NTNU

Norwegian University of
Science and Technology

Online Meat Cutting Optimisation

Uno Wikborg

Master of Science in Computer Science
Submission date: June 2008
Supervisor: Arne Halaas, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In a meat processing plant for warm cutting of cattle carcasses, different cutting patterns can be used to make different products. The possible cutting patterns depend on the attributes of the carcass, such as fat percentage and gender. Only a limited number of carcasses are in storage between measurements are taken and cutting. Information such as the fat percentage is not available for the unmeasured carcasses.

This thesis will look at online optimisation techniques for determining which cutting patterns to use. The goal will be to choose the best cutting patterns to satisfy the plants production plan.

Assignment given: 15. January 2008
Supervisor: Arne Halaas, IDI

Abstract

Nortura, Norway’s largest producer of meat, faces many challenges in their operation. One of these challenges is to decide which products to make out of each of the slaughtered animals. The meat from the animals can be made into different products, some more valuable than others. However, someone has to buy the products as well. It is therefore important to produce what the customers ask for.

This thesis is about a computer system based on online optimisation which helps the meat cutters decide what to make. Two different meat cutting plants have been visited to specify how the system should work. This information has been used to develop a program which can give a recommendation for what to produce from carcasses during cutting.

The system has been developed by considering both the attributes of the animals and the orders from the customers. The main focus of the thesis is how to deal with the fact that the attributes are only known for a small number of the animals, since they are measured right after slaughtering. A method has been made to calculate what should be made from the different carcasses, and this method has been realised with both exact and heuristic algorithms.

Preface

This report presents my Master's Thesis in Computer Science at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). The problem was proposed by SINTEF with cooperation with Nortura. The work has been conducted during the spring 2008 and is a continuance of a project performed by me during the fall of 2007, which will from now on be referred to as [Wikborg07].

I would like to thank my supervisor at NTNU, Arne Halaas, for allowing me to choose the problem from SINTEF.

I would like to thank my supervisors at SINTEF, Arnt-Gunnar Lium and Kristin Tolstad Uggen, for suggesting the problem and for their invaluable advice and thorough inspection of this report.

Finally I would like to thank Trond Malmo and Klas Forfang at Nortura for their great response and their hands-on knowledge of the meat cutting industry. The tour of the meat cutting plant at Rudshøgda was particularly helpful.

Trondheim, 9th June 2008

Uno Wikborg

Table of Contents

Table of figures

Table of tables

1 Problem description

1.1 Nortura

The case studied in this thesis is one of Nortura's meat cutting plants. Nortura is Norway's largest producer of meat and a cooperative consisting of more than 30,000 farmers.

"The group's purpose is to sell the members' slaughtered animals, eggs, live animals and wool in the best possible manner. The group shall through its operations contribute to the members receiving the best possible financial result from their livestock production." – Nortura's objective clause [Nortura08]

By being a cooperative Nortura has to work in order to benefit all of its members, not only by giving profit to its owners, but Nortura is obliged to accept animals for slaughter from its members and to pay a fair price for it. This is what creates the framework for their operation.

To accept animals from all farmers creates various challenges. The animals are not allowed to be transported more than 8 hours by truck [Schütz05], which means slaughterhouses have to be located in proximity to the farmers, while the slaughterhouses also have to be efficient enough for Nortura to obtain a profit from their operation. The profit is made by selling final products to paying customers. These challenges make up a value chain stretching from every farmer all the way to the grocery stores, where every part is crucial for the success of Nortura.

1.2 Hierarchical planning

A meat cutting company makes decisions on many levels, as can be seen in Figure 1. This section will explain the hierarchical structure of the decision process to make it easier to understand the role of online optimisation.

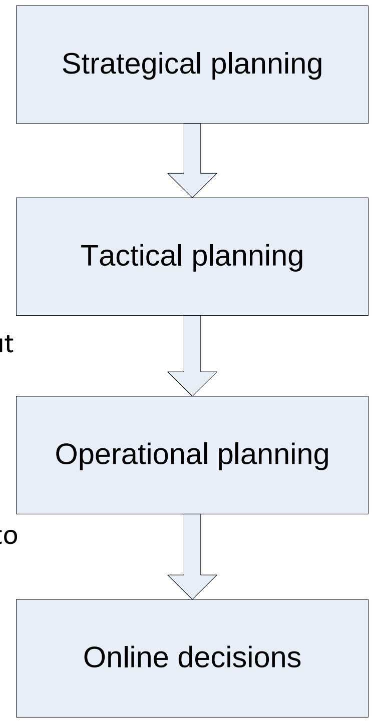

Figure 1 Hierarchical planning

Strategic planning - This level of planning decides the long term strategies of the company. These kinds of decisions involve where to locate new slaughterhouses in the future, which slaughter houses to shut down and which markets to focus on.

Tactical planning - On this level of planning, the whole value chain is in scope. The value chain of meat production includes slaughtering, cutting, processing, logistics and sales. The time horizon is long enough to deal with annual variations. Some products are more popular during certain time periods such as Christmas. Tactical planning should make sure that enough is produced to cope with these variations and make an overall plan for the production.

Operational planning - On this level of planning, the production for a shorter period of time is in scope, typically from one to three days. The slaughtering, cutting and demand for a limited period is used to determine exactly what to produce during a day. The final production plan is decided on this level, based on the estimates from the tactical planning.

Online decisions – Online decisions are decisions which have to be taken with limited amount of information. These decisions cannot wait for more information because they are crucial for the production. One such decision is which products to produce from an animal.

1.3 Making decisions for meat cutting

The last section described the scope of online decisions. This section will describe decisions that have to be done while cutting meat.

After cattle are slaughtered, some parts of the animals such as the hooves, intestines and the head are removed. The rest of the carcass is sent to cutting. Here it is first measured for characteristics such as fat percentage and weight. During cutting, the carcass is cut into different kinds of products. Which products the carcass should be made into has to be decided before the production can continue. Cutting patterns have estimates of what can be produced from a carcass. This pattern will tell what to make out of each part of the carcass. When a cutting pattern is chosen, it will give an estimate of how much of each product the carcass will be made into. One pattern can have the estimates that the carcass will turn into 1.4% tenderloin, 3% sirloin, 49% meat containing 14% fat, 21% round steak, 7% shoulder steak and 4% waste. These estimates are not accurate and will therefore not always sum up to 100%. This is because various factors will determine exactly how the cutting is done, which can result in significant variations. A newly recruited cutter will not be as skilled as a more experience cutter. He or she would leave more waste and degrade more of the meat into lesser quality.

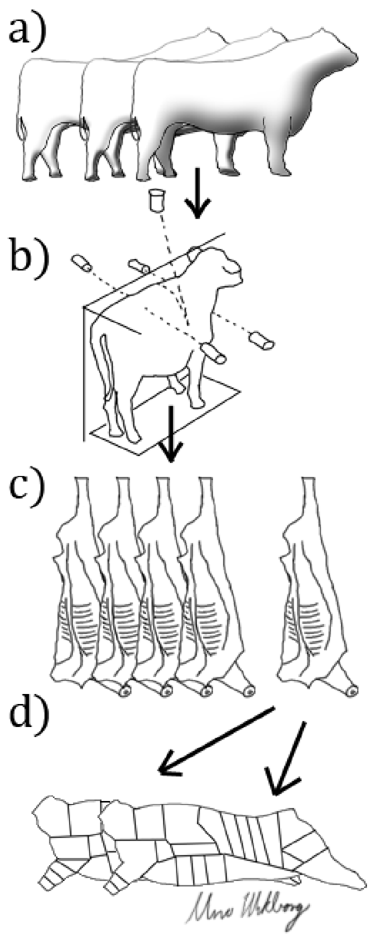

Figure 2 Production line a) Animals lined up for slaughtering b) Animals are measured c) Carcasses ready for cutting d) Cutting pattern is chosen

A good choice of cutting patterns will make a trade off between making valuable products and fulfilling the production plan. If there is a large demand for meat with 7.5% fat, it might be a good idea to choose cutting patterns which make 7.5% fat meat out of the round steak. If the production plan has no need for more 7.5% meat, it might be better to keep it as a round steak, since this is a product with higher sales value.

1.4 Cold cutting

There are two different techniques used for cutting meat. The most common is cold cutting. The animals are slaughtered, cooled down and then moved for storage. The meat does not go bad as fast after having been cooled down, and this makes it easier to plan ahead. If too many animals are slaughtered, they can be stored in the storage and cut later. A period with less slaughtering will make it necessary to reduce the size of the storage. This has been the most common practice for years because of the flexibility the storage provides, and due to the technical difficulties with cutting warm meat. When the meat stays on the carcass during storage, the meat will not be able to shrink since it is attached to the bone structure of the carcass. This has until now made it less desirable to cut the meat before storage, since the meat will become less tender when it shrinks.

1.5 Warm cutting

Another technique is warm cutting. During warm cutting, the carcasses are being cut right after being slaughtered. The carcass has not been cooled down and is still warm. Cutting through warm carcasses improves the working environment since the cutters do not have to worry about keeping their fingers or their body warm while touching the meat and standing in a cold room. But the comfort of the workers is not the main reason for warm cutting. Warm cutting does not need a storage facility for the carcasses, but might need additional storage for live cattle. This is because warm cutting is less flexible to vary the speed of the slaughtering since the slaughtered animals have to be cut right away. Another advantage is the fact that the carcasses are being processed much faster. This leads to less stock line and a higher rate of turnover, which is good for profitability. While carcasses are being stored, some of the water dries out from the meat. This leads to a weight loss of the carcass which will eventually lead to a weight loss in the products. When the carcasses are being cut right away, this weight loss will not happen since the moisture do not have time to leave the meat. Cattle cut with warm cutting will therefore have less waste than those cut with cold cutting. Packing the meat right away does also create some challenges. The meat will shrink because it has no bone structure to keep it stretched. This problem has been solved with a new vacuum technology which capsules the steak in a special wrapping which prevents the steak from shrinking. Because of these advantages, more and more of the cutting is warm cutting and almost all new cutting plants which Nortura builds support warm cutting.

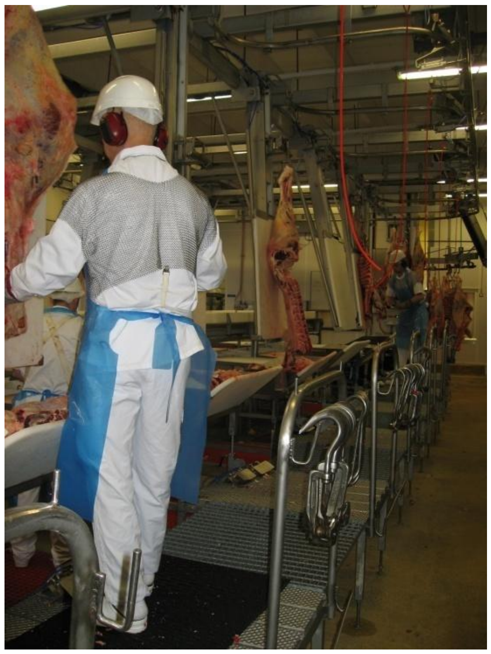
Figure 3 Meat cutting

1.6 Packing

The products are not quite ready for delivery even if the meat has been cut from the carcass. The meat has to go through one more step where the final products are made. Except for the vacuum packed steaks, cold cutting and warm cutting will produce the same type of meat. The quality is the same regardless of the technique used under production. Products such as sausages, ground meat and bacon will be made out of the meat. The meat will then either be packaged into the final wrapping or sent to another cutting facility. The reason why meat is sometimes shipped between cutting facilities is to even out the capacity or to enable different cutting facilities to produce different products. At some stages of the production the meat can be sent to freezing to be stored for later use.

1.7 Choosing the cutting pattern

Which cutting patterns are used can have drastic results on what is being produced. If this task is left for each cutter to decide, it will be impossible for him to have an overview of which products are needed the most. The orders for the day have to be fulfilled, but how much is left to fulfil will vary during the day because of the continuous production. Today this is solved by having one person responsible for monitoring the production. He or she will know which products are needed the most and give the cutters recommendations on how to cut the carcasses. It is not possible for him to tell each and every cutter what to do. Instead; one recommendation is given for all the cutters to follow. This is done on a whiteboard with directions to follow as to what should be made out of certain parts of the animal. It is therefore not possible to make decisions for individual carcasses with this system. A large part of the carcass will be used for different selections of meat. Selections are mixtures of different kind of meat and are graded by the fat percentage. As shown in Figure 4, excess meat should be added to the 18% fat selection, the shoulder part should be used for the 5% selection and so on.

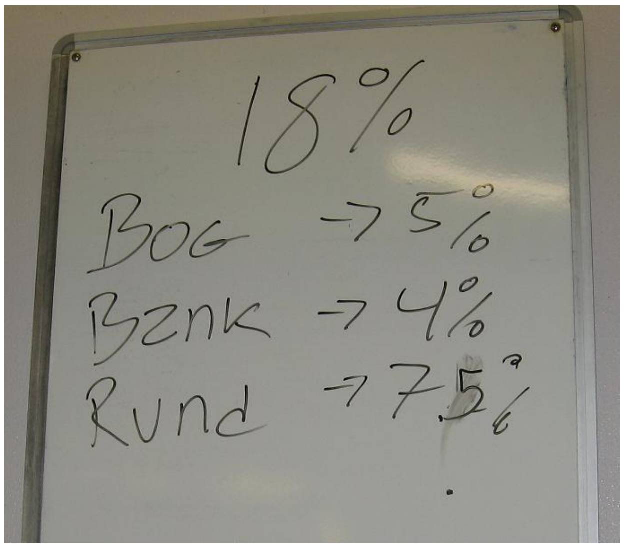

Figure 4 Today's solution

1.8 Computer aided decisions

There has already been developed a system to help chose cutting patterns for cold cutting. This method uses the measured carcasses in storage to calculate how many carcasses should be cut with each cutting pattern. An offline algorithm is an algorithm which has all necessary information available at the start of the execution [Albers97]. The method for cold cutting can therefore be called an offline algorithm since all the carcasses used for the calculations are already measured. These measurements measure the fat percentage of the carcass and give the carcass a classification. This classification defines which cutting patterns can be used on the carcass. The offline algorithm can use a large number of carcasses for its calculations. It is therefore possible to view the carcasses as a continuous amount of carcasses instead of discrete numbers without losing too much precision. Such problems can be solved with linear programming. The offline algorithm might conclude that 102.6 carcasses will be cut with a certain pattern, which is rounded off to 103. This can help the supervisor to know what to recommend to fulfil the production of the day.

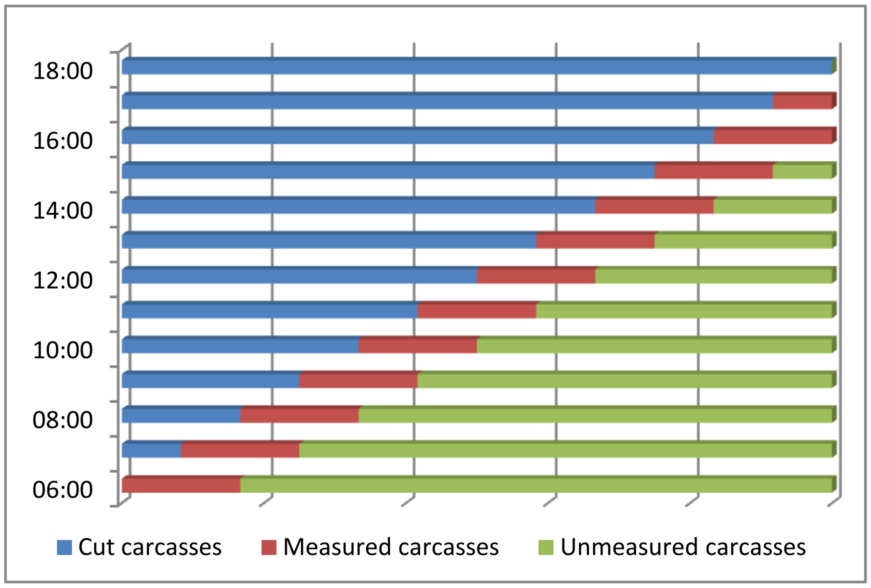

Figure 5 Carcasses during production

Not all the carcasses are measured at the same time when using warm cutting. Only a few carcasses are measured, and only these can be used to calculate which cutting pattern to use. These are represented by the red sections in Figure 5. A decision has to be made from this limited knowledge. An online algorithm is an algorithm which makes decisions before all the data concerning the problem is available [Albers03]. The method for choosing the cutting pattern to use for warm cutting is therefore an online algorithm. How many carcasses are being measured at a time can vary between production plants. Nortura's plant at Rudshøgda is supposed to have 10 measured carcasses at any given time during production. To calculate each carcass individually becomes much more important with as few as 10 carcasses, since rounding off 0.5 carcass is a significant portion of the total. More and more carcasses will have been cut throughout the day. Since the total number of carcasses for a day is constant, the amount of unmeasured carcasses will shrink accordingly. This can be seen in Figure 5 which illustrates how unmeasured carcasses become measured carcasses and then cut carcasses.

1.9 Finding good solutions

For the computer aid to be of any use, it has to give recommendations which will make it easier for the cutters to fill the production quotas from the production plan whilst making the most valuable products. Nortura has already experience with defining the goal of the computer system for cold cutting. To produce a product without an explicit need will be given a lower value than to produce a product which fills the production quota. The same system can be used for warm cutting, since warm cutting and cold cutting produce the same products.

The challenge with choosing cutting patterns for warm cutting is that only a few of the carcasses have been measured. What to produce can only be decided for the measured carcasses, since the quality of the unmeasured carcasses is unknown. It is impossible to know if a later carcass is even better suited for making a certain product than the one which is about to be cut, but it is necessary to make a decision fast. A decision support system should recommend cutting patterns which are likely to give good results when the production from all the carcasses is summed. The recommendation has to be given before new carcasses are queuing up, since this would create a bottleneck in the production. To be sure that the recommendation system does not delay the production, the maximum time for giving the recommendation should be in a matter of seconds, typically 6 seconds for a plant like the one at Rudshøgda.

1.10 Adjustments at the work place

The purpose of computer aid is to make it easier to produce close to the production plan whilst not degrading quality meat into less valuable products. The system for cold cutting makes it easier to give good recommendations for what to cut; it does not however take into account individual differences between the carcasses. The system for warm cutting would give an individual recommendation for every carcass. This would require a new system to inform the cutters. This could be done by stamping each carcass with the cutting pattern to use or by installing a monitor for each cutter.

Requirements:
-give recommendations which in average will be better than the system used today
-execute fast enough in order to not create a delay in the production
-not add a significant amount of extra work for the employees

2 Model description

This section will describe how the meat cutting problem is represented as a model. The model will represent the problem as an optimisation model using the following definitions.

2.1 Definitions

Indexes

c:	carcass, every carcass being cut is defined as an individual carcass.
a:	cutting pattern.
p:	product, what is produced from the carcasses.

Variables

$x_{c,a}$:	$\begin{Bmatrix} 1 \text{ if carcass } c \text{ is cut with cutting pattern } a \\ 0 \text{ otherwise} \end{Bmatrix}$
v_p:	regular production of product *p*. This is production which is planned in the production plan.
s_p:	surplus production of product *p*. This is production which is not specified in the production plan.
u_p:	unsatisfied demand for product *p*. This is the part of the production plan which is not fulfilled.

Constants

$P_{a,p}$:	percentage of the carcass which becomes product *p* while using cutting pattern *a*.
W_c:	weight of carcass *c*.
D_p:	quota for product *p*, how much it has been planned to be produced of this product.
L_p:	loss or penalty for each kg of demand for p not fulfilled.
V_p:	value of planned production of product *p* per kg.
T_p:	value of surplus production of product *p* per kg.

2.2 Data

This section will describe the data used in the model. The model is based on the data from Nortura's data system. Only data available from their system will be used in the model since this is the only information which can be easily gathered automatically. By using only the information from the database without any manual adjustments, less maintenance will be needed when the data is altered. It is therefore a goal to make the program as generic as possible concerning the input and be oblivious to the actual properties of the data. A generic solution will not differentiate between what the input represent as long as it uses the same format as the original problem. This means the program will be able to solve the same problems for pigs, sheep or any animal which is cut according to a cutting pattern.

Products – all the products from Nortura's databases is extracted and included in the model even if they are not relevant for today's production. To extract only the relevant products would require added complexity and make the program less flexible to changes. As long as the excess products will not slow down the calculations significantly all will be considered, otherwise the number will be reduced after loading.

Each product has a price per kilo. This price is used to calculate the value of the production. The value of surplus production can be set individually to reflect the anticipated need of the product in the future. If no value is set, a base value of B% is set while a penalty of F kroner is subtracted representing the cost of freezing the product. Typical values for B and F would be 60% and -5 kroner respectively.

Carcass type – the animals are classified according to their age, fat percentage and gender.

Cutting patterns – cutting patterns tell which products to make from a carcass. Only a few cutting patterns can be used for each carcass type. A cutting pattern contains a list of the products which will be produced when using that particular pattern. For each product in the list, there is also a percentage which tells how much of the carcass is expected to turn into that product. The sum of all the products should turn into the weight of the original carcasses. This is not always the case since there will be inaccuracies in practice. Not all possible cutting combinations are described in a cutting pattern. The most useful combinations have been picked out to limit the number of patterns. There exist other cutting patterns which would give even better results, but experience has shown that present cutting patterns are satisfactory.

Carcasses – production data is necessary to run a model of the production line. The carcasses being measured will be fed to the system one by one. Historical data of slaughtered carcasses will be used for test cases. The historical data will then be revealed one carcass at a time for the online system to simulate real time production.

Production plan – a production plan is a goal for what should be produced of each product during a time period. This time period is for Rudshøgda one day. The production plan does not have to be satisfied strictly, but it should be followed as long as no large sacrifices are made. Large underproduction will hurt the relationship to customers, while sacrifices will hurt the profit. The whole production plan for a period is available at the beginning of the period. Historical data can be used for production plans as long as the time period matches the time period of the carcasses slaughtered.

2.3 Offline Model

The offline model describes how the problem would be solved if measurements of all the carcasses were available before the optimisation model starts.

The objective function consists of three parts. The first part tries to maximise the value of the regular production. This is the production which is scheduled in the production plan. The second part tries to maximise the value of the surplus production. This production is not in the production plan, but it is still important not to let resources go to waste even if it is not an immediate demand for the product. The value for surplus production is typically a portion of the value of regular production of the same product. The third part subtracts a penalty for each product in the production plan which is not satisfied. This is a similar function to what has been used for optimisation for meat cutting of cold carcasses. There is no fundamental difference in the objectives between cold and warm cutting, and what has been learnt from cold cutting should therefore be used to make this objective function.

Objective Function 1

$$Max \sum_p V_p * v_p + \sum_p T_p * s_p - \sum_p (L_p * u_p)$$

Constraint 1 defines the relationship between chosen cutting pattern and produced products. The left side says that each carcass turns into the amount of each product which is defined by the chosen cutting pattern. The right side says the production is either turned into regular production or surplus production. Regular production will always have higher priority than surplus production since it is valued more highly in the objective function.

Subject to:

Constraint 1

$$\sum_c \sum_a (x_{c,a} * P_{a,p} * W_c) = v_p + s_p, \forall p$$

Constraint 2 makes sure only one pattern is chosen for each carcass.

Constraint 2

$$\sum_a x_{c,a} = 1, \forall c$$

Regular production cannot be greater than what is planned in the production plan.

Constraint 3

$$v_p \leq D_p, \forall p$$

The sum of the regular production and the unsatisfied production equals the production plan.

Constraint 4

$$v_p + u_p = D_p, \forall p$$

Constraint 5 and Constraint 6 define the range of the variables.

Constraint 5

$$v_p, u_p, s_p \geq 0, \forall p$$

Constraint 6

$$x_{c,a} \in (0,1), \forall c, a$$

This optimisation model is a representation of the consequences the choices of cutting patterns have. With the assumption that the data and the model represent the real problem well, a maximisation of this model will decide which cutting pattering is the best choice for each carcass.

2.4 Problem size

How easy it is to find the optimal solution depends on the size of the problem. The size of the problem can be described by the number of variables and constraints. This model contains both binary and continuous variables. Binary variables will in worst case pose the largest computational difficulties in this problem, since this problem is related to 0-1 integer programming which is NP-hard [Karp72].

Table 1 shows the typical number of carcasses, products and cutting patterns for Nortura.

Table 1 Example of problem size

	Number of elements
Carcasses (c)	200
Products (p)	1000
Cutting patterns (a)	400

The number of non-trivial constraints is shown inTable 2. Non-negativity constraints and binary constraints are here considered trivial constraints. The total number of constraints are 3 * p + c. This will with the sample from Table 1 give a total of 3200 constraints.

Table 2 Number of constraints

Constraint type	Number of constraints
Constraint 1	p (product)
Constraint 2	c (carcass)
Constraint 3	p (product)
Constraint 4	p (product)
Total	3 * p + c (3 * product + carcass)

There are three types of continuous variables in the model, v_p, u_p and s_p. This will with the sample from Table 1 give a total of 3000 continuous variables.

Table 3 Number of continuous variables

Continuous variable	Number of continuous variables
v_p	p (product)
u_p	p (product)
s_p	p (product)
Total	3 *p (3 * product)

$x_{c,a}$ are the only type of binary variables. The number of binary variables is with the numbers from the sample in Table 1 give a total of 80000 binary variables.

Table 4 Binary variables

Binary variables	Number of binary variables
$x_{c,a}$	c * a (carcass * cutting pattern)
Total	c * a (carcass * cutting pattern)

The 80000 binary variables can pose a problem. The optimisation would take very long to finish if a large number of combinations of these variables have to be explored. The numbers used here is a theoretical worst case. The numbers can be reduced considerably if domain knowledge is used to exclude irrelevant information and impossible states.

Some of the variables and constraints can be removed without changing the results. Not all of the 1000 products in Nortura's example are relevant for the problem. Most of these products are not made before packing stage of the production. Only a limited number of the 40 products are made during the cutting stage. The number of carcasses cannot be reduced since every carcass is unique when it comes to weight and carcass type combination. Even though every carcass in theory can be combined with any cutting pattern, this is not the case in practice. Only a limited number of the cutting patterns can be used for each carcass. This number is usually between 5 and 20. By only considering the possible number of cutting patterns and not every cutting pattern, the elements in c will be reduced from 1000 to 20. With these reductions, the numbers in Table 1 can be reduced to those in Table 5.

Table 5 Example after problem size reduction

	Number of elements after reduction
Carcasses (c)	200
Products (p)	40
Cutting patterns (a)	20

Even after this reduction, the time to find optimum is very much dependent on how many of the binary variables which have to be explored. It could in worst case be impossible to solve with 4000 binary variables, but the nature of the problem will most likely not be close to a worst case.

Table 6 Reduced problem

	Numbers after reduction
Constraints	320
Continuous variables	120
Binary variables	4000

2.5 Performance analysis

This section will look at a theoretical analysis of the performance of methods for choosing cutting patterns. It is a continuance of the work done in [Wikborg07] and gives insight to the problems with creating a model for the online algorithm.

2.5.1 Minimising unsatisfied demand

The objective function of the model which minimises the unsatisfied demand can be written as Objective Function 2 [Wikborg07]. The problem is written as a minimisation of cost instead of a maximisation of value because this makes competitiveness analysis easier.

Objective Function 2

$$Min \sum_{p} (u_p * L_p)$$

The competitiveness of the online algorithm A is defined as Inequation 1, where f_A is the cost of the online algorithm, f_0 is the cost of the optimal offline algorithm, b is a constant, C is the competitiveness factor and $p_1, p_2, \dots, p_n$ is a sequence of requests [Motwani95].

Inequation 1

$$f_A(p_1, p_2, \dots, p_n) - C * f_0(p_1, p_2, \dots, p_n) \leq b, \exists p_1, p_2, \dots, p_n$$

A special characteristic of this problem is the fact that the offline algorithm can have zero cost. In the case where the offline algorithm has zero cost and the online algorithm has a positive cost, the competitive ratio will always be unbounded. In the following example there will be two cutting patterns and two products with the output shown in Table 7.

Table 7 Two carcass example

	Product 1	Product 2
Pattern 1	100%	0%
Pattern 2	0%	100%

In this example there is a sequence of two carcasses, both weighing w kg. The demand for each product is also 200 kg. The first carcass can be cut with any of the cutting patterns. Which cutting pattern the second animal can be cut with is unknown at the moment the first cutting pattern is chosen.

With an adaptive adversary, the available cutting patterns for carcass #2 will simply be set to the same pattern as the carcass #1 has been cut with, see Figure 6. This will force the online algorithm to use the same pattern for both carcasses and produce w kg of one product and 0 kg of the other. Since the offline algorithm would know which pattern was available for carcass #2 it would simply have chosen the other for carcass #1 and therefore satisfied all of the demand. With the assumption that there are penalties for not fulfilling a demand, this will lead to a positive cost for the online algorithm and 0 as the cost for the offline algorithm. The right hand side b of Inequation 1 would therefore have to be equal to the weight w to satisfy the equation. Since w is a variable, no constant for b can ensure this property for every weight w, and the algorithm is therefore unbounded.

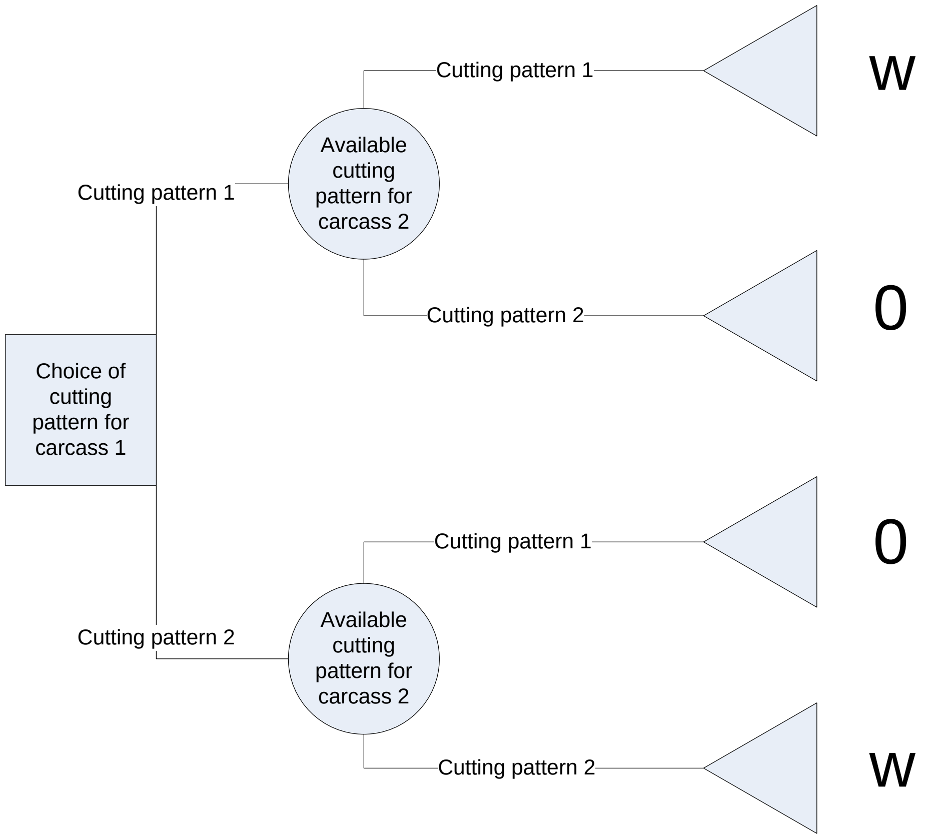
Choice of cutting pattern for carcass 1
Cutting pattern 1
Cutting pattern 2
Available cutting pattern for carcass 2
Cutting pattern 1
Cutting pattern 2
Available cutting pattern for carcass 2
Cutting pattern 1
Cutting pattern 2
w
0
0
w

Figure 6 Unsatisfied demand for the online algorithm

For the oblivious adversary, it is not possible to choose the patterns for carcass #2 to always be the same as the one used for carcass #1. The oblivious adversary can chose a random pattern to be available for carcass #2. Since the algorithm will have no way of anticipating which pattern is available for carcass #2, it will on average choose the same pattern for carcass #1 50% of the time When this happens, the penalty will be the same as with an adaptive adversary, w. When the other pattern is chosen there will be no penalty. The average penalty will therefore be w/2 while the worst case penalty will be w. Since the offline algorithm will always give perfect results without any penalties, this will lead to an unbounded competitiveness in a similar way as for the adaptive adversaries.

2.5.2 Maximising value

As shown in [Wikborg07] see Appendix A, the competitiveness of the online version of the value maximising problem is 1-competitive since it performs identically to the offline version. A value maximising problem is a problem which does not consider the production plan, but only look at the product value.

2.5.3 Combined objective function

The objective function which both minimises the unsatisfied demand and maximises the value can be written as Objective Function 3 [Wikborg07]. The objective function has been converted to a minimisation function to make competitiveness analysis easier.

Objective Function 3

$$Min \sum_p (u_p * L_p) - \sum_p (V_p * v_p) - \sum_p (T_p * s_p)$$

The first part of this function is identical to the objective function for minimisation of unsatisfied demand, while the other part increases the value of the products produced. If the constants can be assigned any value, the values for all of the V_p and T_p could be equal to zero. This will lead to an identical analysis as is the case with Objective Function 2 which leads to an unbounded competitiveness factor. Even with V_p and T_p above zero, the competitiveness will be unbounded without restrictions in the relationship between V_p, T_p and L_p.

2.5.4 Maxmin

Maxmin is a method of maximising the value of the worst case scenario. The method for achieving this is trivial for the online meat cutting problem and the complete objective function as shown in Objective Function 4 will therefore be used right away.

Objective Function 4

$$Max \sum_p (V_p * v_p) + \sum_p (T_p * s_p) - \sum_p (L_p * u_p)$$

It is not possible to guarantee any value from the unmeasured carcasses. This can be proven by the fact that there is no lower bound on the weight of the carcasses. The minimal value will therefore be the value of the measured carcasses. To create the maxmin value for the problem, the minimal value has to be maximised. This can be done by maximising the value of the measured carcasses without considering the unmeasured carcasses.

2.5.5 Minimax regret

Regret is defined as the difference between the online solution and an optimal offline solution. While a competitiveness ratio uses the ratio between the online solution and the optimal offline solution, the minmax regret method will instead minimise the regret. An example would be to choose between two options, A1 and A2.There are two possibilities of how the future will be. In one possible future, A1 will yield 60 in utility while A2 will yield 40. In the other possible future these numbers will be 10 and 20.

From the point of view of competitiveness, the worst case for A1 would be future 2 where it yields 1/2 of the maximum profit. The worst case for A2 would be future 1 where it yields only 2/3 of the maximum profit. A2 would therefore be chosen as the best alternative. By using minimax regret, the worst case for A1 would be future 2 with a regret of 10. The worst case of A2 would be future 2 with a regret of 20. This means that a different solution will be chosen by using the minimax method instead of a competitiveness ratio.

Table 8 Minimax regret example

	A1	A2
Future 1	60	40
Future 2	10	20
Maxiumum regret	20 – 10 = 10	60 – 40 = 20

Minimax regret can also be used for the meat cutting problem. With the purely profit maximising objective function, the results will be just the same as with competitiveness ratio. The offline and online algorithm will perform identical and the regret will be zero.

The example described in Figure 6 can be used to analyse how minimax regret will perform with Objective Function 3, which only emphasises on satisfying the demand. There is no guarantee that the online algorithm will be able to fulfil any more of the demand after the first cutting pattern has been chosen. The offline algorithm may on the other hand fulfil all of the demand. This can be used as an upper bound for the regret. The online algorithm will know for sure how much of the demand it can fulfil with the measured carcasses, but it cannot guarantee that any of the unmeasured carcasses will count towards the unfulfilled demand. This upper bound is easy to achieve by only considering the measured carcasses and assume that the unmeasured carcasses will produce nothing while doing the optimisation. This method will most likely not perform very well in practice. It will only look at short term benefit and in the beginning produce mainly products which have a large penalty for unsatisfied demand. This can lead to overproduction of certain products.

2.5.6 Relationship with the real problem

The production plan is usually made to reflect the animals being slaughtered that day. This means the extreme cases discussed above are very unlikely to occur. Even if it could happen, what is important for the Nortura is the average profit or the expected value and not to minimise the regret or competitiveness ratio. Since the production plan usually is achievable or close to achievable with normal production, overproduction is likely to happen. This can be prevented by adjusting the production plan to only consider a portion of the total plan. If only 10% of the animals are measured, these animals can be expected to produce 10% of the total plan. This would be a fair estimate as long as the variation between the early and the later animals is not too large. This technique will from now on be referred to as an **adjusted production plan**.

3 Online model

The online model describes how the problem would be solved if measurements of the carcasses become available one at a time while the optimisation model is run, as can be seen in Figure 5. The problem of determining what to make out of the first of the measured carcasses will from now on be called the **sub problem**, while the problem of finding out what to make out of every carcass will be called the **master problem**. The master problem therefore consists of one sub problem for each carcass which is being cut.

As shown in the performance analysis section, there is no guarantee that the unmeasured carcasses will make any valuable products at all. A natural approach to an online model would therefore be to maximise the value of the measured carcasses without considering the unmeasured carcasses. The model for this sub problem would be very similar to the offline model, but only consider the measured carcasses instead of all carcasses.

The goal of this model would be the same as for the offline model, to maximise the total value. The objective function will therefore be identical. The restrictions will look identical as well, but the difference lies in the number of variables. The $x_{c,a}$ variables will only be created for the measured carcasses and the set of carcasses c will therefore include a lot fewer members in the sub problem than in the offline model. While the offline model decide which cutting patterns to use for all the carcasses, the model for the sub problem only makes a decision for the first carcass. The model will be run again with new data for each carcass to solve the master problem.

Objective Function 5

$$Max \sum_{p} (V_p * v_p) + \sum_{p} (T_p * s_p) - \sum_{p} (L_p * u_p)$$

Subject to:

Constraint 7

$$\sum_{c} \sum_{a} \left(x_{c,a} * P_{a,p} * W_c\right) = v_p + s_p, \forall p$$

Constraint 8

$$\sum_{a} x_{c,a} = 1, \forall c$$

Constraint 9

$$v_p \leq D_p, \forall p$$

Constraint 10

$$D_p = v_p + u_p, \forall p$$

Constraint 11

$$v_p, u_p, s_p \geq 0, \forall p$$

Constraint 12

$$x_{c,a} \in (0,1), \forall c, a$$

The online model will look identical whether the adjusted production plan is used or the full production plan. The difference will be in D_p.The value of D_p will be adjusted between each instance of the sub problem. After the cutting pattern has been determined for one carcass, the production from this carcass will be added to what have been produced so far. With a full production plan, the difference between the production plan and the production so far will be used directly as a goal for the sub problem. With an adjusted production plan, only a fraction of the production plan will be used. This fraction is equal to the fraction of the remaining carcasses which are measured. An example would be if there were 60 uncut carcasses and 10 of these are measured. Only 10/60 or 17% of the unsatisfied production will be used in the adjusted production plan.

3.1 Heuristics

Heuristics can be used to create a solution for an optimisation problem without a guarantee of an optimal solution. The core of many heuristics is a local search. A local search requires a neighbourhood structure for the states of the problem. Many neighbourhood structures can be used to describe the same problem, and how the neighbourhood structure is defined can have large consequences on the performance of the search [Gendreau02]. A neighbourhood structure describes how to move from one solution to another, even if these solutions are not feasible.

A feasible solution for the meat cutting problem described in this paper is when each carcass has assigned one cutting pattern. Every carcass can be represented with a vector for all the available cutting patterns. One value in the vector will be 1 while the rest is 0. This binary value will determine which cutting pattern is used for the carcass, see Table 9. Not only is this a neighbourhood structure, but the structure also span all feasible, but no infeasible solution. It will therefore not be necessary to check for feasibility when exploring the neighbourhood.

Table 9 Data structure

	Carcass 1	Carcass 2	Carcass 3	...	Carcass N
Pattern 1	0	1	0	0	1
Pattern 2	0	0	1	0	0
Pattern …	1	0	0	0	0
Pattern N	0	0	0	1	0

By reassigning a carcass to a new cutting pattern, a move in the neighbourhood structure will be performed. This would be done in the data structure by changing one vector by moving the value 1 to another position. The search neighbourhood can be defined by a change in any single vector. For the example in Table 9, every carcass has 4 possible cutting patterns. A single cutting pattern for one carcass can therefore be changed to any of the other cutting patterns. By this definition of search neighbourhood, this can be done for only one carcass at a time. The number of neighbours is therefore equal to the number of unused patterns times the number of carcasses.

3.2 Local search

A local search,can be performed on this search neighbourhood to find a local optimum. One way of doing this is to cycle through the carcasses and to pick the cutting pattern which gives the best value for the objective function at each step. Eventually this method will no longer make any changes. This means that it has found a local optimum since none of the neighbours has a higher value [Rardin98]. A feasible solution can be a good start location for the local search. One possibility would be to assign cutting pattern 1 to all the carcasses. This would lead to the same solution every time. Another possibility would be to assign random cutting patterns to every carcass. By running the model again, a new starting position would be used and a local optimum would be found again. This optimum can be a worse optimum, the same, or hopefully a better one. If the first optimum found is the local optimum B in Figure 7, a new starting position further to the left might find the global optimum A instead.

The easiest method to use the neighbourhood structure to improve a solution is to perform a local search. A local search searches through its neighbourhood for a local optimum. There is no guarantee that the local optimum is the best global solution. This is illustrated in Figure 6. Point B is a local optimum, and none of its neighbours have a higher value. Point A does however have a higher value and is for this graph segment a global optimum.

A local search can be performed on the data structure described for the meat cutting problem. This can be done by cycling through all the carcasses and pick the cutting patterns which maximises the total value. Since changing the cutting patterns will change the total production made of each product, a product which used to be in demand can suddenly become abundant. Since this will make other cutting patterns seem more valuable, changes might have to be done to carcasses already assigned a cutting pattern. The changes will always increase the objective value. Only a limited number of changes can therefore be made before a local optimum is found. The local optimum might not be the global optimum. To increase the chance of finding a good optimum, the algorithm can be run several times with different starting positions.

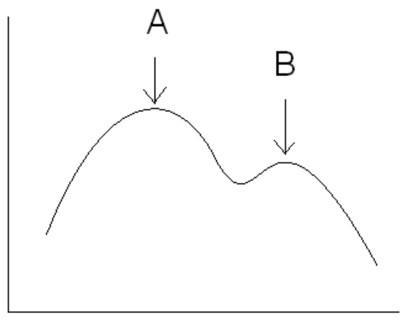

Figure 7 Global and local optimum

An advantage of the local search is that it is generally a simple algorithm which will execute fast. The results can vary a lot depending on the problem's neighbourhood structure and how easily the algorithm can get stuck in a local optimum which is much worse than the global one.

3.3 Genetic Algorithms

One type of heuristics is based on the evolutionary processes seen in nature. The genes of an organism form the basis traits of the organism. Changing these genes will also change the traits of the organism. The genes can change through different methods from generation to generation. Reproduction will let the next generation inherit some of the genes from the ancestors, while mutations can form new genes which has never been in any of the ancestors before.

Genetic algorithms represent the data as a gene structure [Reeves93]. Each gene represents a part of the solution. In the data structure described in Table 9, each column can be considered a gene while the whole table is an individual. An individual can therefore be looked upon as one possible solution, although it might not be a good solution. Methods for improving these solutions have been developed with inspiration from the evolution seen in nature.

An initial population is needed to base the evolution on. This population consists of solutions which are made by assigning all the genes randomly. This can be seen in Figure 10 where there is no common origin for the genes.

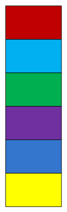
Figure 8 New individuals

Mutations are made by choosing some genes from an existing individual while randomly assigning the other genes. Figure 9 shows how some of the genes are kept from the white individual, while 3 new genes are assigned random values.

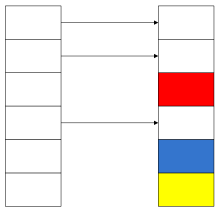
Figure 9 Mutations

Crossovers from the survivors can be seen as children of the survivors. They are made by taking genes from two other individuals and mixing them together into a new individual. Figure 8 shows how a white and a blue individual are crossed to make a third individual with genes from the two other.

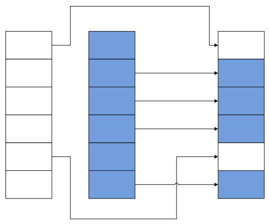
Figure 10 Crossovers

3.3.1 **Evolution through generations**

Different individual will have different combinations of genes. A group of individuals is called a population. The goal of genetic algorithms is to evolve fit individuals through evolution. A fitness function is used to evaluate how fit an individual is. A fit individual is the term used for good solutions. The fitness function for the online problem would therefore be the objective function seen in Objective Function 5.

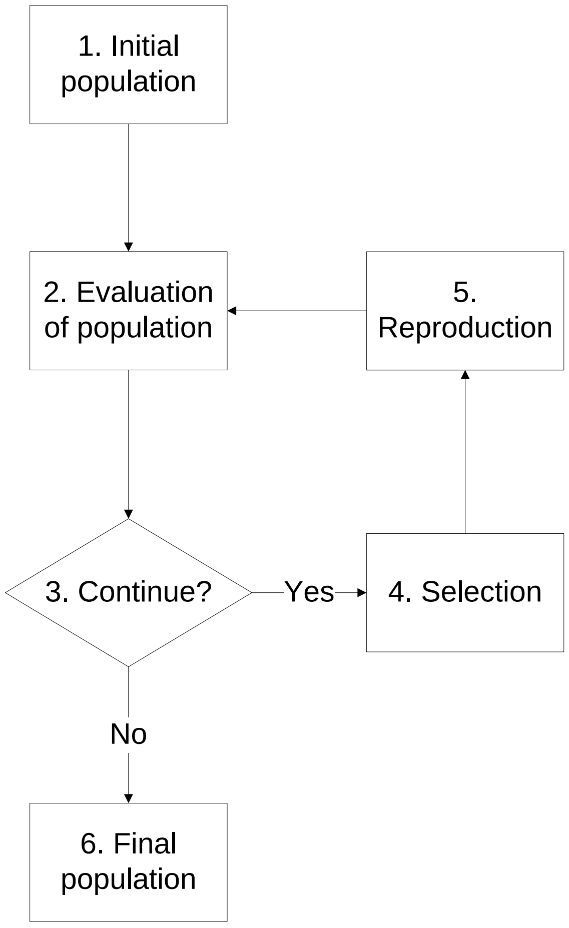

Figure 11 Genetic Algorithm

Genetic algorithms can be broken down into 6 steps as seen in Figure 11.

1) An initial population is generated. This can be done randomly by generating a number of individuals by giving them a set of random genes.
2) Evaluation of population. In this step the population is evaluated by calculating the fitness value of the individuals in the population.
3) Continue? In this step a choice has to be made to either accept the current solution or continue to look for better solutions. When to quit is often decided either on the fitness of the best solution or after a certain number of iterations of the algorithm.
4) Some of the individuals are chosen to survive to the next generation. Each iteration of the loop consisting of steps 2, 3, 4 and 5 is considered a generation. The fittest individuals are typically chosen to survive while the others are removed from the population.
5) The individuals which got selected in step 4) are used to form the basis of a new generation of individuals. There are 4 types of individuals in the next generation.

   i. Survivors from the last generation
   ii. Brand new individuals, see Figure 8.
   iii. Mutations from the survivors, see Figure 9.
   iv. Crossovers from the survivors, see Figure 10.

   The population can be kept constant by always creating as many new individuals as the ones being removed in step 4. The algorithm will go back to step 2 after step 5 is finished and a new generation is ready to be evaluated.

6) The final population is the current population at the last generation. The population becomes better and better with each generation, and the final population is therefore likely to consist of

fit individuals. The best of these individuals can be chosen as the final solution and be considered the output of the algorithm.

4 Implementation

4.1 Offline solution

The offline solution is very closely related to the offline model. Any implementation of the offline model would result in an offline solution. Since the model is a mixed integer programming model, a method for solving mixed integer programs is needed. A specialised language for mathematical programming is very well suited for solving the problem. Regular programming languages such as C++ or Java can also be used, but would require additional programming since the model is harder to represent without a mathematical programming language.

4.1.1 Implementation tools

Various tools can be used to implement the model. Certain requirements have to be fulfilled for a tool to be well suited.

Requirements:

1. Easy to learn
2. Giving good results
3. Fast development
4. Affordable
5. Well suited for the model

Xpress from Dash Optimization has to be evaluated based on these requirements in Table 10. Xpress is optimiser software for solving various kinds of optimisation problems such as linear programming problems and mixed integer problems. Xpress includes its own development environment called Xpress-IVE for writing constriction based programs with the programming language Mosel.

Table 10 Evaluation of Xpress for offline model

Requirement	Description	Degree of fulfilment
Easy to learn	Already known by the developer	High
Giving good results	Solves the model to optimality.	High
Fast development	The developer has experience with the tool and will therefore work faster than with new tools	High
Affordable	Sintef has already paid for a licence for the product. Other companies which want to develop the program further would have to acquire their own licence.	Medium
Well suited for the model	Xpress has efficient algorithms for solving linear programming problems and mixed integer problems [Dash08]. This is the kind of problems which has to be solved for the model.	High

Xpress scores overall very well on the requirements. The largest disadvantage is the license price other developers would have to pay if they want to further develop the program. This would have been less of a problem with open source tools or less expensive tools. The goal of the offline problem is to make an optimal solution which the online solution can be compared with. As a benchmark for the online solution, there will be little need to develop the offline solution further after the online model is finished since it will not be used for production.

Xpress fulfil requirements 1 and 3 in a way no other tool would be able to, since Xpress is the only tool for solving mixed integer programs which the developer has previous experience with and SINTEF has been using. The time it would take to learn a new tool and the risk of not performing as well as anticipated is too large for any other tool to be considered. Xpress will therefore be chosen without evaluating other options.

4.1.2 Input data

All the input data needed for the offline model is available from Nortura's databases. The most challenging data to format is the cutting patterns. The cutting patterns are originally represented by a list of entries with 4 values. The ID of the cutting pattern, the ID of the product produced, the carcass type the cutting pattern is used on and how much will turn into this product. Multiple entries can have the same cutting pattern ID, because the cutting pattern can produce more than one product. A new entry will therefore be added for each product a cutting pattern produces. An example with 2 cutting patterns, 3 products and only one carcass type can be seen in Table 11.

Table 11 Cutting pattern format

Cutting pattern ID	Product ID	Carcass Type	Yield percentage
1	100	10	50%
1	101	10	20%
1	102	10	30%
2	101	10	60%
2	102	10	40%

The data in Table 11 is represented by $P_{a,p}$ in the offline model. $P_{a,p}$ can be represented by a two dimensional matrix with cutting patterns and products as the two indexes. Each cell in the matrix will represent how much of product p will be produced by using cutting pattern a. The data from Table 11 will be represented by Table 12 in this format.

Table 12 Cutting pattern matrix

	Product 100	Product 101	Product 102
Cutting pattern 1	50%	20%	30%
Cutting pattern 2	0%	60%	40%

Some information has not represented in the new table. There is no longer a carcass type associated with each cutting pattern. This information will be represented elsewhere by only creating $x_{c,a}$ variables for the carcasses which can be cut by cutting pattern a, see Pseudo Code 1.

```
forall (Carcasses c) do
    forall (Cutting patterns a) do
        if ( Usable pattern (c, a) equals true ) then
            create( x(c, a) )
        end-if
    end-do
end-do
```

Pseudo Code 1 Creating relevant variables

The other data such as product price, planned production and the input carcasses are stored in the database in a similar format as the one used in Xpress with only minor adjustments needed.

4.1.3 **Programming**

The offline model is programmed in a language named Mosel. Mosel is a programming language designed for mathematical modelling. The constraints in the offline model can therefore be modelled directly in Mosel. Constraint 1 can be written as Pseudo Code 2. The pseudo code is identical to the constraint in the original model, but is no longer written as a mathematical formula.

```
forall (Products p) do
    SUM(Carcasses c, Cutting Patterns a) x(a, c) * P(a, p) * W(c) = v(p) + s(p)
end-do
```

Pseudo Code 2

All the constraints have to be modelled in a similar way before any optimisation can be done. The optimisation is done by defining the objective function and asking the Xpress solver to either minimise or maximise this function. The objective function can be written as Pseudo Code 3. No specific value for surplus production has been defined, and the value is therefore set to B of regular price –F kroner per kilo.

```
Value = SUM(Products p)( V(p) * v(p) + ((V(pr) * B)- F) * s(pr) -  u(pr)*L(pr) )
```

Pseudo Code 3 Objective function

The value defined in Pseudo Code 3 will be maximised to calculate which cutting patterns each carcass should be cut with to create the most value. The value of x(c, a) determines whether the cutting pattern should be used for carcass c or not, while the amount produced of each product is stored in the variables for regular production v(p) and the variables for surplus production s(p).

4.2 Online solution

The model can be solved optimally like the offline model or heuristics can be used to find approximate solutions.

4.2.1 Implementation tools

The model to be solved has no fundamental differences from the offline model. The only difference is the data the model uses, since the data is adjusted according to which carcasses have been measured. There is however additional requirements for the implementation of the online model. It has to be run for each carcass during production. That means the calculations have to be finished and present the results for the cutter in a matter of seconds. A larger execution time can create

The requirements

Requirements:

1. Easy to learn
2. Giving good results
3. Fast development
4. Affordable
5. Well suited for the model
6. Fast execution time

Xpress can be used to find optimal solutions to the sub problems in the online model. An evaluation has been done in Table 13. Xpress is a good choice for all the same reasons it was for the offline solution, but for the additional requirement, fast execution time, it does not perform as well. An exact solution can take more time than what is available at the cutting plant, which will result in a bottle neck in the production. Another problem is the licensing costs. A solution will need adjustments when the cutting plant request changes and licenses for Xpress is a considerable expense.

Table 13 Evaluation of Xpress for optimal solutions of the sub problem

Requirement	Description	Degree of fulfilment
Easy to learn	Already known by the developer	High
Giving good results	Solves the model to optimality.	High
Fast development	The developer has experience with the tool and will therefore work faster than with new tools.	High
Affordable	Sintef has already paid for a licence for the product. Other companies which want to use the program further would have to acquire their own costly licence.	Medium
Well suited for the model	Xpress has got efficient algorithms for solving linear programming problems and mixed integer problems. [Dash08]. This is the kind of problems which has to be solved for the model.	High
Fast execution time	Xpress will search for an optimal solution for the model. Since the model is a mixed integer problem, the execution time can be hard to predict with no certain upper limit.	Low

The heuristics described in the online model section can be realised with various programming tools. This could for example be done in Visual Studio with C++ or Eclipse with Java. C/C++ is the de facto standard for implementing heuristics and will therefore be used.

Table 14 Evaluation of Visual Studio with C++ for heuristic solution of the sub problem

Requirement	Description	Degree of fulfilment
Easy to learn	Already known by the developer	High
Giving good results	Heuristics can give varying degree of optimality, but execution time can be sacrificed for better results.	Medium
Fast development	The developer has experience with the tool and will therefore work faster than with new tools. C++ is often considered more vulnerable to bugs than other programming languages, which can prolong the development time.	Medium
Affordable	Visual Studio available through NTNU or through the free Visual Studio Express Edition.	High
Well suited for the model	C++ is well suited for implementing heuristics since it has got libraries including the needed algorithms and data structures.	High
Fast execution time	Heuristics can be adjusted to execute fast by sacrificing guaranteed optimality. C++ is a programming language which is known to run fast because of how it is compiled into machine code.	High

As can be seen from Table 13 and Table 14, the different methods have different advantages and disadvantages. The heuristic solution is assumed to be faster than the exact method, while the exact method gives better results. Both methods are well known to the developer and implementation should therefore not be a problem. However, a major concern is the cost of Xpress. Licensing of commercial optimisation solutions can be a costly affair, and a free alternative should be used if the performance is equal.

4.2.2 Program structure for the master problem

Various optimisation methods can be implemented in C++. Some functionality will always be shared by these implementations. The data has to be loaded into the program, the results have to be written somewhere. This makes up a common framework which the optimisation methods have to communicate with. This framework should provide the optimisation method with the measured carcasses and the production plan, while the optimisation method will provide the cutting pattern with which to use for the first carcass. The framework will then update the production plan according to the products made from the carcass, and ask the optimisation method to calculate what to do with the next carcass. The structure of the program can be seen in Figure 12.

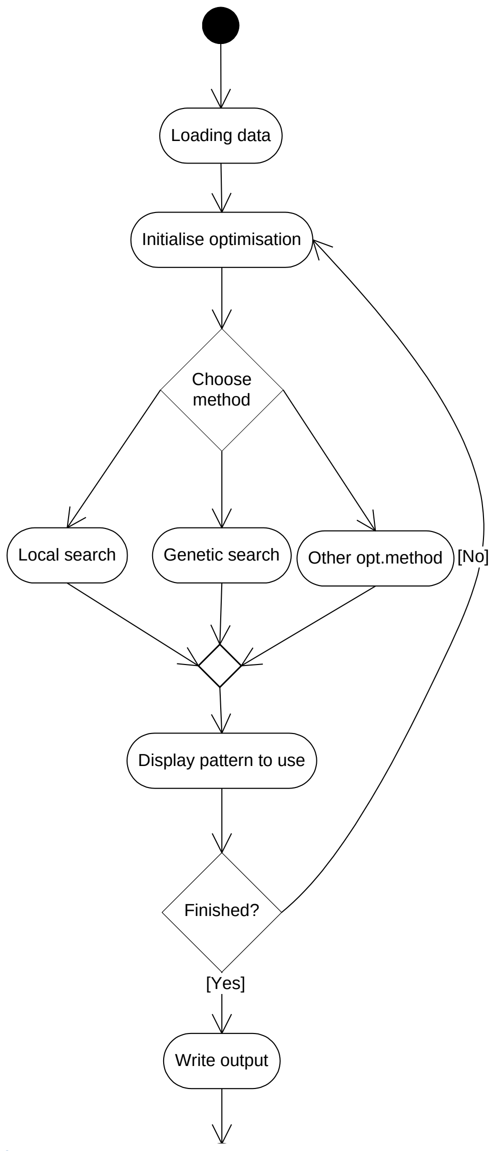

Figure 12 Program structure

Loading data: The data described in section 2.2 is loaded in this phase and loaded into appropriate data structures.

Initialise optimisation: This phase prepares the data for the optimisation method. The produced amount of each product will be summed. This amount will increase for each carcass which has been cut. Then the measured carcasses will be updated. For the first iteration, a given number of carcasses will be added as measured carcasses to be used for the optimisation. For subsequent iterations, the first carcass will be removed from the list of measured carcasses, because it has been cut, and a new carcass will be added to keep the number of measured carcasses constant. The number of unmeasured carcasses will be reduced by one.

Choose method: This phase will simply choose which optimisation method to use. This is typically predefined by the user.

Optimisation method: This phase solves the sub problem by determining which cutting pattern to use on the next carcass. Many methods can be used to achieve this goal, these methods are further explained in section 3.

Display pattern: This phase simply displays the result from the optimisation method. This is where the cutters get the information as to which cutting pattern should be used for the current carcass.

Finished? If there are no more carcasses to be cut, the program will start writing the outputs. If there are still more carcasses left, another iteration will be performed to determine what to do with the next carcass.

Write output: The program has now calculated the production of each product after all the carcasses have been cut. This can either be displayed as the total production of each product or the production compared with the production plan.

4.2.3 Data structures

The data described in section 2.2 has to be represented efficiently to allow fast access and execution. A simplified figure of the relationships between the data types can be seen in Figure 13. "Carcasses" is a list of the measured carcasses. Each carcass has a weight and a carcass type. Each carcass type can be cut with a predefined number of cutting patterns. No cutting pattern can be used for more than one carcass type. Each cutting pattern turns a percentage of the carcass into each product. The production plan defines how much of each product is the production target.

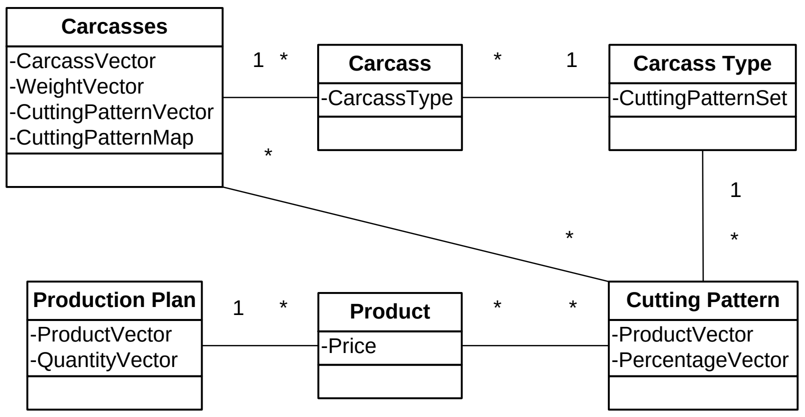

Figure 13 Data relationships

While loading the data, the cutting patterns are read line by line in the format show in Table 11. New elements have to be added each time a new carcass type or cutting pattern is encountered. Each carcass type is placed in a map linking its ID to the appropriate object. The same is done for cutting patterns. This makes it much easier to set up the appropriate linking since the numbers read in from the data sources can be used to look up the objects in the maps instead of traversing long arrays.

During execution of the heuristics, the most common operations are to assign cutting patterns to the carcasses and to calculate the value of the chosen cutting patterns. The cutting pattern is assigned by putting the Cutting Pattern in the CuttingPatternVector which matches the Carcass in the CarcassVector. To find which cutting patterns can be used for that particular carcass, the following steps have to be followed:

-access the carcass from the CarcassVector in Carcasses
-find the carcass type from CarcassType in Carcass
-access the Cutting Patterns in the CuttingPatternSet in CarcassType

This set will contain all the cutting patterns which can be used for the carcass.

To calculate the value of a set of cutting patterns, the total production of each product has to be summed up, as can be seen in Pseudo Code 4. The value of the production depends on the production plan, since some of the products are considered regular production while others are surplus production. Penalties will also be added for unsatisfied demand.

```
forall( Carcasses a ) do
    forall( Products p ) do
        production(p) += weight(a) * PercentageVector(CuttingPatternVector(a))
    end-do
end-do
```

Pseudo Code 4 Summation of the production

The calculation of the total value of the normal production, surplus production and penalties is done in Pseudo Code 5. The normal production is given full price, while the price of the surplus production is reduced to B% of the regular production and given a penalty of F kroner per kilo.

```
forall( Products p ) do
    value += product.price * MIN( production( p ), ProductionPlan( p ) )
    value += ((product.price *  B ) –  F ) * MAX(  production( p ) –     ProductionPlan( p ) , 0 )
    value -= product.penalty*(ProductionPlan( p ) – MIN(production( p ), ProductionPlan( p ))
end-do
```

Pseudo Code 5 Calculating value

4.2.4 **Local search**

This section will explain how the local search is implemented.

```
continue = true
while(continue) do
    continue = false
    bestValue = calcValue(currentSolution)
    forall(Carcasses c in currentSolution) do
        tempSolution = currentSolution
        forall(Cutting Patterns a in c) do
            c.assign(a)
        if(calcValue(tempSolution) > tempValue(currentSolution) do
            currentSolution = tempSolution
            continue = true
            end-if
        end-do
    end-do
end-do
```

Pseudo Code 6 Local search

The local search is a simple algorithm which goes through every measured carcass and assign the cutting pattern which makes the total value the highest. Each change can make chain reactions for the other cutting patterns, and every carcass therefore has to be evaluated again when a change is done. How this is done can be seen in Pseudo Code 6.

4.2.5 Genetic algorithm

The genetic algorithm described in section 3.3 has been implemented from scratch in C++. This section will explain the implementation with pseudo code. The first stage is to create an initial population. This can be done as shown in Pseudo Code 7. New solutions are created by assigning random cutting patterns to all the carcasses in the solution.

```
while(population.size < desired size) do
    new solution
    forall(Carcasses c in solution) do
        c.assign(Random cutting pattern)
    end-do
    population.add(solution)
end-do
```

Pseudo Code 7 Generating a population

Each solution has to be evaluated after the initial population has been generated. The results from these evaluations can be used to rank the solutions according with their fitness.

```
forall(Solutions s in population) do
    calculateValue(s);
end-do
sortByValue(population)
```

Pseudo Code 8 Evaluation population

The next generation is first populated by the best from the last population, Pseudo Code 9. These individuals make sure the best genes survive and make the basis for the crossovers and the mutations.

```
forall(Survivors) do
    nextPopulation.add( population.remove(best) )
end-do
```

Pseudo Code 9 The best solutions survive

Crossovers are made by taking two solutions from the surviving population are chosen as parents. For each gene in the new solution, there is 50% chance of taking the gene from the first parent and an equal chance of taking the gene from the other parent, as can be seen in Pseudo Code 10.

```
forall(Crossovers) do
    new solution
    parent1 = population.get(Random Survivor)
    parent2= population.get(Random Survivor)
    forall(Carcasses c in solution) do
        if( random(50%) ) do
            c.assign(parent1.get(c))
        else do
            c.assign(parent2.get(c))
        end-if
    end-do
    nextPopulation.add(solution)
end-do
```

Pseudo Code 10 Crossovers

Mutations are done by creating new solutions which are partly based on a single parent and partly based on random mutations, see Pseudo Code 11. The degree of mutation can be chosen by adjusting the mutation chance. Most of the genes will typically be taken from a parent while some are mutated.

```
forall(Mutations) do
    new solution
    parent = population.get(Random Survivor)
    forall(Carcasses c in solution) do
        if( random(Mutation Chance) do
            c.assign(Random Cutting Pattern)
        else do
            c.assign(parent.get(c))
        end-if
    end-do
    nextPopulation.add(solution)
end-do
```

Pseudo Code 11 Mutations

Some newborns are added to the solution, see Pseudo Code 12. Newborns are created exactly like they were for the initial solution.

```
forall(New born) do
    new solution
    forall(Carcasses c in solution) do
        c.assign(Random cutting pattern)
    end-do
    nextPopulation.add(solution)
end-do
```

Pseudo Code 12 New born

The nextPopulation will become the current population, and the whole process will be done all over again. This will continue until enough generations have passed away. The best solution from the current population will be chosen as the final solution.

5 Results and discussion

This section will present and discuss the results of the optimisation program. The data used to produce these results are taken from the production of one of Nortura's cutting plants in June 2007.

5.1 Results of the offline algorithm

The results of the offline algorithm will be a goal for the online algorithm. Results of the exact offline algorithm can be found in the appendix.

The offline model has been run with carcass data and the production plan from 4th June 2007 in Figure 14. Take notice of the logarithmic scale. The total value from this production is 912 174 kroner and 95 carcasses were cut during the day.

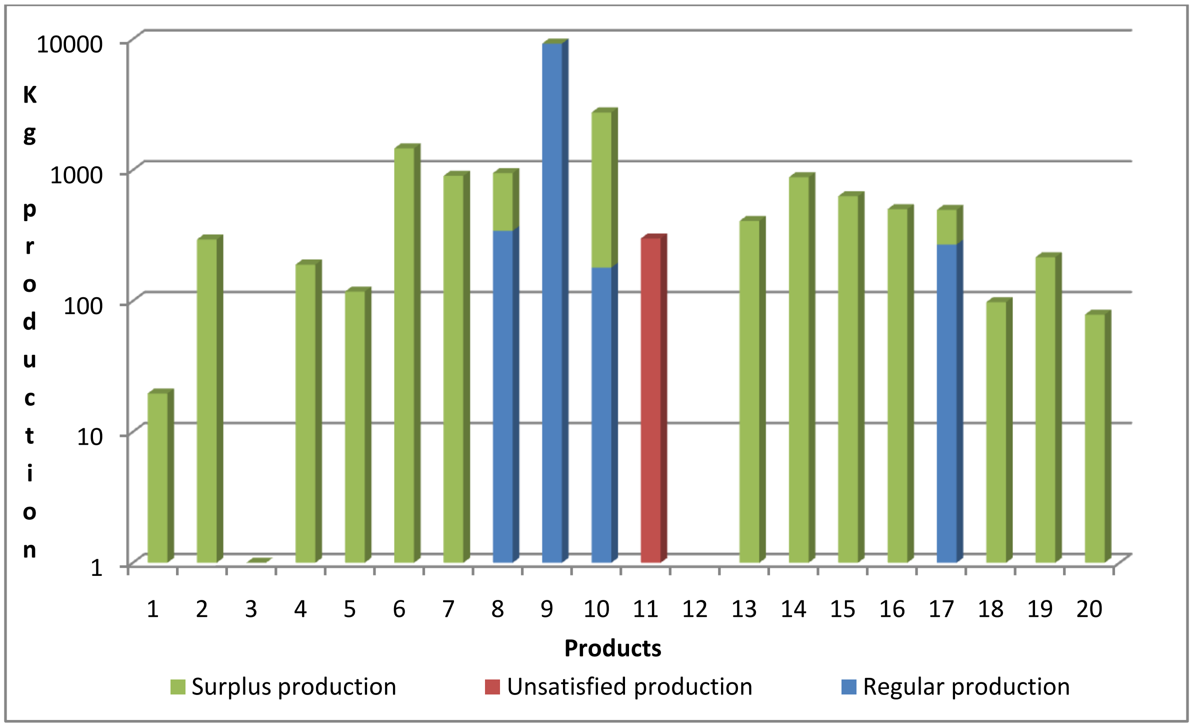

Figure 14 Offline results from 4th June 2007

Product 11 stands out particularly in this graph. It is plenty of demand for this product, but nothing is being produced. This reflects a weakness in the model. Product 11 is tendons, which will always be produced during production of other products. It is therefore not included in all of the cutting patterns, and the optimisation algorithm will not register any production of it. Some of the cutting patterns have tendons included, but the value of tendon is very small compared with the value of other products. This will not affect the production since the production of tendons is independent from the cutting pattern that was used. It does however make it harder to use the system for planning since these products have to be handled separately.

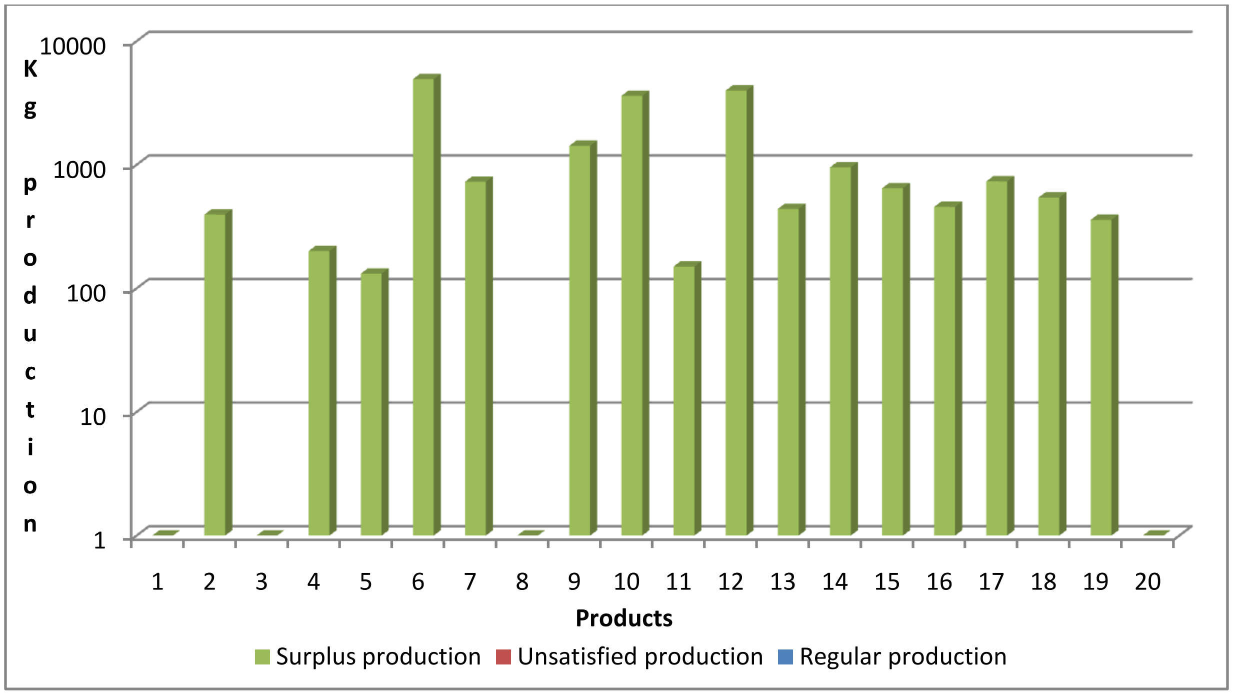

Figure 15 4th June 2007 without a production plan

Figure 15 shows the production for the same day without a production plan. The most valuable products will always be produced since there is no production plan to consider. In this situation, each carcass of the same type will be cut with the same cutting pattern since there is no production plan to fulfil.

The production of product 9 matches the production plan closely in Figure 14. It seems like this production is forced up by the production plan, and it is therefore likely that the production would be lower without the plan. Figure 6 shows that this is exactly what happened. The production of product 9 fell in favor of product 6 and 12. Nothing at all is produced of product 1 without the production plan. This is because product 1 is a byproduct of some of the cutting patterns which produce the products demanded in the production plan.

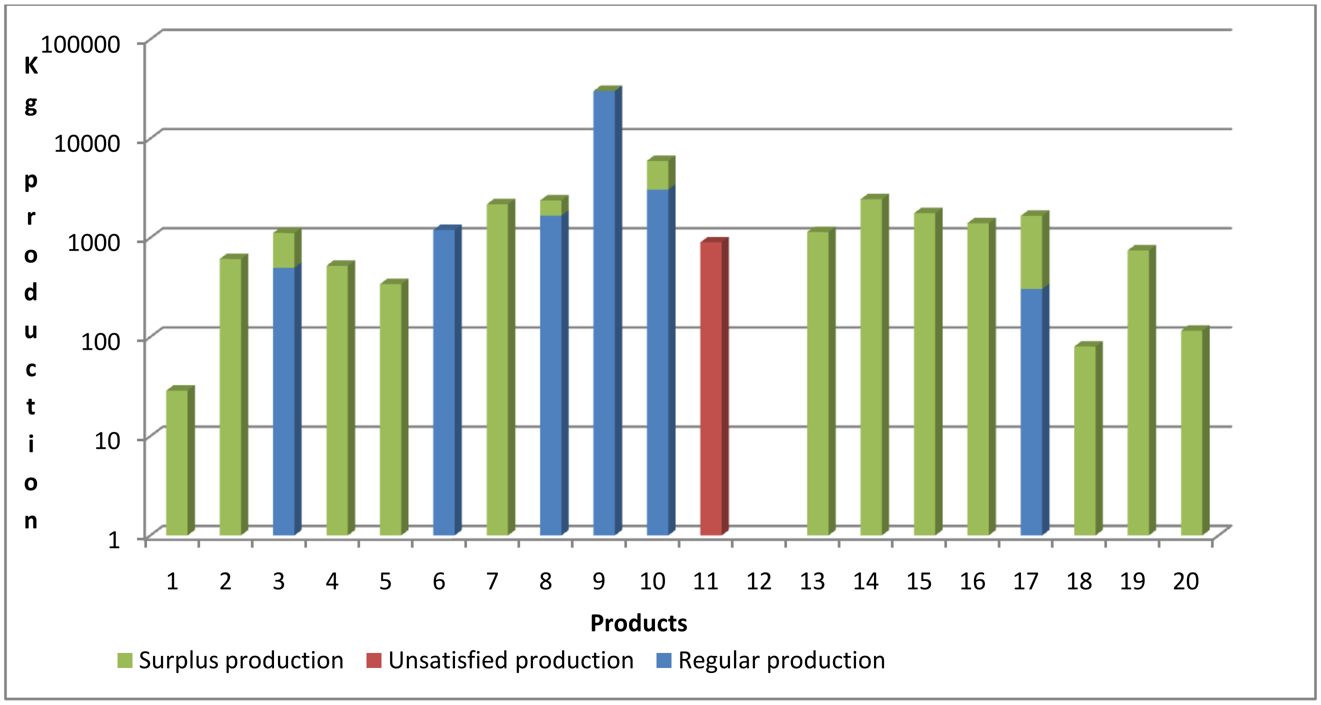

Figure 16 Production from 4th of June to 6th of June

Except for the tendons, the offline algorithm fills up the complete production plan in all the test cases. This can be seen in Figure 16 which shows the total production when the production for 3 days are calculated together. The plan for product 3 and 8 are fulfilled even if none of these products would have been produced without the production plan. Product 6 and 9 is matched even closer to the plan with little surplus production.

The conclusion of the results from the offline algorithm is that the results are what the cutting plant need. There can be small mistakes for certain products, but in this case it is because of the lack in the cutting patterns and not because of the algorithm.

5.2 Convergence of the genetic algorithm

This section will evaluate the genetic algorithm described in section 3.3. The genetic algorithm can give varying results because it is a heuristic. The parameters in the algorithm should be adjusted to give a good trade off between speed and solution quality. Speed can be sacrificed for better results by either performing more generations of the algorithm or by having a larger population in each generation. Both mutations and crossovers can be important for the population. How important one is over the other varies a lot depending on the problem type [Luke98]. The offline problem using the production on 4th of June will be used as an example to test the algorithm. Different settings shown in Table 15 are used to evaluate how the algorithm converges.

Table 15 Parameter settings for the genetic algorithm

	Regular population	Small population	No Mutations	No crossovers	No new born	Fewer survivors	More survivors
Survivors	200	100	200	200	200	50	400
Crossovers	400	200	400	0	400	400	400
Mutations	400	200	0	400	400	400	400
Newborns	400	200	400	400	0	400	400

The exact results can be seen in Table 22 in the Appendix. Two graphs will be used to represent these results. Figure 17 shows the first 10 generations while Figure 18 shows the next 90 generations. This is done to improve the readability due to the large variations in scale when the algorithms converge.

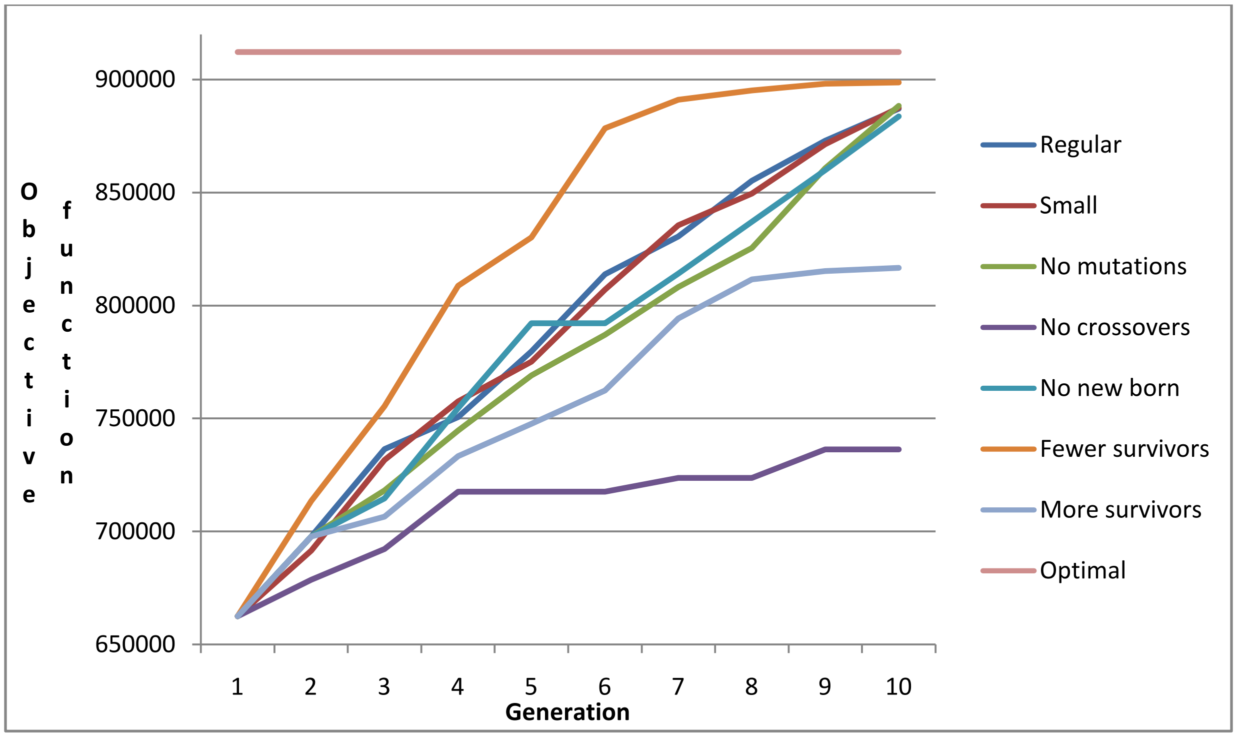

Figure 17 Generation 1-10 of offline 4th of June

A few settings stand out the most during the first 15 generations. The setting with fewer survivors converges much faster than the other settings. The new generations are in a larger degree based on good solutions since only the very best are picked out to survive. The setting with increased number of survivors does the opposite. There is less change with each generation since solutions with bad objective value are allowed to survive. This is what can be expected based on literature on genetic algorithms, since there is a higher degree of selection while the diversity is lower. By far the worst setting is the one without any crossover. Little is brought with it from generation to generation since the surviving solutions are not allowed to be merged, and the algorithm resembles closer to a guessing game by generating random changes. The rest of the settings perform almost equally well, and it is hard to say if one is better than the other.

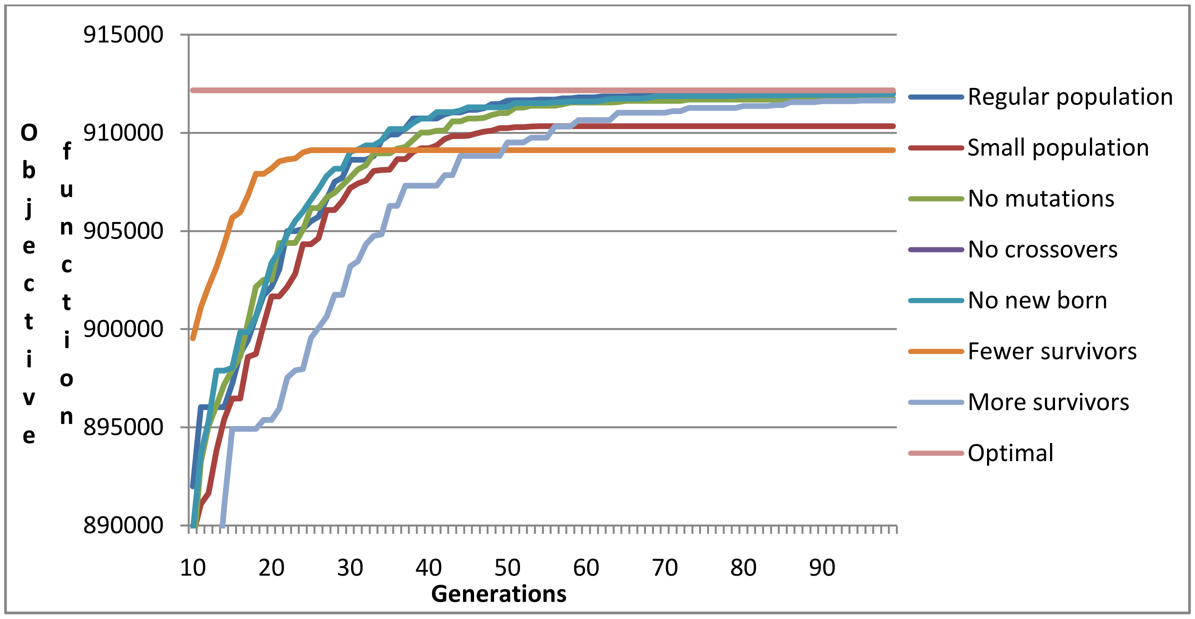

Figure 18 Generation 11-100 of offline 4th of June

The setting without crossovers has been omitted from the graph for the last 90 generations because it is too far behind. The setting with fewer survivors starts to slow down soon and converges prematurely. Few survivors have caused the diversity to be too low and the right gene combinations for a better solution cannot be found in the gene pool. The setting with more survivors takes longer, but catches up with the other solutions slowly. The small version lags behind and comes to a halt substantially worse than the regular solution. The regular solution ends up as the best heuristic, but the differences without mutations or new born are very small. This clearly shows that new born and mutations are much less important than the crossovers. Research shows that mutations or new born are vital to keep the diversity of the gene pool [Mauldin84]. Diversity generally increases the value of the point of convergence, but requires more generations to give a good solution. Keeping many solutions between each generation also helps to maintain diversity. The graph shows that reducing the number of survivors has bad effect on the long term results. From what can be learned from these conclusions, a good combination of speed and good solutions would be to keep the regular setting but decrease the number of mutations and new born. These settings will have to be tweaked for each problem size. For the online problem with only 10 carcasses, fewer generations and smaller populations will be needed. Testing has showed that 20 generations with 100 survivors, 200 crossovers and 50 of both mutations and newborns is adequate for convergence in cases with 10 carcasses.

The genetic algorithm performs a global search in the search space. Even though the algorithm seems to have converged, there might be better solutions in the nearby neighbourhood. This can be exploited by running a local search after the genetic algorithm has finished. That would drive a good global solution into a local optimum. The "Regular" setting in Figure 18 performs best of all the heuristics in the graph with a final value of 912033. This is however still not the local optimum. If a local search is performed on the solution with value 912033, it will be further increased to 912104. To perform a local search on the solution from the genetic algorithm will never make the solution worse, and is therefore a safe improvement to the genetic algorithm.

5.2.1 Discussion of the genetic algorithm

The quality of a genetic algorithm often depends on the data structure which it is based on and the problem it solves. The problem solved here reaches 99.992% of the optimal objective function. 0.008% is insignificant for this kind of problems since the inaccuracies elsewhere is very large in comparison. The inaccuracies in the measurement equipment and the variation between the cutters can be 100 times larger [NorStaff08]. The algorithm can be considered as good as an exact algorithm for all practical purposes.

Today's solution for cold cutting solves a similar problem to that of the genetic algorithm. It has therefore been asked by Trond Malmo [NorStaff08] from Nortura if it is possible to also use this method for cold cutting of carcasses. Cold cutting optimisation does not handle each carcass individually, but is based on a LP solution which is then translated into whole carcasses by a heuristic. To solve the cold cutting problem for individual carcasses should therefore theoretically be able to outperform today's solution. It has been shown here that the genetic algorithm is able to solve problems with a hundred carcasses, and it will be able to solve the amount of carcasses used for cold cutting in a matter of minutes. The question will therefore be how different are the problems. The cold cutting optimisation program does today include prognoses for future demand and spans over a larger operational field than what has been discussed in this thesis. Storage and logistics is as much a focus as the cutting patterns in itself. Even though cold and warm are partly overlapping problems, the differences are too large for any direct comparison.

5.3 Results from the online algorithm

The online algorithm uses the genetic algorithm which has been tested out in section 5.2 to solve the sub problem. This is then done once for each carcass to solve the master problem such as explained in section 3. Whether to use the full production plan for the sub problem or adjust the production plan was discussed in section 2.5. To use the full production plan would give the theoretically minimum regret, but it was acknowledged that this did not relate well to the actual problem. A comparison between the methods is shown in Figure 19. The exact results can be found in Table 23 in the appendix. The horizontal axis shows the number of measured carcasses, while the vertical axis shows the objective value for the master problem. To the far right at 95 carcasses, both methods perform identically since both methods will be a heuristic solution to the offline problem. To the far left, the method using an adjusted production plan outperforms the methods using a full production plan significantly. Using the full production plan lacks any foresight into the future, and is therefore outperformed by the method which depends on the quality of future carcasses. The method using the full production plan will not be considered any further because of its poor performance.

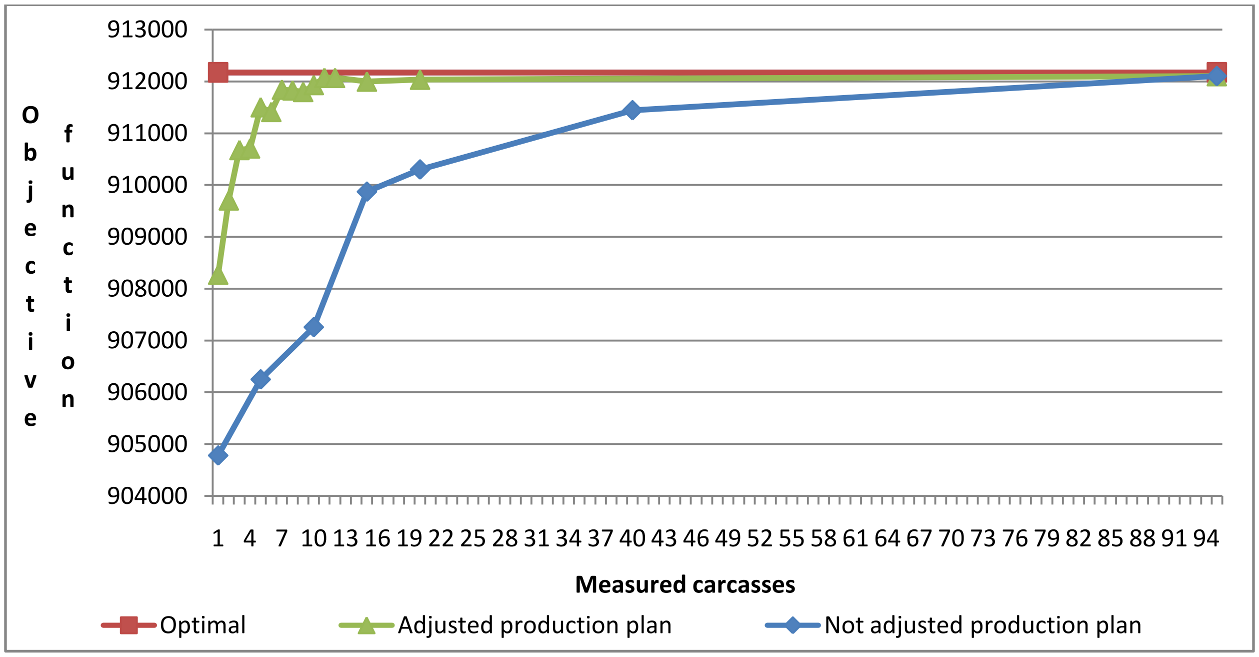

Figure 19 Online algorithm results 4th of June

As can be seen in Figure 19, the quality of the online algorithms depends on the number of measured carcasses it uses for the sub problem. More measured carcasses clearly improve the solution, as the sub problem becomes a bigger part of the master problem. Figure 20 gives a more detailed view of the effect. In the beginning, the solution improves strongly with more measured carcasses, but after a while the quality of the solution stabilises. The improvements are largest from 1 to 3 measured carcasses, noticeable from 3 to 7 carcasses and only minor from 7 carcasses and more.

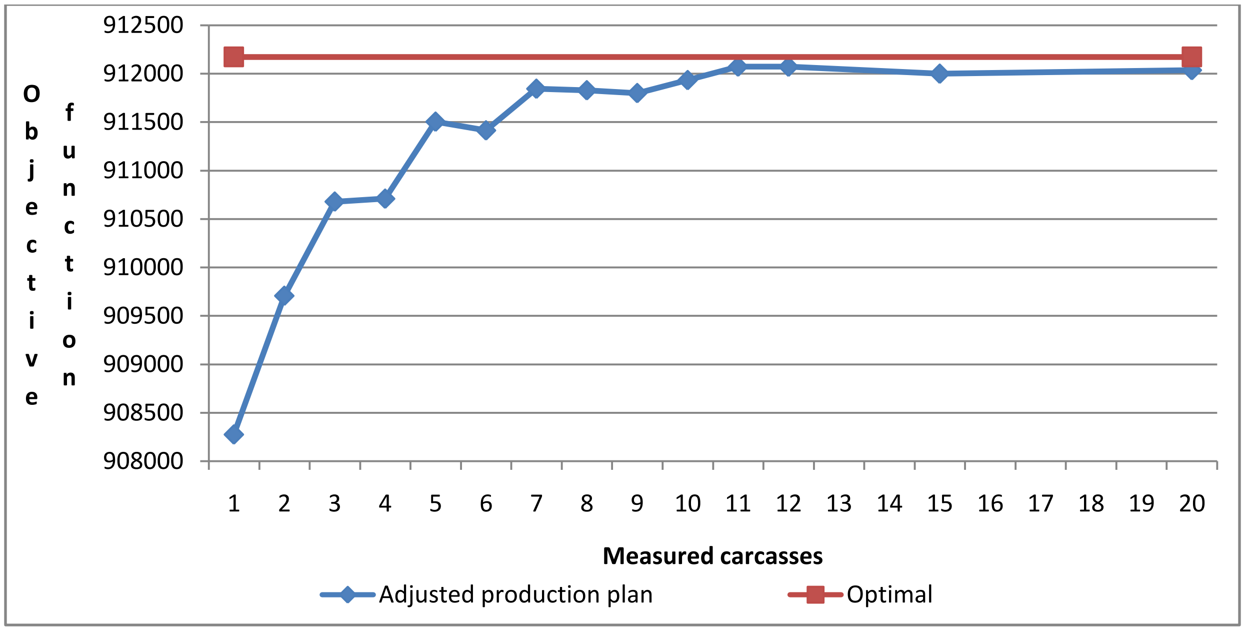

Figure 20 Effect of measured carcasses

More detailed numbers are given in the appendix. As can be seen, the difference from the optimal solution quickly decreases when more measured carcasses are used in the sub problem. For this example, 7 measured carcasses give a close to perfect solution with only 0.034% different from the optimal solution. The number of measured carcasses needed to achieve this accuracy will depend on the specific problem and will vary each time. 3 test cases have been used in Table 16 to create an average with less statistically variance. The trends still resemble what have been seen in the graphs above. Quite large improvements can be seen in the beginning while less is gained after the 7th measured carcass.

Table 16 Performance of online algorithms

Measured carcasses	1	3	7	10	15	Optimal offline solution
6th of June	964951	966514	968244	968714	968784	969629
5th of June	668271	671620	673230	673449	673121	673460
4th of June	908275	910679	911844	911933	911999	912174
Sum	2541497	2548813	2553318	2554096	2553904	2555263
% of optimal	99.461%	99.748%	99.924%	99.954%	99.947%	100.000%

5.3.1 Local search comparison

A local search can be used to solve the sub problem instead of the genetic algorithm. A local search alone will be less likely to find good solutions to the sub problem. The results of solving the master problem with a local search can be seen in Table 17. The performance of the local search is much worse than that of the genetic search. Because the local search will look for the first local optimum, more measured carcasses may not enhance the performance as much. The local search will always adjust the cutting pattern for the first carcass first, which leads to certain cutting patterns being chosen more often for the first carcass in the sub problem.

Table 17 Local search comparison

Local search	908275	908771	910462	908722	908401
% of optimal	99.573%	99.627%	99.812%	99.622%	99.586%
Genetic search	908275	911504	911933	911999	912036
% of optimal	99.573%	99.927%	99.974%	99.981%	99.985%
Measured carcasses	1	5	10	15	20

The local search gives the same results as the genetic search when only one carcass is measured. Only one carcass is being evaluated by the sub problem, and the local search will evaluate every possible cutting pattern for this carcass. Every possible solution will therefore be explored by a single iteration of the local search. An optimal solution will therefore be found for the sub problem each time. Not even the genetic search can guarantee as good results as the local search, since it is based on randomness. However, it is very likely that the genetic search will find the same solution since the mutations will make a random walk towards the same optimum.

Even if only test data from one day is used in Table 17, it is safe to conclude that the local search is unsuited for solving sub problems with many measured carcasses since the performance is well below the worst performance which has been seen by the genetic algorithm. Nonetheless, the local search is a good choice if only one carcass is measured, or to improve the best solution found by the genetic algorithm.

5.3.2 **Production comparison**

Since the objective function from the online algorithm is almost the same as the objective function for the offline value, it is likely that the variation between what is produced with the two algorithms is small.

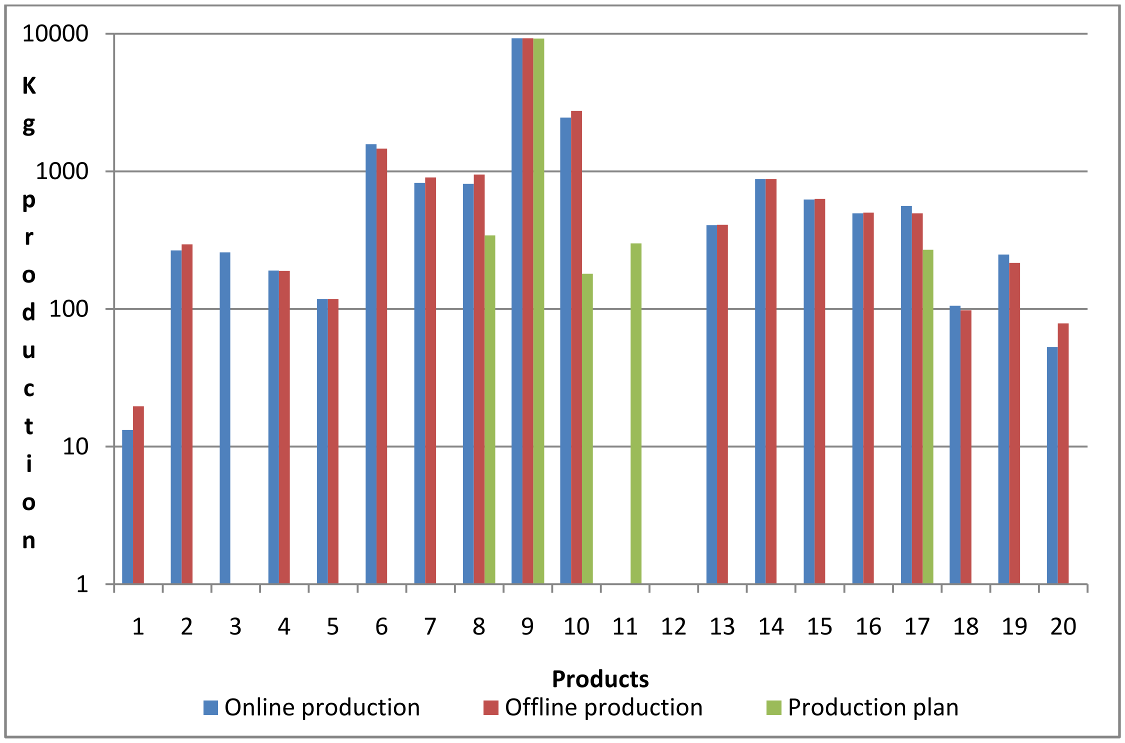

Figure 21 Comparison of the offline and online production

Tendons are still not produced by any of the algorithms, as can be seen in Figure 21. Product 3 is only produced by the online algorithm. Product 3 is actually a low priced product compared with the others. It would therefore seem like the offline algorithm has been able to do the optimisation without degrading the quality of the meat into product 3.

The production plan is accurately matched in production by both the online and offline algorithm for product 9. To produce any more of this product is clearly not worth it, while penalties for not fulfilling the demand forces the algorithm to produce everything that is planned. From Nortura’s point of view, this is one of the purposes of using an optimisation method, since it would be very hard to fill the production plan for product 9 without surplus production by manually picking the cutting patterns.

5.3.3 **The chosen cutting patterns**

Only a few of the cutting patterns are being used in this example. Most of the carcasses of the same type will be cut with the same pattern. While some carcasses will be cut with another pattern to prevent overproduction, there are only 2 or 3 patterns which are used for each carcass type. The online algorithm has a tendency of varying the type of cutting pattern more than the offline algorithm. This is quite logical, since the online algorithm will use different cutting patterns to fill up the different demands earlier in the production, while the online algorithm will know how to fulfil the demand with future carcasses.

It can seem like the unused cutting patterns could have been removed from the calculations completely. However, not all products can be produced by the most popular cutting patterns. If the production plan was different, a different set of patterns would have been the most popular ones. It would in theory be possible to analyse the production plan to discover which of the cutting patterns are needed, and with this information reduce the number of possible cutting patterns before doing the optimisation. Even if this is theoretically possible, it is not a trivial task to calculate which cutting patterns are needed for every combination of production plans. Some cutting patterns can easily be removed if they are neither the most valuable cutting pattern for the carcass type nor consist of any of the products in the production plan.

5.3.4 **Discussion of the online algorithm**

The relationship between the number of measured carcasses and the quality of the solution of the master problem has been shown in the previous sections. How close to optimality the solution should be to be useful depends on what it is used for. If simplicity of the implementation is the main focus, the results with only 1 measured carcass 0.5% from optimal value can be adequate. However, if the results have to be as good as possible, it will be well worth it to measure more carcasses.

Nortura's system is however already designed to measure multiple carcasses, and the benefit of 10 measured carcasses instead of 1 is noticeable. The following arguments are made based on the results in Table 16. If the model and data is a good representation of the real world, the difference between the 0.5% optimality gap with 1 measured carcass and the 0.05% optimality gap with 10 carcasses, this makes up 0.45% of the total revenue of the cutting facility. With the estimated value for the production of 4th of June at approximately 900 000 kroner, 0.45% would equal 4050 kroner. This is for a single cutting plant, and similar profits could be gained from every warm cutting plant.

The difference of 0.45% is however huge in comparison with the difference between the online algorithm with 10 measured carcasses and the offline algorithm. The optimality gap is only 0.05% which would make out only 450 kroner out of the 900 000 kroner revenue. This gap is so small that there is in practice no reason to use more than 10 carcasses, and the online algorithm has been demonstrated to work as intended.

5.4 Execution time

The critical execution time is the execution time of the sub problem. The program will have plenty of time to load the data before the production starts. What is important is how long it takes to solve the sub problem. The sub problem has to be solved for each carcass. The cutter would have to wait if the algorithm takes too long to calculate which cutting pattern to use.

All runs in this section has been performed on a 1.2 GHz Intel Dual Core running Windows XP.

How long the genetic algorithm will use before it terminates mainly depend on two factors, the number of generations and the number of individuals in each generation. Each individual has to be evaluated for each generation, which is the time consuming operation.

Execution time = number of generations * size of population

Equation 1 Execution time of the genetic algorithm

For convergence on the sub problem with 10 carcasses, 20 generations with a population of 400 has been used. To solve this sub problem takes between 5 and 6 seconds. Tests have shown that this scales up linearly when more generations are performed, which supports Equation 1. By using the upper limit of 6 seconds, the algorithm takes 75 µs per individual per generation.

The genetic algorithm should be compared with the exact method implemented in Xpress. The differences in the results between the two methods are so small that it makes little difference which one is used. However, it should be taken into consideration what happens with the execution time if the problem changes.

The timing of the algorithms has been taken by running various sub problems used in the master problem for 4th of June 2007. An upper limit is set by rounding up the longest execution time. The genetic search is performed with 20 generations and 400 population size.

Table 18 Execution time

Algorithm	Execution time
Local search	0.5 seconds
Genetic search	6 seconds
Exact method	6 seconds

All of the methods evaluated in Table 18 perform well enough for the time limit of 6 seconds. A faster computer could easily have brought the results further below the time limit. These results mean both the genetic search and the exact method can be used for the sub problem with 10 measured carcasses. In practice, the problem might be larger than what is described in this thesis. Nortura has visions of including multiple cutting plants in the optimisation and include more of the value chain. This would make the problem larger and the execution time would increase.

The genetic algorithm is much more flexible to different objective function than the exact solution. As long as the objective function can be calculated from the production, any function can be used. This means the objective function could be much more complex. One example is to make a non-linear objective function. Large overproduction is a large problem, while some overproduction is tolerated. The objective function could therefore penalise large overproduction more severely than minor overproduction. Similar methods could be used for underproduction.

The execution time of the genetic algorithm can easily be adjusted for any time limit. The number of generations and the population size can simply be adjusted for the execution time in Equation 1 to be below the time limit. Another approach is to make the genetic algorithm to finish when the time limit is approaching instead of performing a predefined number of generations. The best solution from the genetic algorithm is still likely to give a good result even if it has not converged completely. This is not the case for the exact method. In many instances no solution will be available before the final and optimal solution is found. This means it is impossible to stop the optimisation when the time limit is reached to get an approximate solution. This is a serious weakness of the exact method.

Xpress, which is used to find the optimal solution, is highly tweaked and optimised to perform fast. The genetic algorithm is however not thoroughly optimised for performance. The inner loop of the algorithm is the value calculation. An improvement of the method used to calculate the value of a solution could improve the execution time of the genetic algorithm significantly. Another advantage of genetic algorithms is the fact that they are naturally parallel. This means the algorithm can be performed on multiple processors without any fundamental changes in the algorithm. Different processors can work with different populations and exchange survivors regularly.

Conclusion: the exact method gives slightly better results. However, its execution time cannot be adjusted. It works well for the sub problem with only 10 carcasses and today's cutting patterns, but would too slow lacking if the problem size increases. Licences for Xpress are a considerable expense and makes the solution dependant on the Dash Optimizations which provides Xpress. The genetic algorithm performs almost as good as the exact method and will be just as good in practice. The method can easily sacrifice optimality for fast execution time, and will therefore be able to uphold tighter time limits for larger problems. The source code of the genetic algorithm will be owned by the developer and there will be no license costs or external dependencies. It can therefore be concluded that the genetic algorithm is a better choice.

5.5 Sources of error

How well the data and model resembles the reality is not a source of error for the algorithm, but it is a huge source of error when making the algorithm work in real production. It is not possible for the optimisation program to get perfect information about the reality. These inaccuracies have been noticed when the results from the algorithm were compared to the actual production of a cutting plant. Even if the measurements received by the optimisation program were the same as the ones done in the real cutting plant, the results could have large differences. In some occasions the real production would report a significantly larger production than the input weight to the optimisation program. Reasons for this can be additional production procedures. These kinds of variations have to be taken into account by the program, by systems which are continuously reporting actual production.

Only 3 days of production has been used as test sets for testing the online algorithm. These test sets use real data from one of Nortura's actual cutting plants. The results could vary if test sets from a larger time period or from different cutting plants were used. However, there are strong points suggesting that this algorithm is likely to work for other test sets as well. The algorithm has by no means been adjusted to perform well with these test sets and has been developed before the test sets were known. For all of the test sets the optimality gap from the offline solution has been less than 0.1%, which is insignificant in this context.

5.6 Other uses of the online algorithm

Many production processes have similar characteristics as the one discussed in this thesis. The algorithm is flexible enough to be used on other problems as long as the basic conditions are the same.

Conditions for a problem to be suitable for the online algorithm:

1) Input resources cannot be uniform
2) The resources can be turned into different products.
3) There has to be a production plan.
4) Production in addition to what is specified in the production plan has to be valued less than or equal to the products in the production plan.
5) Production starts before all resources are known.

One problem which suits these conditions are sawmills. Logs can be cut into various types of boards. Some types of boards can only be made from certain logs. Wide boards can typically only be made from large logs. Orders from the customers have to be fulfilled and makes up the basis for a production plan. All the conditions are fulfilled as follows:

1) Logs vary in quality and size.
2) Different kind of boards can be made from the logs.
3) Orders of boards create a production plan.
4) Products which the customers do not demand are not valued as highly.
5) The logs can be measured at the saw mill during production.

The value of using the online algorithm depends on the logistics of the production. If large storages can be made of the products and there is little time pressure to keep the turnover rate high, the problem will resemble cold cutting. However, just in time production will resemble warm cutting and gain more from the online algorithm.

6 Conclusion

Online algorithms have to our knowledge not been applied to meat cutting. This thesis has shown that online algorithms absolutely are competitive with offline algorithms for this problem type. It has also shown the importance of measuring carcasses, how a few measured carcasses give a large improvement while a large number of measured carcasses will only give slightly better results.

Nortura has shown great enthusiasm about the project and are very satisfied with the results [NorStaff08]. The program developed can serve as a prototype if Nortura decides to make an operational system with online optimisation. What will make or break this system is how accurately the model resembles the reality and how robust it will be to manually override. The cutters will be able to see what the optimisation system cannot see, and will at times have to ignore the systems recommendations. Feedback from the production system is required for each product, since using estimates from the cutting patterns alone will result in large inaccuracies.

The test data shows that the genetic algorithm comes extremely close to the optimal solution. This provides a fast and flexible solution method for the sub problem, which makes sure each carcass will be calculated in time even if the system become more complex than what is described in this thesis.

7 Further work

To make the prototype into an operational system is the largest challenge in the future. The system has to work with the measurement instruments and deliver the recommendations to the cutters, which require both integration with other systems and interaction with humans. For this to work well, extensive testing of usability and robustness will be needed, as well as the feedback systems discussed earlier.

Today's cutting patterns do not include all possible patterns. They are only a selection of patterns which has been shown to work well. With more cutting patterns, even better results could have been made since it would have been easier to make the products fit the production plan. An even more advance system could decide what to make for each part of the carcass instead of using predefined cutting patterns.

Not all cutters will know every possible cutting pattern and some cutters may want to use particular patterns for reasons unknown for the optimisation program. It would therefore be very helpful for the cutters if the optimisation program could suggest more than one cutting pattern. There are often a few cutting patterns which contribute almost equally to the objective function. If all of them were presented to the cutter, it would be up to he or she to choose the best suited among them.

Various improvements can be done to the algorithms. The genetic algorithm can be optimised through parallelisation, improvements to the code and adjustments to the algorithm. While the two first improvements are to the code, the last one has to do with finding the best combination of population size, reproduction style and number of generations. The selection function can increase its diversity by also letting some of the unfit individuals survive.

Since the genetic algorithm can solve non-linear as well as linear objective functions, this advantage should be exploited. The objective function can be tweaked and adjusted to be more helpful for the cutting plant. One such improvement would be to let the value of products drop continuously for production beyond the production plan instead of today's simple view of regular and surplus production.

The results can be tested by applying more test cases from a larger time span and from different cutting plants. Other cutting plants may produce a larger variety of productions and can have production plans which are more challenging to fulfil.

8 Bibliography

[Albers97]
Competitive Online Algorithms
Optima Mathematical Programming Society Newsletter No 54 1997
Susanne Albers
Max-Planck-Institut für Informatic

[Albers03]
Online algorithms: a survey
Susanne Albers
Max-Planck-Institut für Informatic 2003

[Dash08]
http://www.dashoptimization.com/home/products/products_optimizer.html 10. May 2008

[Gendreau02]
An Introduction to Tabu Search Michel Gendreau **-** Centre de recherche sur les transports and Département d´informatique et de recherche opérationnelle - Université de Montréal July 2002

[Karp72]
Reducibility among Combinatorial Problems 1972
Richard M. Karp

[Lewis03]
Performance of Java versus C++
J.P.Lewis and Ulrich Neumann
Computer Graphics and Immersive Technology Lab
University of Southern California
Jan. 2003

[Luke98]
A Revised Comparison of Crossover and Mutation in Genetic Programming
Sean Luke
Lee Spector
Department of Computer Science
University of Maryland
1998

[Mauldin84]
Maintaining Diversity in Genetic Search

Michael L. Mauldin
Department of Computer Science Carnegie Mellon University
August, 1984

[Nortura08]
http://www.nortura.no/organisasjon/formal/ 2. April 2008

[NorStaff08]
Discussions with staff from Nortura, including Trond Malmo, Paul Hosen and Klas Forfang

[Rardin98]
Optimization in Operations Research
Ronald L. Rardin
1998

[Reeves93]
Modern heuristic techniques for combinatorial problems
Coling R. Reeves
1993

[Wikborg07]
Online Meat Cutting Optimisation
Pre-study project conducted by Uno Wikborg
Fall 2007

9 Appendixes

9.1 Appendix A

This is a proof of the competitiveness of a profit maximising objective function disregarding demand. It is copied directly from [Wikborg07]

The value of each cutting pattern can be determined by summing the value of all the products it produces. An online algorithm can simply pick the most valuable cutting pattern available for each carcass. The result of this online algorithm will be exactly the same as the result of the profit maximising offline objective function.

Proof:

(1) $$Max\sum_{pr} V_{pr} \sum_{ca}\sum_{pa}(x_{ca,pa} * P_{pa,pr} * W_{ca})$$

Subject to:

(2) $$\sum_{pa} x_{ca,pa} = 1, \forall ca$$

(3) $$x_{ca,pa} \in (0,1), \forall ca, pa$$

(1) can be rearranged to (4).

(4) $$Max\sum_{ca}\sum_{pr}\sum_{pa}(x_{ca,pa} * V_{pr} * P_{pa,pr} * W_{ca})$$

(2) means that only one cutting pattern can be used for each carcass. It would therefore be equivalent if the maximisation function could choose one cutting pattern for each carcass. (4) can be written as (5) to be forced to choose between the cutting patterns directly.

(5) $$\sum_{ca} Max(\sum_{pr}(V_{pr} * P_{1,pr} * W_{ca}), \sum_{pr}(V_{pr} * P_{2,pr} * W_{ca}), ..., \sum_{pr}(V_{pr} * P_{N,pr} * W_{ca}))$$

By replacing the constants with $C_{pa,ca}$ for readability, the results will be (6), which is exactly what the algorithm does. The most valuable cutting pattern is chosen for each carcass.

(6) $$\sum_{ca} Max(C_{1,ca}, C_{2,ca}, ..., C_{N,ca})$$

Since the results are identical, this algorithm is a 1-competative online algorithm. It will, just like the offline algorithm, not make sure that the production fits the demand.

9.2 Appendix B Results

9.2.1 Results for optimisation program

Table 19 Results from 4th of June 2007

Product	Regular Production	Unsatisfied Production	Surplus Production
1	0	0	19.6463
2	0	0	294.886
3	0	0	0
4	0	0	189.526
5	0	0	117.878
6	0	0	1463.29
7	0	0	902.92
8	343	0	604.78
9	9250	0	10.4617
10	180	0	2573.5
11	0	300	0
12	0	0	408.347
13	0	0	880.885
14	0	0	630.898
15	0	0	502.432
16	270	0	225.953
17	0	0	97.8575
18	0	0	215.779
19	0	0	78.5851

Table 20 Offline production 4th June 2007 without production plan

Product	Regular	Unsatisfied	Surplus
1	0	0	0
2	0	0	395.698
3	0	0	0
4	0	0	200.366
5	0	0	131.63
6	0	0	4932.09
7	0	0	729.494
8	0	0	0
9	0	0	1422.7
10	0	0	3611.51
11	0	0	150.294
12	0	0	3987.21
13	0	0	439.816
14	0	0	952.767
15	0	0	645.633
16	0	0	456.737
17	0	0	734.283
18	0	0	542.996
19	0	0	358.214
20	0	0	0

Table 21 Offline production 4th - 6th June 2007

Product	Regular production	Unsatisfied production	Surplus production
1	0	0	28.8433
2	0	0	612.032
3	500	0	618.457
4	0	0	523.282
5	0	0	340.829
6	1199.9	0	0
7	0	0	2175.99
8	1674	0	704.01
9	30247.8	2.20418	0
10	3060	0	2905.32
11	0	900	0
12	0	0	1143.91
13	0	0	2449.45
14	0	0	1768.65
15	0	0	1405.5
16	305	0	1360.89
17	0	0	80.2436
18	0	0	745.928
19	0	0	115.373

9.2.3 Results for genetic algorithm used on offline problem

Table 22 Genetic algorithm results for 4th of June

Generation	Regular	Small	No mutations	No crossovers	No new born	Less survivors	More survivors	Optimal
1	662442	662442	662442	662442	662442	662442	662442	912174
2	697754	691486	697754	678560	697754	713485	697754	912174
3	736447	731654	718216	692302	714640	755437	706570	912174
4	750615	757588	744619	717688	754592	808765	733293	912174
5	779617	775079	769001	717688	792077	830044	747562	912174
6	813729	807013	786974	717688	792077	878396	762378	912174
7	830518	835451	808149	723777	814055	891038	794253	912174
8	855193	849520	825496	723777	837048	895105	811584	912174
9	872871	871358	860818	736365	859895	898086	815223	912174
10	887064	887450	888394	736365	883714	898691	816664	912174
11	891984	889458	888394	736365	889733	899531	851850	912174
12	896031	891089	893232	736365	893675	901121	851850	912174
13	896031	891640	895059	742698	895308	902172	886504	912174
14	896031	893831	896121	742698	897880	903171	886504	912174
15	896031	895405	897176	761361	897880	904321	890975	912174
16	897210	896467	897929	761361	898027	905681	894927	912174
17	898755	896467	898603	761361	899842	905958	894927	912174
18	899441	898591	900337	761361	899842	906811	894927	912174
19	900626	898730	902160	761361	900653	907908	894927	912174
20	901709	900212	902512	765291	901943	907908	895373	912174
21	902153	901660	902515	765291	903378	908186	895373	912174
22	903030	901660	904395	783172	903986	908556	895949	912174
23	904989	902157	904395	783172	904833	908653	897525	912174
24	904989	902833	904395	783172	905511	908692	897905	912174
25	905112	904326	905061	783172	905988	909003	897964	912174
26	905490	904326	906166	783172	906597	909113	899548	912174
27	905733	904642	906166	783172	907147	909118	900078	912174
28	906624	906073	906701	783172	907788	909120	900649	912174
29	907496	906073	906956	783172	908176	909120	901757	912174
30	907721	906525	907331	783172	908176	909120	901757	912174
31	908630	907191	907728	803345	909035	909120	903206	912174
32	908630	907414	908144	803345	909143	909120	903465	912174
33	908630	907562	908345	803345	909366	909120	904321	912174
34	908799	908050	908957	803345	909366	909120	904748	912174
35	909622	908105	908957	803345	909598	909120	904833	912174
36	909906	908126	908957	803345	910186	909120	906290	912174
37	909916	908668	909191	803345	910186	909120	906290	912174
38	910238	908668	909282	803345	910186	909120	907313	912174
39	910731	909023	909637	803345	910502	909120	907313	912174
40	910731	909220	910015	803345	910733	909120	907313	912174
41	910731	909220	910015	803345	910733	909120	907313	912174
42	910731	909361	910119	803345	911060	909120	907313	912174
43	910934	909691	910119	803345	911060	909120	907850	912174
44	911031	909840	910577	803345	911060	909120	907850	912174
45	911031	909840	910577	803345	911141	909120	908833	912174
46	911174	909843	910739	803345	911304	909120	908833	912174
47	911176	909973	910739	803345	911304	909120	908833	912174
48	911248	910068	910770	803345	911304	909120	908833	912174
49	911456	910104	910911	803345	911304	909120	908833	912174
50	911457	910246	911015	803345	911304	909120	908833	912174
51	911639	910246	911015	803345	911323	909120	909510	912174

52	911664	910288	911285	803345	911511	909120	909510	912174
53	911664	910291	911285	803345	911511	909120	909510	912174
54	911664	910317	911385	803345	911511	909120	909756	912174
55	911692	910340	911385	803345	911511	909120	909756	912174
56	911692	910340	911385	803345	911511	909120	909756	912174
57	911692	910340	911387	821795	911567	909120	910323	912174
58	911749	910340	911451	821795	911567	909120	910323	912174
59	911749	910340	911540	821795	911591	909120	910323	912174
60	911797	910340	911540	821795	911596	909120	910657	912174
61	911800	910340	911540	821795	911596	909120	910657	912174
62	911800	910340	911540	821795	911596	909120	910657	912174
63	911848	910340	911552	821795	911639	909120	910657	912174
64	911848	910340	911552	821795	911726	909120	910657	912174
65	911848	910340	911571	821795	911744	909120	911030	912174
66	911848	910340	911628	821795	911744	909120	911030	912174
67	911880	910340	911628	821795	911744	909120	911030	912174
68	911893	910340	911628	821795	911744	909120	911030	912174
69	911893	910340	911628	821795	911784	909120	911030	912174
70	911914	910340	911628	821795	911874	909120	911030	912174
71	911921	910340	911628	821795	911874	909120	911030	912174
72	911921	910340	911628	821795	911874	909120	911111	912174
73	911921	910340	911628	821795	911874	909120	911111	912174
74	911921	910340	911688	821795	911874	909120	911277	912174
75	911921	910340	911688	821795	911874	909120	911277	912174
76	911934	910340	911688	821795	911874	909120	911277	912174
77	911934	910340	911688	822786	911874	909120	911277	912174
78	911934	910340	911688	822786	911874	909120	911277	912174
79	911934	910340	911688	822786	911874	909120	911277	912174
80	911934	910340	911688	822786	911888	909120	911277	912174
81	911947	910340	911688	822786	911888	909120	911379	912174
82	912003	910340	911688	822786	911888	909120	911379	912174
83	912003	910340	911688	822786	911901	909120	911379	912174
84	912003	910340	911688	822786	911901	909120	911379	912174
85	912003	910340	911688	822786	911901	909120	911417	912174
86	912003	910340	911688	822786	911901	909120	911417	912174
87	912003	910340	911688	822786	911915	909120	911573	912174
88	912003	910340	911688	822786	911929	909120	911573	912174
89	912003	910340	911688	822786	911929	909120	911573	912174
90	912003	910340	911688	822786	911929	909120	911573	912174
91	912003	910340	911688	822786	911933	909120	911622	912174
92	912003	910340	911688	823794	911933	909120	911622	912174
93	912003	910340	911688	823794	911933	909120	911622	912174
94	912003	910340	911688	823794	911940	909120	911622	912174
95	912003	910340	911713	830852	911940	909120	911622	912174
96	912033	910340	911713	830852	911945	909120	911651	912174
97	912033	910340	911713	830852	911958	909120	911651	912174
98	912033	910340	911713	830852	911958	909120	911651	912174
99	912033	910340	911759	830852	911969	909120	911651	912174
100	912033	910340	911759	830852	911969	909120	911651	912174

Table 23 Online algorithm 4th of June 2007

Measured carcasses	Optimal	Adjusted production plan	Not adjusted production plan
1		908275	904784
2		909707	
3		910679	
4		910709	
5		911504	906252
6		911414	
7		911844	
8		911828	
9		911799	
10		911933	907260
11		912072	
12		912071	
15		911999	909873
20		912036	910302
40			911445
95	912174	912104	912104