LIMIDs for decision support in pig production

Merete Stenner Hansen*† Anders Ringgaard Kristensen*‡

*Department of Large Animal Sciences, Royal Veterinary and Agricultural University
Grønnegårdsvej 2, DK-1870 Frederiksberg C, Copenhagen, Denmark
†msth@kvl.dk, ‡ark@dina.kvl.dk

1 Introduction

With the current pressure on pig production to be increasingly effective in order to remain profitable, a large effort is put into developing systems that can increase efficiency and/or decrease the demand for man hours. One such system is automatic weighing equipment. In order to best utilize an investment in automatic weighing equipment, a decision support system (DSS) may be used. There are several methods available for construction of such a DSS, but a general problem is the complexity of the decision problem that causes the required optimization calculations to become either impossible or at best very slow. In this paper the relatively new method of *LImited Memory Influence Diagrams* (LIMIDs) is tested in order to establish whether it can handle the delivery decision problem.

Since the main purpose is to evaluate the possibilities in using LIMIDs, the focus of this paper is put on the constructed LIMID, called PigLimid, and its abilities and problems. The growth model for the pigs is presented but not discussed, since it only has minor influence on the conclusions about the use of LIMIDs.

2 Influence diagrams and LIMIDs

An *influence diagram* (ID) is represented graphically as a directed acyclic graph (DAG) consisting of a set of nodes; *chance nodes* (representing the discrete variables), *decision nodes* and *utility nodes* and a set of *directed edges* connecting them. A node is called *parent* of another node if there is a directed edge from the parent node to the *child* node. Each variable has a number of mutually exclusive states, and a corresponding potential table. Edges into chance nodes represent the variable's dependence of the parents, and this dependence is quantified in the potential table. A utility node contains a utility table representing the utility gained by each element in the cartesian product of the parental states. An ID has a strict chronological decision node order, and operates under the "no forgetting" assumption, where all previous decisions and observed variables are known (and taken into account) in the following decisions.

In order to use the ID for optimizations, it must be compiled. A part of the compilation is

triangulation, where a perfect *elimination sequence* is created. An elimination sequence is the order in which the nodes can be removed from the graph without loosing information about the probability distributions of the remaining variables. When a node is eliminated, there has to be edges between all its neighbours, if an edge is missing it will be added. In a perfect elimination no extra edges are added [2]. A *clique* is defined as a complete node set that is not a subset of another complete set. A complete node set is a set of nodes that are all pairwise linked. A *separator* is the remaining nodes in a clique after the nodes in the clique, that only had neighbours in the clique, have been eliminated [2]. After the triangulation, the resulting cliques are ordered in a strong *junction tree* [3]. Due to the "no forgetting" assumption, many extra edges are added to the ID, this results in large cliques, which in turn results in very complex optimization calculations.

The triangulation is not a unique process and a graph might be triangulated in different ways. The triangulation will affect the complexity of the following optimization, so it is sensible to choose the triangulation that minimizes the computational costs [4].

LIMIDs were introduced by Lauritzen and Nilsson [4] in an attempt to create an alternative to traditional IDs, which grows very complex when the number of variables included is increased. LIMIDs do not have the "no forgetting" assumption included in IDs and thus only use the specified variables (the parents of the decision node) in the optimization of a given decision. The decision nodes are only influenced by their parents, and only variables that can be observed are used as parents for decision nodes [4], thus the large cliques that hamper the IDs are avoided. Based on the parental states, a strategy for the decision is given.

A LIMID finds an approximately optimal strategy (local optimum) through single policy updating (SPU). SPU can find locally optimal decision strategies. This means, that at a local maximum no single change of policy can increase the utility value. SPU uses message passing in the junction tree created in the compilation [4].

During the SPU the LIMID is treated as a single action network (i.e. the decisions are optimized one at a time). The last decision is considered first (this is not mandatory) and the optimal decision policy is found based on the initial values of the preceding decisions. Then the last decision but one is optimized and so on, until all decisions have been optimized. The process is iterative, and after considering all decisions the optimization is repeated based on the new decision policies. This is repeated until convergence [2].

3 The model

In the PigLimid, a pen of pigs is represented as a collection of individually modeled pigs in order to be able to slaughter each pig at the optimal time. All pigs in the model are initially identical, but by insertion of evidence (i.e. weight observations), the pigs are differentiated.

The PigLimid, is divided into stages, the initial stage corresponds to the weeks from insertion in the pen until start of observation (the first week where the pigs can be delivered for slaughter), the rest of the stages each represents a week. The PigLimid is based on a growth model that assumes linear growth in the weeks where delivery is possible.

The current live weight for the i'th pig ($w_{i,t}$) is a function of its previous weight ($w_{i,t-1}$)

and growth capacity ($g_{i,t-1}$), as shown in Eq (1).

$$w_{i,t} = \begin{cases} w_{i,t_0} & \text{for } t = t_0 \\ w_{i,t_0} + (g_{i,t_0} + \nu_t) \cdot t_{SO} & \text{for } t = t_{SO} \\ w_{i,t-1} + g_{i,t-1} + \nu_t & \text{for } t > t_{SO} \end{cases} \quad (1)$$

Where t_0 is the time of insertion of pigs in the pen, t_{SO} is the time for start of observation (measured in weeks), g_{i,t_0} is the growth capacity for the i'th pig in the weeks from insertion until start of observation, ν_t is the random variation in weekly gain over time and it is assumed that $\nu_t \sim \mathcal{N}(0, \sigma_\nu^2)$, i.e. ν_t is assumed normally distributed with mean 0 and standard deviation σ_ν. Every variable included in the model is assumed normally distributed. In Table 1 the assumptions regarding the herd is shown.

Table 1: Assumptions concerning the herd

Parameter	Assumptions
Start weight	The pigs will have a weight at insertion in the pen of approximately 30 kg, with a standard deviation of 1.5 kg
Weeks for delivery	Each pen will be emptied over 4 weeks, and the pen must be empty before insertion of new pigs.
Production type	The production is of Danish standard pigs, and the pigs are mixed gilts and castrates.
Pen size	There is a maximum of 16 pigs per pen.
Weighing	The pen has an automatic weighing system with individual pig registration.
Feeding regime	The pigs are fed ad lib pelleted dry feed.
Slaughter prize	The optimal slaughter weight interval is 70.0 - 84.9 kg.

4 The PigLimid

The PigLimid is constructed in the "Esthauge LIMID software system"[1]. In order to make the PigLimid as flexible as possible, a Java program is written, that uses the Esthauge API to generate the PigLimid. In Figure 1 the PigLimid is shown, and in Table 2 the variables included in the model are listed.

Each node has a number of states representing equally sized intervals. The live weight and observed live weight nodes also have states representing dead and delivered pigs.

In order to be able to investigate whether it might be profitable to terminate (slaughter the remaining pigs) the pen earlier than planned and insert a new batch of pigs, the termination value (p_{T_t}) is used. The termination value is the average weekly gross margin per pen if the pen is empty, else it is zero.

The number of pigs left in the pen is counted by the pig counter nodes (c_t). The pig counter nodes are divided into 4 levels, each with 2 parents, in order to keep the size of the

[1] http://www.esthauge.dk/

Table 2: Variables included in the model

Variable	Symbol
Current live weight (for the i'th pig at week t), kg	$w_{i,t}$
Observed live weight (for the i'th pig at week t), kg	$w_{o_{i,t}}$
Growth capacity (for the i'th pig at week t), kg/week	$g_{i,t}$
Slaughter weight (for the i'th pig at week t), kg	$w_{s_{i,t}}$
Feed consumption (for the i'th pig at week t), FEsv	$f_{i,t}$
Dead or alive (for the i'th pig at week t)	$a_{i,t}$
Pig counter (for the pen at week t)	c_t
Slaughter price (for the i'th pig at week t), DKK/pig	$p_{s_{i,t}}$
Feed price (for the i'th pig at week t), DKK	$p_{f_{i,t}}$
Price of young pigs (for the i'th pig), DKK/pig	p_{p_i}
Value of terminating the pen (for the pen at week t), DKK	p_{T_t}
The slaughter decision (for the i'th pig at week t)	$s_{i,t}$

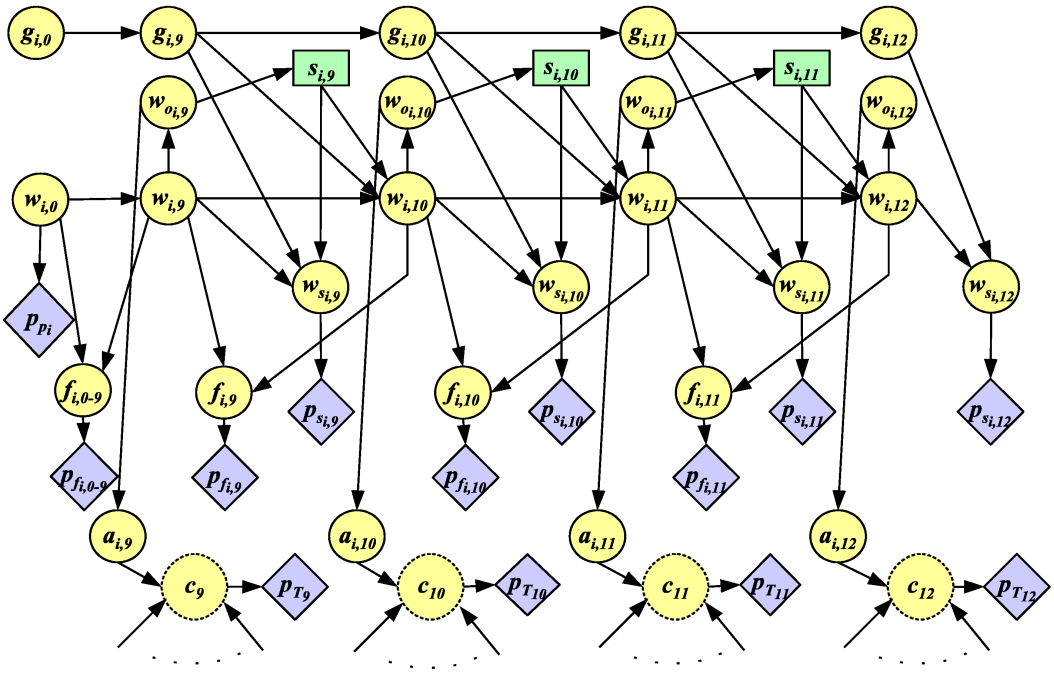

Figure 1: Representation of one pig in the PigLimid. Circles are chance nodes, rectangles are decision nodes and diamonds are utility nodes. See Table 2 for definition of the node names.

final potential table manageable. This principle is called divorcing [2] and consists of dividing the parents of a node into smaller groups and thus preventing the potential table of the child from growing too large as well as keeping the cliques small.

Each node has a potential table representing the probabilities for each state in the node given the state of the parents. The probabilities for the states of the node sums to 1 for every combination of parent states. All values in the potential tables are calculated with the use of normal probability distributions.

5 Results

After the construction of the PigLimid several problems have been encountered. The first problem is the fact, that the PigLimid is impossible to compile with the intended 16 pigs. This was found to be due to a very complex triangulation. During the triangulation many edges were added between nodes belonging to different pigs, some nodes got up to 25 extra edges added to them, which made the rest of the compilation impossible. A triangulation criterion that punished edges between nodes belonging to different pigs was tested, but it had no effect. Hence all tests of the PigLimid has been carried out with a maximum of 5 pigs.

When the SPU was performed on the PigLimid, the resulting strategies in some cases turned out to be neither globally nor locally optimal, and in other cases the local optimum found was far from the global optimum. Part of the problem turned out to be, that the SPU considers one pig at a time. Logically the optimal strategy must be the same for all pigs, since they are all identical before insertion of evidence. In some cases the SPU found strategies that were not identical for all pigs, these were not locally optimal. This phenomenon was investigated, and is considered to be due to a numerical instability in the software. In other cases the local optimum found was far from the global optimum, because most strategies that are identical for all pigs are locally optimal even though they are far from the global optimum.

In order to solve these problems a new optimization algorithm, that forces identical strategies between pigs, was made. It does not use real SPU, but is inspired by dynamic programming. Through calculations in the junction tree, called a full propagation, the utility value for a given strategy can be found. The structure of the optimization routine is as follows:

- First all decisions are set to No and a full propagation is performed
- Then, starting from the last stage of decisions;
  - The decision for the heaviest state is set to Yes for all pigs and a full propagation is performed.
  - If this increased the utility value, the next state is set to Yes as well, and so on.
  - Else the state is set back to No, a full propagation is performed, and the procedure is continued with the previous stage.
- When all stages have been done, the process is repeated until the utility value is no longer being increased.

This optimization utilizes the inherent logic in the PigLimid by slaughtering the heaviest pigs first, which will in all but perhaps the most extreme cases be the optimal solution. This optimization provides globally optimal strategies, and thus solves the problem, unfortunately it is much slower than the SPU.

6 Discussion

During the construction of the PigLimid it became obvious, that the LIMID method as it is currently used in the "Esthauge LIMID software system", is inadequate at handling large models. The triangulation becomes too complex, which results in a junction tree with very large cliques, which in turn results in huge optimization calculations.

A possible solution to this problem may be object oriented LIMIDs. Bangsø [1] presented an object oriented framework for Bayesian networks. The Bayesian network is split into objects built from classes. A pig could be a class, and then for each pig needed in the network, a pig object would be created as an instance of the pig class. The computational advantage of object orienting is the possibility for triangulating each class separately, which will help keep the cliques, and thus the joint probability tables, at a manageable size. Based on the similar structure of Bayesian networks and LIMIDs, this method should be applicable on LIMIDs as well. Some experimentation with an object oriented PigLimid has been done, and is so far showing promising results. The object oriented approach is judged to be the best way to solve the complexity problems. If it is implemented successfully it will be possible to handle pens with more than 16 pigs, have more variables for each pig, and more states in the nodes.

The LIMID method requires discretization of the variables, and some investigation of the optimal number of states might be required, especially since the discretization will always be an approximation of the truth. The way the discretization is done should also be considered, currently it uses equal sized intervals, but equally probable intervals might be better.

The problems observed with the original SPU algorithm, which is probably occurring due to numerical instabilities, might be caused either by the way the algorithm is implemented or by the nature of the algorithm itself. This might be worth investigating further before an optimization algorithm for object oriented LIMIDs is created.

If the problems with complexity and numerical instabilities are solved, the LIMID method shows promising abilities. It gives the possibility of modeling the pig in a natural and rather precise way, where the same modeling of a pig using dynamic programming would result in huge state spaces. On the other hand the pen dynamics are not handled optimally due to the static nature of the method.

The decisions in the PigLimid is only influenced by the current observed live weight. In order to utilize the termination value better, it would make sense to include the pig counter nodes as parents to the decision. Then a strategy would be provided for each combination of weight and number of pigs left, which would probably lead to slaughter of smaller pigs if there is only a few pigs left in the pen, hence increasing the probability for emptying the pen and getting the termination bonus. This is difficult to implement with the new optimization algorithm, but a solution may be found.

A possibility for combining the strengths of LIMIDs and Markov processes was introduced by Nilsson and Kristensen [5]. Their idea is to have the founder level modeled as a Markov process, and the child level represented as a LIMID. In this case the PigLimid would be the child process, that is generated each time a new batch of pigs is inserted in a pen. The founder level would be representing the strategic long term planning, and would have the possibility of chancing the input values of the PigLimid from batch to batch. In this way the static nature of the LIMID method would not be a problem since it is implemented in a dynamic system, hence the model could be used for optimizations under infinite time horizons as well. At the same time the Markov process could operate with unobserved variables at the child level, which normally is problematic, and the number of variables involved would not lead to unreasonably large transition matrices.

The general conclusion is, that LIMIDs show promising abilities for solving the delivery problem, but further development of the method is necessary before a working DSS can be made.

References

[1] Bangsø, Olav (2004): *Object Oriented Bayesian Networks*. PhD thesis. Aalborg University, Denmark.

[2] Jensen, Finn V. (2001): *Bayesian Networks and Decision Graphs*, Springer-Verlag, New York.

[3] Jensen, Frank and Jensen, Finn V. and Dittmer, Søren L. (1994): "From Influence Diagrams to Junction Trees". In: R. Lopez de Mantaras and D. Poole (eds.): *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann. San Francisco. 367–373.

[4] Lauritzen, Steffen L. and Nilsson, Dennis (2001): "Representing and Solving Decision Problems with Limited Information". *Management Science*. **47**, 1235–1251.

[5] Nilsson, Dennis and Kristensen, Anders R. (2002): "Markov LIMID processes for representing and solving renewal problems". In: *First European Workshop on Sequential Decisions under Uncertainty in Agriculture and Natural Resources*. INRA. September 19-20, 2002, Toulouse, France. 97–104.