimes of the four algorithms. Here we split the comparison into two categories - number of variables less than 100 and greater than 100, i.e $p \leq 100$ and $100 < p \leq 1000$. This is done as two of the algorithms - MMHC and Gobnilp - either can not handle more than 100 variables or the computation time is many orders of magnitude greater than that taken by CausNet. Specifically, MMHC takes over an average of 300 min, and Gobnilp terminates without producing any output network in most cases. Running all the simulations for these two algorithms would require an inordinate amount of time and was not considered

worth the effort. The average runtimes in seconds are summarized in Table 3. As we can clearly see, Causnet has the best runtimes for $p \leq 100$ and Gobnilp the worst. For $100 < p \leq 1000$, CausNet is better than Gobnilp and MMHC. Although HC is faster, sections above demonstrated it to be inferior to CausNet in terms of performance metrics of FDR and Hamming distance.

Application to clinical trial data

We applied our method to recently published data involving gene expression and ovarian cancer prognosis among participants in multiple clinical trials [30]. The aim of this study was to develop a prognostic signature based on gene expression for overall survival (OS) in patients with high-grade serous ovarian cancer (HGSOC). Expression of 513 genes, selected from a meta-analysis of 1455 tumors and other candidates, was measured using NanoString technology from formalin-fixed paraffin-embedded tumor tissue collected from 3769 women with HGSOC from multiple studies. Elastic net regularization for survival analysis was applied to develop a prognostic model for 5-year OS, trained on 2702 tumors from 15 studies and evaluated on an independent set of 1067 tumours from six studies. Results of this study showed that expression levels of 276 genes were associated with OS (false discovery rate < 0.05) in covariate-adjusted single-gene analyses.

We applied our method CausNet to this gene expression dataset of 513 genes and survival outcome. For dimensionality reduction and parent set identification, we used a three-level phenotype driven search. For the disease node 'Status', dead or alive at censoring or end of follow-up, we carried out Cox proportional hazard regression for each gene separately, adjusted for age, stage, and stratified site. Then we computed analysis of variance tables for the fitted models and created a list of p values based on the χ^2 distribution. FDR was then computed using the Benjamini & Hochberg (BH) method. We choose the 5 genes, *ZFHX4*, *TIMP3*, *COL5A2*, *FBN1*, and *COL3A1*, with the most significant p-values as possible parents of the disease node. At the next level, we used correlations between these and rest of the genes to identify possible parent sets. Parent sets were also identified for possible grand-parents of the disease node. These three levels of ancestors of the disease node resulted in 16 genes with non-null parent sets. This process substantially reduced the dimensionality of the dataset. The sets of possible parents were then used by CausNet with the BIC score to find a best network. Using the in-degree of 2, we identified a best network as shown in Fig. 10. On a personal computer with a 2.3 GHz Intel Core i9 processor with 16 GB RAM, this processing took about

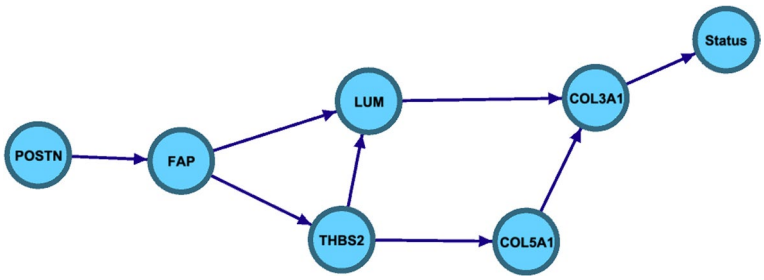

Fig. 10 Ovarian cancer network

3.4 min. As no other methods in our study or otherwise work for survival outcome, our method is the only method able to handle such large data, and survival outcome in such short runtime to produce an optimal BN.

Discussion

We implemented a dynamic programming based optimal Bayesian network (BN) structure discovery algorithm with parent set identification with 'generational orderings' based search for optimal networks, which is a novel way to efficiently search the space of possible networks given the possible parent set.

Our main novel contribution aside from providing software is the revision of the SM Algorithm 3 [6] to incorporate possible parent sets and 'generational orderings' based search for a more efficient way to explore the search space as compared to the original approach based on lexicographical ordering. In doing so, we cover the entire constrained search space without searching through networks that don't conform to the parent set constraints. While the basic algorithm can be applied to any dataset from any domain, the phenotype based algorithm is particularly suitable for disease outcome modeling.

The simulation results show that our algorithm performs very well when compared with three state-of-art algorithms that are widely used currently. The parent set constraints reduce both the search space and the runtime significantly, while delivering better results, especially for greater than 60 variables.

The application to the recently published ovarian cancer gene-expression data with survival outcomes showed the algorithm's usefulness in disease modeling. It yielded a sparse network of 6 nodes leading to the disease outcome from gene expression data of 513 genes in just 3.4 min on a personal computer with a 2.3 GHz Intel Core i9 processor with 16 GB RAM. This disease pathway may generate guiding insights and hypotheses for further biomedical studies.

Important features of the algorithm include specifiable parameters—correlation, FDR cutoffs, and in-degree—which can be tuned according to the application domain. Choice of two scoring options, BIC and Bge, and implementation of survival outcomes and mixed data types makes our algorithm suitable for identifying disease pathways from a broad range of biomedical data types, e.g. GWAS and other omics.

Acknowledgements
The authors would like to thank Professor Duncan C. Thomas for helpful inputs during the development of this work. The authors would also like to thank two anonymous reviewers for their helpful comments that helped improve the exposition.

Author Contributions
N.S. and J.M. conceived of the method and wrote the manuscript; J.M. and N.S. developed the software; N.S. conducted the analyses and prepared the figures and tables. All authors read and approved the final manuscript.

Funding
This research was supported by the National Cancer Institute [P01CA196569 to N.S. and J.M.]; the National Institute of Child Health and Human Development [1R01HD098161 to J.M.]; and the National Institute on Aging [P01AG055367 to J.M.].

Availability of data and materials
The CausNet software package in R is available at https://github.com/nand1155/CausNet.

Declarations

Ethics approval and consent to participate
Not applicable

Consent for publication
Not applicable

Competing interests
The authors have no competing interests as defined by BMC, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Received: 18 July 2022 Accepted: 24 January 2023
Published online: 14 February 2023

References

1. Friedman N, Linial M, Nachman I, Pe'er D. Using Bayesian networks to analyze expression data. In: Proceedings of the fourth annual international conference on computational molecular biology. RECOMB '00. New York: Association for Computing Machinery; 2000. pp. 127–135 https://doi.org/10.1145/332306.332355.
2. Bielza C, Larrañaga P. Bayesian networks in neuroscience: a survey. Front Comput Neurosci. 2014;8:131. https://doi.org/10.3389/fncom.2014.00131.
3. Agrahari R, Foroushani A, Docking TR, Chang L, Duns G, Hudoba M, Karsan A, Zare H. Applications of Bayesian network models in predicting types of hematological malignancies. Sci Rep. 2018;8(1):6951. https://doi.org/10.1038/s41598-018-24758-5.
4. Su C, Andrew A, Karagas MR, Borsuk ME. Using Bayesian networks to discover relations between genes, environment, and disease. BioData Min. 2013;6(1):6. https://doi.org/10.1186/1756-0381-6-6.
5. Chickering DM, Heckerman D, Meek C. Large-sample learning of Bayesian networks is NP-hard. J Mach Learn Res. 2004;5:1287–330.
6. Silander T, Myllymäki P. A simple approach for finding the globally optimal Bayesian network structure. In: Proceedings of the twenty-second conference on uncertainty in artificial intelligence. UAI'06. Arlington: AUAI Press; 2006. pp. 445–452.
7. Singh AP, Moore AW. Finding optimal Bayesian networks by dynamic programming. 2018. https://doi.org/10.1184/R1/6605669.v1.
8. Darwiche A. Modeling and reasoning with Bayesian networks. Cambridge: Cambridge University Press; 2009. https://doi.org/10.1017/CBO9780511811357.
9. Korb KB, Nicholson AE. Bayesian artificial intelligence. Chapman & Hall/CRC: Boca Raton; 2004. p. 364.
10. Schwarz G. Estimating the dimension of a model. Ann Statist. 1978;6(2):461–4. https://doi.org/10.1214/aos/1176344136.
11. Carvalho AM. Scoring functions for learning Bayesian networks. 2009.
12. Geiger D, Heckerman D. Learning gaussian networks. In: Proceedings of the tenth international conference on uncertainty in artificial intelligence. UAI'94. San Francisco: Morgan Kaufmann Publishers Inc.; 1994. pp. 235–243.
13. Kuipers J, Moffa G, Heckerman D. Addendum on the scoring of gaussian directed acyclic graphical models. Ann Statist. 2014;42(4):1689–91. https://doi.org/10.1214/14-AOS1217.
14. Koivisto M, Sood K. Exact Bayesian structure discovery in Bayesian networks. J Mach Learn Res. 2004;5:549–73.
15. Koivisto M. Advances in exact Bayesian structure discovery in Bayesian networks. In: Proceedings of the twenty-second conference on uncertainty in artificial intelligence. UAI'06. Arlington: AUAI Press; 2006. pp. 241–248.
16. Robinson R. Counting labeled acyclic digraphs. In: Harary F, editor. New directions in the theory of graphs. New York: Academic Press; 1973. p. 239–73.
17. Millstein J, Battaglin F, Arai H, Zhang W, Jayachandran P, Soni S, Parikh AR, Mancao C, Lenz HJ. fdrci: FDR confidence interval selection and adjustment for large-scale hypothesis testing. Bioinform Adv. 2022;2(1):vbac047. https://doi.org/10.1093/bioadv/vbac047.
18. Mosca E, Bersanelli M, Gnocchi M, Moscatelli M, Castellani G, Milanesi L, Mezzelani A. Network diffusion-based prioritization of autism risk genes identifies significantly connected gene modules. Front Genet. 2017;8:129. https://doi.org/10.3389/fgene.2017.00129.
19. Bersanelli M, Mosca E, Remondini D. Network diffusion-based analysis of high-throughput data for the detection of differentially enriched modules. Sci Rep. 2016. https://doi.org/10.1038/srep34841.
20. Bartlett M, Cussens J. Integer linear programming for the Bayesian network structure learning problem. Artif Intell. 2015. https://doi.org/10.1016/j.artint.2015.03.003.
21. Tsamardinos I, Brown LE, Aliferis CF. The max-min hill-climbing Bayesian network structure learning algorithm. Mach Learn. 2006;65(1):31–78.
22. Scutari M. Learning Bayesian networks with the bnlearn R package. J Stat Softw. 2010;35(3):1–22.
23. Ainsworth HF, et al. A comparison of methods for inferring causal relationships between genotype and phenotype using additional biological measurements. Genet Epidemiol. 2017;41(7):577–86. https://doi.org/10.1002/gepi.22061.
24. Scutari M. Learning Bayesian networks with the bnlearn R package. J Stat Softw. 2010;35(3):1–22.
25. Margaritis D. Learning Bayesian network model structure from data, phd thesis. pittsburgh: Carnegie-Mellon university, school of computer science. 2003.
26. Tsamardinos I, Brown LE, Aliferis CF. The max-min hill-climbing Bayesian network structure learning algorithm. Mach Learn. 2006;65(1):31–78. https://doi.org/10.1007/s10994-006-6889-7.
27. Bhattacharjee MC, Dhar SK, Subramanian S. Recent advances in biostatistics: false discovery rates, survival analysis, and related topics. 2011.
28. Butts C, Carley K. Some simple algorithms for structural comparison. Comput Math Organ Theory. 2005;11:291–305. https://doi.org/10.1007/s10588-005-5586-6.

29. Hamming RW. Error detecting and error correcting codes. Bell Syst Tech J. 1950;29(2):147–60. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x.
30. Millstein J, Budden T, Goode EL, et al. Prognostic gene expression signature for high-grade serous ovarian cancer. Ann Oncol. 2020;31(9):1240–50. https://doi.org/10.1016/j.annonc.2020.05.019.

Publisher's Note