An Improved Bayesian Network Super-Structure Evaluation using *Physarum polycephalum* Bio-inspiration

Vitor P. Ribeiro [*] **Benvindo R. Pereira Jr.** [**] **Carlos D. Maciel** [**]
José Antônio P. Balestieri [*]

[*] *School of Engineering, São Paulo State University (UNESP), Guaratinguetá, SP, Brazil (e-mail: {vitor.p.ribeiro;jose.perrella}@unesp.br).*
[**] *Department of Electrical Engineering, São Paulo University (USP), São Carlos, SP, Brazil (e-mail: {brpjunior;cdmaciel}@usp.br)*

Abstract: Bayesian Networks are a powerful tool to assess association relationships between variables on a given observed system. Despite their usage on several areas, the task of obtaining such structure exclusively from observational data is a well-documented NP-hard problem. *Physarum Learner* is a method to obtain a Bayesian Network from a data set, based on the foraging behaviour of the *Physarum polycephalum*, a member of the Myxomycetes class. However, the original method overlook crucial characteristics of this combinatorial problem. In the work here presented, we implement three changes to the *Physarum Learner* original algorithm and apply this improved method on three well-known benchmark data sets. Our results indicate a faster convergence point identification, while achieving good structures from a scoring standpoint.

Keywords: Equivalence class; Bayesian networks; Physarum Autonomous Optimisation; Bio-inspired algorithm.

1. INTRODUCTION

Bayesian Networks (BNs) are a type of Probabilistic Graphical Model that can be used to build models from data or expert knowledge, and provide a robust and mathematically coherent framework for the analysis of several kind of problems (Uusitalo, 2007). This method is becoming increasingly popular to model uncertain and complex domains such as medicine (Shen et al., 2022), traffic engineering (Wang et al., 2022), and risk assessment (Ruiz-Tagle et al., 2022). This wide usage is mostly due to its function to point at associations between a large set of variables from the observed system, in an explainable format.

Despite the usefulness of BNs in analysing complex systems, the construction of the structure which best comprehends the associations between a variables of a given system, referred to as an optimal solution, is a well-known NP-hard problem (Chickering et al., 2004). Most of the algorithms proposed to build BNs from data fall into three categories: constrain-based (CB) methods, search-and-score (SS) methods, and hybrid methods.

The constrain-based (CB) methods employ conditional independence (CI) tests to create their structure (Koller and Friedman, 2009), which in one hand scale up better to problems with many variables, but on the other provide a structure with lower accuracy when compared to the next two methods (Natori et al., 2015).

The score-and-search (SS) methods use any suitable scoring function to rate a candidate Bayesian network, and search for a structure that optimises said score through adjustments on the elements of said network (Cooper and Herskovits, 1992). Although the results with this approach are usually more accurate for smaller networks, it becomes unfeasible when dealing with a problem with many variables, due to the super-exponential increase on the number of candidate structures that it would have to score (Gross et al., 2019).

Lastly, hybrid methods such as the one presented in Dai et al. (2018) make use of techniques from the CB approach in order to reduce the search space, then use procedures based on the SS approach to examine this constrained search space and provide a solution. This class of methods allow for a fast reduction of the search space, which is then used as a small subset of structures to be scored and searched upon.

The *Physarum Learner* algorithm employs a hybrid approach to provide a Bayesian Network exclusively from data (Schön et al., 2014), based on a Physarum Autonomic Optimisation (PAO) (Tero et al., 2007) methodology. Although this algorithm provides good structures according to a given score, as provided in the original article, the procedure described there presents an approach that can be improved in terms of score convergence, computational time, and coverage of the evaluated structures.

As described in Chickering (2013), the procedure to build a BN exclusively from fitting the structure to the data

according to a given score function generates a family of graphical structures with the same or very similar scores. This family of structures is called an *equivalence class*, where the most complete structure that encompasses all possible graphs from the same equivalence class is referred to as the *super-structure* for the association relationships on that data set.

The original *Physarum Learner* performs an extensive search on the space of possible structures given a starting point, this initial configuration is then used throughout all iterations of the method, which causes the search procedure to be trapped in a local optima, a common issue with heuristics with a single starting point for the solution (Mao et al., 2021).

The exhaustive nature of the PAO approach on *Physarum Learner* taken causes the algorithm to conduct the evaluating process for new candidate structures regardless on whether or not the score has reached a plateau. Given the nature of super-structures from the learning process exclusively from data, one could interrupt the learning process once small changes in the network provide a negligible change in score, or none at all (Wang et al., 2021).

In this paper we present a modified version of the *Physarum Learner* algorithm, which is comprised of three new steps in order to tackle some of the perceived issues on the original method. First, our approach is to abort the search procedure once the score has stagnated for a number of iterations. Second, as the main goal is to minimise a given score, we include one step to change the direction or presence of edges after each iteration, to cover a larger search space for a suitable super-structure. Third, in order to minimise the issue of premature convergence to a local optima, our approach uses different starting points across iterations, which helps to create diverse starting points to the subsequent search-and-score steps.

Our method is shown to achieve promising results on small and medium benchmark data sets, with a fast convergence and sound structures when compared to the expected result. We also provide our code for our improved implementation of the *Physarum Learner* using *Python* in an open-source approach, as well as the data sets used for the testing and validation of the method.

Section 2 summarises the theoretical background on Bayesian networks, and presents the relevant basis on the original *Physarum Learner* method, as well as the description of the data sets used. Section 3 contains the algorithm description and the improvements made when compared to the original version. Section 4 shows the results of the algorithm applied to the previously mentioned data sets and the pertinent discussion. Finally, Section 5 contains the conclusions achieved from the presented results.

2. THEORY

2.1 Bayesian Network

Formally, the function of a Bayesian Network (BN) is to estimate a joint probability distribution (JPD) based on n discrete random variables, from a data set D taken from the observed system (Koller and Friedman, 2009).

A BN $B = (G, \Theta)$ is defined by a Directed Acyclic Graph (DAG) G, and a set of conditional probability tables Θ. These structures are referred to as BN's *qualitative* and *quantitative* parts, respectively (Neapolitan, 2004).

For the *qualitative* part of a BN, let G be a DAG comprised of a set of vertices $V = \{v_i\}_{i=1,2,...,n}$, and a set of directed edges $E = \{e_{ij}\}_{i,j=1,2,...,n;i\neq j}$. Each v_i represents a random variable from the data set D, and each $e_{ij} \in \{0, 1\}$ indicates the absence or presence of a directed edge pointing from the vertex v_i to the vertex v_j.

For the *quantitative* part, $\Theta = \{\theta_i\}_{i=1,2,...,n}$ is the set of conditional probability distributions for each one of the n observed variables. As such, $\theta_i = p(v_i|P_i)$, where P_i denotes the set of k vertices where $e_{ki} = 1$ from the aforementioned graph G, also known as the parents' set of the vertex v_i on the graph.

Figure 1 represents a Bayesian Network of a toy problem known as the "Sprinkler Network", presented in Murphy (1998). Each vertex represents a binary variable of interest, the edges illustrate associations among the variables, and every conditional probability table for every random variable is explicitly presented.

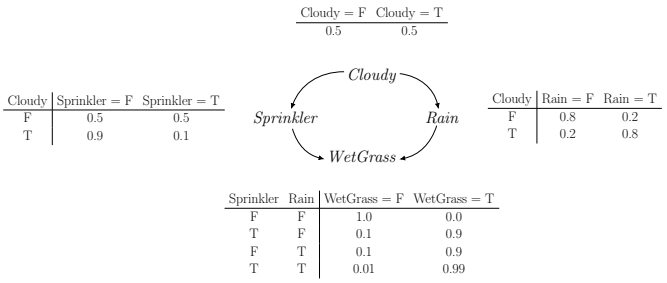


Figure 1. Bayesian Network for the "Sprinkler" data set, adapted from Murphy (1998). The fact that all four variables are binary allow for a representation of false (F) or true (T) for the occurrence of each one. Every variable on this system has a conditional probability table associated with them, which represents the chance of a given outcome taking into account the variables associated with the one being observed and their outcomes, defined on the main text as $p(v_i|P_i)$.

Given both the *qualitative* and *quantitative* parts of the network, the joint probability distribution of the variables on B can be estimated through the Bayes' chain rule, defined as follows:

$$p(v_1, v_2, ..., v_n) = \prod_{i=1}^{n} p(v_i|P_i) \tag{1}$$

The main task of an algorithm designed to create a BN from data revolves around an optimisation problem: find a structure B whose joint probability distribution best fits the one from the given data set D provided.

Ref. Broom et al. (2012) shows that one approach to this optimisation problem is to find the DAG which best represents the association relationships between variables present on the data set. The adequacy of a given graphical structure can be estimated through a fitness function that associates a candidate graph with the provided data

set. Examples of fitness functions for this scoring procedure include the *Bayesian Information Criterion* (BIC) (Schwarz, 1978), and the *Bayesian Dirichlet equivalence with uniform prior metric* (BDeu) (Buntine, 1991).

2.2 Physarum Learner

Physarum polycephalum ("multi-headed slime mould") belongs to the Amoebozoa group, in the class Myxomycetes (Stephenson and Stempen, 2000). During the active vegetative stage of its life cycle, called "plasmodium", it can extend for up to $30x30cm^2$, and move to a maximum speed of $4cm$ per hour, as a multi-nuclear unicellular being (Vogel and Dussutour, 2016).

The seminal work in Nakagaki et al. (2000) presented the ability of the *Physarum polycephalum* in consistently finding the shortest path between two sources of food in a maze, during its plasmodium stage. This behaviour kindled the interest of researchers on several areas into analyse, understand, and model the behaviour of this brainless being (Oettmeier et al., 2020). A survey on the subject can be found in Gao et al. (2018).

One particular area that benefited from the maze-solving experiment was mathematical modelling and route optimisation. *Physarum Solver* is an adaptive mathematical model which simulates the foraging behaviour of the *P. polycephalum* as presented in Nakagaki et al. (2000), and it is first proposed in Tero et al. (2006).

Physarum Solver's original purpose is to find the optimal path between two vertices on a weighted undirected graph, similar to the classical Dijkstra's algorithm, and was proven to provide the optimal solution for a graph on a Riemannian surface (Miyaji and Ohnishi, 2008). As the work in Nakagaki et al. (2000) describes the behaviour of the *P. polycephalum* on a maze, the weighted undirected graph used on the *Physarum Solver* algorithm is referred to as *Physarum* maze, and the initial and final vertices on the path-finding problem are called source and sink nodes, respectively.

Physarum Learner (Schön et al., 2014) is an algorithm which employs the *Physarum Solver* modelling as an auxiliary structure to build a BN from a data set. Figure 2 presents a flow chart for the original *Physarum Learner* algorithm, containing its main steps.

Given a data set containing n variables, the *Physarum Learner* algorithm starts by creating two graph structures: a fully-connected undirected graph, which will act as the "maze" for the *Physarum Solver* computations, and a directed graph, upon which the BN will be built on. Both structures contain n nodes each.

For the initial set up of the *Physarum* maze, the parameters for the edges on the *Physarum* maze are set to standard values. This structure will serve as an "external memory" for the creation of the BN, and its parameters will be updated accordingly on the next steps.

As for the directed graph, the algorithm creates edges connecting every pair of nodes, on a random direction. This decision presents no issues from a structural standpoint, given that the structure will be rebuilt after the first iteration of the algorithm.

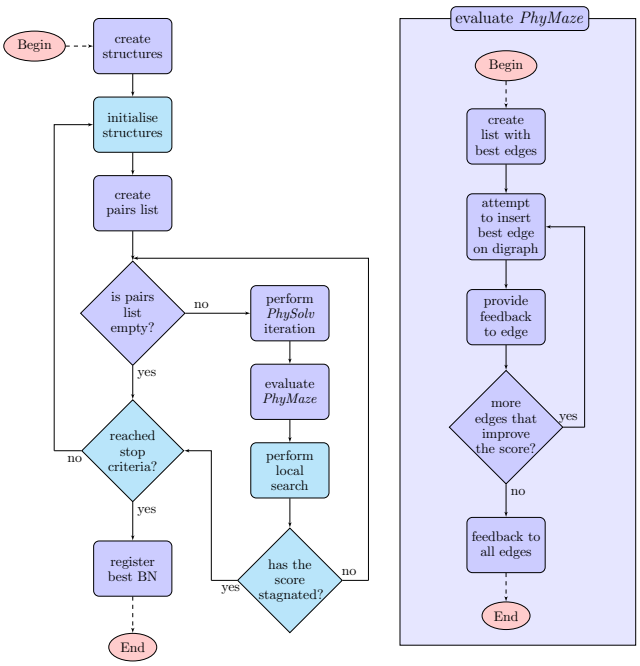


Figure 2. Flow chart depicting the *Physarum Learner* (Schön et al., 2014) algorithmic structure, with improvements presented in this article. On the left, the main structure is presented. The internal structure for the "evaluate *PhyMaze*" block is expanded on the right, as this block comprehends the interactions between the *PhyMaze* – an undirected graph – with the Bayesian Network been built. The boxes in blue are from the original algorithm, while the boxes in cyan show where the improvements from this work were made.

The original *Physarum Solver* algorithm finds the optimal path between a pair of variables on an undirected graph. For this reason, the *Physarum Learner* algorithm creates a list of every pair of nodes contained on its *Physarum* maze. This ensures that every pair of nodes and, consequently, every pair of variables, will be evaluated during execution.

One pair is chosen at random to represent the input source/sink nodes for the *Physarum Solver* algorithm. The parameters on the edges on the *Physarum* maze will be updated, reflecting the attempt to find the best path between this pair. The edge directly connecting the pair taken as source/sink from the previous step has its parameter set above a threshold value, even if the result of the *Physarum Solver* iteration did not set it as such. This ensures that all the edges will be considered at least once to be part of the BN on the next step.

Once the *Physarum Solver* iteration ends, the algorithm performs an *evaluation* on the resulting maze. This is the step which represents the interfacing between the undirected graph and the Directed Acyclic Graph (DAG), representing the BN.

The evaluation of the maze consists of creating a list containing every edge from the *Physarum* maze whose parameters surpasses the given threshold. All the other edges that don't comply with this criteria based on the *Physarum* maze are removed from the BN, if present.

For every edge present on the edges list, the algorithm attempts to insert the same edge on the DAG. Considering both directions for the edge, if the inclusion of it does not result in a cycle on the graph, and there is an improvement on the score from inserting this edge on given direction, this edge is considered as candidate to be permanently set on the DAG. Once the best connection is found among all the edges on the list, this connection is removed from the list, and an edge is set on the DAG considering the direction which provides the best improvement.

If a connection that provides the best improvement is found, then a feedback is applied to this edge on the *Physarum* maze. This feedback ensures that this edge will be obtained for the next evaluation step, and will not be removed from the DAG structure.

Finally, when there are no more edges on the list that provide a score improvement for the BN, a feedback function is applied to all of them, lowering their parameters in order to prevent the *Physarum Solver* algorithm to include those edges on the shortest path for the next iteration.

One iteration of the algorithm is defined by the list of every pair of nodes on the *Physarum* maze being depleted. Once an iteration has end, the *Physarum* maze as the "external memory" will contain the best edges found so far on the BN, coded on the parameter values for its edges. For the next iteration a new pairs list is created, and this time the learning process take into account the edges already identified as "best" for the BN from the previous iteration.

Let T be the total number of iterations on the *Physarum Learner* algorithm. Considering one time unit as the time for the algorithm to compute all the steps for a single pair of nodes on the complete pairs list, one can describe the time complexity for the algorithm $O(n)$ as:

$$\begin{aligned} O(n) &\approx T \cdot n^2 \\ O(n) &\propto n^2 \end{aligned} \tag{2}$$

As the algorithm performs all T iterations over the entire pairs list, the best case is the same as the worst case, as is the average case.

3. MATERIAL AND METHOD

3.1 Improved Physarum Learner

The original *Physarum Learner* algorithm exhibits good performance to build a Bayesian Network (BN) from a data set. Nevertheless, some of its presented features lead to excessive time consumption to perform the task at hand, as can be observed from the time complexity estimated on Equation 2.

The blocks coloured in cyan on Figure 2 represent steps modified from the original or entirely new blocks, on the algorithm here named *Improved Physarum Learner*

First, the evaluation step on the original article assumes a "greedy" approach to build a BN based on the *Physarum* maze, as it considers the best edge one at a time to include on the DAG. This procedure, although similar to the classical algorithm K2, does not contemplate the fact that, according to Equation 1, the joint probability is function of *every* parent for every variable on the network.

This definition implies that, once a new set of edges is inserted onto the graphical structure, a local search should be performed over the obtained graph. This extra step is required to determine the best direction for each edge, given that all the other edges are already present on the directed graph.

The implementation here proposed tries to change the direction of every edge on the obtained BN, on a random order, searching in the graph neighbourhood for a structure with a better score.

Second, the original algorithm ends an iteration only when its pairs list is empty. This ensures that every one of the approximately n^2 edges is considered to be inserted on the BN at least once. This approach does not take into account that the best structure might have already been found during previous computations, achieving a local optima super-structure.

A procedure to check for the eventual structure stagnation across several evaluation steps is to verify if the score for the obtained structure is kept constant, or within an error margin, for a given number of pairs analysed. The *Improved Physarum Learner* contains a structure which checks for this score stagnation. If the score is identified as constant, or within an error margin, after a percentage of pairs have been analysed, the current iteration is then finished.

Third, the original algorithm relies on the analysis of nodes on the pairs list in a random order, and both the external memory and the obtained BN are kept constant across iterations. The order in which the pairs are considered during an iteration may result in a bias for both structures, to be carried across iterations.

In order to lessen this eventual bias, and to enable new structures to be found, *Improved Physarum Learner* resets the *Physarum* maze as well as the BN structure to their original conditions. The best BN found so far is stored on a separate structure.

Lastly, the original algorithm proposes the edges on the first DAG to be set between all pairs of nodes, in random directions. The computational cost involved in creating all those edges and picking a random direction for each one is prohibitive, specially when dealing with networks with larger number of nodes.

Improved Physarum Learner creates its DAG structures containing only the nodes for the random variables, and no edges connecting them. This is a valid approach due to the structure been overridden at each evaluation step.

This version of the algorithm is implemented in the *Python3* language, due to its broad usage on scientific research, as well as to its open-source licensing. The complete code is readily available upon request.

3.2 Data sets

Here we describe the data sets used as benchmarks for the testing of our improved method.

Asia network This Bayesian Network is proposed in Lauritzen and Spiegelhalter (1988), as a tool to illustrate how the method on the original paper works. The original structure is reproduced here as Figure 3a. The structure consists of eight nodes, and eight edges. Each node codifies a binary variable, which can only assume the values of *True* or *False*.

We chose this structure as one of our benchmarks due to its small number of nodes, and the fact that it is a weakly connected graph, i. e. there is a path between every pair of nodes if the direction of the edges is omitted.

Sachs network This BN is presented in Sachs et al. (2005), where the authors study possible causal relationships on a protein-signalling environment. The structure obtained on the original article is reproduced here as Figure 5a.

This structure is a good candidate to test the method based on its relatively small size, and the fact that this graph is not connected as the one presented in the *Asia* structure. This is an interesting characteristic due to the usage of a fully-connected undirected graph as an auxiliary structure for the method.

Alarm network This network is presented in Beinlich et al. (1989), and encodes a medical monitoring system, built from the authors' expertise and medical textbook literature.

The graph contains thirty-seven nodes, each representing one out of three category of variable: a diagnostic, a measurement, or an intermediate variable. The values for all nodes are represented as a discreet value, with two or three levels each.

This is a particularly interesting network to be used as a benchmark due to the expert knowledge that it encodes. Each of the "diagnostic" type of variables is independent from the others, and are supposed to have no parents on the graphical structure, given the assumption that an identified condition characterised by a diagnostic is the cause of the variables associated to it. As the method here presented deals primarily with a data set, this *a priori* knowledge from the structure is omitted, being used afterwards to analyse the obtained structure.

4. RESULTS AND DISCUSSION

This section is reserved to the presentation and discussion of the results obtained from applying the *Improved Physarum Learner* on the benchmark data sets already presented. The learning process was undertaken on a desktop computer with Fedora OS, processor Intel® Core™ i7-4970 CPU at 3.60GHz, and 32GB of RAM at 1600MHz.

The Bayesian Networks (BNs) super-structures obtained are presented here on their graphical structure, with edges direction purposefully omitted. The colour scheme for the colouring of the edges is as follows: a green edge represents a correct edge, pointing in the correct direction; a blue edge represents a correct edge, but pointing in the opposite direction of the original; and a red edge represents an edge that should not exist. This correctness is based on the original graphical structures for each data set.

The justification for this representation is based primarily on the works of Pearl and Mackenzie (2018) and Cox Jr. (2018). Ref. Pearl and Mackenzie (2018) arguments that a finite data set can never encompasses all of the relationships of association from a system within itself. This also includes the direction of the edges in the obtained structure.

Furthermore, Cox Jr. (2018) argues that a Bayesian Network when built exclusively from data should be used as a starting point for an expert to analyse the system, as a result of the data set insufficiency on incorporating all of the correct associations.

4.1 Results for the Asia data set

Figure 3b presents the graph obtained for the *Asia* data set. As the data set with the least variables, the method was able to find all correct edges, on the correct directions, with exception of the edge *Smoking* $\rightarrow$ *Lung cancer*, which was found to be on the opposite direction.

The extra edge *Asia* $\rightarrow$ *Either* can be validated as an expected edge from the concept of "d-separation". The same reasoning can be applied to the extra edges *Tuberculosis* $\rightarrow$ *Xray*, and *Lung cancer* $\rightarrow$ *Xray*.

Ref. Koller and Friedman (2009) defines d-separation as a collateral edge resulting from the setting of the middle random variable on a association chain. Consider the chain $X \rightarrow Y \rightarrow Z$, with X, Y, and Z random variables. As X influences Y, and Y influences Z, if the variable Y has its value fixed, the influence from X is "felt" by Z as a collateral effect.

Although the extra edge *Tuberculosis* $\rightarrow$ *Bronchitis* can not be directly explained using the concept of d-separation, the association between these medical conditions is well established on the medical literature (Martin et al., 1968).

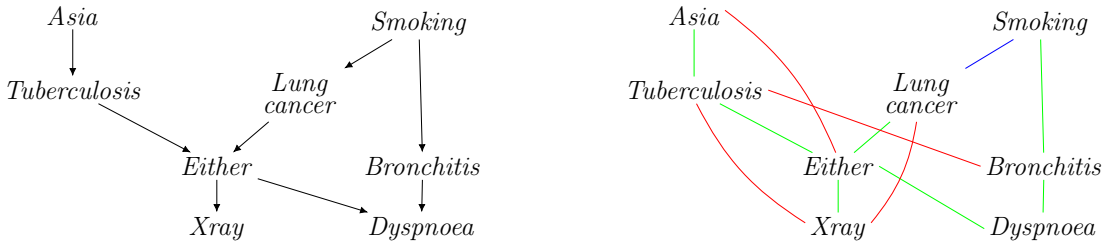
For the fast convergence of the method, Figure 4 presents the convergence of the BDeu score for the best BN structure obtained after each iteration of the algorithm, in red, compared to the score for the original structure, in blue. The convergence was obtained for a score lower than the expected from the original after only three iterations, whereas the original algorithm would require at least twenty-eight iterations to provide an answer for a network with eight nodes. The structure learning process for this structure took on average less than one minute across several tests.

4.2 Results for the Sachs data set

Figure 5b presents the graphical structure obtained for the *Sachs* data set. The *Improved Physarum Learner* algorithm managed to find the same number of edges from the original structure, although some edges were identified in the wrong direction, compared to the original.

Despite the algorithm employing a fully-connected maze as its external memory, it was able to allow the variables *Plcg*, *PIP3*, and *PIP2* to be disconnected from the remaining graph, as the original structure requires.

Figure 6 present the score convergence for the best network obtained after each iteration of the algorithm, in red. In



(a) Original structure.

(b) Obtained structure.

Figure 3. Bayesian networks for the *Asia* data set. On the left, the original structure as described in Lauritzen and Spiegelhalter (1988). On the right, the structure found with the improved method. Although the obtained structure has more edges than the original, the absolute value of its BDeu score is marginally lower (better) than the score from the original structure, given the same data set.

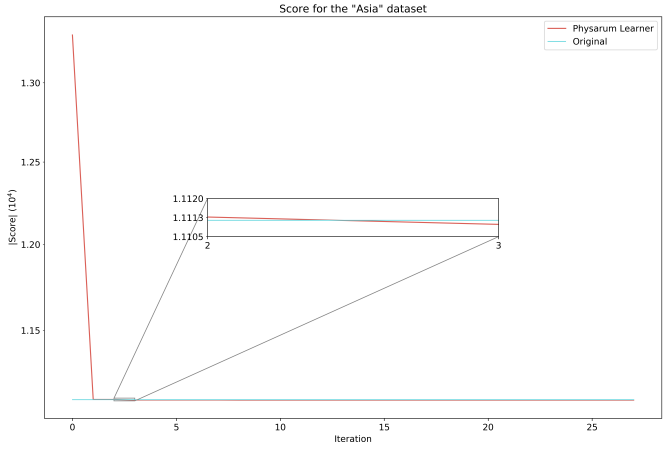


Figure 4. Semi-log representation for the convergence of the BDeu score absolute value for the *Improved Physarum Learner* applied to the *Asia* data set, in red. In blue, the score for the original structure, applied to the same data set. The detail presents the iteration after which the obtained structure presents a score marginally better than the original.

blue, the score for the original network on the same data set. After the thirty-seventh iteration, the score for both structures were exactly the same. The original algorithm requires at least fifty-five iterations to provide an answer for this network. The structure learning process for this structure took on average fifteen minutes across several tests.

This result raises a question concerning the relevance of the direction of the edges when obtaining a score for given structure. From the super-structure approach, both the structures are the best from a score standpoint, requiring an expert in the modelled system to point out which one is the structure which more accurately comprises the associations observed in the real world.

4.3 Results for the Alarm data set

For the *Alarm* data set, our method obtained twenty-four correct edges with the right orientation, eighteen correct edges with the wrong orientation, and eighteen edges that are not present in the original structure. The fact that it contains several extra edges reinforce the argument from Pearl and Mackenzie (2018), considering the details from this data set that were presented in Section 3.2.3. The algorithm did not take into account the fact that, in the real world, one diagnostic influences another diagnostic, although the expert on this medical system presented this constraint on the original article for the network, for instance.

Several instances of wrong edges present on the obtained structure could be dismissed from an expert on the system. For instance, the associations obtained between different diagnostics due to the knowledge that the events represented by these variables should not directly interact with one another. On a similar manner, some instances of wrong edges might be dismissed based on the concept of d-separation presented before.

The obtained model did not encompass only four edges present on the original network. To evaluate the correctness of the structure missing these four edges, conditional independence tests were run on each pair of variables, given the values on the data set. For the four cases the test pointed to independence between the variables within a threshold, which illustrate the impact that the data can have on the perceived best super-structure.

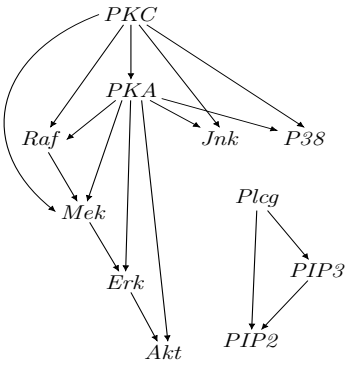
Figure 7 presents in red the score for the best structure found after each iteration, compared to the score for the original structure, presented in blue. As a result of the high number of variables on this data set, the score achieved several plateau levels during the execution. The structure learning process took on average seventy-two hours for this data set. This jump in computational time is expected as the internal *Physarum Solver* algorithm takes on a complete graph containing thirty-seven nodes, and the list of source/sink pairs contains six hundred and sixty-six different combinations.

The convergence test implemented was most valuable for this data set. It allows for the reset of structures after several iterations have passed with stagnated scores, instead of waiting for the end of the pairs list, which may take up to almost six hundred and seventy iterations.

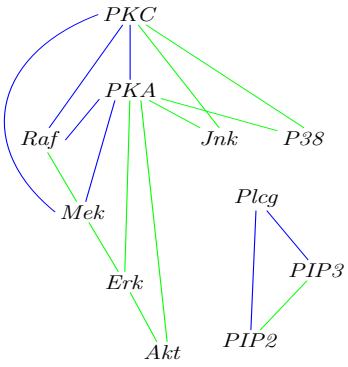
5. CONCLUSION

In this paper we presented a version of the *Physarum Learner* algorithm re-implemented and improved to tackle computational weaknesses presented in the original method.

The *Improved Physarum Learner* achieved sound results when obtaining the Bayesian Networks from benchmark



(a) Traditional structure



(b) Obtained structure.

Figure 5. Bayesian networks for the *Sachs* data set. On the left, the original structure as proposed in Sachs et al. (2005). On the right, the structure found by us. Even with some of the edges on the opposite direction from the original, both structures present equal BDeu score given the same data set. The method was also able to produce a disconnected graph, in conformity to the original structure.

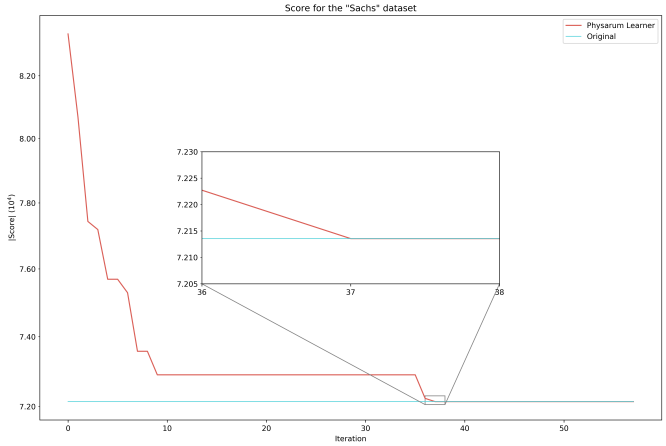


Figure 6. Semi-log representation for the convergence of the BDeu score absolute value for the *Improved Physarum Learner* applied to the *Sachs* data set, in red. In blue, the score for the original structure, applied to the same data set. The detail presents the iteration after which the obtained structure presents the exact same value for score as the original.

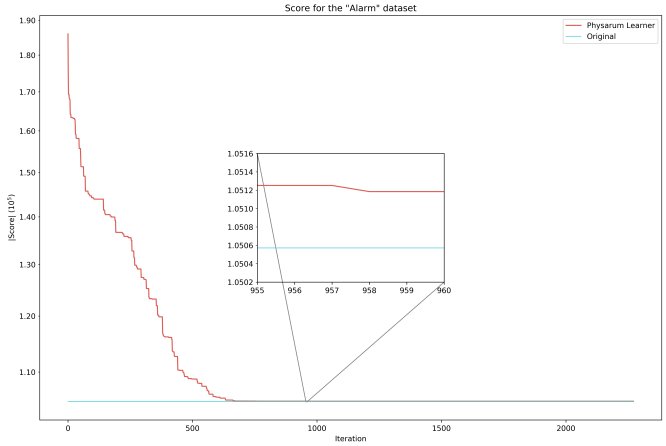


Figure 7. Semi-log representation for the convergence of the BDeu score absolute value for the *Improved Physarum Learner* applied to the *Alarm* data set, in red. In blue, the score for the original structure, applied to the same data set. The score for the obtained structured become stable on several plateaus, before becoming stable on a score slightly higher than the score for the original structure.

data sets, in a reasonable amount of iterations when compared to the original method.

Our results also provides evidence for the intrinsic association between the quality of a structure and the data set used to learn it. Results in the *Asia* network show that is possible to obtain a *marginally* better score with the structure here presented when compared to the structure on the original article. Results for the *Alarm* data set show that a Bayesian Network is only as good as the data allows it to be, given the perceived conditional independence on a subset of variables that have an edge in the original structure.

The results also corroborates the notion that multiple directed acyclic graphs can represent the same association structure from a scoring standpoint. In our case, both the expected and obtained DAG are slightly different on structure – direction of edges – but both present the exact same score given the same data set.

As limitation of the presented method, and future works within this context, we perceive a necessity for a mechanism that reduces the number of the nodes on the *Physarum maze* for each iteration of the *Physarum Solver*, as the total time required for this step across all iterations is exponential according to the number of nodes on the maze.

ACKNOWLEDGEMENTS

Vitor P. Ribeiro and Jose Antonio P. Balestieri are grateful to the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); Carlos Maciel is partial supported in processes 142390/2019-4, 301853/2018-5. FAPESP 2014/50851-0, CNPq 465755/2014-3 and BPE Fapesp 2018/19150-6.

REFERENCES

Beinlich, I.A., Suermondt, H.J., Chavez, R.M., and Cooper, G.F. (1989). The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In J. Hunter, J. Cookson, and J. Wyatt (eds.), *AIME 89*, 247–256. Springer Berlin Heidelberg, Berlin, Heidelberg.

Broom, B.M., Do, K.A., and Subramanian, D. (2012). Model averaging strategies for structure learning in bayesian networks with limited data. *BMC Bioinformatics*, 13(S13), S10.

Buntine, W. (1991). Theory refinement on bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'91, 52–60. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Chickering, D.M., Heckerman, D., and Meek, C. (2004). Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5, 1287–1330.

Chickering, D.M. (2013). Learning equivalence classes of bayesian networks structures. *arXiv:1302.3566*.

Cooper, G.F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4), 309–347.

Cox Jr., L.A. (2018). Modernizing the bradford hill criteria for assessing causal relationships in observational data. *Critical Reviews in Toxicology*, 48(8), 682–712.

Dai, J., Ren, J., Du, W., Shikhin, V., and Ma, J. (2018). An improved evolutionary approach-based hybrid algorithm for bayesian network structure learning in dynamic constrained search space. *Neural Computing and Applications*.

Gao, C., Liu, C., Schenz, D., Li, X., Zhang, Z., Jusup, M., Wang, Z., Beekman, M., and Nakagaki, T. (2018). Does being multi-headed make you better at solving problems? A survey of Physarum-based models and computations.

Gross, T.J., Bessani, M., Darwin Junior, W., Araújo, R.B., Vale, F.A.C., and Maciel, C.D. (2019). An analytical threshold for combining Bayesian Networks. *Knowledge-Based Systems*.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*.

Lauritzen, S.L. and Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2), 157–224.

Mao, J., Pan, Q., Miao, Z., and Gao, L. (2021). An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance. *Expert Systems with Applications*, 169, 114495.

Martin, C.J., Cochran, T.H., and Katsura, S. (1968). Tuberculosis, emphysema, and bronchitis. *American Review of Respiratory Disease*, 97(6P1), 1089–1094.

Miyaji, T. and Ohnishi, I. (2008). Physarum can solve the shortest path problem on riemannian surface mathematically rigorously. *International Journal of Pure and Applied Mathematics*, 47(3), 353–369.

Murphy, K. (1998). A Brief Introduction to Graphical Models and Bayesian Networks. URL `https://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html`.

Nakagaki, T., Yamada, H., and Tóth, (2000). Maze-solving by an amoeboid organism. *Nature*, 407(6803), 470–470.

Natori, K., Uto, M., Nishiyama, Y., Kawano, S., and Ueno, M. (2015). Constraint-based learning bayesian networks using bayes factor. In J. Suzuki and M. Ueno (eds.), *Advanced Methodologies for Bayesian Networks*, 15–31. Springer International Publishing, Cham.

Neapolitan, R.E. (2004). *Learning Bayesian networks*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall.

Oettmeier, C., Nakagaki, T., and Döbereiner, H.G. (2020). Slime mold on the rise: the physics of physarum polycephalum. *Journal of Physics D: Applied Physics*, 53(31), 310201.

Pearl, J. and Mackenzie, D. (2018). *The book of why: the new science of cause and effect*. Basic Books, first edition edition.

Ruiz-Tagle, A., Lewis, A.D., Schell, C.A., Lever, E., and Groth, K.M. (2022). Bantera: A bayesian network for third-party excavation risk assessment. *Reliability Engineering & System Safety*, 223, 108507.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D.A., and Nolan, G.P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721), 523–529.

Schön, T., Stetter, M., Tomé, A.M., Puntonet, C.G., and Lang, E.W. (2014). Physarum Learner: A bio-inspired way of learning structure from data. *Expert Systems with Applications*, 41(11), 5353–5370.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 461–464.

Shen, J., Liu, F., Xu, M., Fu, L., Dong, Z., and Wu, J. (2022). Decision support analysis for risk identification and control of patients affected by covid-19 based on bayesian networks. *Expert Systems with Applications*, 196, 116547.

Stephenson, S.L. and Stempen, H. (2000). *Myxomycetes: a handbook of slime molds*. Timber Press.

Tero, A., Kobayashi, R., and Nakagaki, T. (2006). Physarum solver: A biologically inspired method of road-network navigation. *Physica A: Statistical Mechanics and its Applications*, 363(1), 115–119.

Tero, A., Kobayashi, R., and Nakagaki, T. (2007). A mathematical model for adaptive transport network in path finding by true slime mold. *Journal of Theoretical Biology*, 244(4), 553 – 564.

Uusitalo, L. (2007). Advantages and challenges of bayesian networks in environmental modelling. *Ecological Modelling*, 203(3), 312 – 318.

Vogel, D. and Dussutour, A. (2016). Direct transfer of learned behaviour via cell fusion in non-neural organisms. *Proceedings of the Royal Society B: Biological Sciences*, 283(1845).

Wang, S., Patwary, A., Huang, W., and LO, H.K. (2022). A general framework for combining traffic flow models and bayesian network for traffic parameters estimation. *Transportation Research Part C: Emerging Technologies*, 139, 103664.

Wang, Z., Gao, X., Tan, X., and Liu, X. (2021). Determining the direction of the local search in topological ordering space for bayesian network structure learning. *Knowledge-Based Systems*, 234, 107566.