Communication-free Widened Learning of Bayesian Network Classifiers Using Hashed Fiedler Vectors

Oliver R. Sampson, Christian Borgelt, and Michael R. Berthold

Chair for Bioinformatics and Information Mining
Department of Computer and Information Science
University of Konstanz, Germany

Abstract. *Widening* is a method where parallel resources are used to find better solutions from greedy algorithms instead of merely trying to find the same solutions more quickly. To date, every example of Widening has used some form of communication between the parallel workers to maintain their distances from one another in the model space. For the first time, we present a communication-free, widened extension to a standard machine learning algorithm. By using Locality Sensitive Hashing on the Bayesian networks' Fiedler vectors, we demonstrate the ability to learn classifiers superior to those of standard implementations and to those generated with a greedy heuristic alone.

1 Introduction

Moore's Law has begun to run up against harder physical limits, and parallel processing has taken over the continuing increases in computing performance. Whether it is from multiple cores in potentially multiple CPUs on desktops or thousands of individual cores available in GPGPUs (general-purpose graphics processing units) to seemingly unlimited parallel computing resources available from commercial cloud computing providers, little research [2] has been performed on applying those parallel resources to finding better quality solutions. WIDENING [1] has demonstrated an ability to describe parallelized versions of greedy machine learning algorithms, while using *diversity* between the parallel workers, that are able to find better solutions than their standard counterparts. The guiding philosophy is "Better. Not Faster." Although the demonstrated examples, such as WIDENED KRIMP [24], WIDENED HIERARCHICAL CLUSTERING [11], WIDENED BAYESIAN NETWORKS [25] and BUCKET SELECTION [12] have been able to find superior solutions, i.e., "better," they have been unable to demonstrate this ability in a run-time that is comparable to the standard versions of the greedy algorithms. "Not faster" is not intended to mean "slower."

This is because all of the demonstrated examples have used some form of communication between the parallel workers to enforce a distance between them while they move through the model space. In this paper we present the first example of WIDENING that enables the workers to traverse the model/solution space without communication between them–*communication-free widening*.

Communication-free widening can be realized through the use of a hash function, where each model and its refinements form a potential path through the model space only when they have all been hashed to the same hash value. The sets of models that hash to the same values form a partitioning of the model space and are the mechanism WIDENING uses to maintain diversity between the parallel workers in the model space.

The hash function used here is a variant of a LOCALITY SENSITIVE HASHING [13] hash family and is used to hash the Fiedler vectors of the matrix representations of Bayesian networks which are evaluated for use as classifiers.

2 Background

2.1 Learning Bayesian Networks

A Bayesian network is a probabilistic graphical model that represents conditional dependencies between features of a dataset. More formally, given a dataset, $\mathcal{D}$, with features, $\mathcal{X}$, a Bayesian network, B, is a pair $\langle G, \Theta \rangle$, where $G = \langle \mathcal{X}, \mathcal{E} \rangle$ is a pair representing a directed acyclic graph, where the nodes of the graph are represented by $\mathcal{X}$, $\mathcal{E}$ are the edges, and Θ is the set of conditional probability tables for each of the features. Each edge, $E = \langle X_i, X_j \rangle$, where $E \in \mathcal{E}$ and $X_i, X_j \in \mathcal{X}$, is an ordered pair reflecting the conditional dependency of one feature on another, where a *child node*, X_j, is conditionally dependent a *parent node*, X_i.

Algorithms for learning the structure of Bayesian networks are, at their core, search algorithms that vary in how they score changes to a network's structure along the search path in the super-exponentially sized, i.e., $\mathcal{O}(|\mathcal{X}|!2^{\binom{|\mathcal{X}|}{2}})$, model space. Largely, they vary in the starting network configuration and in the assumptions they make about the relationships between the features of the Bayesian network and in their method of scoring the network. The algorithms can be divided into four categories: *constraint-based*, *search-and-score*, *hybrid*, [15] and *evolutionary algorithms* [17]. Constraint-based algorithms derive network structure based on dependency relationships between features. Search-and-score algorithms refine the network topology by adding, deleting, or reversing the edges in the network and then score and select the network in a greedy manner. Hybrid algorithms integrate techniques from both of the search-and-score and constraint-based methods; a partially-directed-acyclic graph is created using constraint-based techniques, and the network is evaluated using search-and-score methods, while giving a direction to each undirected edge. These methods usually use either Bayesian methods, by calculating the posterior probability of a network given the dataset or likelihood-based information theoretic scores. Evolutionary algorithms follow a typical evolutionary algorithmic pattern of mutation, reproduction, selection, and random sampling, where the classification performance accuracy for classification is often used for the selection (fitness) function.

When using a Bayesian network as a classifier, the calculated probabilities are solely influenced by the *Markov blanket*, i.e., the target variable, its parents, its children, and the other parents of its children [22]. To calculate the predicted target variable value, each value of the target variable is evaluated using a vector, $\mathbf{x}$, of instantiated values for all of the other features in the dataset, i.e., $\mathcal{X} \setminus C$, using Equation 1 [3],

$$\hat{c} = \underset{u=1,\ldots,|C|}{\operatorname{argmax}} P(c_u, \mathbf{x}) = \underset{u=1,\ldots,|C|}{\operatorname{argmax}} P(c_u|\boldsymbol{pa}(C)) \prod_{v=1}^{|\mathbf{x}|} P(x_v|\boldsymbol{pa}(X_v)) \qquad (1)$$

where $C \in \mathcal{X}$ is the target variable, c_u are the values that C may assume, X_v is the variable corresponding to the value $x_v \in \mathbf{x}$, and $\boldsymbol{pa}(\cdot)$ is the set of parents of a given node. The target variable value, $\hat{c}$, with the highest probability is the predicted value.

2.2 Widening

WIDENING is a framework that describes a method for using parallel resources for potentially finding better solutions with greedy algorithms than the algorithms would find using their standard implementation.

Given an initial model, $m_0 \in \mathcal{M}$, from the set of all models in a model space, a refinement operator, $r(\cdot)$, and a selection operator based on a performance metric, $s(\cdot)$, a greedy algorithm can be defined as a series of iterations, $m_{i+1} = s(r(m_i))$, which continues until a stopping criterion is met. More exactly, m_i is refined to a set of derivative models, $\boldsymbol{M}_i' = r(m_i)$, and from this set one model is selected, $m_{i+1} = s(\boldsymbol{M}_i')$. A simple extension to this is BEAM SEARCH, where the top k models are selected at each iteration. The widening framework terms this *Top-k-widening*, i.e., $\boldsymbol{M}_{i+1} = s_{Top-k}(r(\boldsymbol{M}_i)) : |\boldsymbol{M}_{i+1}| = k$. WIDENING begins to widen the search paths beyond a simple greedy mechanism when diversity is brought into play. The notion of diversity can be implemented in either the refining step as in [24,25] or in the selection step as in [11,12]. Given a diverse refinement operator, $r_{\Delta}(\cdot)$, as in [24,25], where a diversity function, Δ, is imposed on the output, DIVERSE TOP-K WIDENING is described by $\boldsymbol{M}_{i+1} = s_{Top-k}(r_{\Delta}(\boldsymbol{M}_i))$.

Depending on how this diversity is imposed, it can either be communication-free or not. WIDENED KRIMP evaluated the best models from all parallel workers at the end of each iteration. The P-DISPERSION-MIN-SUM measure [21] is used by WIDENED BAYESIAN NETWORKS to find maximally disperse, i.e., diverse, members of the refined sets at each refinement step. WIDENED HIERARCHICAL CLUSTERING, in contrast, uses a clustering method to find diverse subsets and selected a top member from each of the clusters. BUCKET SELECTION uses a hashing mechanism and transfers models between the parallel workers. All four examples require non-parallelized communication between the parallel workers for a comparison of their results.

2.3 Communication-free Widening

Communication-free widening can be thought of as a partitioning of the model space, where a refinement operator either yields only models that are part of the partition, or yields models to many partitions and discards those that do not belong to the partition.

Many problems have large model spaces that are characterized as having large plateau-like structures that are difficult for greedy algorithms to progress within. Often there are many local optima distributed throughout the solution space. Given a partitioning of such a model space, which restricts the search of each parallel worker within a partition (See Figure 1), we hypothesize (1) that as the number of partitions increases, i.e., as the width increases, that WIDENING will be able to find better solutions, and (2) that with too many partitions, the solutions will deteriorate as more of the partitions do not cover a better solution or will not cover a complete path to a solution.

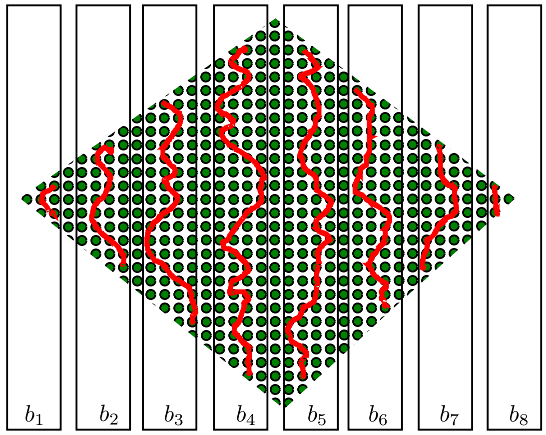


Fig. 1: Solution paths (red) are limited to regions of the model space defined by the output of a hash function, i.e., buckets, denoted by $b_1, \ldots, b_8$.

Partitioning the model space introduces a potential problem to the greedy search–*reachability*. The problem of reachability describes when the parallel worker is unable to find any better solution in a partition or, in the extreme case, any better solution in any partition. It also describes when a good or best solution path is not *complete* under any given partition and would need to "jump" partitions.

A hash function, $H(\cdot)$, is a natural method for partitioning a model space. (See Figure 1.) The refined models are hashed with $H(\cdot)$, where refined models with different hash values from the original model are discarded. The best of these models are in turn selected by a selection operator for the next iteration. Continuing with the notation from above, each parallel path, denoted by j, is described by WIDENING as $\boldsymbol{M}_{j,i+1} = s(\{m' \in r(\boldsymbol{M}_{j,i}) | H(m') = j\})$ where j is the output of the hash function $H(\cdot)$.

2.4 Locality Sensitive Hashing

LOCALITY SENSITIVE HASHING (LSH) has shown excellent results in similarity search, with applications in image retrieval [16], genetic sequence comparison [5], melody-based audio retrieval [20], among others. LSH reduces the dimensionality of a dataset by hashing the dataset's entries with a hash function, $h(\cdot) \in \mathcal{F}$, from a hash family $\mathcal{F}$, which has a high probability of giving the same value to similar examples [6].

Several different hash families, $\mathcal{F}$, are found in the literature including datapoints on a unit hypersphere [28], angle-based distance [6], and p-stable distributions [9].

In the R-NEAREST NEIGHBOR (R-NN) problem, LSH is used with a number, L, of sets of concatenated hash functions from $\mathcal{F}$ in order to increase the probability of a collision between a query example and examples already hashed from the database. Typically, the results of g different examples of each hash function, $\{h_1(\cdot), \ldots, h_g(\cdot)\}$, are concatenated together to create a hash value for one hash function, $H(\cdot)$. L hash functions are used to hash each example, $x \in \mathcal{D}$, into L hash tables. When there is a collision between examples, a collision list is kept for each hash entry in the appropriate hash table. When searching for R-NN examples, a query item x_q is hashed using each of the L hash functions, and the previously bucketed values from each of L hash tables are retrieved. These are then compared to x_q with a standard similarity measure. Those falling within some distance, r, of x_q are considered to be matches for R-NN [9].

The L2 Gaussian hash family [9] hashes an example, $\mathbf{v}$, of dimension d using the function $h(\mathbf{v}) = \left\lfloor \frac{\mathbf{v}\cdot\mathbf{a}+b}{w} \right\rfloor$, where $\mathbf{a}$ is a d-dimensional vector whose elements are randomly sampled from a Gaussian distribution with $\mu = 0$ and $\sigma = 1$ and b is a value randomly sampled from a linear distribution between $[0, w)$, where w is an input parameter. g different examples of $h(\cdot)$ derived from g different vectors $\mathbf{a}$ and values b are concatenated together to compose $H(\cdot)$. In this work we only use one ($L = 1$) set of hash functions, as opposed to the larger number used for R-NEAREST NEIGHBOR in [9], because we are only interested in testing a single partitioning of the model space.

2.5 Fiedler Vectors

An *adjacency matrix*, $\mathbf{A}$, of an undirected network graph is defined to be an $n \times n$: $n = |\mathcal{X}|$ matrix with entries, $a_{ij} \in \{0, 1\} : i, j \in \{1, \ldots, |\mathcal{X}|\}$. a_{ij} is set to 1 if there is an edge between nodes i and j in the network, and to 0 where no edge exists. A *node degree matrix*, $\mathbf{D}$, is an $n \times n$ diagonal matrix where the diagonal, i.e., the entries d_{ii}, are set to the number of edges incident to the node $i \in \{1, \ldots, |\mathcal{X}|\}$; all other entries are set to 0. The unnormalized *Laplacian matrix* of the undirected graph is simply $\mathbf{L} = \mathbf{D} - \mathbf{A}$, which, when $deg(i)$ is the node degree, gives [7]:

$$\mathbf{L}_{UN} = \begin{cases} deg(i) & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } X_i, X_j \text{ are adjacent,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Normalized Laplacian matrices exist, such as the *symmetric norm* from Chung [7]

$$\mathbf{L}_{SN} = \begin{cases} 1 & \text{if } i = j \text{ and } deg(i) \neq 0, \\ -\frac{1}{\sqrt{deg(i)deg(j)}} & \text{if } i \neq j \text{ and } X_i, X_j \text{ are adjacent,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

and the *random walk* from Doyle and Snell [10], $\mathbf{L}_{RW}$, which differs from that of Chung by the value for adjacent nodes: $-\frac{1}{deg(i)}$.

The eigenvalues of symmetric matrices are both real and positive, with the number of eigenvalues equal to 0 reflecting the number of connected components in the graph. The *Fiedler vector* is the eigenvector associated with the second smallest, i.e., first non-zero, eigenvalue, or *Fiedler value*, of a connected graph's Laplacian matrix [7]. The Fiedler value is associated with a graph's algebraic connectivity; the Fiedler vector reflects graph's structure, in that graphs cannot be isomorphic if they do not have the same Fiedler vector. If the graphs do have the same Fiedler vector, then the probability of their being isomorphic is very high and seems to trend to 100% as $|\mathcal{X}| \to \infty$ [29].

Assuming the property that the Fiedler vector approximates a unique identifier for a graph,[1] we hypothesize (3) that the LSH function as described above, when hashing

[1] Isomorphic graphs have similar Fiedler vectors. The converse is not necessarily true.

Dataset	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{X}\rvert$	$\lvert C\rvert$	max $\lvert H(\cdot)\rvert$
car	1728	7	4	22
connect4	67556	43	3	29
ecoli	336	8	8	23
glass	214	10	7	24
ionosphere	351	35	2	28
pima	768	9	2	24
waveform	5000	22	3	28

Table 1: Dataset Characteristics. $|\mathcal{D}|$ is the number of entries in the dataset, $|\mathcal{X}|$ is the number of features including the target feature, $|C|$ is the number of target classes, and max $|H(\cdot)|$ is the width where refined models are more likely to be refined to the same hash than random, obtained from the experiments as depicted in Figures 3c and 3d.

the Fiedler vector allows similar Bayesian networks to stay together in a partition, and that the disparate refinement paths will lead to a superior solution, thereby realizing COMMUNICATION-FREE WIDENING.

2.6 Related Work

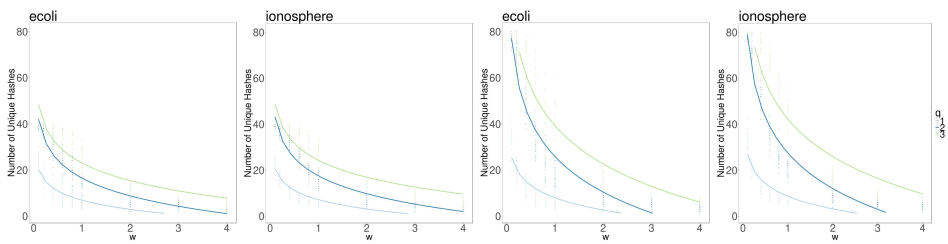
This work relies implicitly on work related to the SUBGRAPH ISOMORPHISM PROBLEM, which is an area of active research into efficient methods for finding common subgraphs. The use of graph spectra is a popular method with applications in clustering [26], chemistry [30], and image retrieval [23], among others. Luo et al. in [19] used spectral properties with other graph theoretical values for graph clustering. Qiu and Hancock in [23] used graph spectral properties, and the Fiedler vector in particular, for graph matching by decomposing a graph into subgraphs.

Zhang et al. in [31] used LSH on graphs for K-NEAREST NEIGHBOR SIMILARITY SEARCH. Their method is based on using a hash function of differences between graphs in the database and prototypes either randomly selected beforehand or calculated by clustering. Variants of LSH exist that use only one hash function, such as the Single Hash MINHASH [4]. To our knowledge no examples exist in the literature of implementing LSH for graphs with the Fiedler vector as the value to be hashed.

3 Experimental Setup

The datasets used for the experiments were chosen for their wide variety of dimensionality and number of target classes and for their lack of missing values (See Table 1). They are all available from the UCI Machine Learning Repository [18] and were discretized using the LUCS-KDD DN software [8].

Each experiment tested the response to WIDENING by varying the input variables w, which controls the number of different output values for each function $h_g(\cdot)$ and g, which is the number of hash values concatenated together. Experiments with every combination of w and g for each dataset were conducted using 5-fold cross-validation



(a) (b) (c) (d)

Fig. 2: Number of hash values related to w and g. The number of different initial hash values measured for two datasets (`ecoli` and `ionosphere`) with two different sizes of sets of initial models, $|\boldsymbol{M}_0| = 40$ in Figures 2a and 2b, and $|\boldsymbol{M}_0| = 80$ in Figures 2c and 2d, are plotted with values for $g \in \{1, 2, 3\}$ and $w \in \{0.1, 0.2, 0.4, 0.8, 1.0, 2.0, 3.0, 4.0\}$. Small values of w result in a large number of hashes quickly approaching the number of initial values.

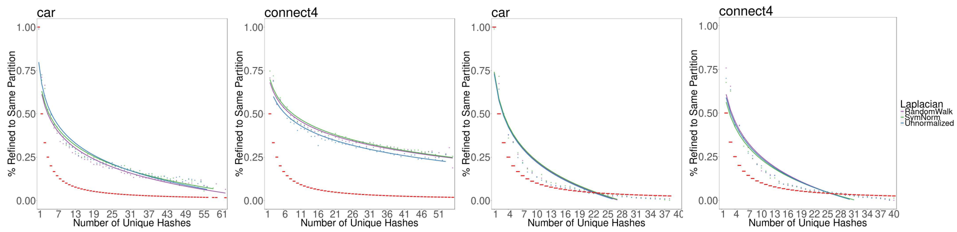
and repeated five times, resulting in 25 individually scored trials with different random values for the hash functions $h(\cdot)$ for each trial. The 5-fold cross-validation is naturally an 80/20 train/test split. The iterative refine-score-select steps in each of the five training folds are also learned and scored using an 80/20 partitioning, resulting in an overall 64/16/20 train/validate/test split. All experiments were conducted using KNIME v3.5.

3.1 Initialization

In the experiments presented here, the initial models are single-component networks with up to two edges being added from every node to other random node(s). All experiments were performed with an initial model set, $\boldsymbol{M}_0$, with $|\boldsymbol{M}_0| = 40$ initial models and with $w \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.5, 2, 3, 4\}$ and $g \in \{1, 2, 3\}$. For each initial model, the Fiedler vector for the Markov blanket is calculated and hashed. The number of initial hashes for the experiment is determined from the initial set of hashed values.

Because of the stochastic nature of the hashing scheme, it is impossible to predict exactly how many different partitions will be created from the initial set of models, but we can measure the response for the number of generated hash values. What is certain is that, in our application as w decreases and g increases there will be a tendency for the number of hash values to increase (See Figure 2).

For applications where the exact width needs to be known beforehand, several different rounds of initialization are performed, and the round with the model(s) with the correct number of unique hashes is used as the starting point for WIDENING. The primary relationship being evaluated here is that between the amount of widening, i.e., the number of unique hash values, and the resulting classification performance of the derived Bayesian networks. When there is no widening, i.e., there is no hashing and partitioning of the model space, the refined model path is a simple, greedy search.



(a) (b) (c) (d)

Fig. 3: Percentage of refined models with Fiedler vector/LSH to the same hash value for two datasets: `car` and `connect4` and three Laplacian normalizations: SymNorm (green), RandomWalk (purple), Unnormalized (blue). Figures 3a and 3b show the percentage of models refined to the same partition when comparing the Fiedler vector for the entire network. Figures 3c and 3d show the percentage of models refined to the same partition when using the Fiedler vector only for the network's Markov blanket.

3.2 Refinement

Because the Markov blanket is the portion of the network that, when changed, can cause changes in classification accuracy, the refinement strategy first attempts to add or delete edges from non-Markov blanket nodes to the Markov blanket, depending on the constraints of the network's being acyclic and a single component. If that fails, similar attempts for any edge in the network are made. Because this work is interested in demonstrating WIDENING via the use of the Fiedler vector as a good hashable descriptor of a Bayesian network, and how its use with an LSH-based hashing scheme will find better solutions than standard greedy algorithms, only one model is refined per iteration—this corresponds to the use of $l = 1$ in [25]. The number of parents for any given node in a model is limited to 5, because conditioned probabilities can degrade to 0 for datasets where $|\mathcal{D}|$ is insufficiently large.

The Fiedler vector for the refined model is filled with zeroes for the nodes that are not included in the Markov blanket, and hashed using g concatenated values of $h(\cdot)$. Any refined model with a hash value differing from its preceding model is discarded. The preceding model may have a differing set of nodes in the Markov blanket.

3.3 Partitioning

To determine the efficacy of the Fiedler vector/LSH method of refining models within the same partition, we performed some preliminary experiments. Twenty-five repetitions of $|\boldsymbol{M}_0| = \{40, 80\}$ initialized models were refined through 50 iterations as described above. The new model's hash value is compared to the previous model's hash value. When equal, the new model is kept for further refinement; when unequal, it is discarded. No scoring of the resulting Bayesian network is performed, so in this case, the only difference considered between datasets is the number of features.

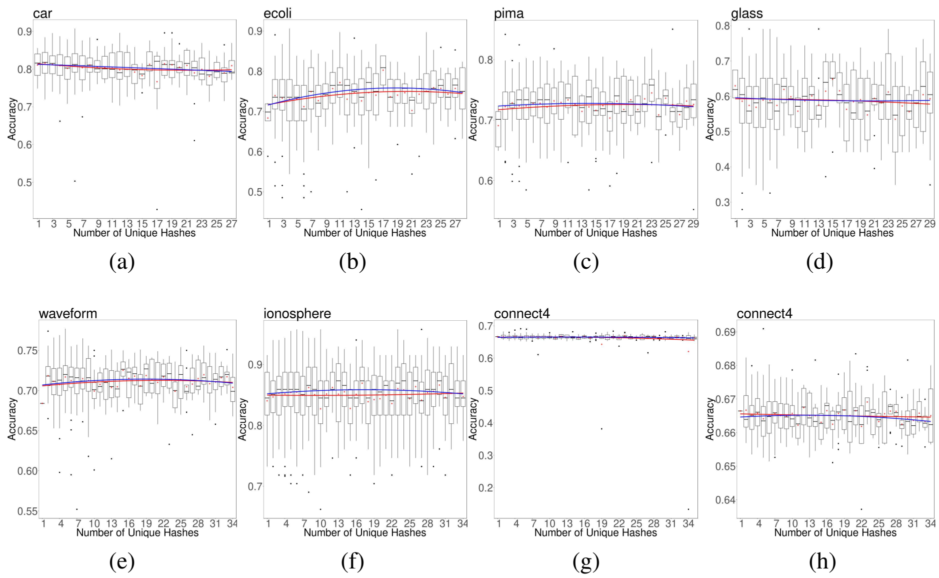


Fig. 4: COMMUNICATION-FREE WIDENED BAYES accuracy versus the number of unique hashes. The red and blue lines are second degree polynomials fitted to the mean (red dots) and median accuracy, respectively, for each value of the number of unique hashes (width). When the lines are concave facing down, it supports the hypothesis of better performance with WIDENING to a certain point with worsening performance thereafter. `connect4` is shown twice (Figures 4g and 4h, once each with and without outliers ($\sigma \geq 3$), to better show the trend.) The x-axis shows the max number of hashes from Table 1 plus 20% thereof allowing for a decline in accuracy afterwards.

Figure 3 shows how well the Fiedler Hash/LSH technique performs with refining models to the same hash value for three different types of normalization for the Laplacian matrix compared to a $1/n$ baseline, which would be expected with a purely random hashing scheme. Two different Fiedler vector/LSH hashing schemes are shown in Figure 3 to illustrate the effect of using just the Markov blanket compared to the entire network. In the cases (Figures 3a and 3b) where the Fiedler vector from the entire network is hashed, the larger datasets have a higher number of hashes for which the models are refined to the same partition, and a higher number of hashes which perform better than the baseline. This is because small perturbations to the larger network can have smaller effects on the Fiedler vector. In the cases (Figures 3c and 3d), where only the Markov blanket is considered, the Markov blanket is (usually) smaller than the total network, and small changes may eliminate a node or nodes entirely from the Markov blanket resulting in larger changes to the Fiedler vector and its hash value. The crossover to performance worse than the baseline is between 23 hash values for smaller datasets, e.g., `car` and 29 for larger datasets, e.g., `connect4`. This value (max $|H(\cdot)|$ in Table 1) is used for the maximal widening in later experiments.

Dataset	R:HILL-CLIMBING	R:MMHC	R:TABU	GREEDY	COMM.-FREE WIDENED BAYES	Best number of Partitions	p-value
`car`	0.715 ± 0.037	0.700 ± 0.002	0.718 ± 0.035	0.682 ± 0.125	$\mathbf{0.816 \pm 0.029}$	2	< 0.01
`connect4`	0.678 ± 0.012	0.658 ± 0.000	$\mathbf{0.684 \pm 0.002}$	0.589 ± 0.152	0.669 ± 0.006	23	< 0.01
`ecoli`	0.632 ± 0.044	0.495 ± 0.088	0.602 ± 0.109	0.677 ± 0.100	$\mathbf{0.803 \pm 0.032}$	17	< 0.01
`glass`	0.501 ± 0.112	0.388 ± 0.036	0.500 ± 0.057	0.532 ± 0.107	$\mathbf{0.649 \pm 0.079}$	15	< 0.01
`ionosphere`	0.807 ± 0.055	0.641 ± 0.011	0.810 ± 0.057	0.826 ± 0.055	$\mathbf{0.869 \pm 0.037}$	17	< 0.01
`pima`	0.706 ± 0.053	0.716 ± 0.065	$\mathbf{0.760 \pm 0.027}$	0.693 ± 0.050	0.745 ± 0.051	23	< 0.01
`waveform`	0.504 ± 0.119	0.339 ± 0.000	0.612 ± 0.016	0.630 ± 0.058	$\mathbf{0.725 \pm 0.017}$	15	< 0.01

Table 2: Experimental results comparing simple greedy search (one partition) to the best results from COMMUNICATION-FREE WIDENED BAYES and three algorithms from the **R** `bnlearn` package. The p-values are for Student's t-test, two-tailed, 95% confidence level with equal variances assumed, comparing COMMUNICATION-FREE WIDENED BAYES NETWORKS to the purely greedy variant.

Additionally, the different Laplacian matrix normalizations described in Section 2.5 were compared with an unnormalized Laplacian matrix in these experiments. The three different types of Laplacian matrix normalization performed similarly to one another, but Chung's $\mathbf{L}_{SN}$ (See Equation 3) slightly yet consistently outperformed $\mathbf{L}_{RW}$ and $\mathbf{L}_{UN}$ (See Equation 2), and is therefore used in the classification evaluation experiments.

3.4 Scoring and Selection

At each iteration, the model is scored using 20% of the training data subset. For WIDENING in general, the best models are selected, but, here only a single model is being evaluated—this corresponds to the use of $k = 1$ in [25]. If the performance score is better than that of the model from the previous iteration, the model is passed into the next iteration, otherwise the old model is kept and refined anew. The iterations stop when the improvement in performance is less than 0.01%. A Laplacian correction of 1 is added to the entries in the conditional probability table when a count is 0.

4 Results

A summary of the experimental results for the seven datasets is shown in Table 2. COMMUNICATION-FREE WIDENED BAYES was able to find superior solutions when compared to three standard Bayesian network learning algorithms (HILL-CLIMBING (both `perturb` and `restart = 100`)), MAX-MIN HILL-CLIMBING (MMHC) (`perturb` = 100), and TABU from the **R** `bnlearn` v4.2 package [27]) for five of seven datasets. However, for all seven datasets COMMUNICATION-FREE WIDENED BAYES was able to demonstrate, as hypothesized, finding better solutions when compared to a purely greedy learning method.

As depicted in Figure 4, five of the seven datasets, (`ecoli`, `pima`, `waveform`, `ionosphere`, and `connect4`) show the predicted curves for both the mean and the median with the exception of `ionospheres`'s mean. `ecoli` and `pima` show the clearest examples whereas `glass` shows a sharp peak in the middle that the smoothing lines oversmooth. `connect4` shows minimal variation in response to WIDENING, and its results from the different algorithms differ relatively little, indicating that good solutions are relatively easy to find along the solution surface; we do not expect all datasets to respond equally well, either because of the nature of the dataset, or because of the reachability problem described in Section 2.2. `car` found the best solutions with only two partitions, but, like `connect4` showed little variability overall.

5 Conclusion and Future Work

The results demonstrate for the first time the successful implementation of a communication-free, widened version of a class of popular machine learning algorithms. Additionally, the experiments compared two methods of normalizing the Laplacian matrix and the unnormalized Laplacian matrix, and while no large differences between the three were found, Chung's symmetric normalization [7] slightly outperformed the other two. The results verify the Fiedler vector as a viable descriptor of mixed-sized Markov blankets from Bayesian networks for use with LOCALITY SENSITIVE HASHING.

A drawback to these experiments is the use of the undirected adjacency matrix for calculating the Laplacian matrix. Hashing the complex values that are the result of the eigendecomposition of skew-symmetric adjacency matrices or of a Hermitian adjacency matrix [14], or even of variations to the Laplacian matrix calculated with them could result in a stricter partitioning. Furthermore, within each hash region, a *Top-k* could be used to find slightly better models at each refinement step, thereby accelerating the search. Schemes that affect the refining step, such as preventing an edge that has contributed to a better score from being deleted in the next refining step, could also speed up the search. Experiments involving other LSH hash families could also be useful.

References

1. Zaenal Akbar, Violeta N. Ivanova, and Michael R. Berthold. Parallel data mining revisited. Better, not faster. In *Proceedings of the 11th International Symposium on Intelligent Data Analysis*, pages 23–34, 2012.
2. Selim G. Akl. Parallel real-time computation: Sometimes quantity means quality. In *Proceedings International Symposium on Parallel Architectures, Algorithms and Networks, 2000. I-SPAN 2000.*, pages 2–11. IEEE, 2000.
3. Concha Bielza and Pedro Larrañaga. Discrete Bayesian network classifiers: a survey. *ACM Computing Surveys (CSUR)*, 47(1):5, 2014.
4. Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of Sequences 1997*, pages 21–29. IEEE, June 1997.
5. Jeremy Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.
6. Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on theory of computing*, pages 380–388. ACM, 2002.
7. Fan-Roon Kim Chung. *Spectral Graph Theory*. Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, 1997.

8. Frans Coenen. LUCS-KDD DN software, 2003.
9. Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.
10. Peter G. Doyle and J. Laurie Snell. *Random walks and electric networks*. Mathematical Association of America,, 1984.
11. Alexander Fillbrunn and Michael R. Berthold. Diversity-driven widening of hierarchical agglomerative clustering. In *Advances in Intelligent Data Analysis XIV*, pages 84–94. Springer, October 2015.
12. Alexander Fillbrunn, Leonard Wörteler, Michael Grossniklaus, and Michael R. Berthold. Bucket selection: A model-independent diverse selection strategy for widening. In *Proceedings of the 16th International Symposium on Intelligent Data Analysis (IDA 2017)*, 2017.
13. Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.
14. Krystal Guo and Bojan Mohar. Hermitian adjacency matrix of digraphs and mixed graphs. *Journal of Graph Theory*, 85(1):217–248, 2017.
15. Timo J. T. Koski and John M. Noble. A review of Bayesian networks and structure learning. *Mathematica Applicanda*, 40(1):53–103, 2012.
16. Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *12th International Conference on Computer Vision*, pages 2130–7. IEEE, 2009.
17. Pedro Larrañaga, Hossein Karshenas, Concha Bielza, and Roberto Santana. A review on evolutionary algorithms in Bayesian network learning and inference tasks. *Information Sciences*, 233:109–125, 2013.
18. Moshe Lichman. UCI Machine Learning Repository, 2013.
19. Bin Luo, Richard C. Wilson, and Edwin R. Hancock. Spectral feature vectors for graph clustering. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2002.
20. Matija Marolt. A mid-level representation for melody-based retrieval in audio collections. *IEEE Transactions on Multimedia*, 10(8):1617–1625, December 2008.
21. Thorsten Meinl. *Maximum-Score Diversity Selection*. PhD thesis, University of Konstanz, Konstanz, Germany, July 2010.
22. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.
23. Huaijun Qiu and Edwin R. Hancock. Graph matching and clustering using spectral partitions. *Pattern Recognition*, 39(1):22–34, 2006.
24. Oliver Sampson and Michael R. Berthold. Widened KRIMP: Better performance through diverse parallelism. In *Advances in Intelligent Data Analysis XIII*, volume 8819 of *Lecture Notes in Computer Science*, pages 276–285. Springer, October 2014.
25. Oliver R. Sampson and Michael R. Berthold. Widened learning of Bayesian network classifiers. In *Advances in Intelligent Data Analysis XV*, pages 215–225. Springer, October 2016.
26. Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.
27. Marco Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.
28. K Terasawa and Y Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. In *Workshop on Algorithms and Data Structures*, pages 27–38. Springer, 2007.
29. Edwin R Van Dam and Willem H. Haemers. Which graphs are determined by their spectrum? *Linear Algebra and its applications*, 373:241–272, 2003.
30. Saraswathi Vishveshwara, KV Brinda, and N Kannan. Protein structure: insights from graph theory. *Journal of Theoretical and Computational Chemistry*, 1(01):187–211, 2002.
31. Boyu Zhang, Xianglong Liu, and Bo Lang. Fast graph similarity search via locality sensitive hashing. In *Pacific Rim Conference on Multimedia*, pages 623–633. Springer, 2015.