Modelling Ocean Parameters Through Graphical Models

Felix Schneider, Yanira Guanche Garcia, Joachim Denzler
Computer Vision Group
Friedrich Schiller University Jena
Jena, Germany
Email: felix.schneider@uni-jena.de

***Abstract*—Ocean parameter modelling is an important task for many fields. While using simulations and simple statistical models may not yield desired results in reasonable time, using graphical models like Bayesian networks can address this problem. In this paper, we show the application of Bayesian networks to the tasks of estimating and predicting significant wave heights in the North Sea. Additionally we present the K2 IO algorithm, a modification to the K2 algorithm developed for the prediction task. Experiments show the possibilities and problems of estimation and prediction with Bayesian networks. They also show that the K2 IO algorithm produces a structure that is suitable for prediction in a shorter time than the K2 algorithm.**

I. Introduction

Research on environmental processes is vital for a wide range of modern applications in engineering and sciences. The need for knowledge about these processes is present in research related to climate change, renewable energies and coastal defence planning as well as flood and storm risk assessment. All of these areas contain spatio-temporal multivariate problems. On the one hand, these problems are too complex to model with basic statistical tools. On the other hand a detailed simulation is often not reasonable, because this takes too much time to get the outcomes or there is not enough process knowledge to create a model with plausible results.

In many scientific fields methods of computer vision and machine learning have acquired great relevance due to the strong improvement in computational capabilities and the increasing amount of available data. While these methods are already widely applied in many industrial branches and in medical sciences, there remain many possible applications in fields like environmental sciences.

The fields of computer vision and machine learning encompass a multitude of techniques and models. Many of them are fitting for multivariate data as well as for modelling time series. Especially the methods from computer vision are applicable for spatial data, as a grid filled with information can be processed in the same way as an image. Graphical models and in this case Bayesian networks are a widely used machine learning method, that combines graph theory and probability theory to model systems that are too complex to describe with simple statistical methods.

Ocean parameter modelling is a part of environmental sciences. The field is characterised by complex interactions of different factors as the *significant wave height* and the *wind speed* at different locations. Creating a simulation to accurately estimate and predict wave heights takes a lot of time and expert knowledge. This paper showcases that creating Bayesian networks and training them on ocean data can produce models that are capable of accurately estimating and predicting wave heights.

One of the subfields of Bayesian networks is the creation of the graph structure. One possibility is letting experts design the network structure by hand, but knowledge and the time of experts is needed to do so. Another possibility is learning the network structure from data. As this is a time consuming task, this paper proposes a method called the *K2 IO algorithm* that decreases the time of the common *K2* structure learning algorithm when used for prediction.

In section II we present an introduction to Bayesian networks. Section III explains our modification to a common Bayesian network structure learning algorithm. The application and experiments we did are described together with the data in section IV-A and section IV-B. Conclusions and suggestions for further work are presented in section V.

II. Bayesian Networks

Probabilistic graphical models combine graph theory and probability theory. [KF09] Bayesian networks are a type of graphical models that use a *directed acyclic graph (DAG)* to model the dependencies between random variables. The *conditional probability distribution (CPD)* $P(x)$ is dependent on the parents of the random variable in the DAG pa_x:

$$p(x) = p(x|pa_x) \tag{1}$$

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|pa_{x_i}) \tag{2}$$

A random variable in a Bayesian network can be discrete or continuous.

In the discrete case, the CPD can be stored using a *tabular CPD*, where every configuration x_i of the random variable X with k being the number of its configurations has a probability $P(x_i)$ with $P(x_i) \geqslant 0 \forall i \in \{1, ..., k\}$ and $\sum_{i=1}^{k} P(x_i) = 1$ assigned.

In the continuous case, the CPD can store the parameters of the distribution function of the random variable. In the case of a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ this would be the mean value μ and the variance σ^2.

A discrete random variable with discrete parents can be represented as a joint probability table of all combinations of configurations of the parents, while the parameters of a continuous random variable with continuous parents can be represented as a function of the outcomes of the parents. If a continuous random variable has discrete and continuous parents, a combination of these methods can be used. For every combination of configurations of the discrete parents, the CPD can be represented by a distribution function whose parameters are functions of the outcomes of the continuous parents.

The most common case of these mixed nodes is the *conditional linear Gaussian CPD (CLG CPD)*. For a continuous random variable X with the discrete parents $D = \{D_1, ..., D_m\}$ and the continuous parents $C = \{C_1, ..., C_k\}$ a *CLG CPD* has for every configuration d of D a set of $k+1$ coefficients $\mu, w_{d,1}, ..., w_{d,k}$ and the covariance σ^2 the form:

$$p(X|d,c) = \mathcal{N}(\mu + \sum_{i=1}^{k} w_{d,i}c_i;\ \sigma^2) \tag{3}$$

A. Inference

Given a Bayesian network $\mathcal{B}$, with the random variables X, a set of known evidence nodes E, a set of queried nodes Q, and the unobserved nodes $U = X \setminus (E \cup Q)$, we can formulate a query as:

$$P(Q|E=e) = \frac{P(Q,e)}{P(e)} \tag{4}$$

In the easiest way and in case of a fully discrete Bayesian network, this can be solved by creating the joint probability table over the whole Bayesian network and summing out the unobserved variables:

$$P(q,e) = \sum_{U} P(q,e,u) \tag{5}$$

$$P(e) = \sum_{Q} P(q,e) \tag{6}$$

However, creating the full joint probability table is not only undesired, but in the case of Gaussian or mixed Bayesian networks even impossible. To solve this problem, there exists a variety of different exact and approximate inference algorithms. The most basic algorithm that exploits the graph structure of the Bayesian network is the *variable elimination* algorithm. The basic idea of this algorithm is to follow the graph structure of the Bayesian network and to sum out the variables along the way. This can be done not only in discrete models but also in Gaussian and mixed Bayesian networks. While Bayesian network inference is still $\mathcal{NP}$*-hard*, in most application cases the problem of inference can be solved in reasonable time.

B. Parameter Learning

One of the strengths of the Bayesian network model is that its parameters can be learned from data. As the CPDs can consist of parametrised distribution functions, these distributions can also compensate for cases where only small training datasets are available.

If the training data D is *complete*, meaning that in every sample the value of every random variable is given, the parameters can be learned by using *maximum likelihood estimation*.

The *likelihood* $\mathcal{L}$ of a distribution $P(D:\theta)$ over the data D with the parameters θ from the *hypothesis space* Θ is defined as:

$$\mathcal{L}(\theta : D) = P(D : \theta) \tag{7}$$

The likelihood describes how good parameters fit for a certain distribution over data. Consequently the *maximum likelihood* is defined as:

$$\mathcal{L}(\hat{\theta} : D) = \max_{\theta \in \Theta} \mathcal{L}(\theta : D) \tag{8}$$

According to our definition of the Bayesian network in equation 2, we can split the likelihood of the random variables $x \in X$ into:

$$\mathcal{L}(\theta : D) = \prod_{x \in X} P(x|pa_x : \theta_x) \tag{9}$$

This allows us to estimate the maximum likelihood for every random variable separately by varying the parameters of the CPD of the variable.

C. Structure Learning

While the graph structure of a Bayesian network can be learned by hand, it was also mentioned that learning the structure from data is possible. The *K2 algorithm* is a simple and greedy algorithm that searches for parents for each node separately [CH92]:

```
1: K2 algorithm
2: Input: A set of n random variables, the maximum number
   of parents u, the training dataset D with m samples.
3: Output: A list of parents for each random variable.
4: for $i \leftarrow 1$ to $n$ do
5:     parents(i) := $\emptyset$
6:     score_old := score(i, parents(i))
7:     potential_ps := all random variables before i
8:     while potential_ps $\neq \emptyset$ do
9:         score_new := score_old
10:        for $p \in potential\_ps$ do
11:            if score(i, parents(i) $\cup\{p\}$) $>$ score_new then
12:                score_new := score(i, parents(i) $\cup\{p\}$)
13:                new_p := p;
14:            end if
15:        end for
16:        if score_new $>$ score_old then
17:            parents(i) := parents(i) $\cup$\{new_p\}
18:            potential_ps := potential_ps $\setminus\{new\_p\}$
```

```
        end if
        if score_new == score_old or |parents(i)| == $u$ then
            potential_ps := $\emptyset$
        end if
    end while
end for
return parents
```

To avoid cycles the K2 algorithm takes a list of ordered nodes. For every node X_i the set of potential parents $P_{\text{pot},i}$ is the previous nodes $X_1, ..., X_{i-1}$ in the list. The algorithm assigns a score to every potential parent and adds the node with the best score to the set of parents P_i. This step is repeated until no potential parent can improve the score or a defined maximum number of parents is reached. Compared to other scoring algorithms the K2 algorithm is fast, but with a growing number of nodes and a high maximum number of parents this algorithm still takes significant time as shown in section IV-B.

III. K2 IO

A Bayesian network can also be trained for prediction tasks. To do this, we can use a set of nodes for the past and one set for the prediction. The DAG for this can be trained with the K2 algorithm. However, when training the DAG for the prediction, the algorithm usually creates connections that are not necessary for this task. On the one hand connections between the past nodes are created. As the values for the past nodes are always given as evidence when predicting values, this is not necessary. On the other hand connections between prediction nodes are created. This could influence the results as the wave heights at one time step are always correlated to some point. However, these connections are not mandatory, because these influences are already captured by choosing the parent locations in the past set, especially for short term prediction.

To use this information we modified the K2 algorithm so that only connections between past and prediction nodes were allowed. This resulted in the *K2 IO* algorithm.

```
K2 IO algorithm
Input: A set of random variables divided in an input set of k random variables rv_input, one output set of n random variables rv_output, the maximum number of parents u, the training dataset D with m samples.
Output: A list of parents for each random variable.
for $i \leftarrow k+1$ to $k+n$ do
    parents(i) := $\emptyset$
    score_old := score(i, parents(i))
    potential_ps := rv_input
    while potential_ps $\neq \emptyset$ do
        score_new := score_old
        for $p \in potential\_ps$ do
            if score(i, parents(i) $\cup\{p\}$) $>$ score_new then
                score_new := score(i, parents(i) $\cup\{p\}$)
                new_p := p;
            end if
        end for
        if score_new $>$ score_old then
            parents(i) := parents(i) $\cup$\{new_p\}
            potential_ps := potential_ps $\setminus\{new\_p\}$
        end if
        if score_new == score_old or |parents(i)| == $u$ then
            potential_ps := $\emptyset$
        end if
    end while
end for
return parents
```

In contrast to the normal K2 algorithm, this algorithm takes two sets of nodes and no ordering. One set of nodes is the *input* set and another is the *output* set, hence the name *IO*. The algorithm then searches for parents only for the output nodes. The set of potential parents for each node is the set of input nodes. This way, every output node is guaranteed to be connected only to input nodes and the time needed to search for connections is reduced.

IV. Case Study: Wave Height at the North Sea

In this section we show how we tested the application of Bayesian networks and the new K2 IO algorithm on North Sea data.

A. Data

The data we used was the CoastDat-1 dataset [HZG12]. This dataset encompasses various variables on a grid with hourly observations on points in the distance of approximately 5 kilometres from 1958 to 2007 in the North Sea.

We chose 12 locations to observe as shown in Figure 2. Three offshore locations were chosen in places with high maximum wave height and high variance, 9 coastal locations were chosen near cities at the North Sea coast. The cities are Calais, Antwerp, the Hague, Amsterdam, Groningen, Bremerhaven, Cuxhaven, Husum and Esbjerg. In all experiments we used 20% of the data for testing and 80% for training. The data was divided in alternating segments of one year of testing data and 4 years of training data.

You can see the locations in Figure 2. The distribution of the wave heights are shown in Figure 1. After logarithmisation, the wave distributions mostly followed a Gaussian distribution. Notable exceptions from this are especially the locations near Amsterdam, Bremerhaven and Cuxhaven. In these cases the water was shallow compared to the other locations which led to big waves being broken and becoming small waves. In the distribution plot this can be seen very well as the wave height density rises left to the supposed Gaussian mean value.

For the discrete experiments we discretised the data into 5 bins using Lloyd's algorithm. This algorithm is used in various fields of computer science for discretisation of continuous values and iteratively minimises a quadratic noise function to find fitting bin centres and borders [Llo06]. As the common wave

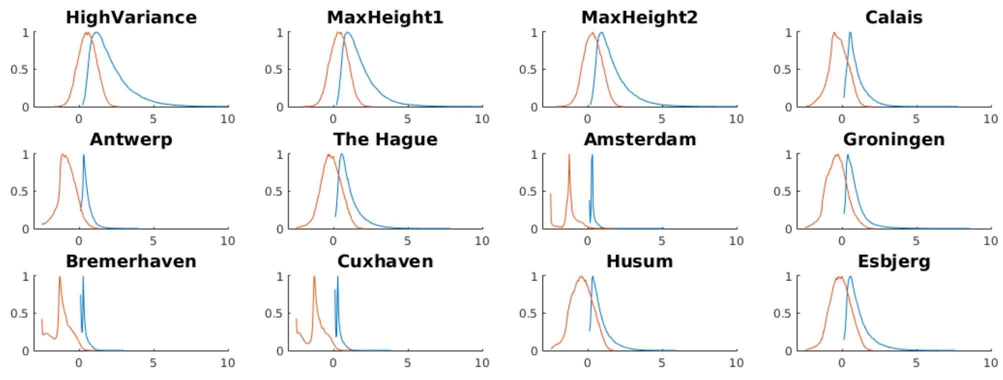

Fig. 1. Normalised wave height distributions of the different locations. Blue are the wave heights, red are the wave heights logarithmised, the x axis is the wave height in meters..

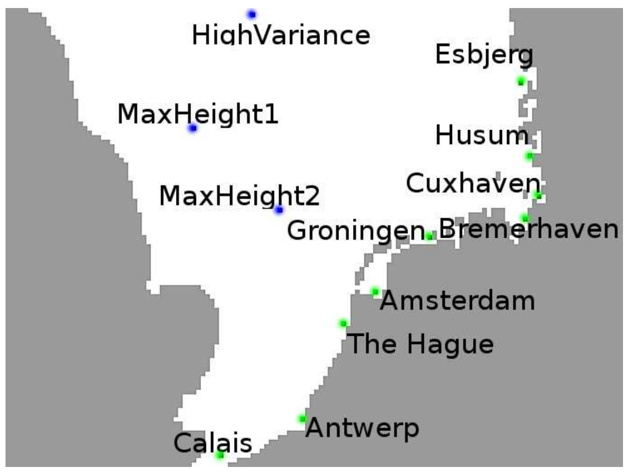

Fig. 2. Locations in the north Sea. Offshore nodes are blue, coastal nodes are green.

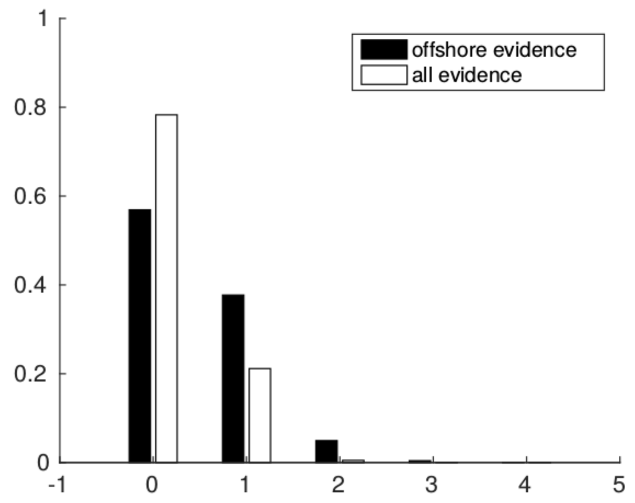

Fig. 3. Bin deviation rates of the discrete estimation.

heights differ from location to location, we have discretised the waves for every location independently. In the case of The Hague, the bins are *very low* from $0 - 0.71$ meters, *low* up to 1.25 meters, *medium* until 2.00 meters, *high* up to 3.06 meters and *very high* everywhere above.

B. Experiments

In this section we present the results of our experiments.

The first experiment was to train a Bayesian network on the discrete wave heights to be able to estimate them. This was done in two configurations: In the first configuration only the 3 offshore nodes were used as evidence, in the second configuration all nodes besides the queried one were used as evidence. In the first case the *overall recognition rate (ORR)* is 57%, in the second case the ORR is 78%. This means that this percentage of waves are correctly detected. You can see in Figure 3, that the most waves that are not detected correctly are only one bin off.

Subsequently we evaluated the possibilities for training a Bayesian network with continuous wave data. In Figure 4 you can see the probability density functions for the estimation errors. The errors are relative to the mean wave height at their respective location. The variance of the tests with the offshore nodes used as evidence is 0.18, the variance of the tests with all unqueried nodes used as evidence is 0.05. This means that in the first case 95% of the errors do not exceed 36% of the mean wave height and in the second case 95% of the errors do not exceed 10% of the mean wave height.

The prediction of wave heights was another task. To predict values the nodes were doubled into one past and one prediction set. The results in Figure 5 show the prediction capabilities of this structure for up to 100 hours. We tested for the first 10 hours hourly, up to 20 hours every second hour, up to 50 hours every fifth and up to 100 hours every tenth hour. This was done because the biggest changes in the prediction error happen in the first hours, later on the the error does not change as fast. As expected the prediction quality decreased with time. However it is notable that the prediction quality for short term prediction is better than the estimation. The

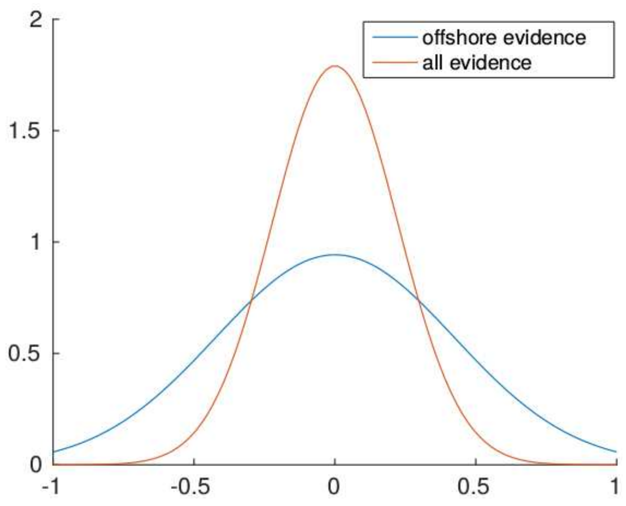

Fig. 4. Probability density function of the relative continuous estimation errors.

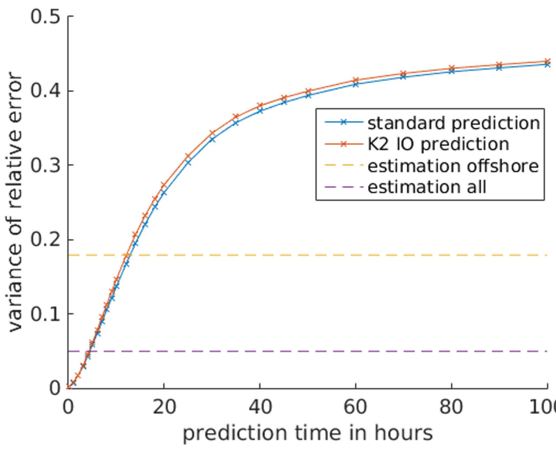

Fig. 5. Variances of prediction and estimation deviation from the ground truth data. The prediction is shown per prediction time up to 100 hours.

estimation results are represented by the horizontal lines. The short term prediction quality being better than the estimation is due the system having information not only from offshore or neighbouring locations but also from the queried locations themselves, just from the past.

Examples of predictions can be seen in Figure 6. In *a* you can see the example of an easy and accurate prediction. However *b* shows that the prediction can cover general trends, but the details of the curve are not anticipated. The plot in *c* shows an example of a bad prediction. While the model recognises that the wave development begins with a small ascend, the long term prediction is unrelated to the ground truth. There seems to be an event, as the wave height exceeds the usual values. Example *d* shows another case of a fairly good prediction. While the details aren't captured, the general development fits to the ground truth data.

The other prediction task was to use a DAG that was trained with the K2 IO algorithm. In Figure 5 you can also see that the

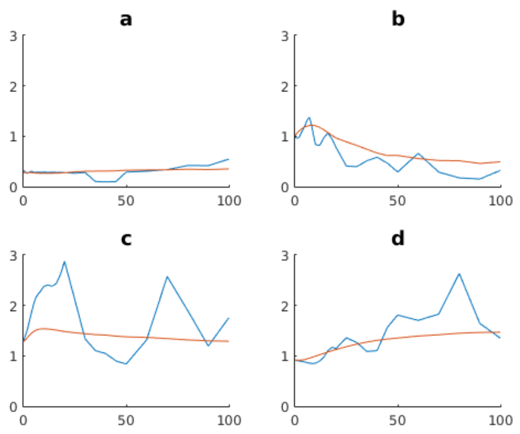

Fig. 6. Examples of predictions. Blue is the ground truth value, red is the prediction.

prediction quality decreased only marginally when connections between nodes of one time slice were forbidden and the DAG was trained with the K2 IO algorithm.

As expected the resulting DAG has a much lower number of connections. In Figure 7 you can see that there are no interconnections in one time step. The upper nodes always represent the past nodes, the lower nodes are the prediction nodes. In this case a maximum number of 4 parents per node were allowed. The standard K2 algorithm generated a DAG with 86 edges while the K2 IO algorithm created a DAG with only 48 edges. This means that there are 38 less influences that have to be learned when training the Bayesian network weights.

This algorithm also fulfilled our expectations in its speed up compared to the normal K2 algorithm. In figure 8 you can see that the K2 IO algorithm is significantly faster than the K2 algorithm. The algorithm had a mean speedup of 1.96 compared to the conventional K2 algorithm. Together with the low loss of prediction quality this shows that this algorithm is suitable for the creation of DAGs for prediction tasks.

V. Conclusion and Further Work

This paper shows that Bayesian networks are a useful tool for analysing and modelling ocean parameters as the significant wave height. It also shows that the K2 IO algorithm significantly outperforms the conventional K2 algorithm in terms of the time needed for structure learning while only marginally impacting the prediction quality.

However there are still tasks to solve and methods to try. With the current DAG structure we assume that the waves always follow the same creation process, independent from other factors like storms, changes in big streams and similar influences. To deal with this problem and increase the model quality, a method named *extending the conversation* can be used. This means adding additional nodes as parents of the influenced nodes that represent these events like storms. This way the Bayesian network can distinguish between the wave

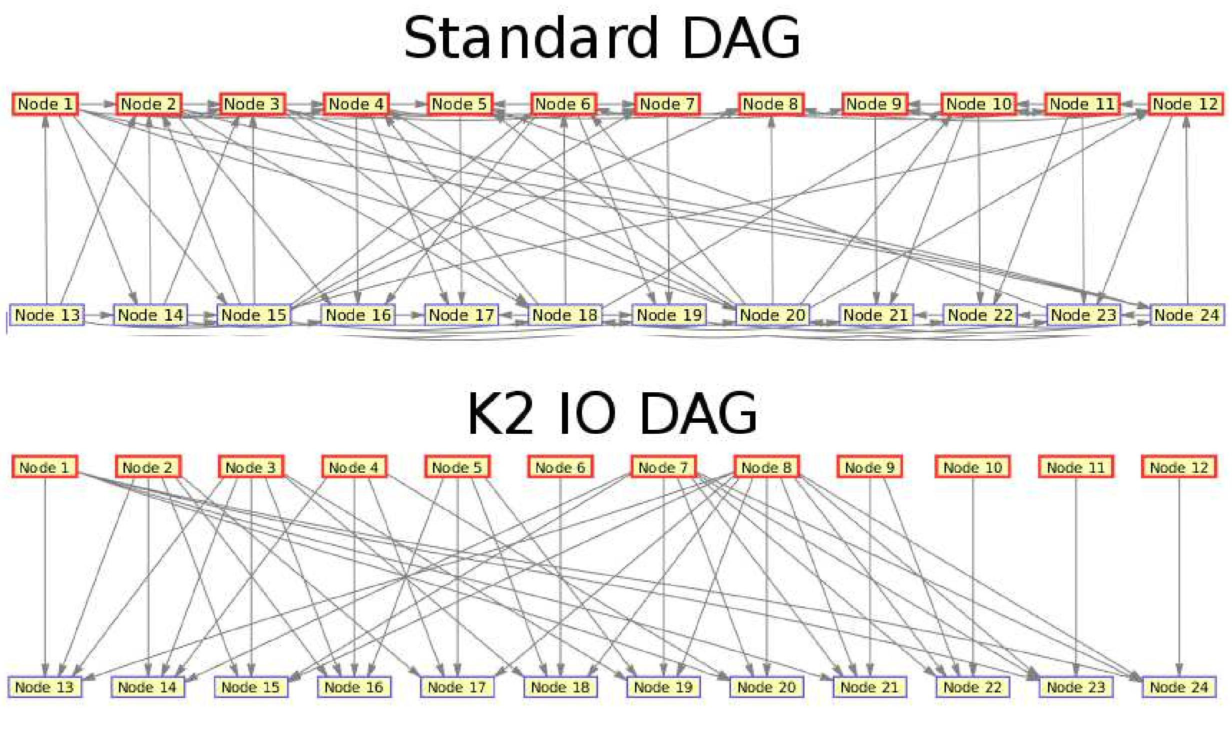

Fig. 7. DAG created with the standard K2 algorithm vs. DAG created with the K2 IO algorithm.

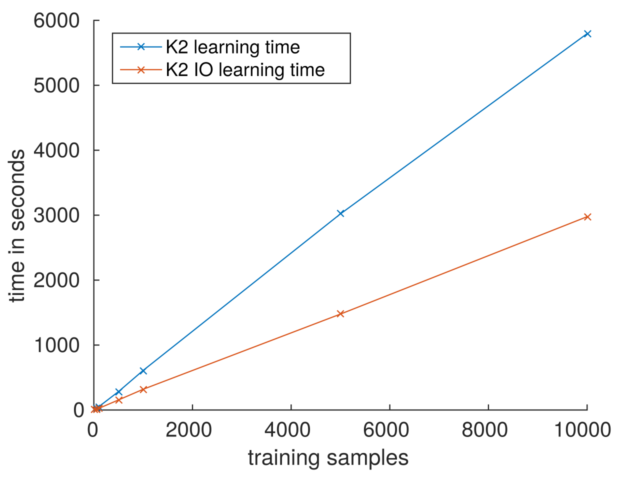

Fig. 8. The learning times of the standard K2 and the K2 IO algorithms.

height probabilities during these events and during normal

For the creation of the DAG there are also different possibilities. Besides the K2 algorithm there exist various algorithms with different approaches to this subject. Also

conditions. Additionally this structure can be used to detect storms and other events in the data.

Another idea is the addition of extra variables like the wind speed and the wave period to the model. A model that takes also these influence factors into account should have better estimation and prediction properties when trained on sufficient data.

domain knowledge can be used for a better DAG creation process. An example would be to incorporate knowledge about the directions of the wave movement to create the connections in the DAG.

References

[CH92] Gregory F. Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[HZG12] Zentrum für Material-und Küstenforschung GmbH Helmholtz-Zentrum Geesthacht. coastdat-1 waves north sea wave spectra hindcast (1948-2007), 2012.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[Llo06] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 2006.