Learning Bayesian Networks from Data
Weapons in schools problem

Bayesian Networks - Second Assignment

Alejandro González Rogel (s4805550)
Athena Iakovidi (s4694368)
Juraj Šušnjara (s4846559)

12 December 2016

1 Introduction

This second assignment is focused on applying structure learning algorithms to a real-world problem. In our case, this problem is the relation between teenagers and weapon carrying in school. In order to derive a network from data for this problem, we test the results of two different structure learning algorithms based on different approaches. One way to infer the structure is using a constraint-based learning. It performs independence tests between variables and builds a Bayesian network according to the relations found. The second approach is called score-based approach, and focuses on defining a global measure (score system) able to compare two different network models and, based on that comparison, search in the space of all possible combinations until it finds the most suitable.

1.1 The Semi-Interleaved HITON-PC (SI-HITON-PC) algorithm

This algorithm is based on the constraint-based paradigm. The specific version of the algorithm we will use for this practice was proposed by [1], even though the HITON algorithm had already been presented years ago [2].

The main reason why we chose this algorithm, is that we wanted to experiment with an algorithm that outputs undirected edges (this method does not output a directed graph but a skeleton of a network). It has been tested mostly on biomedical data [2] [3], but we expect that the problem we proposed (defined again in section 1.3) is not that different as far as defining the structure of the network is concerned.

Another important characteristic of the HITON-PC algorithm is that it is highly scalable according to the original paper (alike MMHC that we will be explained in section 1.2). Although in our network we only use eleven nodes, because these are the data we have been working with since the beginning, we are aware that modeling a problem like this could and should require many other nodes and variables. Thus, it makes sense to choose for an algorithm that could be used if our problem was trying to be solved for "the real world".

It is important to note that the HITON-PC algorithm outputs just a skeleton, which makes it easy to combine it with some other, maybe more complex or computationally expensive, algorithm. The output of HITON-PC would then be the input for this new algorithm, saving time and resources.

The behaviour of this algorithm is based on Markov blankets. Let us remember that a Markov blanket of a node T is a set of nodes composed by T's parents, children and other parents of the children. Pseudo-code in 1 tries to explain its behaviour. There is one thing that requires an explanation. When calculating which variables could be related, we used what it is called an association function. This function will perform a conditional independence test between the variables using the available data to infer their relation. In the original paper they used the p-value of the G_2 test as this function, however, it was not possible to perform this test using the actual library and we will use some others that will be specified when showing the results.

The factor that has the greatest influence on this algorithm is the algorithm used to test the conditional independence between two variables. We will see bellow that this value can be changed. We would also try to modify the precision by which two variables are considered to be related, to see if we end up with very different graphs.

Algorithm 1 Semi-Interleaved HITON-PC algorithm

```
1:  Input: A dataset D, a set of variables V
2:  Output: A skeleton for the dataset D
3:  for each T in V do
4:      $RELATED_V AR(T) = \{\}$
5:      $OPEN = V - T$
6:      $SORT\_LIST$ =Sort in descending order all other variables according to their association with T
7:      Remove any variable with Zero association with T
8:      while OPEN is not empty do
9:          $X = SORT\_LIST(0)$
10:         $RELATED\_VAR(T).append(X)$
11:         $SORT\_LIST.remove(X)$
12:         $OPEN.remove(X)$
13:         if There is a set Z in RELATED_VAR(T) such as {T, X—Z} then
14:             $RELATED_V AR(T).remove(Z)$
```

1.2 The Max-Min Hill-Climbing (MMHC) algorithm

The second algorithm we chose is the Max-Min Hill-Climbing (MMHC) one [4]. This algorithm was proposed in 2006; at that time, it outperformed all the most promising structure learning algorithms, while being much more efficient runtime-wise. It was only outperformed by the other algorithms, that was compared with, for very specific sets of parameters and the difference in performance was minute, while the MMHC algorithm still had a great advantage when it came to speed of execution. We chose this algorithm for its efficiency and also its exceptional performance. Albeit this algorithm does sacrifice a small amount of performance for scalability (use on Bayesian networks with many variables) and this is not a requirement for our project (as we don't use many variables), this algorithm is often used as a reference on (or comparison with) many algorithms. Hence, it seems to be a landmark in Bayesian strucure learning and we, as students, thought it was wise to familiarize ourselves with this algorithm for educational purposes.

It is important to highlight that the algorithm combines ideas from local learning, constraint-based, and search-and-score techniques. The first step of the algorithm is to learn the skeleton (edges without direction) of a Bayesian network. It uses the Max-Min Parents and Children (MMPC) local discovery algorithm [5] to do that. Then, it determines the directions of the edges by applying a greedy Bayesian-scoring hill-climbing search.

The first step effectively improves the performance of the algorithm (compared to, e.g., the greedy search algorithm), because it allows for the algorithm to identify potential parents of each node, instead of examining the whole search space (all nodes in the network). Parents of a node can be only nodes that are connected with the former node (with an undirected edge). An inspiration for this algorithm was the Sparse Candidate (SC) algorithm, which also identifies potential parents. However, the SC does that in a different manner that causes several problems which often result in non-sensical networks (see [6] for more details and [4] for a comparison).

Once we have a skeleton, we try to orientate the edges. In order to perform the hill-climbing (second step on this algorithm), the algorithm uses a method for escaping local, sub-optimal maxima (such as random restarts, simulated annealing, or a TABU list). In our project, we use the random restarts hill-climbing method. The algorithm explores new structures by adding, removing edges, or reversing the direction of existing edges. It also scores each explored structure and saves the score of the best one.

Generally, the MMHC algorithm was designed so that it doesn't require finetuning of parameters. Having to choose the correct values for parameters was considered a downside for previously proposed algorithms and the authors' intention was to create an algorithm that doesn't have this requirement. However, implementations of the algorithm can choose which hill-climbing method to use, as well as which score calculation algorithm is used when evaluating each discovered Bayesian network structure.

Since we use the random restarts method for hill-climbing in our project, we can tune the parameters of that algorithm (number of restarts and number of iterations within a restart). We decided to test the algorithm with 0, 5, 10 and 20 restarts and 200 iterations for each restart. The number of restarts increases the chance of finding the optimal solution (escaping local maxima), and the number of iterations defines how many structures will be examined after the highest scoring structure is discovered, which, if high enough, can improve the results for each restart.

1.3 Data

As in our last report, we use data from the 2015 from [7], which analyses health-risk behaviours in teenagers in the United States. This data is public and contains the answers given by more than 15.000 students to almost 100 different questions. All the information (together with the data) can be found at [7].

After discarding all those students whose answers were not completed (because they did not answer some questions or their answers were already discarded by the YRBS because they were inconsistent) we had a dataset containing 12.669 entries.

After some binning, we show our variables in Table 1. They are all related to some question asked in the mentioned survey.

Name	Type	#Levels	Description	YPBS question
Race	categorical	8	Race	Raceeth (Q4 & Q5)
Age	categorical	5	Age	Q1
Sex	categorical	2	Sex (Gender)	Q2
Weapon	categorical	2	Carry weapon in school	Q15
Unsecure	categorical	2	Feel unsecure in school	Q16
Fight	categorical	2	Participate in fight last year	Q20
Bulled	categorical	2	Bullied in school	Q24
Depression	categorical	2	Suffered depression last year	Q26
Alcohol	categorical	2	Alcohol use last month	Q43
Sports	categorical	4	Sports practiced	Q80
Grades	categorical	4	Grades	Q89

Table 1: The variables of our network. The table includes the name, type and number of levels of each variable (categorical variables with only 2 levels can be considered as binary).

2 Methods

As in our last report, we will use the programming language `R` for testing the networks. Both algorithms were already implemented in the library `bnlearn` [8]. We also used the library `igraph` when performing some distance measure calculations between the graphs. We will further discuss these methods ahead on this same section.

We don't perform any preprocessing on the data other that the mentioned binning of variables. This was not necessary to do for this assignment. However, because we used the same strategy (binning) for our first assignment and we want to make a fair comparison with the results of this study with the ones obtained in the previous one, we use the data with binning.

Before comparing different graphs, however, we perform a moralization of the output produced by the MMHC algorithm. This is so we can make the comparison between two undirect graphs, instead of between a directed and an undirected graph.

The objective is to compare two undirected graphs based on their structure. For this comparison we assume that we do not know any relation between the variables, since that could bias the comparison according to possibly wrong facts. The comparison will be explained in section **??**

3 Results

Before we start this section, it is important to highlight that, even though it is possible to add some prior information to the structure learning algorithms to include/avoid specific connections, this option is not used in this project. Since this practice was not specified on the original assignment, we use only the information contained in our dataset to build the final model.

We start by running the algorithms separately so that we can analyse their behaviour according to some parameters.

We start with the MMHC algorithm. Please keep in mind that all the figures shown here correspond to the moral graph of the output produced by MMHC (albeit this algorithm produces a directed graph).

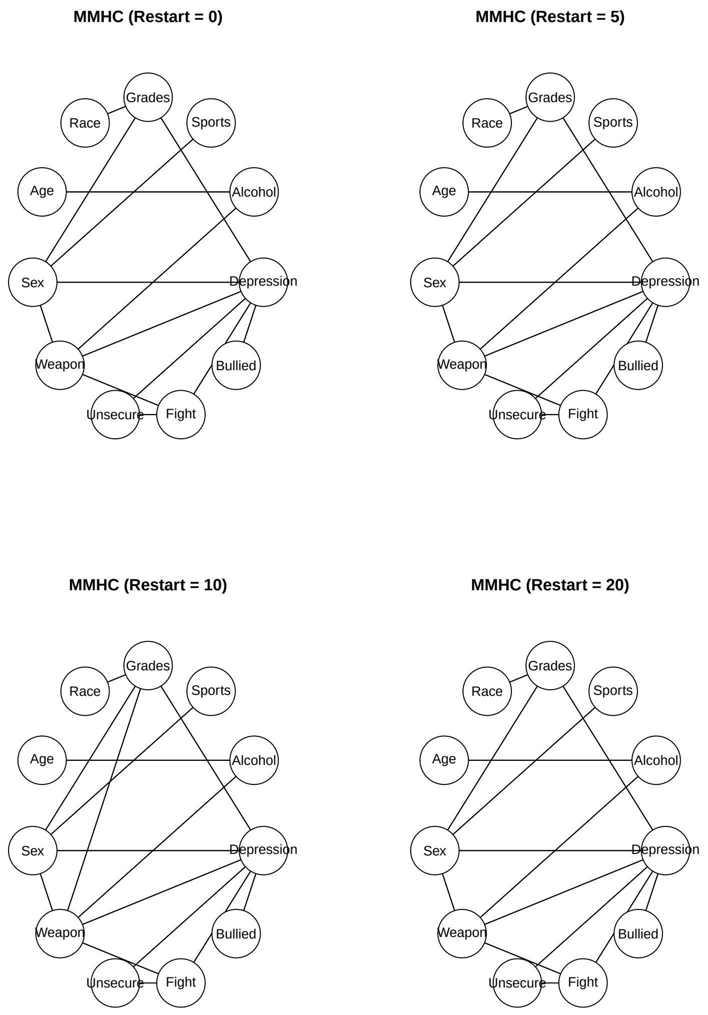

Figure 1: Output of the MMHC algorithm (moralized) depending on the number of restarts.

Since the biggest difference between this algorithm and our other choice (SI-HITON-PC) is the Hill Climbing part, we focus our efforts on examining how the changes on parameters that affect Hill Climbing modify the final result. We start by modifying the number of restarts the algorithm does so it can avoid getting stuck in a local maximum. Results for different configurations can be seen in 3. As we can see, there is a local maximum where the two first configurations get stuck, but with a few random restarts the algorithm seems to find a better solution. However, the final graph is only slightly changed.

We can now try to modify the number of edge modifications (insert, remove, or reverse) that are done after performing a random restart. Considering the last results, we choose a low number of restarts and see if we can find a difference when changing this parameter. This parameter has been called “perturb” to use the same term with the library we are using. Results can be seen in 3. Again, the problem seems really easy for this algorithm to solve and we do not need big numbers in the parameters to get the same results we got in our last picture (3).

Finally, to be sure that we are not getting stuck in a local maximum when running the algorithm, we will run it using 1000 random restarts and modifying the edges 200 times after every restart. This does not guarantee that we will find the global maximum but it greatly decreases the probability of getting stuck to a local maxium. The result (in figure 3) is the same with many other of the graphs plotted before.

For the Semi-Interleaved HITON-PC algorithm, we first try to see if there is any difference between solutions when we change the conditional independence test that decides if there is a correlation (or how strong it is) between two variables. We will use three different tests: the asymptotic χ_2 test (or Chi-squared test), for being the default one and a well known test, a shrinkage estimator for the mutual information test, for being an improvement over χ_2 test, and the Monte Carlo permutation test. When

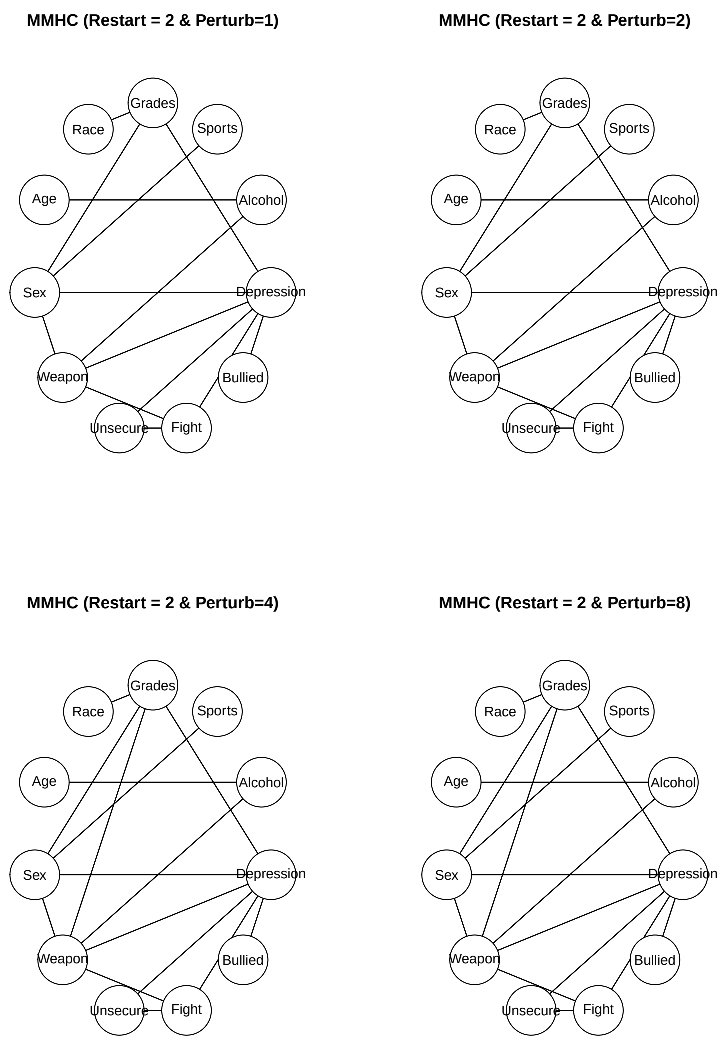
MMHC (Restart = 2 & Perturb=1)
MMHC (Restart = 2 & Perturb=2)
MMHC (Restart = 2 & Perturb=4)
MMHC (Restart = 2 & Perturb=8)
Grades
Race
Sports
Age
Alcohol
Sex
Depression
Weapon
Bullied
Unsecure
Fight

Figure 2: Output of the MMHC algorithm (moralized) depending on the number of attempts to modify (insert, remove, or reverse) an edge on every random restart.

all the other variables maintain their default values, the results produced can be seen in figure 3.

We now try modifying the maximum error rate allowed for a connection to be considered associated with other variable when performing the test. Results can be seen in figure 3.

3.1 Comparison

Together with this assignment we hand in a script that performs the following comparisons.

We compare the three graphs: MMHC with 10 restarts and 200 iterations, SI-HITON-PC with $\alpha = 0.05$, and our manually created graph. When doing the comparison, we first measure the number of relations on each graph. This piece of information is not very useful, as far as the structure of the graphs is concerned, but it is a good first indication that the final network may be too simple or too complicated. We also consider the number of relations that are actually present in both graphs. MMHC found 18 edges and SI-HITON-PC found 21 edges between the nodes, while the manually created graph had 40 edges.

As a second factor, we need to count for the number of changes that one network needs to perform to be similar to the other one. We understand a "change" as an addition or removal of a relation in a graph. We found 13 differences between MMHC and SI-HITON-PC, 28 between MMHC and manual graph, and 27 between SI-HITON-PC and manual graph.

In order to detect if there is any strange structure in the network we measure the mean length path between all variables. Notice that "strange" in this context does not mean necessarily bad, but

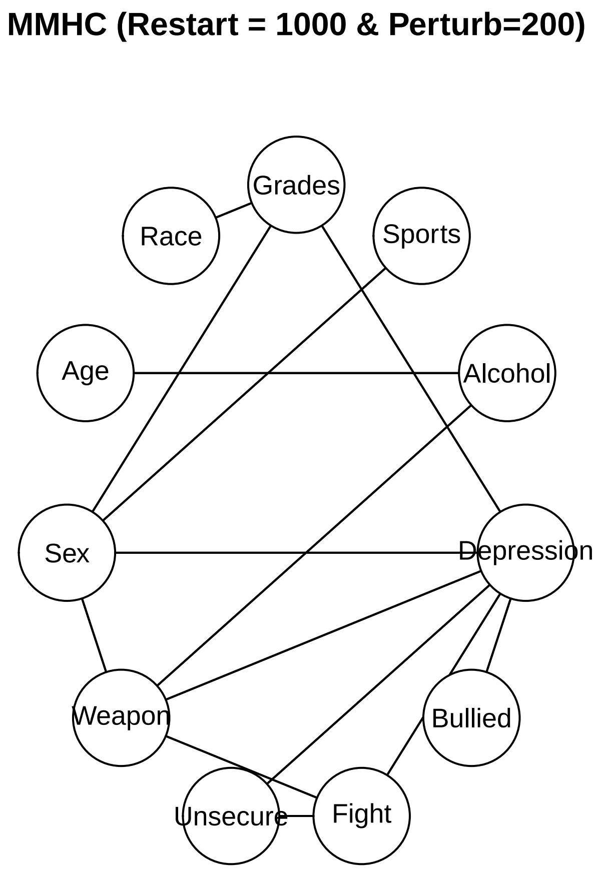

Figure 3: Output of the MMHC algorithm (moralized) after 1000 random restarts with a perturb value of 200.

something that is worth to look at to clarify if the structure is not valid. If a variable is not connected to any other variable, this will greatly influence the score, because we presume that, to reach that variable from any other variable, we need to take a path $\#_variables + 1$ long. In this test, MMHC scored 1.84, SI-HITON-PC scored 1.7, and the manual graph scored 1.27.

4 Discussion

First thing to notice is that most of the graphs proposed by these algorithms (independently of the configuration used) output a more simple graph than the ones we manually created for our first assignment. After doing that first assignment we already concluded that the network had so many parameters that was almost impossible to correctly predict the value we were looking for (probability of a teenager carrying a weapon to school). Now we already see that most of those connections we added were not well supported by the data we use.

About the algorithms themselves, and looking at the results, we can conclude that the MMHC algorithm was more sophisticated than the problem's requirement. The problem is solved nicely and the results generated by the algorithm seem reasonable. However, we tried to tune the parameters related with the hill-climbing part, and we saw that there isn't a big difference between the different outputs. Thus, maybe a simpler algorithm that does not perform hill-climbing would suffice for our problem. Regardless, the MMHC remains a very efficient algorithm, and it does produce a slightly better graph.

We observed that comparing the networks when only having undirected graphs can be complicated. Of course, we are still able to see the relation between two variables, but we lack of any other information that could be useful for the comparison. In the beginning, we proposed these two algorithms so as to get as different outputs as possible. In retrospect, having two directed graphs or two CPDAG (Completed

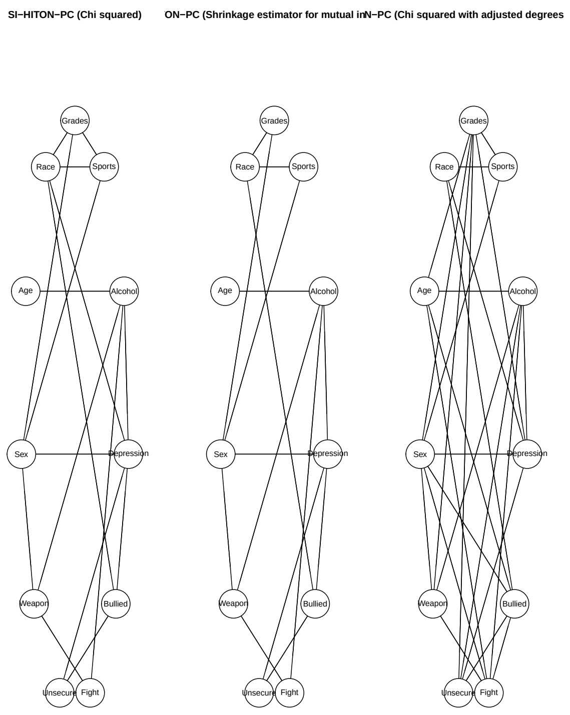
SI−HITON−PC (Chi squared)
ON−PC (Shrinkage estimator for mutual in
N−PC (Chi squared with adjusted degrees
Grades
Race
Sports
Age
Alcohol
Sex
Depression
Weapon
Bullied
Unsecure
Fight

Figure 4: Output of SI-HITON-PC depending on the test used for associate variables.

Partial DAG) would have helped much more when making the comparison (we could have used the concepts of V-structures or Markov Equivalence Classes).

Finally, the algorithms used here were already implemented in the chosen library. However, several versions or improvements of them could be used to deal with the structure learning problem. For example, MMHC was proposed in the original paper using a tabu list that could help the algorithm converge. An improvement to this same algorithm was also proposed in 2012 [9]. This algorithm alters the first step of the MMHC algorithm (defining the potential parents), but maintains the second one (search and score algorithm). The HITON-PC algorithm has been maintained and updated mostly by its creator. The version used here (Semi-Interleaved HITON-PC) is actually one of these improvements [1].

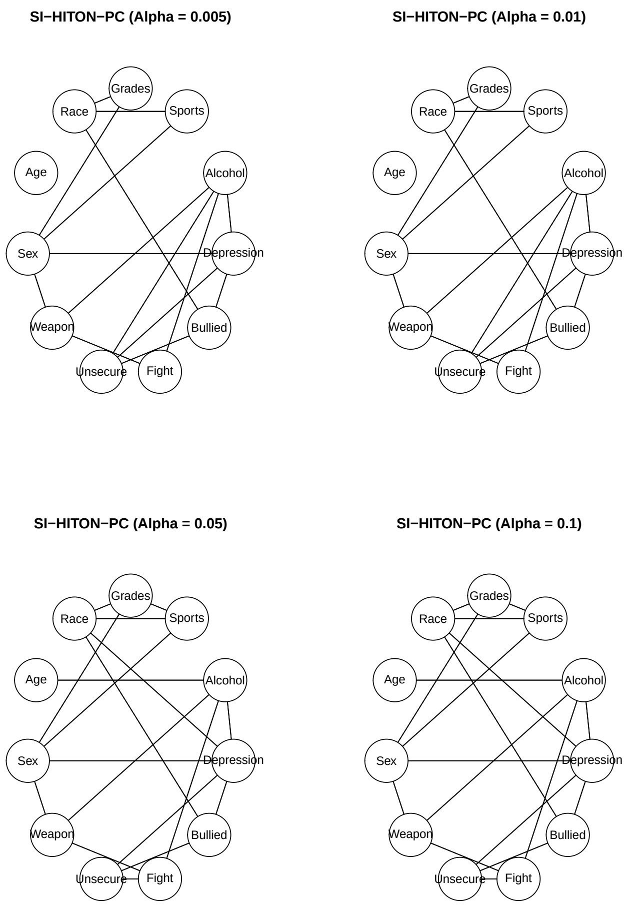
SI−HITON−PC (Alpha = 0.005)
SI−HITON−PC (Alpha = 0.01)
SI−HITON−PC (Alpha = 0.05)
SI−HITON−PC (Alpha = 0.1)
Grades
Race
Sports
Age
Alcohol
Sex
Depression
Weapon
Bullied
Unsecure
Fight

Figure 5: Output of SI-HITON-PC depending on the error rate allowed.

References

[1] Constantin F Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D Koutsoukos. Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11(Jan):171–234, 2010.

[2] Constantin F Aliferis, Ioannis Tsamardinos, and Alexander Statnikov. Hiton: a novel markov blanket algorithm for optimal variable selection. In *AMIA Annual Symposium Proceedings*, volume 2003, page 21. American Medical Informatics Association, 2003.

[3] Andrea Sboner and Constantin F Aliferis. Modeling clinical judgment and implicit guideline compliance in the diagnosisof melanomas using machine learning. In *AMIA*. Citeseer, 2005.

[4] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[5] Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale markov blanket discovery. In *FLAIRS conference*, volume 2, pages 376–380, 2003.

[6] Nir Friedman, Iftach Nachman, and Dana Peér. Learning bayesian network structure from massive datasets: the sparse candidate algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc., 1999.

[7] Youth risk behavior survey (yrbs), 1995-2015. URL `http://www.cdc.gov/healthyyouth/data/yrbs/data.htm`.

[8] Marco Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010. URL `http://www.jstatsoft.org/v35/i03/`.

[9] Maxime Gasse, Alex Aussem, and Haytham Elghazel. An experimental comparison of hybrid algorithms for bayesian network structure learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 58–73. Springer, 2012.