Inferring Models for Subsystems Based on Real World Traces

Rutger Kerkhoff, Aleš Tavcar, Boštjan Kaluza
Jozef Stefan Institute, Jamova cesta 39
1000 Ljubljana, Slovenija
rutger.kerkhoff@gmail.com
ales.tavcar@ijs.si
bostjan.kaluza@ijs.si

ABSTRACT: *Creating simulations for smart cities is a complex and time consuming task. In this paper we show that using traditional Bayesian networks and real world data traces it is possible to infer models that can simulate the original domain. The created model can provide great insight into the actual subsystems that are considered. We show that given a set of observed values we can successfully use the created model to simulate data and show trends present in the original system.*

Key words: Bayesian Networks, Real-life simulation, Smart Cities, System Models

DOI:10.6025/ed/2019/8/2/43-49

Received: 3 April 2019, Revised 12 June 2019, Accepted 2 July 2019

1. Introduction

With modern cities becoming more complex and ever increasing in size it is of vital importance to control and optimize the different systems present in a city. While the different systems already available in a city, e.g. the power grid, waste management, bus scheduling etc., can and are being optimized, the bigger picture has not been explored yet. Connecting them allows further optimisation and realises emergent behaviour; something that cannot be observed when looking at a subsystem alone.

An important aspect of realising such a system is understanding the relations between the subsystems and even within the subsystems themselves. A concrete example of where this knowledge is needed is simulation. It is of vital importance to thoroughly test such a city-controlling system before applying it, and for that simulations are required. A simulation will also allow faster development of new systems and applications within the smart city. However, simulating a smart city is an immensely complex task, and in all but the simplest cases it is infeasible to specify all the variables and relations concerned in such a simulation. We therefore propose to learn a model, representing all the variables and relations in the smart city, from real-world data traces.

To allow for a better understanding of the system and the ability to simulate new traces we use a graphical model, a Bayesian belief network [1] to be exact. In this article we will focus on different ways of creating the network and inferring the probabilities. The ability to graphically represent the network makes it an ideal tool for understanding the modelled system. Moreover, we can input a set of observed variables and update, according to those values, the probabilities of the unobserved which makes it a suitable tool for creating simulations.

As a proof of concept this article will focus on simulating a single smart house. While this model will be significantly less complex, and even allow for exact simulation, it is ideal for showing the power of this approach. In this article we will show that using a manually defined Bayesian network already allows for quite accurate predictions. Expanding to an automatically learned network improves these even more, showing great promise for modelling a complete smart city.

2. Related Work

Using a Bayesian network to model a system is not a new idea, take for example a water supply network [5], where the authors try to predict when a pipe will burst. It has also been used to model mobility within a city [3] or to predict when to replace parts of the New York power grid [2]. While Bayesian networks have not yet been used to model a complete city, there are approaches which try to tackle large or distributed models. For example a hierarchical object oriented approach [7] or building local networks and identifying which variables likely link the systems [6]. While not needed for the model of the smart house discussed in this paper these techniques will likely be necessary when considering a smart city. As an example of what possible data streams and variables can be found in a smart city one can look at sensor data streams in London [8].

While other papers focus mainly on classification or decision support systems [4], we focus on recreating the original data and trends for a smart environment. We are not aware of any complete simulation and creation of a trace using a Bayesian network.

3. Domain

The domain explored in this paper is that of a smart house, which is less complex than a city, but it is well suited to show the ideas and explore the methodology. The traces used to model, learn and simulate the smart house are obtained from the EnergyPlus simulator [14]. The simulator was developed within the OpUS system [13] and it is based on a real world building. While simulating a simulator seems pointless, it allows for testing of the model in varying situations without having to gather real world data for an extended amount of time. For simulating the city this will be necessary.

The simulation has 6 input variables, all concerning the environment. One should think of values for outside temperature, wind etc. The simulator also supports setting if the occupant is present, the house was set to be empty from 7:00 to 17:00 every day. Redundant output variables were filtered out and we are left with 9 variables, ranging from inside temperature, heating coil power consumption to electricity produced by a solar panel. All variables are recorded in an interval of 15 minutes and are continuous. The subsystems that we want to simulate in the city can be seen as the different devices in the house, e.g. the solar power generator reacting to an increase in solar radiation.

4. Method

To learn a model of the house we use Bayesian networks. We decided on using Bayesian networks as they do not only perform well when predicting, they also have an understandable structure that can give insight to the situation being modelled. We assume the reader to be familiar with the subject as described by Heckerman et al. [1]. In this section we will explain the specific parameters and choices made for our implementation.

Creating a Bayesian network can be seen as a two stage process. First, we define a network structure, nodes and how variables are related as a directed acyclic graph. Second, we must set the probabilities distributions of all the different nodes depending on their ancestors. We define our network as G, our nodes as V and the arcs between the nodes as A.

$$G = (V, A) \tag{1}$$

During the first stage the structure of the network can be defined either manually or automatic. First, we define it manually, leveraging our knowledge of the system to define which nodes should be related. An advantage of this is that we will not get an

over-fitted network based on the coincidences on our training data. A drawback is that we might miss relations we did not know beforehand, and that for larger networks defining a structure quickly becomes a complex task.

Because the drawbacks will become more significant in the smart city we also implement an automated approach to learn the structure from the data. We use a local search based metric, the "Look Ahead in Good Directions, LAGD, hill-climbing algorithm [9]. It looks at a sequence of best future moves, instead of a single move as is usually done for hill climbing. The algorithm to calculate a sequence has exponential time complexity, and therefore it first computes the best moves and then decides on a sequence. It scoring function considers conditional mutual information between the nodes. For a formal definition please see Abramovici et al. [9].

The second problem, learning the probability distributions, is solved by calculating direct estimates of the conditional probabilities using a set of training data. For a single node $X \in V$ we define the probability that $X = x$ as:

$$P(x) = P(X = x \mid Xi, ..., Xj) \quad (2)$$

Here x are the different possible outcomes for that node. Simply counting the occurrences in our training data gives us a probability table for every possible outcome of X depending on every possible outcome of $Xi, ..., Xj$. Because not every possible combination is observed we use a prior ± = 0.5 as an initial count on each variable.

As the observant reader might have noticed the equation 2 requires discrete variables, and thus we discretized all our variables into 10 equal sized buckets. The partition intervals where chosen based on a minimum description length principle. For more information and a formal definition we refer the reader to Fayyad et al. [12].

We input the values of our observed testing variables and use this knowledge to calculate the remaining variables. This is done by the junction tree algorithm [10], it calculates the marginal distribution for each unobserved node based on the values of the observed nodes. We take the most likely value for each node, set it as evidence and update the margins, till all the variables are set.

5. Results

In this section we present the experimental results for the models presented in the previous section. In section 5.1 we will look at the structure of the networks and in section 5.2 we will analyse the performance of the complete models.

The data used was obtained from running the EnergyPlus simulator. We used two days of simulated data for training the model and two days for testing. The variables were all recorded at 15 minute intervals. The concrete implementation was done in Java using the WEKA library [11].

5.1 Network Structure

We first created a manual structure of the network. Since the selected domain is relatively simple with only 17 variables it could be largely comprehended by the analysis of the data. We looked at correlations between the variables, used our knowledge of the physical processes and experimented with a few possible networks to get a network that covered all the variables. Because of computational challenges some relations were simplified to restrict the number of parents of a node. Precautions were taken to make sure no relations were lost, for example, the solar radiation influencing the solar panel which in turn influences the battery. The arc from solar panel to battery was removed and instead it was linked directly to the radiation. This goes at the cost of a little accuracy but does not lose the relation. In Figure 1, a network created using the LAGD hill climber algorithm with at maximum 1 parent is shown. The figure does not show all the variables, as the nodes that do not have any arcs are left out for the sake of readability. The reasons why certain variables do not have arcs will be discussed in the next subsection. It is interesting that a lot of arcs seem to be reversed. Note; however, that setting evidence for any variable will update margins throughout the network, and thus even a reverse relation is still captured. Automatic network generation can model inconsistent relations, for example, *Net Purchased Power* related to *Wind speed* seems to be over fitted on a coincidence in the data. The other curious relations, like *Solar Produced* and *Inside temperature* can be explained by the fact that all the input variables are closely related, on a warm day there will be more solar radiation etc.

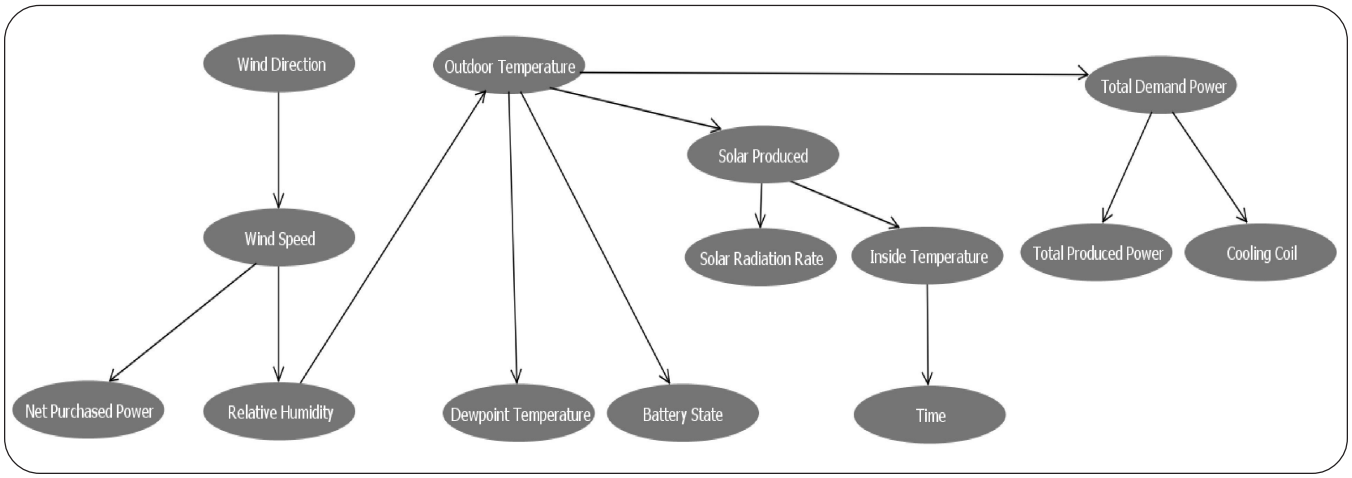


Figure 1. Network generated by LAGD Hill climber

5.2 Performance

To test the performance of the networks we computed the percentage of correctly predicted values per variable and the root mean-square-error (RMSE). However, even more important than predicting the right values is to predict the trends in the data. A few erroneous spikes are of lesser importance then missing a trend. Therefore we also plotted the data and did a visual analysis of the results.

	Baseline		**Manual**		**Automatic**	
Variable	%	E	%	E	%	E
Battery charge [J]	8	85e5	82	21e4	74	47e4
Heating coil	99	1.15	99	1.15	99	1.15
Washing machine	98	4.27	98	4.27	98	4.27
Total demand	51	87.54	51	87.54	60	15.17
Total produced	67	84.32	67	84.32	76	28.52
Cooling coil	69	46.49	35	49.45	77	22.25
Net purchased	65	35.61	65	35.61	65	35.61
Solar produced	60	88.31	81	34.58	82	35.37
Lights	72	35.53	88	12.46	72	35.53
Inside temp [C]	34	2.62	30	0.6	72	0.24
Overall	62		70		77	

Table 1. All variables are power in watt [W] if not stated otherwise. The columns labelled % depicts the percentage correctly classified. E is the RMSE, lower is better

In Table 1 the percentage of correctly predicted values and the RMSE for each variable for three separate classifications is shown. First a baseline where the most probable bucket was chosen based on just priors. Second the manually constructed

network and last the automatically learned network. For the heating coil and the washing machine even the baseline has an exceptionally good prediction power. This is because they are off in all but a few cases, showing the importance of complete training data. The manual network performed better on a subset of the variables, for example power used by lighting. Lighting can be clearly linked to time, this was not found by the LAGD algorithm as it does not occur often in the training data. However, in most cases automatic structure generation did perform better. Some of the interesting variables are the predicted values of inside temperature (Fig. 2b), solar power produced (Fig. 2c) and battery charged state (Fig. 2a). Note that the predicted data was discretized into buckets and for each datapoint the average of the bucket was used for the calculation of the RMSE and as data points in the graphs.

The predicted inside temperature over time shown in Figure 2b follows the general trends for both models, though the manual network is over-fitted on outside temperature and clearly performs worse. Another problem is that because of the discretisation some small changes are amplified. In Figure 2c the prediction of power produced by the solar panel is shown. It is closely correlated to one of the input variables, solar radiation, and is therefore quite accurately predicted. The second production increase is not seen in the graph for the manual network as the values are still in the range of the first bucket.

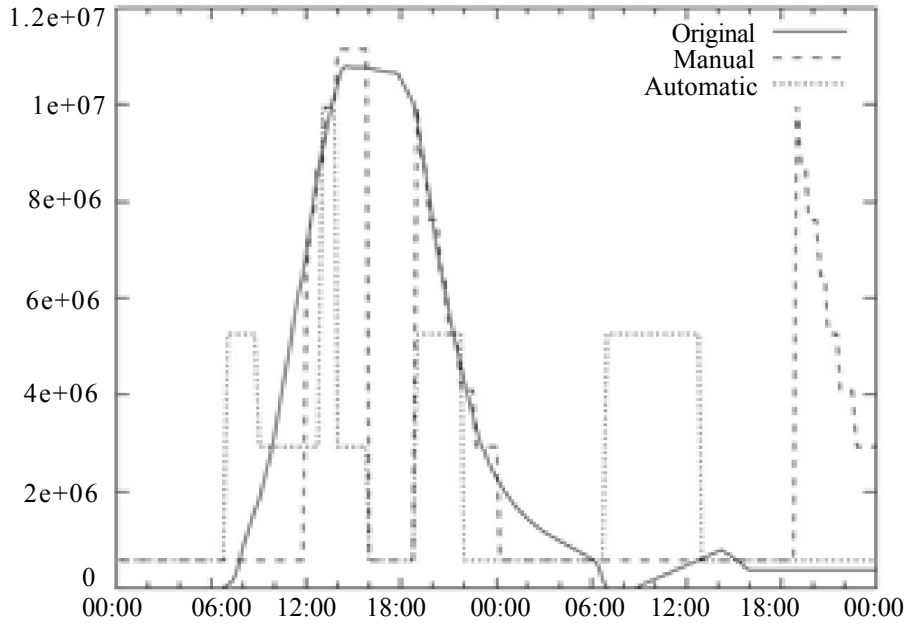
Predicted battery charge state over time can be seen in figure 2a. This is difficult to predict as it depends on many variables. As can be seen the manual network is over-fitted on a wrong variable; time. The first charge peak happens to coincide in the training and testing data and therefore the percentage of correct prediction is still high. The automatic network is not conclusive in predicting a trend. For variables like a battery, which cannot drop quickly and get back up again, it will be a valuable extension to consider the previous state as well.

In general, the automatically constructed network model performed better than the manually constructed one and possibly correct but yet unknown relations were found. The variables could be predicted relatively accurately and most trends present in the original data could also be found in the generated data.

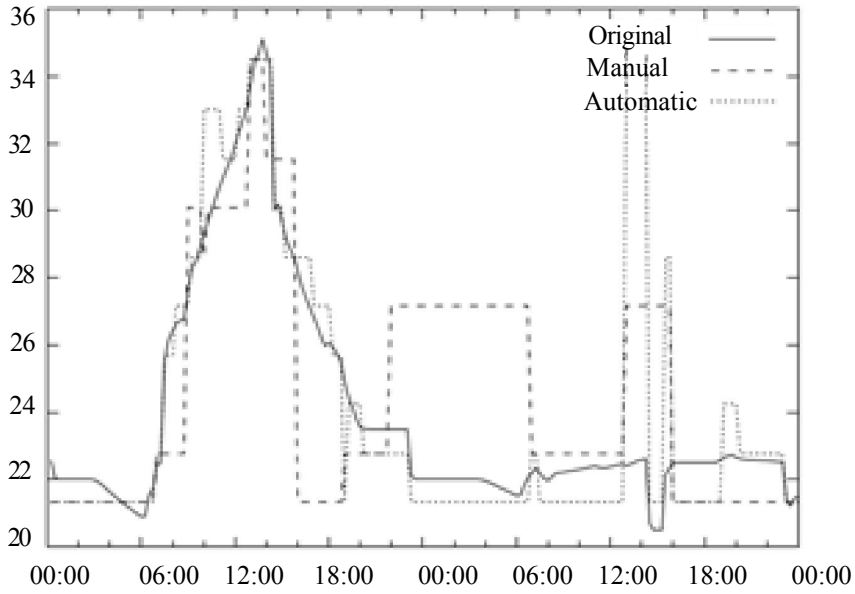
4. Conclusion

We have shown that it is possible to build complex models from real world traces that model relations between subsystems. These models can then be used to simulate the system and generate more data based on a set of input variables. We have seen that trends in the data can be modelled and even single predictions can be used as an indication of expected data.

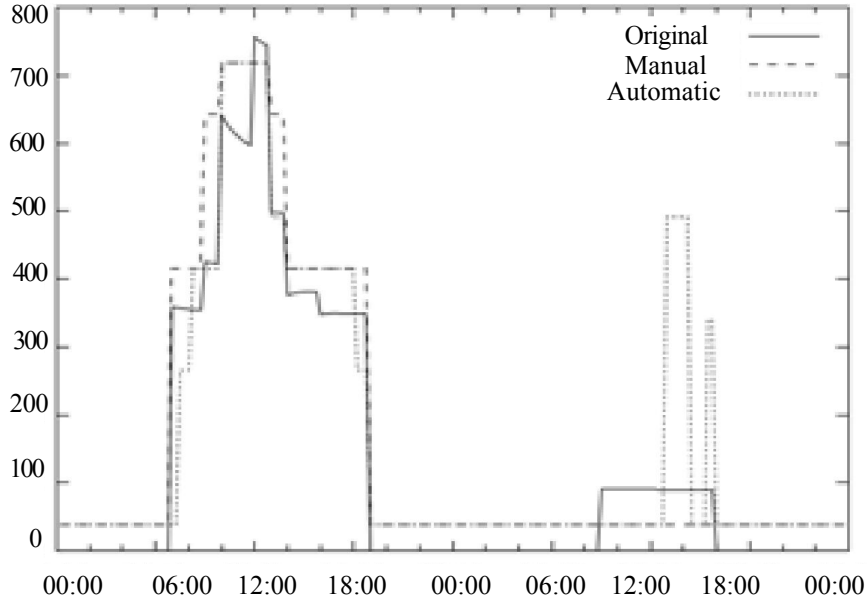
While automatic generation of a Bayesian network for a domain is possible, some expert knowledge will still be required to



(a) Amount of joules in the battery



(b) Inside Temperature



(b) Solar panel production

Figure 2. Predictions of system parameters

reduce over-fitting due to coincidences in the data, and to improve the network. Due to the nature of the Bayesian networks the cooperation with a domain expert can easily be established.

The biggest drawback of this method is that it largely depends on the available data. A lack of data will lead to incomplete or incorrect models.

The most important direction for future work will focus on taking the temporal nature of the network into account. Expanding to dynamic Bayesian networks or Hidden Markov Models will allow for an even more accurate prediction of trends. The challenge

will be to cover the unknown parameter space that is not directly present in the training data. Introducing Gaussian probabilities to closer model the values produced by different sensors is another possible direction for future work. A third extension will handle tackling computational challenges that arise when the network sizes increases, those may be solved by creating a more hierarchical network structure.

References

[1] Heckerman, D. (1997). Bayesian networks for data mining, Data mining and knowledge discovery, 1 (1).

[2] Rudin, C., Waltz, D., Anderson, N. R., Boulanger, A., Chow, M., Dutta, H. (2012). Machine learning for the New York City power grid, *IEEE Transactions Pattern Analysis and Machine Intelligence*, 34 (2).

[3] Fusco, G. (2003). Looking for Sustainable Urban Mobility through Bayesian Networks, Cybergeo: *European Journal of Geography*.

[4] Lanini, S. (2006). Water management impact assessment using a Bayesian network model, *In*: Proceedings of the 7th Conference on Hydroinformatics.

[5] Babovic, V., Drcourt, J., Keijzer, M., Hansen, P. F. (2002). A data mining approach to modelling of water supply assets Urban Water, 4 (4).

[6] Rong, C., Sivakumar, K., Kargupta, H. (2004). Collective mining of Bayesian networks from distributed heterogeneous data, *Knowledge and Information Systems*, 6 (2).

[7] Molina, J. L., John Bromley, J. L., Garca-Arstegui, C., Sullivan, Benavente, J. (2010). Integrated water resources management of overexploited hydrogeological systems using Object-Oriented Bayesian Networks, *Environmental Modelling & Software*, 25 (4).

[8] Boyle, D., Yates, D., Yeatman, E. (2013). Urban Sensor Data Streams: London 2013, *IEEE Internet Computing*, 17 (6).

[9] Abramovici, M., Neubach, M., Fathi, M., Holland, A. (2008). Competing fusion for bayesian applications, *In*: Proceedings of International Conference on Information Processing, 8, 379.

[10] Lauritzen, S. L., 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.

[11] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witte, I. H. (2009). The WEKA Data Mining Software: An Update, SIGKDD Explorations, 11 (1).

[12] Fayyad, U., Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning, Chambery, France.

[13] Tavcar, A., Piltaver, R., Zupancic, D., Sef, T., Gams, M. (2013). Modeliranje Navad Uporabnikov Pri Vodenju Pametnih Hiš, *In*: Proceedings of Information Society 2013, 114-117.

[14] US Department of Energy, EnergyPlus Energy Simulation Software, eere.energy.gov/buildings/energyplus/, accessed 08-2014.