Determining the optimal log position during primary breakdown using internal wood scanning techniques and meta-heuristic algorithms

by
Fritz van Zyl

Thesis presented in partial fulfilment of the requirements for the degree Master of Science in Engineering at the University of Stellenbosch

Supervisor: Me. L. Van Dyk
Co-supervisor: Mr. C.B. Wessels
Faculty of Engineering
Industrial Engineering

March 2011

Declaration

By submitting this thesis/dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2011

Abstract

During the 2009 financial year the sawlog production from plantations in South Africa amounted to 4.4 million m^3 and sawn timber of R4.2 billion was produced from these logs. At the current average price for structural timber, a 1% increase in volume recovery at a medium-sized South African sawmill with an annual log intake of 100 000m^3 will result in additional profit of about R2.2 million annually.

The purpose of this project was to evaluate the potential of increasing in value recovery at sawmills through optimization of the positioning of a log at the primary workstation by considering the internal knot properties. Although not yet commercially available, a high speed industrial log CT scanner is currently in development and will enable the evaluation of the internal characteristics of a log before processing.

The external profiles and the internal knot properties of ten pine logs were measured and the whole log shape was digitally reconstructed. By using the sawmill simulation program Simsaw, explicit enumeration was performed to gather data. This data include the monetary value that can be earned from sawing the log in a specific log position. For every log a total of 808 020 sawing positions were evaluated.

In the sawmill production environment only a few seconds are available to make a decision on the positioning of each log. Meta-heuristic optimization algorithms were developed in order to come to a near optimal solution in a much shorter time than that required when simulating all possible log positions. The algorithms used in this study include the Genetic algorithm, Simulated Annealing, Population Based Incremental Learning and the Cross-Entropy method. An Alternative algorithm was also developed to incorporate the trends identified through analysis of the sawmill simulation results.

The effectiveness of these meta-heuristic algorithms were evaluated using the sawmill simulation data created. Analysis of the simulation data showed that a maximum increase in product value of 8.23% was possible when internal knot data was considered compared to using conventional log positioning rules. When only external shape was considered a maximum increase in product value of 5% was possible compared to using conventional log positioning rules. The efficiency of the meta-heuristic algorithms differed depending on the processing time available. As an example the Genetic algorithm increased the mean product value by 6.43% after 200 iterations. Finally, a method to evaluate the investment decision to purchase an internal scanning and log positioning system is illustrated.

Opsomming

Gedurende die 2009 finansiële jaar is daar 4.4 miljoen m^3 rondehout op plantasies in Suid Afrika geproduseer en saaghout ter waarde van R4.2 biljoen is hieruit vervaardig. Met die huidige gemiddelde prys vir strukturele hout, kan ‘n 1% verhoging in volumeherwinning by ‘n gemiddelde grootte saagmeul in Suid Afrika met ‘n jaarlikse rondehout inname van 100 000 m^3 ‘n bykomende wins van R2.2 miljoen lewer.

Die doel van hierdie projek was om die potensiële verhoging in waardeherwinning by ‘n saagmeul te evalueer, indien die posisionering van ‘n stomp by die primêre werkstasie geoptimeer word deur interne kwas eienskappe in ag te neem. Kommersiële CT-skandeerders word tans nog nie hiervoor aangewend nie, maar ontwikkelinge in tegnologie sal dit moontlik binnekort prakties moontlik maak om die interne karakteristieke van ‘n stomp te evalueer voor prosessering.

Die eksterne profiel en interne kwas eienskappe van tien *Pinus* rondehout stompe is gemeet en die al tien stompe is digitaal geherkonstrueer. Met behulp van die saagmeulsimulasieprogram, Simsaw, is 808 020 verskillende saagsimulasielopies uitgevoer. Elk van hierdie simulasielopies het ‘n ander beginposisie gehad in terme van rotasie, skeefheid en horisontale verskuiwing. Die finansiële waarde wat verdien kan word deur ‘n stomp in ‘n sekere posisie te saag is telkens bepaal.

In die saagmeulomgewing is daar slegs ‘n paar sekondes beskikbaar om ‘n besluit te maak oor hoe ‘n stomp geposisioneer moet word. Meta-heuristiese optimisering algoritmes is ontwikkel om ‘n naby optimale oplossing te bepaal in ‘n baie korter tyd as wanneer alle saagposisies geëvalueer word. Vyf verskillende meta-heuristiese algoritmes is teen mekaar opgeweeg. Vier van hierdie algoritmes is bestaande heuristieke wat vir verskeie ander optimeringsprobleme ingespan word. Die vyfde algoritme is spesifiek vir doeleindes van hierdie projek ontwikkel om die neigings wat tydens die data-analise van die saagmeulsimulasie geïdentifiseer is, te inkorporeer.

Die effektiwiteit van hierdie meta-heuristiese algoritmes is bepaal deur van die saagmeul simulasiedata wat gegenereer is gebruik te maak. Analise van die simulasiedata toon dat ‘n maksimum toename in produk waarde van 8% moontlik is wanneer interne kwaseienskappe ook geïnkorporeer word tydens besluitneming teenoor die konvensionele stompposisioneringreëls. Wanneer slegs die eksterne stompprofiel in ag geneem word, is ‘n maksimum produkwaardeverhoging van tot 5% moontlik teenoor resultate wat verkry word met konvensionele stompposisioneringsreëls.

Die effektiwiteit van die meta-heuristiese algoritmes word beïnnvloed deur die hoeveelheid tyd beskikbaar vir besluitneming. Met die Genetiese algoritme, kan die gemiddelde produk waarde byvoorbeeld met 6% verhoog word na 200 iterasies. Ten einde is ‘n metode om die beleggings besluit in ‘n interne skandering en stomp positionering stelsel geilustreer.

.

Acknowledgements

I want to thank God for giving me the talents and the opportunities to do my studies, and for carrying me through and blessing my whole thesis.

I would like to thank the following people whom without this thesis would not have been possible:

- MTO forestry for sponsoring us with ten logs to use in this study.
- Cape Sawmills in Stellenbosch for debarking all the logs.
- Brand Wessels for all his time and effort with especially the wood science section of this study.
- Liezl van Dyk for all her support and effort especially with the writing of the thesis report, going past what was expected from her.
- Christine Hougaard for editing the thesis document.

1 Table of Contents

List of Figures

List of Tables

1 Introduction

Forestry plantations cover about 1.3 million ha of South Africa's land surface. Of the total plantation area, 51% is covered with softwood trees. Of the total log harvest, 34% of the volume is processed in sawmills. The production from these plantations in the 2009 financial year amounted to 4.4 million m^3 of sawlogs. The value of sales of sawn timber from these logs amounted to R4.2 billion (Forestry Economics Services, 2009)

This study investigated the optimization of the log breakdown process and specifically the positioning of a log in front of the primary breakdown saw when the internal knot characteristics were known. Meta-heuristic techniques were evaluated for their efficiency in reaching optimal or close-to-optimal solutions in a limited number of iterations.

A small increase in the volume (total volume of lumber sawn from a log) or value (total monetary value of lumber sawn from a log) recovery of a sawmill can have a significant impact on the profitability of the operation. At the current average price for structural timber, as listed by Crickmay and Associates (2010) a 1% increase in volume recovery at a medium-sized South African sawmill with an annual log intake of 100 000m^3 will result in additional profit of about R2.2 million annually.

To produce timber from logs in a sawmill, complex actions need to be performed at a number of working stations. The timber production process is shown diagrammatically in Figure 1. This project focuses on the primary breakdown station, also known as the headrig.

Apart from the economic advantage that can be achieved by increasing the volume and value recovery from a log, it also has a positive impact on the environment. If more timber is recovered from all harvested logs, it means fewer trees have to be cut down to produce the same amount of timber.

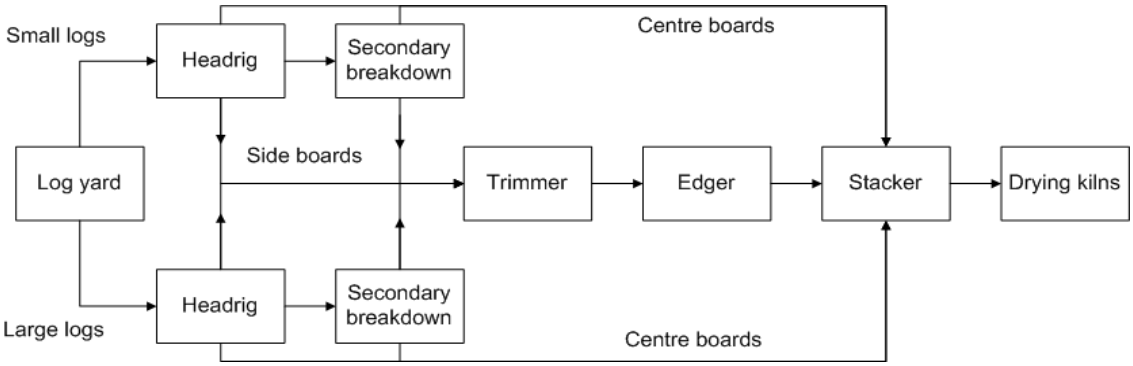


Figure 1: The log breakdown process

1.1 Problem statement

One of the most important decisions during the log breakdown process is at the primary breakdown station. The positioning of the log at this stage largely determines the potential value that can be recovered from that log. Subsequent operations cannot correct poor positioning decisions at the primary breakdown saw.

Most features that affect the quality of timber are distributed irregularly across the log. In South Africa the knot properties of timber largely determine the grade and price of a board. When in the log form, the operator cannot see or even predict where knots might be, as most sawlog trees in South Africa are pruned. The position of the log during primary breakdown determines where knots will be situated in boards and influences the quality and the volume of the timber produced. Both the volume and the quality of the timber produced significantly influence the value earned from the timber (Rinnhofer, 2003).

To make the best decision on how to break down the log, the operator of the machine, firstly, has to know what the defects are inside the log and, secondly, decide which position will yield the optimum utilization (Wessels, 2006). Currently, at most sawmills in South Africa the operator can neither see the defects inside of the log, nor does he have the decision support to decide on the best position. Logs are positioned solely based on external features like the curvature (sweep) and taper. The only way to control the position of the knots in the lumber is to internally scan logs before they are sawn. Internal x-ray log scanners that providing 2-dimensional scans of logs have been commercialised in the last decade and are used to grade logs according to their internal features. Such scanners are not suitable for positioning logs at a saw. The first commercial log CT scanner is currently being built and will be implemented in 2011 in Chile. (pers. comm. Martin Bacher, Microtec, 2010). Such a scanner provides a detailed 3-dimensional internal image of the log. This specific scanner will be

used to assist with bucking decisions of logs but might be used in future for log positioning optimization.

Purpose and objectives

The purpose of this project was to evaluate the optimization of the log breakdown process at the primary workstation by considering the internal knot properties and to develop an optimization algorithm to find the optimal or near-optimal log position. To accomplish this, the following objectives were set:

- Evaluate the effectiveness of log breakdown when internal knot properties are known, compared to the effectiveness if only external characteristics are known.
- Develop meta-heuristic algorithms to arrive at a set of position parameters that would result in a near-optimum log value recovery after breakdown.
- Validate the effectiveness of these meta-heuristic algorithms through experimentation involving actual knot property data from real logs.
- Aid in the selection of such an algorithm.
- Evaluate the economic feasibility of log breakdown through internal scanning.

1.2 Hypotheses

Knowledge of internal knot properties of logs result in more effective breakdown of a log compared to when only external log properties are known.

A meta-heuristics method will result in an improvement in the economic feasibility from implementing an internal log scanning system.

1.3 Scope

The study considered *Pinus radiata* logs from a single compartment in Tookai forest in the Western Cape, South Africa. Since the logs used in this study had to be sawn into small pieces, it could not be utilized as lumber, which limited the amount of logs that could be used. The logs used were sponsored by MTO Forestry - they agreed to sponsor ten logs of 3m each. Five logs came from the pruned section of a tree and five logs from the unpruned section. The logs could not be longer than 3m due to length restrictions of the lathe on which the external profile of the log was measured. Since the number of logs is limited and it is only

from a single location, the results from the study cannot necessarily be applied to other log resources.

Inetrnal log images had to be constructed manually since there were no internally scanned log images of Pine logs from South Africa available to the author. The only scanners available in South Africa to scan logs internally are the CT scanners in hospitals and at airports. For practical reasons, none of these could be used to scan the logs.

The sawing method used in this study is the cant sawing method, which is the most popular method in softwood sawmills in South Africa. This method is described in more detail in section 2.1.

The simulation program that was used to determine the monetary value that can be earned from sawing a log in a certain position in front of the headrig is called Simsaw. To create the whole log shape, including the internal knot information of the log, a new Simsaw version was programmed for purposes of this project. The previous version, Simsaw6, did not have the ability to enter knot information. An independent programmer did the programming of the new Simsaw version. The author helped in the process of developing the new version, which included testing the version and pointing out problems with it, but did not assist in the programming of the new Simsaw version.

The meta-heuristic algorithms used in the study are the Genetic Algorithm, Simulated Annealing, Population Based Incremental Learning (PBIL) and the Cross-Entropy method. An initial literature study rendered these algorithms as the most likely to result in fast and accurate solutions for this type of application.

1.4 Research approach

According to Mouton (2008) statistical modeling and computer simulation study are *"aimed at developing and validating accurate representations (models) of the real world. In statistical modeling, a specification of a model is constructed through a process of abstraction from what are theorized to be the process in the 'real world'."* For this project meta-heuristic methods were used to generate expected values that were compared to actual data. The research design map provided by Mouton (2008) was used.

This study was aimed at developing and validating an accurate representation of the real world (the log breakdown process). Specifically, for this project, meta-heuristic optimization methods to theorize about the most effective log breakdown algorithm were used.

This approach is based on the model building process suggested by Winston (2004). After the problem was formulated, the following steps were followed.(Winston, 2004):

- Collect the data
- Develop the optimization algorithm
- Verify the algorithm and use the algorithm for prediction
- Present the results and conclusion of the study
- Select a suitable alternative
- Implement
- Identify suitable alternative

1.4.1 Collect the data

Log digitizing

To carry out the study, 3-dimensional internal images of South African *Pinus Radiata* logs were used. Images from South African logs were used since differences can be detected in logs from other parts in the world and this research was aimed at a local audience. As mentioned before, there were no scanners available in South Africa to produce such images. Hence, an alternative method was necessary to obtain data that would otherwise have been revealed through scanning technology. This alternative is explained in section 3.2.

Simsaw

The Simsaw program was initially developed by the CSIR in South Africa in 1975 (Wessels et al, 2006). It is a program that predicts the volume, grade and monetary value of the boards sawn from the log, based on specified inputs – such as sawing pattern, kerf size, rotation angle, etc. In this study it was the monetary value that we wanted to optimize. Previous versions of Simsaw did not fulfil in the requirements needed to handle the simulations that needed to be done for this study. The requirements of a new Simsaw version were set out, and a programmer programmed a new Simsaw version.

As the need for more complex calculations to be done with the program grew, more versions of the program were written. The program is freeware and the latest versions were programmed in Delphi programming software.

Simulation scenarios

Simsaw was used to run many simulations for one log. By changing variables such as rotation angle, log skewing and offset, Simsaw gave the monetary value when the log is

sawn under each combination of these variables. This simulation process can also be referred to as explicit enumeration. Figure 2 gives a representation of what these three variables mean in practice. The algorithm which was developed to come to an optimal solution determines what the values must be for these three variables.

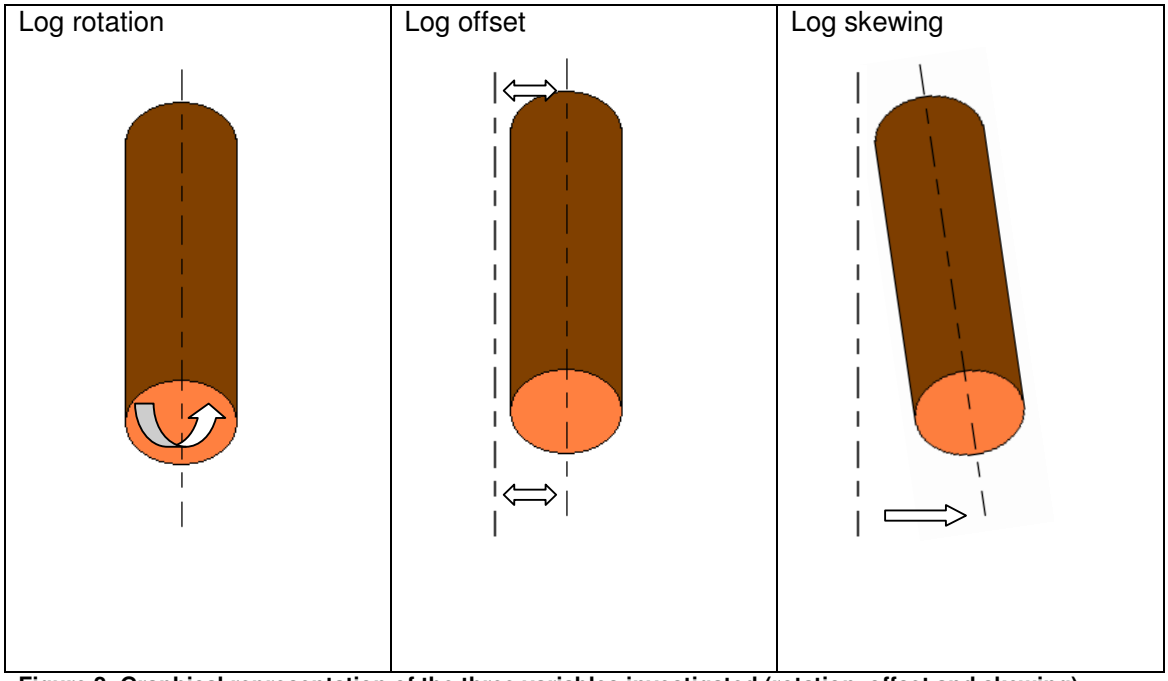


Figure 2: Graphical representation of the three variables investigated (rotation, offset and skewing)

All the variables with the ranges which was considered:

- X_1 – Rotation angle $(0° < X_1 < 360°)$, in increments of 2° ; 180 possibilities
- X_2 – Offset $(-100 < X_2 < 100)$, in increments of 3mm ; 67 possibilities
- X_3 – Log skew $(-100 < X_3 < 100)$, in increments of 3mm ; 67 possibilities

This resulted in 808 020 possible log positions per log.

Since the only product produced from a log that was used to calculate the monetary value from the log were the sawn boards, the value that we wanted to optimize will henceforth be referred to as the board value.

The time it took to calculate the board value for one log position depended on the amount of internal information for the specific log.

It was necessary to compare optimizing for best board value when internal log information was considered with optimizing volume recovery when only external log information was

considered. It was not necessary to run a new simulation. When the original simulation was done, Simsaw also calculated the volume recovery.

1.4.2 Develop the sawing optimization algorithm

Even if 3D images of logs were available it will take a computer with a 3 GHz processor and 2Gb RAM between one and eight weeks, depending on the amount of internal information of the log, to solve all possible solutions. In total, 808 020 log positions were solved in Simsaw for each log. By proving that an improvement in board value from a log can be obtained by correctly positioning the log in front of the primary breakdown saw does not prove it is practically implementable. The whole process of scanning the log and making a decision on how to position the log needs to happen within a few seconds. In one example sawmill, on average 2334 logs are sawn per day, which means 14 seconds are available per log to come to a solution. A meta-heuristic algorithm was developed to come to a near optimal log position within reasonable time.

“Heuristics are fallible and they do not guarantee a correct solution. It is important to understand their limitations when applying them to different equipment and processes. Even though heuristics are limited, they may be of value because they offer time saving approximations in preliminary process design.” (Turton et al, 2003)

For the purpose of this project, the meta-heuristic optimization algorithms were programmed in MATLAB.

Section 4.2 is devoted to the development of the optimization algorithms.

1.4.3 Verify the algorithm and use the algorithm for prediction

Each algorithm was run for a different number of iterations (one iteration means that one combination of the three variables is evaluated) with different values for the constants of each algorithm. The values for the constants which yielded the best results were used to do the final simulation run of the algorithm, which was used to compare the different algorithms against each other. Section 5.3 is devoted to the validation of the meta-heuristic algorithms to increase the utilization of the logs.

1.4.4 Present the results and conclusion

The results include the optimum amount of value in Rand (South African currency) found by the optimization algorithms compared to the absolute maximum which can be earned from a log. The absolute maximum was also compared to the traditional method of log breakdown, which means sawing the log in the “horns-up” position (which is when the log’s maximum

curvature is upright). Based on this information, the value increase that would result from installing an internal log scanning and optimization system could be calculated.

1.4.5 Select a suitable alternative

The different algorithms were evaluated against each other based on the following criteria:

- Speed to reach a solution
- Consistency
- The maximum value found by the algorithm

1.5 Thesis roadmap

The roadmap followed to execute this methodology is shown in Figure 3.

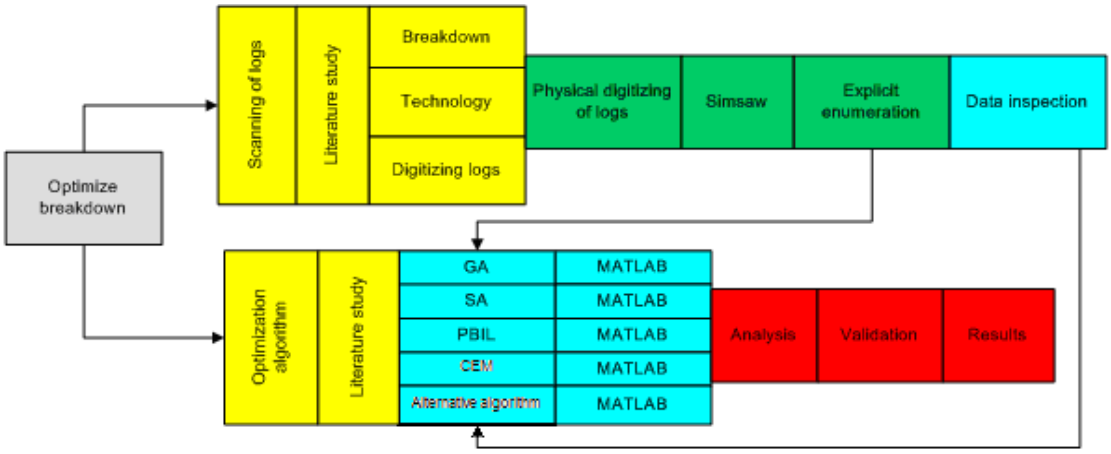


Figure 3: Thesis roadmap

The rest of this study is structured according to this roadmap:

- Chapter two reviews, firstly, literature on the scanning of logs, as well as previous studies on the optimization of the primary breakdown of logs. Secondly, a literature study was conducted on the operation of the different optimization algorithms used.
- Chapter 3 is devoted to the data gathered and used in this study. This includes the digitizing of the logs and the virtual sawing of the logs in all possible positions.
- Chapter 4 describes the working of the optimization algorithms as well as why each algorithm works in a particular way.

- The analysis of the different algorithms and the interpretation thereof are described in chapter 5.

There is a real need to convert logs into timber more effectively. This study aims to improve the utilisation of the logs at the primary breakdown station in the log breakdown process. Statistical modeling and computer simulation is a suitable research methodology to do so.

2 Literature review

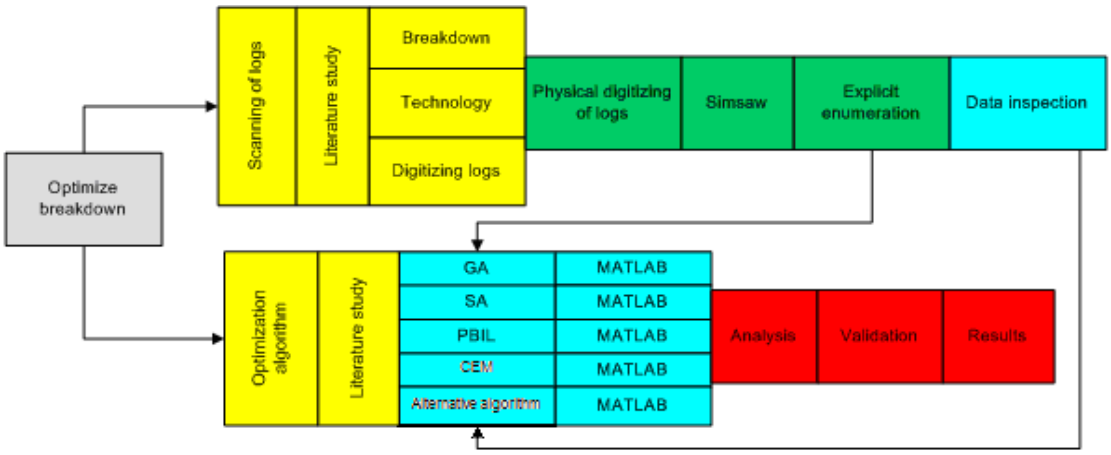


Figure 4: Thesis roadmap

The purpose of this project was to optimize the primary breakdown of a log. To implement such an optimization process in practice, (i) digital images of the log needs to be created, (ii) an optimization decision needs to be made, and (iii) the log needs to be positioned and sawn.

This chapter is devoted to literature on:

1. the log breakdown process, including the sawing pattern used and the scanning of logs;
2. previous techniques considered regarding optimization of the primary log breakdown; and
3. the optimization algorithms used.

2.1 The log breakdown process

A few different sawing techniques are used at the primary breakdown stations of sawmills. The most common patterns include Cant sawing, Live sawing, Quarter sawing, Round sawing, Grade sawing and Radial sawing. These sawing patterns are illustrated in Figure 5 to Figure 9:

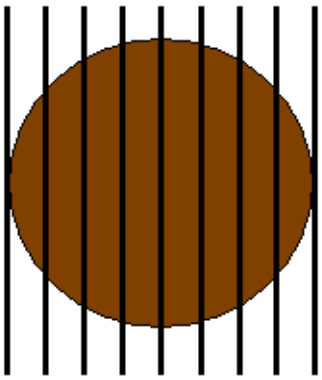


Figure 5: Live sawing

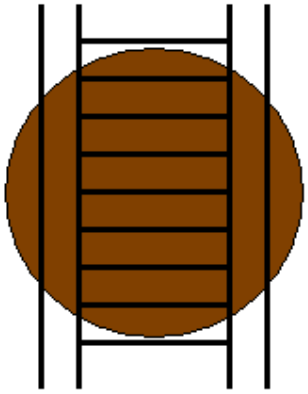


Figure 6: Cant sawing

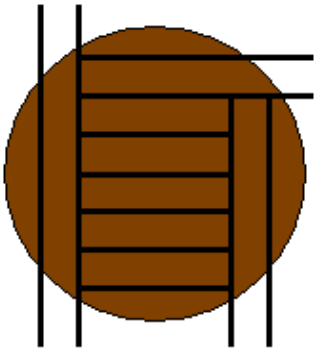


Figure 7: Grade sawing

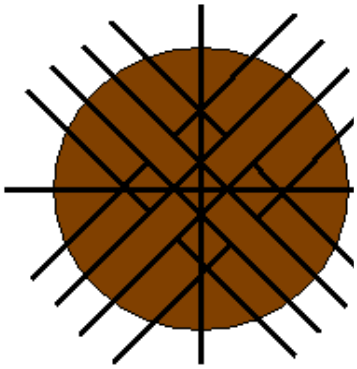


Figure 8: Radial sawing

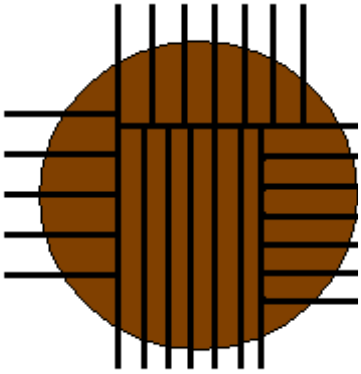


Figure 9: Quarter sawing

2.1.1 Cant sawing method

Most pine sawmills in South Africa use a framesaw (Figure 10) for primary breakdown of logs. These framesaws make use of either the cant or the live sawing method. Only the cant sawing method will be used in this study, since this is the most common sawing pattern used in softwood sawmills in South Africa. A framesaw consists of a number of blades that are fixed in a frame that moves up and down at a high speed. The log is pushed through the saw, which then produces the cant and the side boards. The cant is the centre section which still has bark on both sides. The cant is then turned 90° and sawn into boards by another framesaw. One disadvantage of a framesaw is the fact that the dimensions of the boards sawn cannot be changed during production. To change the dimensions, a whole new blade setup has to be done. This takes a substantial amount of time.

Figure 10: An industrial framesaw

2.1.2 Scanning technology

In the past few decades much effort was put into the development of internal wood scanning technology. X-ray scanners producing internal 2-dimensional images of logs have been commercialised and are sometimes used for grading of logs (see Figure 11). These images, however, does not give accurate 3-dimensional data suitable for positioning of logs.

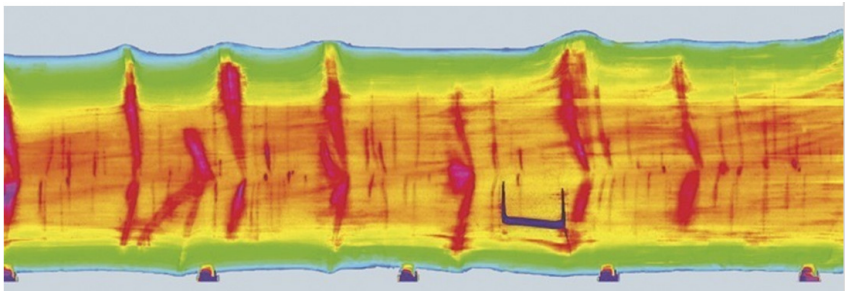


Figure 11: A 2-dimensional x-ray image of a log from a commercial log scanner (Microtec, 2011).

X-ray computed tomography (CT) scanning technology has shown to be the most promising technique to successfully scan logs for 3-dimensional internal images in an industrial environment (Schmoldt, 2000). CT scanners have been implemented for many years in the medical area, but the big challenge is to have such a scanner which can be used in industrial environments, where the time to scan plays an important role. The first industrial internal CT log scanner will be implemented in Chile in 2011. This scanner will only be used for log bucking decisions (pers. comm. Martin Bacher, Microtec, 2010). Log bucking is the process of cutting a whole tree stem into logs. Such a system will determine the optimal lengths and

positions of the logs sawn from the stem. According to Microtec a system is currently being developed to internally scan a log and used for decision support in sawing pattern selection for individual logs (Giudiceandrea, 2010).

Figure 12: The CT log scanner from Microtec (Giudiceandrea, 2010)

2.1.3 Digitizing logs with manual method

When developing a log breakdown optimization system, the way the log is described is one of the most critical aspects of how accurate the solution will be. No matter what algorithm is used to obtain a final solution, the further the log model is from the real shape of the log, the further the solution will be from the optimum (Zeng,1995).

There are several methods for modeling a log on a computer. According to Zeng (as cited in Alleckson, 1980) one can define log models as whole log, cross section, or computer array models.

"Whole log models use a mathematical equation to describe a log. Examples of this approach are cylinder and truncated-cone models." (Zeng,1995)

Zeng states that early systems mostly used such simplified log models instead of real log shapes (as cited in Geerts, 1984; Hallock and Lewis, 1971, 1973, 1978; Lewis, l985a, 1985b; Tejavibulya, 1981). The cylinder model simply used a diameter and length of a log to represent it as a cylinder. The truncated cone method added a small end diameter to include taper and represent the log as a cone.

Cross-section models use a number of cross sections at certain intervals along the length of the log to build up a 3-D representation of the log. The surface of the log is represented by lines that connect each pair of intervals with each pair of cross section. This representation can be seen in Figure 14. Since each cross section is different, models such as these

include irregularities such as crook and sweep. Cross-section models can be divided into circular, elliptical and polygonal cross-section models. Circular cross-section models simply use a radius at each cross-section. In an elliptical cross-section model the cross-sections are made by using the long and short axes of the cross section. In polygonal cross-section modeling, the cross-section is represented by a series of points on the cross-section (Zeng, 1995). The polygonal cross-section method was used in this study, since it is the closest to the real log shape.

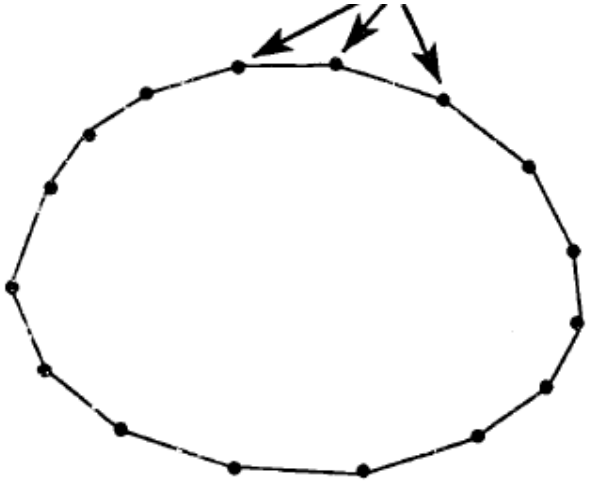


Figure 13: A cross section of the polygonal cross-section model. (Zeng, 1995)

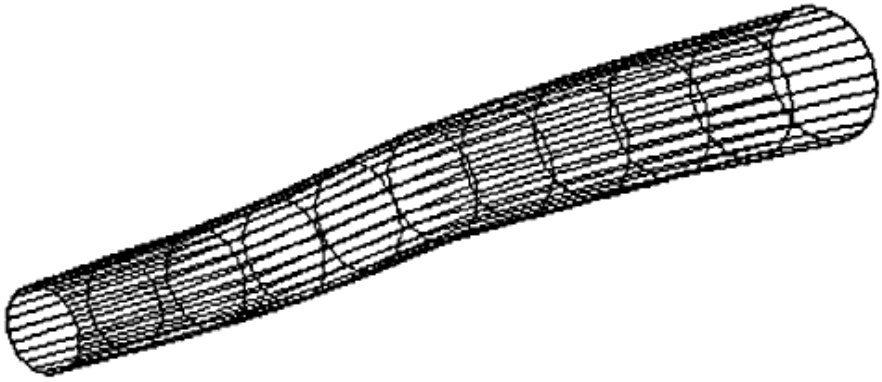


Figure 14 : A log represented by the polygonal cross-section model. (Zeng, 1995)

According to Zeng (1995) the best results can be expected if real shape laser scanning systems are used. Because of the unavailability of such scanners during the current study, polygonal cross-section models had to be constructed on a manual method. Zeng also mentioned that one should consider reality and complexity when choosing a log modeling

method. When using a model that more accurately describes the shape of the model, the final solution will be more accurate, but the computing time will be longer, since more log information must be processed.

Another method to construct models of logs is by mathematical reconstruction of the log from board flitches. This method was used by Pinto et al (2002). They sawed the logs into 25mm thick flitches, which were then scanned and the information stored. Such information included geometric outline of the sawing surface, the log pith line and the location, size, shape and quality factor of each knot. All these measurements were stored as points in xyz co-ordinates. The process can be seen in Figure 15 (Pinto et al, 2002).

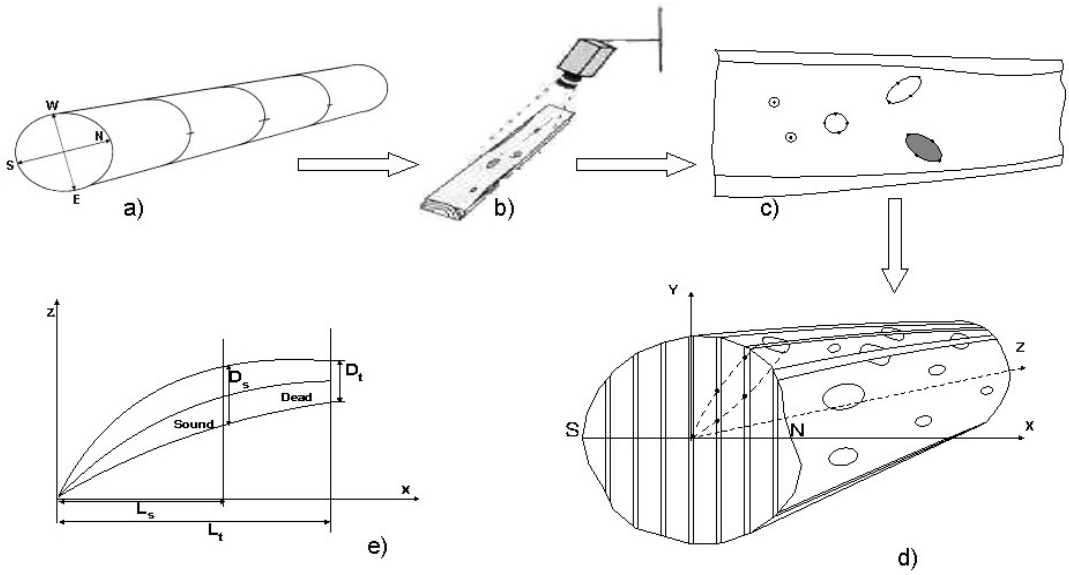


Figure 15: Log shape and internal knots reconstruction: (a) stem cross cutting into logs; (b) scanning of flitches; (c) marking knots on the flitch sawing surface; (d) log and knots reconstruction in the xyz co-ordinate system; (e) knot in the xz plane (Pinto, 2002)

Due to the fact that the equipment needed to make use of this method was not available, this method could not be used in the present study.

According to Smith (2003) there are basically three manual log breakdown methods that do not require any scanning technology and that can be used for the reconstruction of logs for further research, each with its own advantages and disadvantages:

- Firstly, logs can be crosscut. A disadvantage of this method is that knots can be situated in the middle of sawn sections since knots are parallel to the cuts.

- Secondly, logs can be rotary cut. A problem with this method is that the outside parts of the log that fall off before the log is cut to a perfect cylinder may be difficult to keep track of.

- Thirdly, logs can be cut longitudinally. A disadvantage of this method is that the end of a knot can be anywhere within the thickness of a board.

The rotary and longitudinal cutting methods both allow the veneer or timber to be used for further processing so that it does not go to waste. In line with the resources available, it was decided to make use of the crosscut method for this study. The method followed is described in section 3.2.

2.2 Optimizing using internal and external log information

Wessels (2009) did a simulation study to find the optimum or close-to-optimum sawing positions of small-diameter pine logs by using real external log shape information. Three positioning variables were used in the study: rotation, offset and skewing. Simulation scenarios were done at different increments of these variables to determine the effect of the increment sizes. Wessels (2009) found that there was a higher frequency of optimal solutions spread close to the conventional horns-up and horns-down positions (where the plane of maximum log curvature is vertical). In most cases the optimal position was not exactly on the conventional position but close to it. This was also investigated in the current study and is explained further in section 4.1.1.

When 4056 log positions were considered, the log volume recovery increased with 2,51%, compared to the conventional “horns-up” position (Wessels, 2009). He found that when only 90 positioning combinations were used, it resulted in an average increase of volume log recovery of 1,87%, which is only 0,64% less than considering 4056 positions. This is represented graphically in Figure 16.

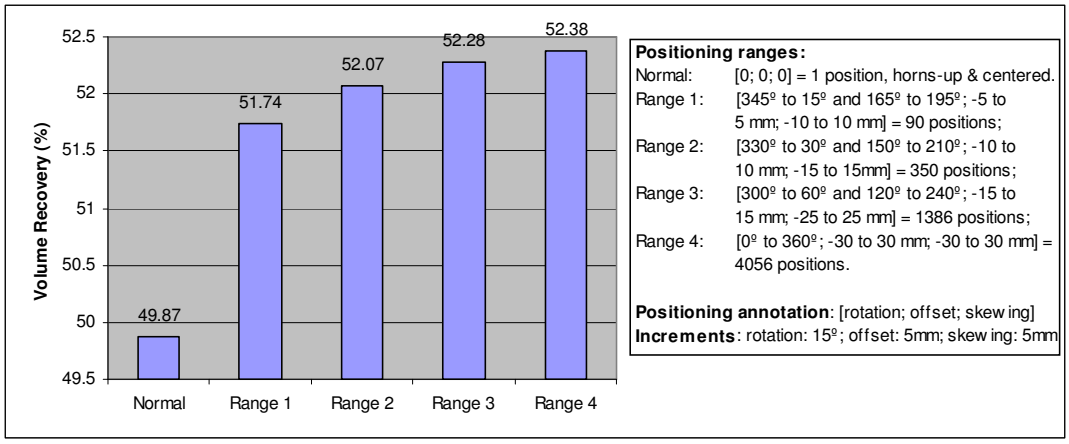


Figure 16: The average maximum volume recoveries obtained within four positioning ranges for the 60 sample logs (Wessels, 2009).

In another study, Du Plessis (2010) compared four meta-heuristic search algorithms to find the optimal or close-to-optimal position of small diameter pine logs for primary breakdown. He found that it was more rewarding to exhaustively search a small space around the high quality positioning regions than to search for solutions in a large search space using search algorithms. These results were very much affected by the sweep in the logs.

Zeng (1995) developed a log breakdown system to find the optimal pattern using external and internal log information. The system included log positioning in front of the headrig. Zeng made use of the programs SAW3DG and SLGRADER to simulate the cutting process and do timber grading. SAW3DG uses dynamic programming algorithms to find the optimal log breakdown pattern.

Zeng stated that certain operations research techniques such as linear programming, integer programming, dynamic programming and network techniques can be used for optimization during log board sawing, but only dynamic programming has been applied successfully. If a log breakdown optimization problem is properly formulated, it can be defined as a dynamic programming problem. A recursive equation can be established to find the optimal value (Zeng, 1995).

Figure 17 indicates the decision making process used by SAW3DG. The program variables were log rotation and skewing. The rotation and skewing increment sizes and the skewing range can be entered by the user. The program determines the total value which will be earned from a log under a current sawing position.

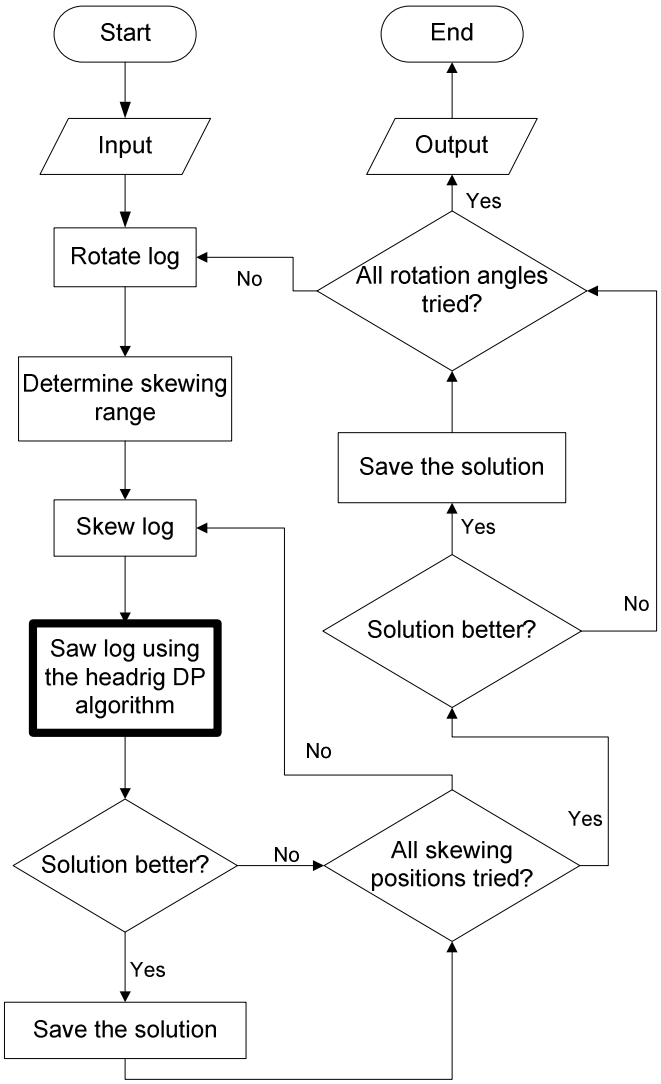


Figure 17: Log breakdown procedure of SAW3DG (Zeng, 1995)

Todoroki (1999) developed a model that integrates the primary and secondary breakdown of pruned *Pinus radiata* logs. Internal as well as external log information was considered during modeling. Todoroki developed dynamic programming formulations to determine the optimal cutting sequence of logs into slabs. She found that when value is maximised instead of volume, an overall increase of 16% in value can be achieved. Todoroki only considered live-sawing cutting patterns.

In a different study Todoroki (2001) acquired three-dimensional images of logs containing the external profile and internal defect information. Sawing simulation was done with the Autosaw program by live sawing and changing the opening face repeatedly for each log. Todoroki (2001) again found that a 16% higher value yield is obtained when value rather than volume was optimized. In the current study the value arrived at when maximizing for

value is compared to the value found when maximizing for volume. In section 5.2 the value determined by Todoroki (2001) is compared to the results of the current study.

Barbour (2003) assessed Douglas-fir and Ponderosa pine logs for product potential by diagramming the location, size and type of knots that are visible on the outside surface of the log. The Autosaw program was used to evaluate the processing options. All the logs were physically sawn, dried and graded. The log sawing were also simulated using Autosaw, which consistently underestimated the volume recovery by 10 to 15 percent. A correction factor could have been applied to compensate for this variance. By evaluating sawing of the logs in different sawing positions with the Autosaw program, it was shown that greater value could have been recovered from the small-diameter Douglas-fir logs.

Since the same Autosaw version is not used in the current study, it cannot be said that the Simsaw software also underestimates the recovery of the log. Because all the logs were sawn into pieces, the accuracy of Simsaw could not be determined. Even if it is inaccurate, it will not necessarily affect the outcome of the current study, since, presumeably all logs and all log positions will be miscalculated with roughly the same percentage. In the current study the different board values are compared with each other.

Occeña (1997) generated three hypothetical logs (grades 1-3) which were simulation sawn using six established log sawing heuristics; with and without using internal log information. Preliminary results showed that in the absence of an optimal log breakdown procedure, but with just internal log information, value recovery can be improved by 8,5% for grade #1 logs. For lower grade logs, timber value does not change significantly. Section 5.2 compares this value to the results of the current study.

2.3 Meta-heuristic optimization algorithms

The large number of log position options available make the exhaustive evaluation of every possibility through simulation impractical. In practice only a few seconds are available to come to a solution, not a few days. Thus a method had to be found to arrive at a solution in a much shorter time.

Rardin (1997) stated that when an optimization model is too large for exhaustive evaluation, heuristic algorithms can often yield very effective results. In many cases their optimality cannot be guaranteed; nor can it be determined how close they come to optimal. Since the full search space was evaluated in our data, the maximum found by the heuristic algorithms can be evaluated by comparing it to the global optimum.

This section shows how the algorithms implemented in this project works.

2.3.1 Genetic algorithm

This section provides the necessary literature background to support the development of the genetic algorithm for the purpose of this project (refer to section 4.2.2).

A genetic algorithm attempts to emulate the biological evolution process. The algorithm operates on a chosen constant-size population. When the algorithm is applied to an optimization problem, the following analogy can be drawn: A population of possible solutions is created (initialization). The population comprises a number of individuals (the population size). Each individual (referred to as a chromosome) represents a solution of the problem. Similar to parents producing children, two chromosomes in the population are combined to produce offspring. To produce these offspring the parents undergo steps of crossover and mutation. If a newly produced chromosome is better than a chromosome in the current population, it replaces that chromosome and becomes part of the current population. In this way, the population improves over time. The steps of the process described above are discussed in more detail below (Rardin, 1997) (Houck, 1995).

Initialization

A genetic algorithm must start with an initial population. The initial population can either be created by using another algorithm or it can be generated randomly. According to Houck et al. (1995) and Bekker and Groves (2001) it is most common to create the entire initial population randomly. However, in section 4.2.7, for purposes of this project, the hypothesis is considered that a non-random initial population based on trends found in the data after data inspection will result in a significant increase of value.

Selection function

There are several ways of selecting individuals to produce offspring with. These methods include: roulette wheel selection, scaling techniques, tournament, elitist models and ranking models (Houck et al, 1995).

In the author's opinion, roulette wheel selection is the best selection method, since this method gives individuals with higher fitness value a bigger probability to be selected to reproduce offspring with. The problem with applying this method in this specific project is the fact that in some cases the fitness values do not differ that much between individuals. For instance: two individuals' fitness value may be 370 and 366 respectively. When the percentage change between these two values is calculated, it is a very small, but we want the first value to be chosen with a much higher probability than the second. In our problem the percentage change in value recovery between options might be small but the financial effect on the business is large.

When this problem is encountered, Luke (2009) recommends that one should use the tournament selection method. The tournament selection method simply chooses a user specified amount of individuals randomly, and selects the fittest solution.

Crossover

Crossover is a step in the genetic algorithm which combines two chromosomes (parents) to produce two new chromosomes (offspring). The intention of crossover is to produce at least one chromosome which combines the good characteristics from each of the parents. Crossover occurs with a user specified probability. Should crossover not occur, the offspring are exact replicas of the parents (NeuroDimensions, 2002).

There are various ways in which the crossover process can be executed, including single-point, two point, uniform, arithmetic and heuristic crossover. For the purpose of this project, the single-point crossover method is used, since this method has the lowest degree of randomness (Obitko, 1998).

Single point crossover: One random point is selected at which to cross over. The binary string from the one parent up to the crossover point is used as the first part of the first offspring. The part just after the crossover position to the end is used as the second part of the second offspring (Obitko, 1998) (NeuroDimensions, 2002).

Parent1: 1001|0101

Parent2: 1100|1010

Offspring1: 1001|1010

Offspring2: 1100|0101

Mutation

Mutation is an operator that takes the offspring chromosomes that were created with the crossover operation and changes the bits of each offspring with a certain probability. Mutation prevents the algorithm from stagnating at a local optima. There are two mutation methods that can be used with binary string chromosomes, the flip bit method and another method suggested by NeuroDimensions (2002).

Flip bit: The value of each bit in a chosen gene is inverted. (a 0 changes to a 1, and a 1 changes to a 0). This method can only be used for binary genes.

Diwekar (2008) suggested a method of changing each bit of each chromosome with a set probability. This probability should be developed by means of trial and error.

NeuroDimensions suggests a good beginning probability is 0.01. The probability at which mutation should occur, should not be set too high, as this will result in the search becoming a primitive random search.

Termination

Termination is the method the algorithm uses to decide whether to go on searching for a better solution or to stop the search.

The following termination methods can be used: generation number, evolution time, fitness threshold, fitness convergence, population convergence and gene convergence (NeuroDimensions, 2002).

Generation number: The algorithm runs for a set number of evolutions. According to Houck et al (1995) this is the most common stop criteria.

2.3.2 Population based incremental learning (PBIL)

The PBIL algorithm was initially developed by Baluja (Bekker, 2008). It consists of the following concepts: set-up of the solution structure, probability vector, mutation, formation of the next generation and convergence.

Each of these concepts is described below:

Initialization

The initial population of the PBIL algorithm is similar to the initial population of the genetic algorithm. The initial population consists of a user specified amount of individual solutions. Each solution is made up of a certain number of bits. These bits represent the different variables of the individual in binary format. Thus each bit can be either a 0 or a 1.

Probability vector

The probability vector is a separate structure which consists of just as many bits as the individuals in the population. But each bit contains a certain probability instead of a 0 or 1. A specific bit in the probability vector indicates the probability that the specific element in a solution vector contains a 1. A low value in the digit indicates a low probability of the element in the solution vector containing a 1. The probability vector is the core of the PBIL algorithm. Initially each element in the probability vector contains 0.5. With every repeat of the algorithm, the probability vector changes.

The algorithm generates an initial population. The evaluation value of all the individual solutions (called solution vectors) is calculated and the solution vector with the highest evaluation value is selected to modify the probability vector. This modification of the probability vector is done with the following formula:

$$P^j(i) \leftarrow P^j(i) \times (1 - LR) + SV_B^j(i) \times LR$$

where

$P^j(i)$ = value of the *i*-th cell in the probability vector in the *j*-th generation

LR = learning rate (typically 0.1–0.4)

$SV_B^j(i)$ = value of the *i*-th digit in the solution vector (0 or 1) yielding the maximum evaluation value (the current 'best') *j*-th generation

Formation of the next generation

A random number is created for each element in each solution vector of the current population. If the number in the probability vector is smaller than the random number in the corresponding element of the solution vector, a 1 is assigned to that element of the solution vector. Otherwise a 0 is assigned. This forces each element in the probability vector to converge to either 1 or 0. This new generation is then used in the same manner that the initial generation is used.

Termination

The algorithm is considered as converged and is stopped when every element of the probability vector is either smaller than 0.05 or higher than 0.95. When the algorithm has converged, all elements with a value below 0.05 are changed to 0, and all elements higher than 0.95 are changed to 1 (Bekker, 2008).

2.3.3 Simulated Annealing

The idea for a Simulated Annealing algorithm was first published by Metropolis et al. in 1953. The algorithm is based on the annealing process in solid materials. When a solid is heated past its melting point and cooled, crystals are formed. When the material is cooled rapidly, the crystals formed will contain imperfections, but when it is cooled very slowly, fewer imperfections are formed, making the material stronger. The algorithm search can be seen as giving the atoms in the material that are trying to find the lowest form of energy enough time to move around and settle into place.

Initialization

Unlike the Genetic Algorithm and the PBIL algorithm that work with a population of solutions at a certain time, the Simulated Annealing algorithm only works with one solution at a time. An initial solution can either be generated randomly or generated by using another heuristic. The initial temperature of the algorithm as well as the cooling rate is set by the user.

New solution

The generation of a new solution and the acceptance of a solution depend largely on the current temperature (Moins, 2002; Ma et al, 2010). If the initial temperature is too low, the algorithm will most likely remain at a local optima. If the initial temperature is too high, the algorithm will basically become a random search (Diwekar, 2008).

Selection

A new solution is generated and if it is a better solution, it is automatically accepted. If the new solution is not better, it is accepted with probability $e^{-\frac{\Delta E}{kT}}$

where

ΔE – difference in fitness values of old and new solution

k – Boltzmann constant

T – Current temperature

Termination

The algorithm can be stopped when either one of the following criteria is met: when the temperature reaches a certain user specified temperature, when a specified number of moves have been made, or when no significant change has been made in a certain number of moves (Diwekar, 2008) (Po Wong, 1995).

2.3.4 The Cross-Entropy method (CE)

The Cross-Entropy method was first introduced by Rubenstein. The main difference between the Cross-Entropy method and the Simulated Annealing and Genetic algorithms is the fact that the Cross-Entropy method performs a global search, whereas the other two perform more of a local search. The Cross-Entropy algorithm generates a sequence of random solutions that converge randomly to the optimal or near-optimal solution. The Cross-Entropy optimization method aims to minimize the variance between solutions; when the variance drops below a certain point, the algorithm is stopped (De Boer, 2005).

Perelman and Ostfeld (2007) did a study to optimize the design of a water distribution system. They set out the following steps for the Cross-Entropy algorithm:

Initialization

Set an iteration counter t = 0.

Create a probability vector p_0 with components $p_{0,i}$ (i = 1,…..,m) where $p_{0,i}$ is the probability of choosing node i at iteration zero and m is the total number of nodes. Normally all nodes in the probability vector are initially set to 0.5.

Probability vector

By using the probability vector, N sample vectors are generated, each of size m and each consisting of a 0 or a 1.

Determine the fitness value for each sample vector.

Sort all sample vectors in decreasing order according to their fitness value.

Select the top user specified percentage (e.g. 10%) performance vectors to update the probability vector.

The probability vector is upgraded as follows:

$$p_{t+1,i} = \frac{Bt,i}{TBt}$$

Where B_{ti} is the total number of times node i was chosen in the solution vectors selected in the previous step. TB_t is the total number of solution vectors selected in the previous step. A smoothing parameter α is used to control the rate at which the performance vector changes. The performance vector is changed with the following equation:

$$p_{t+1,i} = a * p_{t+1,i} + (1 - \alpha)p_{t,i}$$

Termination

Check stopping conditions: if the standard deviation between a user specified amount of iterations remains unchanged, the algorithm is stopped; else return to step 3.

3 Gathering of study data

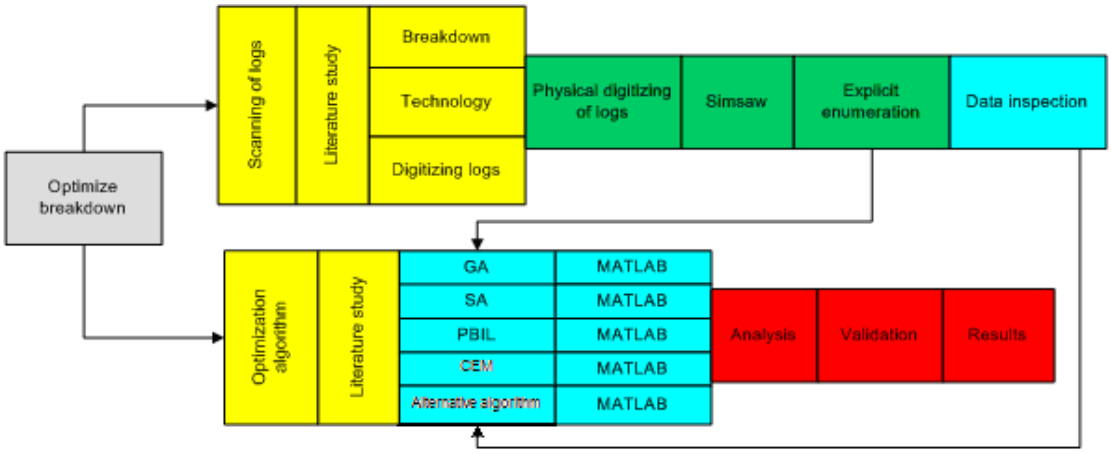


Figure 18: Thesis roadmap

3.1 Logs used

Pinus radiata logs were used for this study. It was decided to use pine logs since 51% of the 1.3 million ha plantations in South Africa is covered with softwood logs, of which most are pine trees (Godsmark, 2006). *Pinus radiata* is the most common specie found in the Southern and Western Cape areas of South Africa (Forestry Economics Services, 2009). The logs used for the study were harvested at Tokai plantation near Cape Town. The logs were debarked with the ring debarker at Cape Sawmills in Stellenbosch before use in the study. Table 1 shows the forestry history of the logs, which include the age of the trees when thinning (cutting down weaker trees to improve the growth of stronger trees) and pruning (cutting off the lower branches) were done.

Thinning is done to increase the amount of sunlight and decrease the competition of the remaining trees, resulting in an increase of the growth rate.

Pruning influences the quality of the timber produced. If pruning is done too often, the tree will have too few branches, thereby decreasing the growth of the tree. If pruning is done too infrequently, the timber will have too many knots. Figure 21 on page 42 shows a knot in a log that has been pruned. Differences in pruning and thinning practices are also one of the major determinants of the differences between *Pinus radiata* from South Africa and softwood

logs from especially the northern hemisphere. It is important to realize that the quality of raw material in a study such as this might play an important role in the outcomes of the study and that all the results might not be applicable to other log resources.

Table 1: History of logs

Operation	Age (years)
Plant	0
Thin	9
Prune	13
Thin	14
Thin	19
Fell	37

3.2 Data collection technique

Because there are no log scanners available in South Africa to scan logs internally, images of logs had to be constructed manually. Different possible methods to create such images were discussed in section 2.1.3.

3.2.1 External shape

Scanners which can scan the external shape of a log are available in South Africa, but the ones available to us did not provide enough data points to be used in this study. For instance, the scanner at Cape Sawmills in Stellenbosch only offers the coordinates of four points at a certain cross section of the log. More detailed images of the logs had to be produced to make accurate calculations.

To develop such images the logs were put on a lathe with a length of 3m (Figure 19). The coordinates of the surface of the logs were measured using a laser distance measure (Figure 20). A Bosch DLE 40 Professional laser measurement device was used. The tool has a typical measuring accuracy of ±1,5 mm.

Measurements were taken at a series of cross-sections at specific intervals along the length of the log and at specific rotational degrees. A measurement was taken every 30cm along the length of the log, and the log was rotated 15° at a time.

Figure 19: Lathe used to measure external shape of log

Figure 20: Laser measure used

When Simsaw (refer to section 3.2.3 for more information) uses the coordinates of the log shape, the area between all the coordinates entered are interpolated. To ensure that the log was represented as accurately as possible, the cross-sectional measurements were sometimes not taken exactly at every 30cm, but moved a few centimetres to one side. This was done to avoid taking a measurement on a branch stub, since this will result in wrongly simulating the end of the branch stub as the diameter of the log. It was important to take measurements where there were big indentations in the log, else the simulation software will cut 'n board where there was actually no wood. Remaining bark was removed where measurements were taken.

When the measurements taken with the laser distance measurement device were subtracted from the constant distance of the laser to the centre rotation axis of the log it only gives the radius of the log at that specific point. To create the log image in Simsaw the coordinates of the points had to be placed on the x and y planes. To calculate these x-y coordinates, the following calculations were made:

To calculate x coordinates:

$X_q = - (K-L_q)*(\cos \Theta_q)$

To calculate Y coordinates:

$Y_q = (K-L_q)*(\sin \Theta_q)$

where

K – Distance from laser measurement device to rotation axis

L_q – Measurement taken with laser measurement device

Θ_q – Angle from 1st measurement to measurement number q

3.2.2 Internal information

To obtain the internal information of the log, the logs had to be sawn. The logs were cut with a chainsaw into approximately 5 cm thick discs. The cuts were made from one side of the log, until a knot whorl was found, then a thicker disc was sawn to try and include the whole length of the knot in the disc.

For Simsaw to draw the knot in the log, the following information for each knot had to be measured: the rotation angle of the knot in the disc (azimuth), the angle between the log's longitudinal axis and the branch's longitudinal axis (knot or branch angle), the maximum diameter of the knot, the length of the knot and the dead knot length. Figure 21 shows a typical cross section of a knot on which most of its properties were measured. To expose the knot to measure all this information, the discs were sawn through the pith of each knot with a bandsaw. All the information was measured with a ruler and a protractor.

Figure 21: Crosscut through a pruned knot which was used to measure the required properties

Knot shape

In a *Pinus radiata* tree, branches are usually formed in whorls and start at the pith or centre of the tree. Some branches called epicormic shoots might start from the outer parts of the stem. The shape of a knot can be described as conical with a bent centrelineas shown in Figure 21. Simsaw modelled a branch as a cone which does not take into account the slight bent shape of the branch. The error resulting from this simplification, however, will be very small since it has a small effect on the largest value determinant which is the knot size. Usually, boards without knots or with the least possible number of knots are preferred and result in higher value boards, though in some cases knots are used for artistic effect. In pine timber, boards are normally sawn to produce boards with the least possible number of knots, since clear timber earns a higher price. In a longitudinally sawn board, a knot will appear as a circular solid piece of wood around which the grain of the rest of the wood "flows". Knots affect the mechanical properties of timber for the worse and roughly 70% of South African sawn timber are used as structural timber (Crickmay and Associates, 2010).

Defects not considered in study

While gathering the internal information of the logs, a few defects were discovered that cannot be modelled by Simsaw. Such defects include resin canals and resin pockets.

3.2.3 Simsaw

Simsaw is a simulation software package that virtually saws wood logs to calculate certain outputs such as volume recovery, board sizes, board grades and value (Rand) for all boards in one log. All the log shape information gathered, including internal and external information, was entered into Simsaw to create 3D images of the logs. Such a log image can be seen in Figure 22.

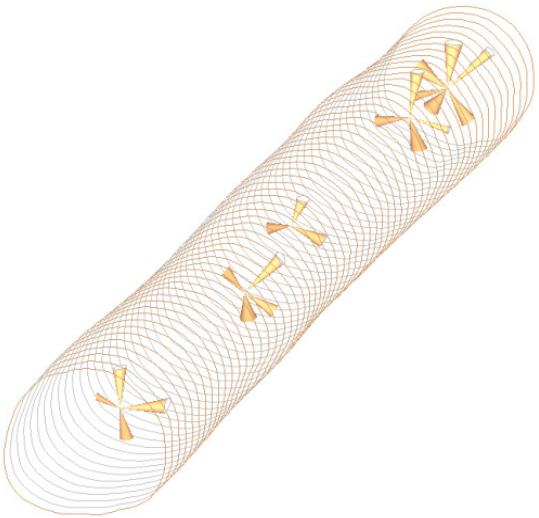


Figure 22: 3D image of reconstructed log as recreated in Simsaw

One disadvantage of this simulation method is the fact that it cannot be verified how accurately the virtual sawing of the logs represent the physical sawing of the logs (all the logs were destructively evaluated). Since we only compare our results for volume recovery and value yield with each other, however, it does not matter, as both will be under or over estimated with the same percentage if there are inaccuracies. It also does not make a difference with the optimization algorithms, since they work with the same generated data.

Just like all machines and processes are physically set up in a sawmill, various settings can be entered into Simsaw. All the settings entered are given below:

Product definitions

Both the wet and dry dimensions of board dimensions must be specified. The wet values are the dimensions of the board directly after it is sawn. But all boards must be dried, and because of the moisture loss during the drying process the boards shrink. The dry value is more or less 95% of the wet value. In the wood industry, boards may only be of a certain thickness or width. The following dry board dimensions were entered into Simsaw as allowable sizes:

Widths: 76, 114, 156 and 228 cm

Thickness: 25 and 38 cm

Board grades

In practice each board in a sawmill is graded according to strict criteria set out by the SABS. Each board grade has a different price per m^3. The prices for all board grades and sizes are listed in Table 16 of Appendix A. Simsaw uses the same criteria as contained in SANS 1783 parts 1-4 to grade boards according to knots.

Grades: Packaging(25), XXX 38), Utility(25), S5(38), S7(38) and Clear(25 & 38). (The number in brackets indicates the allowable thickness for that board grade)

Table 2 contains the maximum allowable knot ratio per cross section for each board grade. The knot size is measured as the distance that the knot extends across the width of the face side, back or edge of the board.

Table 2: Maximum allowable knot ratio as a percentage of the face or edge of a board according to SANS 1783 parts 1-4

	Maximum % knot				
	Worst single face	Combined faces	Worst edge	Combined edges	All surfaces
Packaging	∞	∞	∞	∞	∞
XXX	∞	∞	∞	∞	∞
Utility	67	∞	100	150	83
S5	67	∞	∞	125	112
S7	50	∞	∞	75	58
Clear	0	0	0	0	0

The ∞ sign indicates that there is no limit for that board grade under that condition.

The "All surfaces" grading criteria for the boards in SANS 1783 are given in the form "width of face + 1 ½ the width of edge". Since a board can consist of a few combinations of dimensions, it is not that straight forward to determine the percentage of knot at a certain cross section. The formulas below show how these percentages are calculated.

$$Utility\ grade\ \% = \frac{W}{W+T}$$

$$S5\ grade\ \% = \frac{W+(1.5\times T)}{W+T}$$

$$S7\ grade\ \% = \frac{0.75\times W}{W+T}$$

W – Width of board

T – Thickness of board

The value rendered will differ for every different dimension of the board, but Simsaw only allows for one value for the grading criteria. So these percentage values were calculated for each possible dimension of the board, and the average taken as the criteria. This is only the case for the Utility, S5 and S7 grades. Table 3 shows the highest and lowest values for the different board dimensions, and the average used.

Table 3: Values used for "All surfaces" grading of boards

Board grade	Highest value	Lowest value	Average used
Utility	90	38	83
S5	118	108	112
S7	64	50	58

Machine settings

All settings that can physically be implemented in a sawmill can be set in Simsaw. The settings were set up as follows:

Primary breakdown:

Kerf size: 4mm

Secondary breakdown:

Kerf size: 4mm

Cant guiding (“Round the curve”): Half taper

Edging/X-Cut/Resaw:

Edging for maximum: Volume

Number of edging blades: 2

Edger kerf: 5mm

Sawing pattern

All the logs were sawn with the cant sawing method, as described in section 2.1. The pattern used and the size of the boards during simulation sawing of logs are shown in Figure 23.

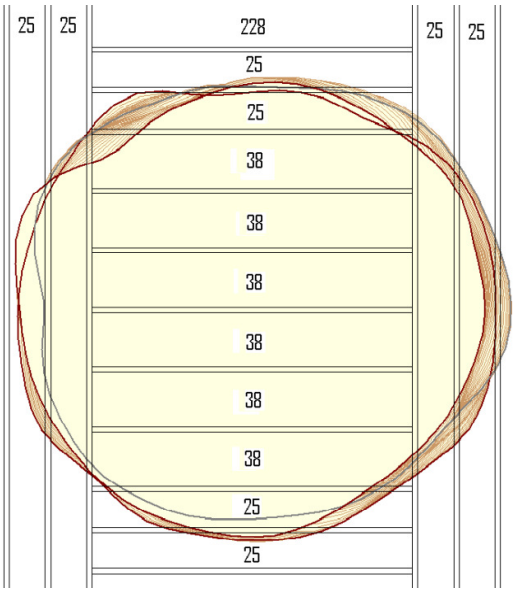


Figure 23: Sawing pattern used for Simulation sawing

3.3 Explicit enumeration

With all the log shape information and the sawing settings in place, the logs could be simulation-sawn. Each log were sawn in 808 020 different positions. The simulation-sawing log positioning information can be seen in section 1.4.1. This process of generating data by simulation-sawing the logs can also be called explicit enumeration.

Table 4 shows the time it took Simsaw to simulation-saw each log.

Table 4: Time it took Simsaw to do simulation sawing

Log	Time per position (s)	Total simulation time (days)
1	0.77	7.2
2	1.5	14
3	1.26	11.8
4	1.3	12.2
5	3.3	30.9
6	5.86	54.8
7	2.15	20.1
8	5.22	48.8
9	3.23	30.2
10	0.8	7.5

An example of the outputs of the simulation sawn logs can be seen in Table 15 in Appendix A. The aim was to optimize Board value. The volume recovery results were used to evaluate the difference for optimizing for optimal volume recovery versus optimizing for best value recovery.

The explicit enumeration data was used by the optimization algorithms. Each time the algorithm reads a board value from the file, it is as if Simsaw has to simulation-saw that specific log position. Explicit enumeration was done since the current Simsaw version does not have the ability to communicate directly with programming software like MATLAB. The whole Excel file with all explicit enumeration data is read into an array in MATLAB. The process through which the optimization programs in MATLAB read from the array is illustrated in Figure 24.

Row ID	Rotation	Offset	Skewing	Board value
14241	78	-3	0	378.1278
14242	78	-3	3	375.3414
14243	78	-3	6	373.4838
14244	78	-3	9	371.6261
14245	78	-3	12	354.2723
14246	78	-3	15	354.2723
14247	78	-3	18	348.5041
14248	78	-3	21	342.4913
14249	78	-3	24	330.1237
14250	78	-3	27	325.2843
14251	78	0	-27	324.1368
14252	78	0	-24	340.4005
14253	78	0	-21	367.7155
14254	78	0	-18	362.1426
14255	78	0	-15	362.1426
14256	78	0	-12	367.8132
14257	78	0	-9	370.6973
14258	78	0	-6	373.4838
14259	78	0	-3	375.3414
14260	78	0	0	378.1278
14261	78	0	3	376.2702
14262	78	0	6	375.3414
14263	78	0	9	377.199
14264	78	0	12	354.2723
14265	78	0	15	354.2723
14266	78	0	18	341.0735
14267	78	0	21	343.7623
14268	78	0	24	322.6932
14269	78	0	27	316.925
14270	78	3	-27	327.8521
14271	78	3	-24	330.7362
14272	78	3	-21	354.8703
14273	78	3	-18	362.1426

The optimization algorithm performs its steps. When the optimization algorithm needs to evaluate a certain log position, the program determines in what row that specific combination of the three variables is. The board value is then looked up at that row in the array. If, for instance, the algorithm wants to evaluate the log position 78;0;-21, it is determined that it lies in row 14253, the specific row is checked up in the array, and the board value in that row is returned to the algorithm.

Figure 24: The process for handling board values from different log positions

4 Mathematical model

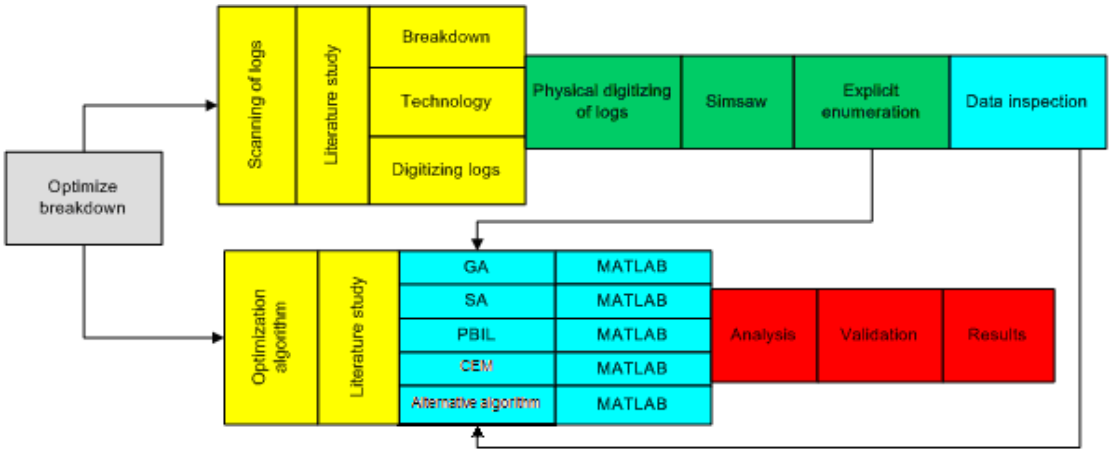


Figure 25: Thesis roadmap

4.1 Data inspection

Before a mathematical model was developed, the data from the explicit enumeration had to be analysed to identify trends in the data that could assist in an algorithm that searched more intelligently. Since there were 808 020 lines of data for each log, this was not an easy task. Data were analysed by looking at the influence of the different variables in the sawing procedure as well as the differences in log shape had on the board value. Variables for the sawing procedure include log rotation, skewing and offset. Variables for log shape include log taper, sweep, ovality and defect core. Other variables that could influence the optimal log position included length and diameter, but these could not be investigated in this study since all the logs had the same length and fell in the same log diameter class.

As previously mentioned, the search space for the three variables was rotation[0º;360º], offset[-100mm;100mm] and skewing[-100mm;100mm]. The maximum board value, in most cases, was within a certain region near the zero offset and zero skewing positions. This is discussed in more detail in section 4.1.2 below. In Table 5, the board values were compared for each of the 10 respective logs when the offset and skewing variables were restricted to only change between -27 and 27 mm. The maximum board value obtained did not change much. The only two logs whose values changed were logs 5 and 7, and log 5 did not change much. Compared to the other logs, log number 7 changed with a fairly large amount. Nevertheless, it was worthwhile to decrease the search space, since the amount of

possible log positions decreases from 808 020 (180 x 67 x 67) to 64 980 (180 x 19 x 19), which considerably decreased search time. The conclusion is that in practice optimization searches will probably benefit from limiting the search space to areas near the zero offset and skewing positions.

Table 5: The maximum board value for searching offset and skewing between -100mm and 100mm compared to -27mm to 27mm

Log	1	2	3	4	5	6	7	8	9	10
Best value	392.25	480.37	453.82	421.31	441.61	241.64	264.86	328.91	298.13	549.87
Best value_27	392.25	480.37	453.82	421.31	440.19	241.64	251.05	328.91	298.13	549.87

4.1.1 Effect of rotation

To detect the effect of log rotation on the outcome of the board value, the rotation at certain fixed offset and skewing positions were obtained. Rotation could not be tested at each offset and skewing combination, since there are 361 (19 x 19) combinations at each rotation position. Combinations of offset and skewing were selected to represent the whole range of possible positions. Combinations selected are shown in Table 6. To test the effect of rotation, the skewing and offset combinations below were kept fixed while running through all possible rotation positions. Figure 26 shows the results for these tests of log 1. The rest of the logs' graphs can be seen in Figure 68 to Figure 77 of Appendix A. The further the offset and skewing variables move from the zero, the lower the yield becomes.

It can also be noted that the maximum rotation position for one offset and skewing combination is not necessarily the maximum position for another offset and skewing combination. If this were the case, it would have greatly simplified the search for the global optimum. Furthermore, had this been the case, a search for the maximum rotation position at any offset and skewing combination could have been conducted, followed by a search for the maximum offset and skewing at that rotation position.

If the skewing or offset variable is changed within a certain range, the value yield graph does follow more or less the same pattern. This trend in the data was the main factor incorporated during the development of an alternative optimization algorithm, which is described in section 4.2.6.

Table 6: Skewing and offset positions to test rotation effect

Series in Figure 26	Skewing	Offset
Series 2	0	0
Series 3	0	-15
Series 4	0	30
Series 5	0	75
Series 6	-15	0
Series 7	-15	30
Series 1	75	75

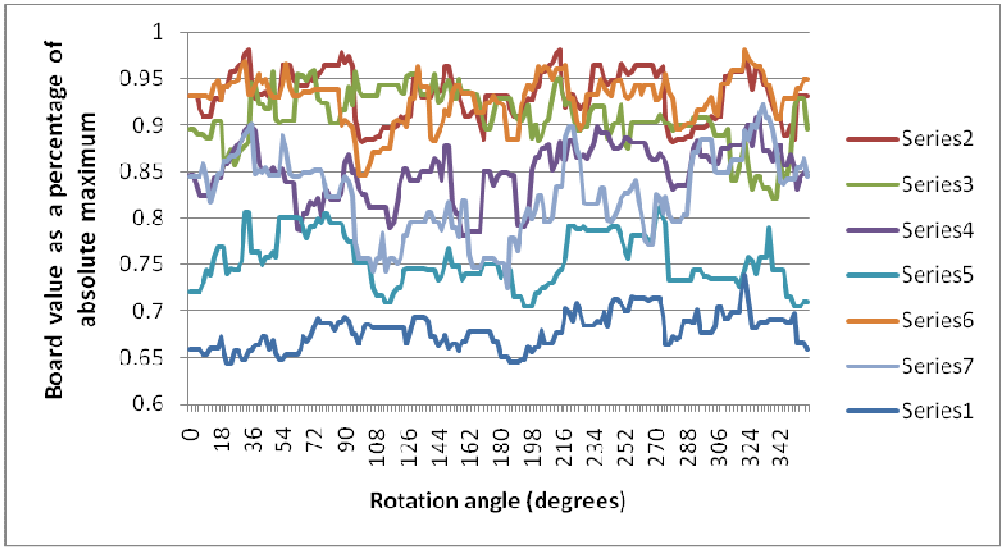


Figure 26: Effect of change in offset and skewing on board value recovered

Figure 27 provides a representation of the situation when all logs are rotated so that the maximum board value is at zero degrees when skewing and offset is zero. The log shape does have a significant influence on the outcome of the value yield. Wessels, (2009) also found this phenomenon in his results. He found that the maximum values are almost always close to the "horns-up" or "horns-down" positions. The "horns-down" position is 180° from the "horns-up" position. Thus, it can be speculated that the results found in Figure 27 can be ascribed to sweep in the logs. The logs used in this study did not have a significant amount of sweep and where there was sweep it was almost impossible to detect the direction thereof. It was also investigated whether or not ovality played a role on the value yield; there was no indication that ovality had a big influence on the optimal log position. Other aspects

such as knot position and taper could also have influenced the optimal log position, but because of all the factors that can influence the optimal log position, it is very difficult to determine which log characteristic made the biggest difference. What can be concluded is that there is a characteristic or a combination of different characteristics of the log that makes that a near optimal log position is 180º from the actual optimal.

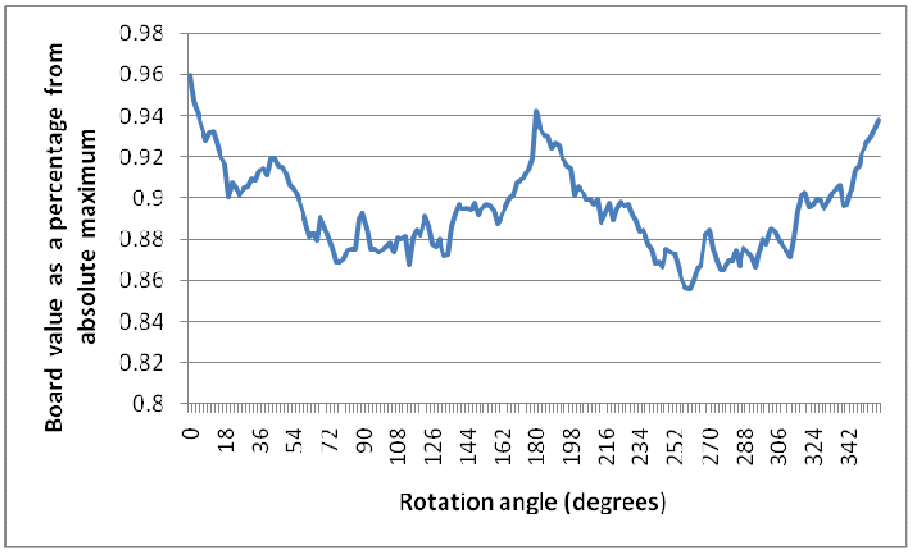


Figure 27: Average of all 10 logs, where the rotation angle is positioned so the maximum board value is at 0°

4.1.2 Effect of offset and skewing

Figure 29 to Figure 38 indicate the board value at 0° rotation. The effect offset and skewing have on the board value is illustrated in these graphs. In Figure 28, the colour intensity of the red area gives an indication of the board value.

The optimum board value does not necessarily lie on the zero skewing and offset positions, but in most cases it is not far from the zero positions. This was also found by Wessels (2009). The maximum position normally lies within the -27 and 27mm range.

The data also normally follow a trend whereby when either offset or skewing increases or decreases in value, the other variable should be changed in the same direction. If a log is skewed in a certain direction, the log has to be moved in the same direction to keep it in the sawing range of the framesaw.

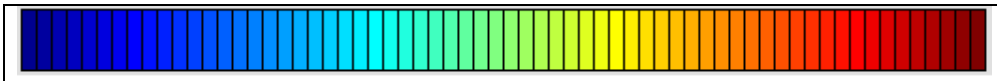


Figure 28: Colour scheme for MATLAB graphs. (Left equals low numbers, increasing to right)

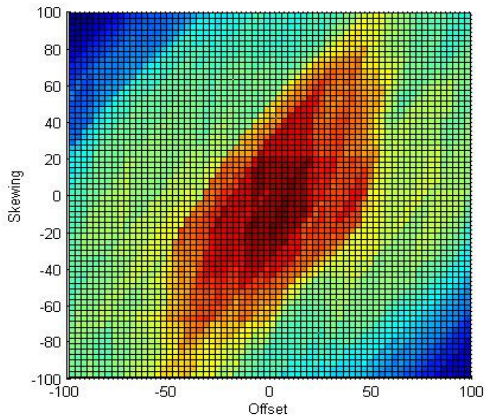


Figure 29: Log 1 at 0° rotation

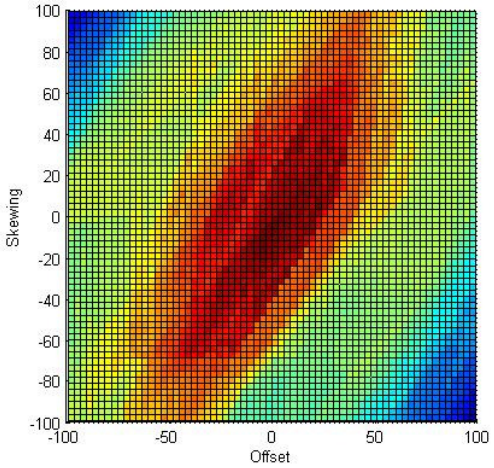


Figure 30: Log 2 at 0° rotation

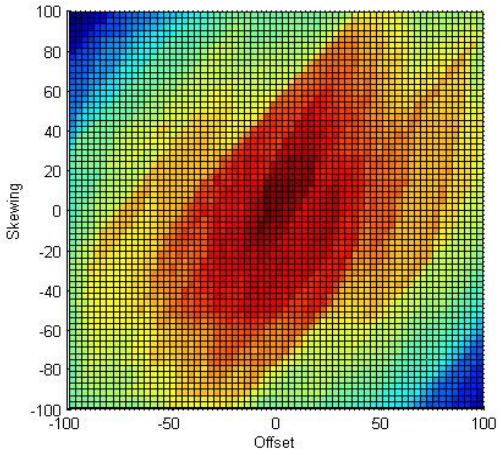


Figure 31: Log 3 at 0° rotation

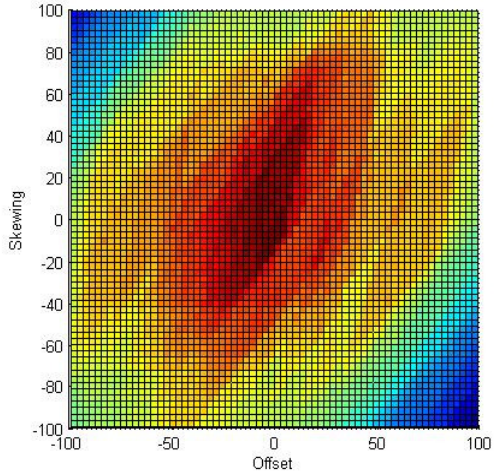


Figure 32: Log 4 at 0° rotation

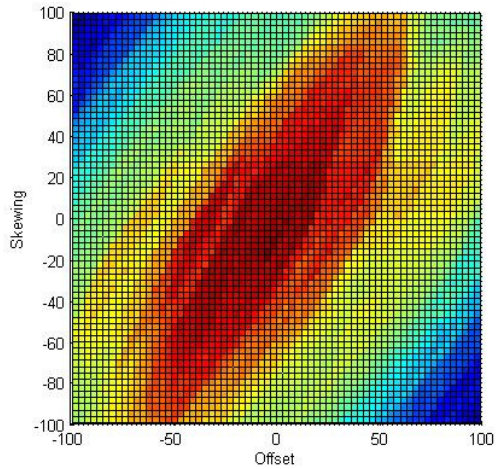


Figure 33: Log 5 at 0° rotation

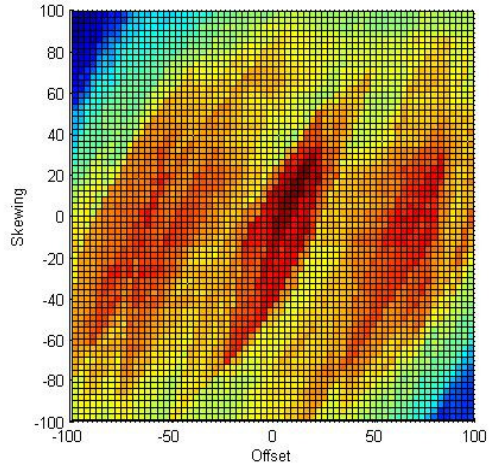


Figure 34: Log 6 at 0° rotation

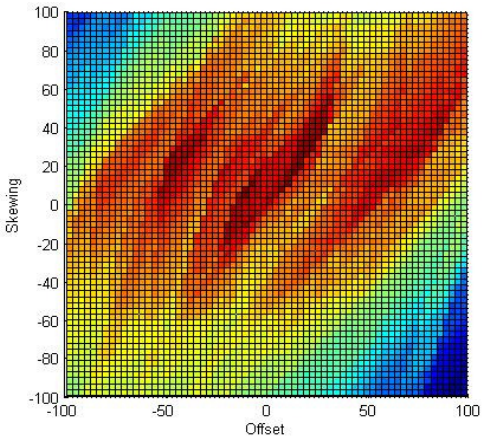


Figure 35: Log 7 at 0° rotation

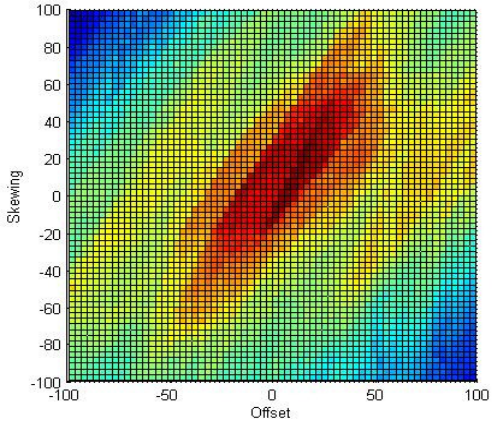


Figure 36: Log 8 at 0° rotation

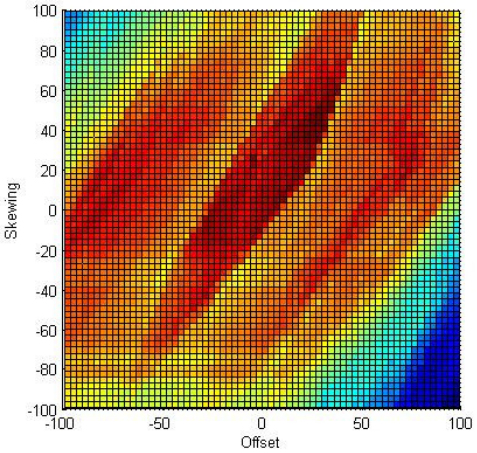


Figure 37: Log 9 at 0° rotation

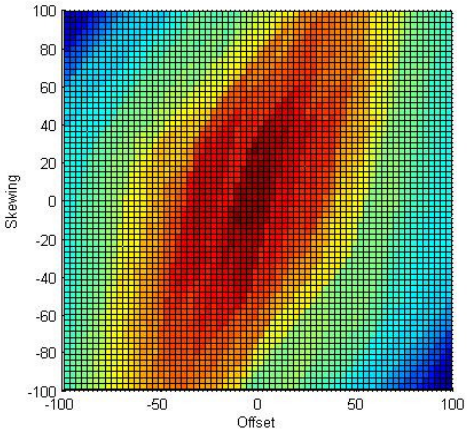


Figure 38: Log10 at 0° rotation

From Figure 34, Figure 35 and Figure 37 it is apparent that these logs have three fairly large local optimums. An explanation for this phenomenon is that when a log is moved beyond a certain offset, it loses recovery of one board on the opposite side than to which the log is moved. But when it is moved past a certain point, the next blade of the framesaw starts to cut an extra board from the log on the side to which the log is moved. This still does not affect the outcome of the optimal position, since the local optimum nearest to the zero offset and skewing positions always outperforms the local optima further away. The above described phenomenon is illustrated in Figure 39 to Figure 42. Figure 39 to Figure 41 illustrate the boards that will be sawn from the log in the given position. The number in block brackets corresponds with the number in Figure 42.

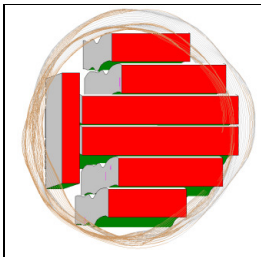


Figure 39: Log9 Position 0;24;39 [1]

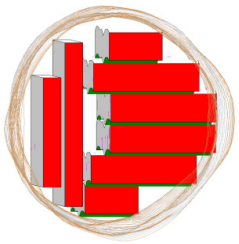


Figure 40: Log9 Position 0;63;18 [2]

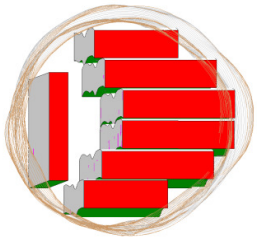


Figure 41: Log9 Position 0;-24;21 [3]

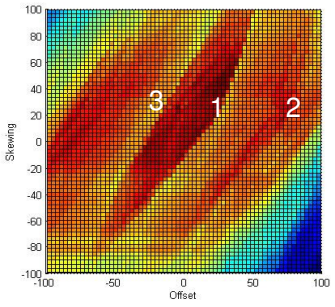


Figure 42: Log 9

4.2 Meta-heuristic optimization algorithms

To determine the constant values in each of the algorithms, except for the Alternative algorithm, each algorithm was run 10 times for each log under the same constant values and with the same number of iterations. The number of iterations was increased from 100 to 1000 with increments of 100 each time. For each amount of iterations and each constant combination, the average of all 10 runs from all logs for a certain amount of iterations and with each constant combination was taken to compare against each other. The initial values tested for these constants, were values that worked well for other applications of these algorithms, but since these values will differ slightly for different applications, the values that were tested in the different versions, was by changing the values in an upward and downward direction.

4.2.1 MATLAB

The optimization algorithms were coded in the software program MATLAB. MATLAB stands for **Mat**rix **Lab**oratory. MATLAB is a numerical computing environment that allows matrix manipulations, plotting of functions and data, and implementation of algorithms. For this reason, MATLAB was chosen as the program to use during this project. All of these features were used during the execution of the project. Large amounts of data had to be plotted during the data analysis stage, the algorithms had to be coded and executed, and the program had to work with large data arrays.

The complete MATLAB code for all the algorithms is shown in Appendix B.

4.2.2 Genetic algorithm (GA)

The literature background concerning this algorithm is given in a previous chapter (refer to section 2.3). Below, the development of this algorithm within the context of the project is discussed:

The results of the different simulation runs described earlier in this section are indicated in Figure 43. Table 7 gives the values for all the constants for the different combinations of constants tested (each combination of constants tested is called a version). Different versions can only be compared with each other if only one of the variables differ. Different versions are compared against each other according to the results they show in Figure 43. When versions 1, 2 and 3 are compared, it is apparent that version 2 rendered the best results, indicating that it is best to have a mutation probability of 0.1. When versions 2 and 4 are compared with each other, version 2, which has a crossover probability of 0.8, yields better results. When version 2 and version 5 are compared, version 5, which has population size of 20, gives better results.

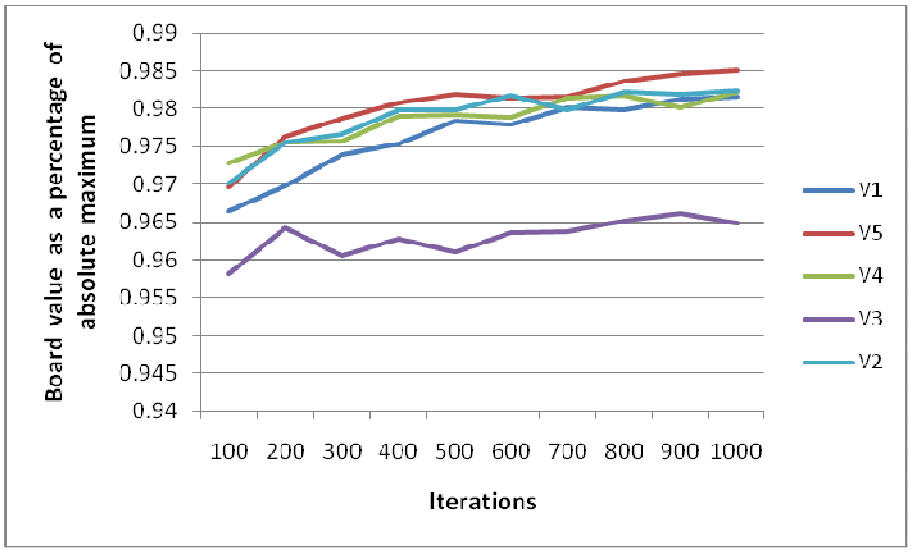


Figure 43: The effect of using different constant values for the genetic algorithm

Table 7: Different values evaluated for constants in genetic algorithm

Version	**1**	**2**	**3**	**4**	**5**
Probability for mutation	0.5	0.1	0.01	0.1	0.1
Probability for crossover	0.8	0.8	0.8	1	0.8
Population size	10	10	10	10	20

The steps followed by the genetic algorithm are stipulated below:

Step 1: A random population is created by randomly assigning 1's or 0's to an array with a size of 20 rows and 18 columns. This population is called the 'Present population'.

```
for i=1:PPL,
   PP(i,1:18) =
round(rand(1,18));   %Creates
array with PPL rows en 18
columns
end;   % i for loop
```

Figure 44: Random population generation

There are 20 rows because this is the size of the population as determined above. There are 18 columns in the array since this is the amount of bits needed to create the three variables in decimal format.

Step 2: Two parents are chosen by means of the tournament selection method (described in detail in section 2.3).

```
  %Choose two parents with tournament
selection method
  Crandom = ceil(rand(1,4) * PPL)
%Generate random numbers from one to PPL
if PP(Crandom(1,1),19) >
PP(Crandom(1,2),19)
    Parent1 = PP(Crandom(1,1),1:18)
else
    Parent1 = PP(Crandom(1,2),1:18);
end
if PP(Crandom(1,3),19) >
PP(Crandom(1,4),19)
    Parent2 = PP(Crandom(1,3),1:18)
else
    Parent2 = PP(Crandom(1,4),1:18);
end
```

Figure 45: Step 2; Choose parents to produce offspring with

Step 3: Crossover is performed on the two parents to create two children. The crossover operation makes use of the single point crossover method. Crossover is done at a random position with equal possibility throughout the whole chromosome. The two chromosomes created after crossover is called the offspring.

```
CrossWF = 0;
while CrossWF == 0
 Cif = rand;
 Cpos = 1;
 if Cif <= PC
    Cpos = ceil(rand * 17); %Choose place
where crossover should be performed
    Child1(1:Cpos) = Parent1(1:Cpos);
    Child1(Cpos+1:18) =
Parent2(Cpos+1:18);
    Child2(1:Cpos) = Parent2(1:Cpos);
    Child2(Cpos+1:18) =
Parent1(Cpos+1:18);
 else
    Child1 = Parent1;
    Child2 = Parent2;
 end
```

Figure 46: Step 3: Crossover procedure

Step 4: Mutation is done with a probability of 0.1. This means that every bit in the offspring chromosomes have a 10% possibility of changing value (changing a 0 to a 1 and *vice versa*).

Step 5: The two children are compared with the two weakest chromosomes in the population. The two best chromosomes of the four are then stored in the place of the two weakest chromosomes selected.

Depending on the number of iterations to be evaluated, Steps 2 – 5 are carried out a certain number of times.

```
MutWF = 0;
Child1_2 = Child1
while MutWF == 0
   Child1_2 = Child1
for y=1:18
   RM = rand
   if RM <= PM
       if Child1(1,y) == 1
           Child1_2(1,y) = 0
       else
           Child1_2(1,y) = 1
       end
   end
end  % y loop
```

Figure 47: Step 4: Mutation procedure

```
Convert = Child1_2(1,1:18)
ConvToDecGa;
Child1_2(1,19) = array(nommer,4)
BVC1 = Child1_2(1,19)
Convert = Child2_2(1,1:18)
ConvToDecGa;
Child2_2(1,19) = array(nommer,4)
BVC1 = Child2_2(1,19)
PP = sortrows(PP, -19)
C1 = Child1_2(1,19)
C2 = Child2_2(1,19)
PP9 = PP(9,19)
if Child1_2(1,19) > Child2_2(1,19)
    if Child1_2(1,19) > PP(9,19) &&
Child2_2(1,19) > PP(9,19)
       PP(PPL-1,1:19) = Child1_2(1,1:19)
       PP(PPL,1:19) = Child2_2(1,1:19)
    else
        if Child1_2(1,19) > PP(PPL,19)
           PP(PPL,1:19) =
Child1_2(1,1:19)
        end
    end
else
    if Child1_2(1,19) > PP(9,19) &&
Child2_2(1,19) > PP(9,19)
       PP(PPL-1,1:19) = Child2_2(1,1:19)
       PP(PPL,1:19) = Child1_2(1,1:19)
    else
       if Child2_2(1,19) > PP
          PP(PPL-1,1:19) =
Child1_2(1,1:19)
          PP(PPL,1:19) = Child2_2(1,1:19)
       else
          if Child2_2(1,19) > PP(PPL,19)
             PP(PPL,1:19) =
Child2_2(1,1:19)
          end
       end
    end
end
```

Figure 48: Step 5: Test if offspring should become part of population

4.2.3 Population based incremental learning (PBIL)

Table 8 shows the values of the constants in the different versions that were tested. **Error! Not a valid bookmark self-reference.** shows the results for all the different versions. When the different versions were compared to each other by considering the outcome in **Error! Not a valid bookmark self-reference.**, it was shown that Version 4 reaches a fairly good solution quite quickly, but versions 1 and 5 did give slightly better results when more iterations were carried out. The values from version 4 were used as the final values for the constants.

Table 8: Different values used for constants to determine best values

Version	**1**	**2**	**3**	**4**	**5**
Learning rate	0.1	0.2	0.3	0.3	0.1
Population size	10	10	10	20	20

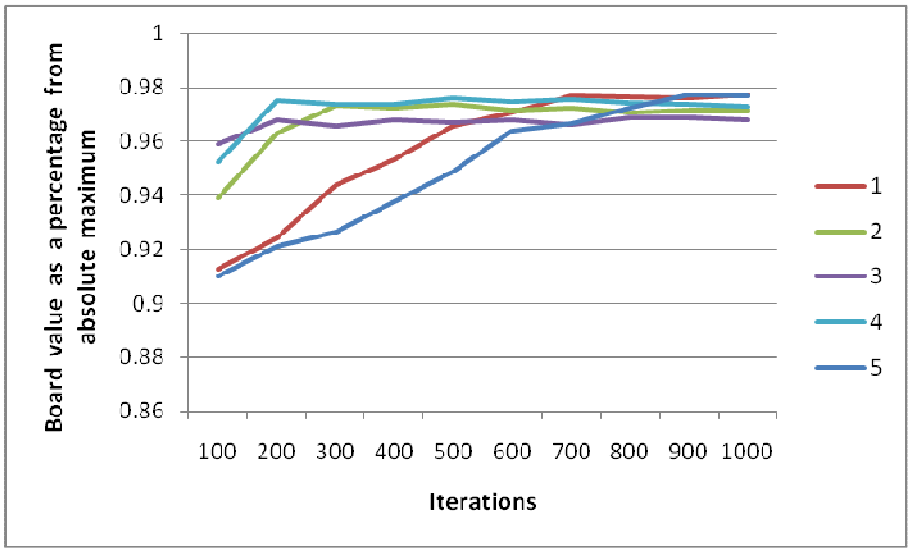


Figure 49: The effect of using different constant values for the PBIL algorithm

The PBIL algorithm works as follows:

Step 1: A probability vector was created with one row and 18 columns; a value of 0.5 was assigned to each cell in the vector. MATLAB code: `PV(1:18) = 0.5;`

```
for x=1:18,
   if SV(maxry,x) == 1
      PV(1,x) = PV(1,x)*(1-LR)+
SV(maxry,x)*LR;
    else
      PV(1,x) = PV(1,x)*(1-LR);
       end
    end;
```

Figure 50: Step 3: Updating the probability vector

Step 2: A random population was created by randomly assigning 1's or 0's to an array with a size of 20 rows and 18 columns. Each row in the matrix represents a binary number which can be converted to three decimal numbers, which are the three variables in the simulation problem. The MATLAB code is the same as this step in the Genetic algorithm.

Step 3: Each row in the population was converted to decimal numbers and the fitness of each was calculated. Each value in the probability vector was changed by using the row vector with the highest board value. Each bit in the probability vector was changed with the following formula: PV(i) = PV(i)*(1-LR) + MaxRow(i)*LR

Steps 2 and 3 were carried out a certain number of times, depending on the number of iterations to be evaluated. In certain applications of this algorithm, the algorithm was stopped once all numbers in the probability vector were either less than 0.05 or larger than 0.95. The author decided to make the number of runs a fixed amount since certain bits converge more slowly, which will make the number of iterations evaluated unnecessarily high.

The probability factor was converted to decimal numbers and the board value was calculated. These decimal numbers were the outcome of the algorithm.

4.2.4 Simulated Annealing

As in the other algorithms, a few simulation runs were done and compared against each other to determine the values of the constants in the algorithm.

Table 9: Values for constants with different simulation runs

Version	**1**	**2**	**3**	**4**	**5**	**6**
Initial temperature	5	5	10	10	10	20
Decrease in temperature	0.1	0.1	0.1	0.1	0.2	0.4
Amount of iterations evaluated before temperature is decreased	5	10	5	10	5	5
Total amount of iterations evaluated	250	500	500	1000	250	250

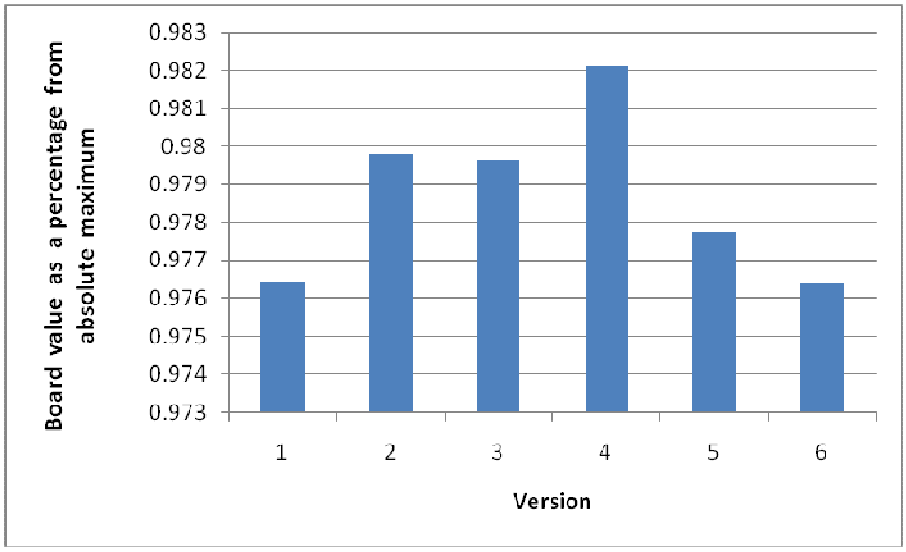


Figure 51: Board value for each version tested in the Simulated Annealing algorithm

Figure 51 gives the results of running the Simulated Annealing algorithm with different values for the constants. Version 4 gave the best results and was used as indicated in Table 9. Only if the number of iterations evaluated was the same can different starting conditions be compared with each other. When considering versions 1, 5 and 6, version 5 gave the best results. This indicates that a starting temperature of 10 degrees and cooling rate of 0.2 degrees was best. When the temperature was set higher, the algorithm accepted too many solutions in step 3 (described below), rendering an outcome that was worse than the current solution.

When versions 2 and 3 were compared with each other, it did not make a difference whether the initial temperature and the number of iterations per temperature were different – as long as the number of iterations evaluated was the same.

The steps of the simulated annealing algorithm are described below:

Step 1: The algorithm ran the Alternative algorithm V1 to get a fairly good starting solution; this made the algorithm more likely to come to a good solution than starting at a random place.

```
    while RoTF == 0
        ChRo = ceil((rand(1)-
0.5)*2*45*(T/Tin));   %Change in
Rotation variable (can change 90° to
each direction)
        if Ro + ChRo < 0
            RoTF = 0;
        else
           if Ro + ChRo > 179
               RoTF = 0;
           else
               RoTF = 1;
               Ro = Ro + ChRo;
           end
        end
    end
```

Figure 52: Step2: Changing the rotation variable

Step 2: Generate a new solution. The following procedure was carried out for each of the three variables (rotations, offset and skewing). A random number from -1 to 1 was generated. This random number was then multiplied with the result of the current temperature, divided by the initial temperature, in order to decrease the amount of change as the temperature of the algorithm decreases. This number was then multiplied by 90 (because the rotation variable may only change up to a maximum of 90 degrees in each direction) to obtain the amount of change in the rotation variable. The same was done with the offset and skewing variables. Each of them was restricted to change up to 27mm in each direction.

Step 3: Evaluate if the new solution must be accepted. If the new solution was better than the old solution, it was automatically accepted. If the new solution was not better than the old solution, it was accept it with the probability $p = e^{\frac{-(OS-NS)}{T}}$

where

OS = Old Solution

NS = New solution and

T = Current temperature

```
    delta = BV - BVN;   % BVN =
Board Value New
    if delta < 0
        Sol = SolN;   % SolN =
Solution New
        BV = BVN;
        m = m + 1
        val(m,1) = Sol;
        val(m,2) = BVN;
    else
        Pe = rand(1)   % Pe =
Probability to accept solution
        if Pe < exp(-delta/T)
            Sol = SolN;
            BV = BVN;
            m = m + 1
            val(m,1) = Sol;
            val(m,2) = BVN;
        else
        end
    end
```

Figure 53: Step3: Accepting new solution

Repeat steps 2 and 3 a certain number of times, depending on the number of iterations to be evaluated.

Step 4: Reduce the current temperature with 0.1 degrees and repeat steps 2 and 3.

Step 5: Repeat steps 2 to 4 until the temperature reaches zero degrees.

4.2.5 The Cross-Entropy Method

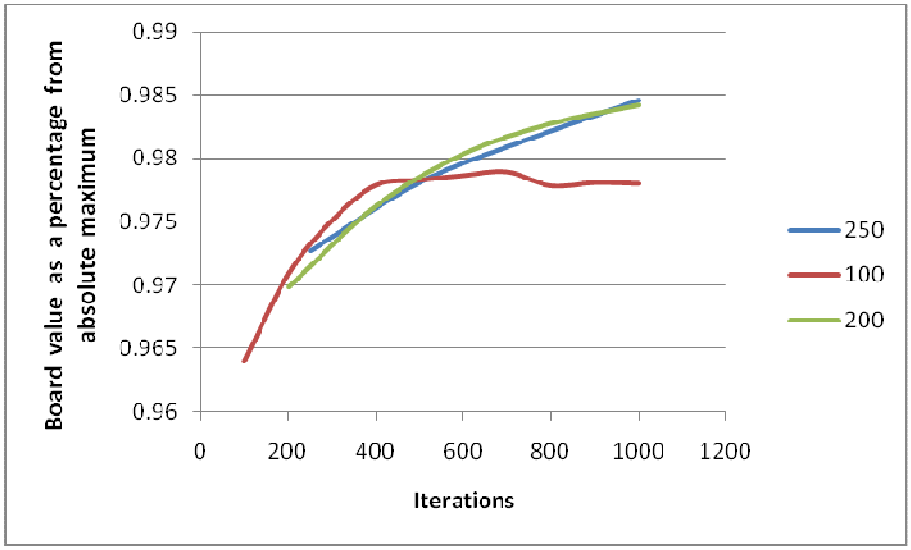


Figure 54: Effect of different population sizes in the Cross-Entropy algorithm

The Cross-Entropy algorithm is the only algorithm that was not programmed by the author. James Bekker from the University of Stellenbosch tested the data used in this study on this algorithm that was programmed by him. For this reason the steps of this algorithm were not described in detail.

The only constant for the Cross-Entropy algorithm that had to be determined was the population size. Figure 54 shows the results for the different population sizes tested. The size eventually used for this algorithm was 200. A population size of 100 initially gave a better solution, but after a certain number of iterations it did not increase anymore. A population size of 250 also initially gave better solutions than when a population size of 200 was used. After a certain number of iterations a population size of 250 again gave better solutions, but using 200 gave better solutions for the biggest part as the number of iterations increased.

4.2.6 Alternative algorithm

Through inspection of the data, it was decided to write a new algorithm that was not based on previously developed algorithms. The aim of this algorithm was to search more intelligently by incorporating the knowledge learned after analysing the data in section 4.1. Ten algorithms were written that all work in the same manner, but that differ in the number of simulation combinations they considered. All the alternative algorithms started with a non-random solution. They all started with all three variables at zero.

Hypothesis: The board value obtained by this alternative algorithm will be higher than the average board value obtained by the other algorithms included in this study.

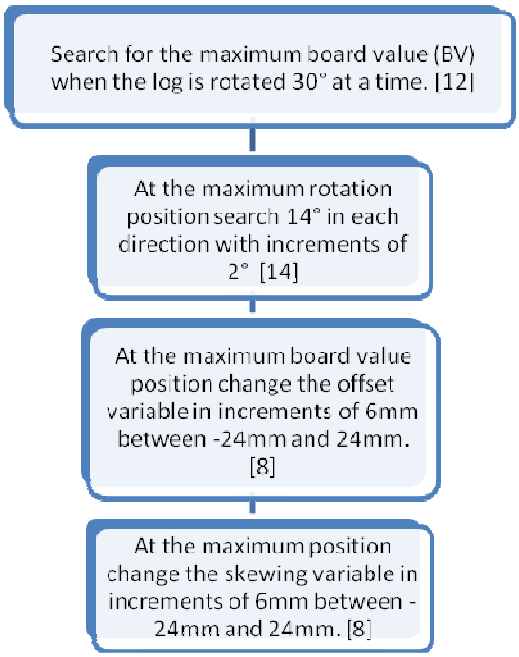
The manner in which the different versions of this algorithm operated are indicated below:

(The number in block brackets indicate the number of iterations it evaluated at each step)

Alternative version 1

Step 1: A search was done for the maximum board value (BV) when the log was rotated 30° at a time and offset and skewing stayed at 0mm. [12]

Step 2: At the maximum rotation position, 14° in each direction was searched with increments of 2° [14]



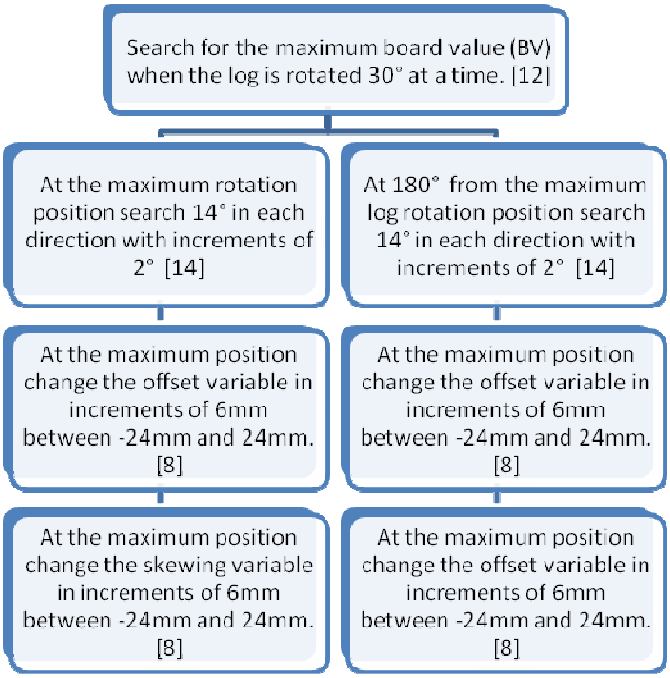
Step 3: At the maximum board value position, the offset variable was changed in increments of 6mm between -24mm and 24mm. [8]

Step 4: At the maximum position, the skewing variable was changed in increments of 6mm between -24mm and 24mm. [8]

Total positions evaluated = 42

Alternative version 2

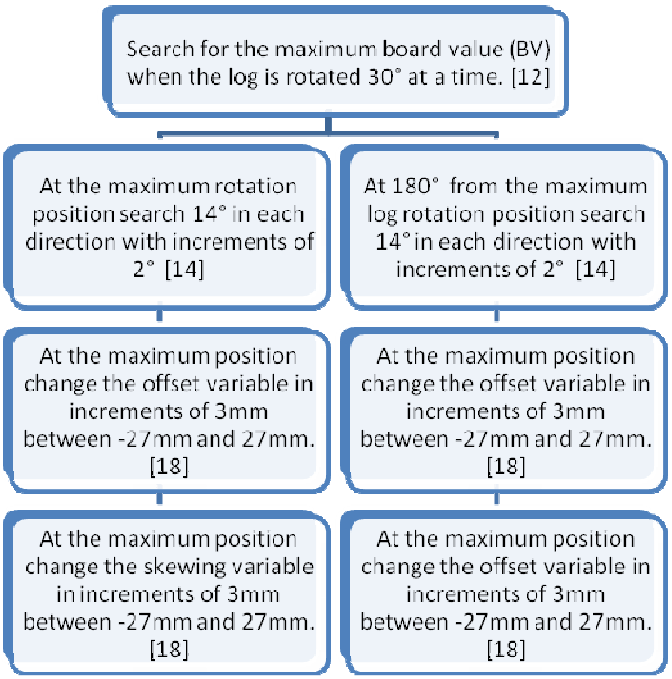
Version 2 did the same as version 1, but it also took an angle 180° from the maximum angle calculated in step 1 to perform steps 2 – 4 on. Total positions evaluated = 72



Alternative version 3

Version 3 did the same as version 2, except at steps 3 and 4. Instead of searching from -24mm to 24mm with increments of 6mm, it searched from -27 mm to 27mm with increments of 3mm.

Total positions evaluated = 112

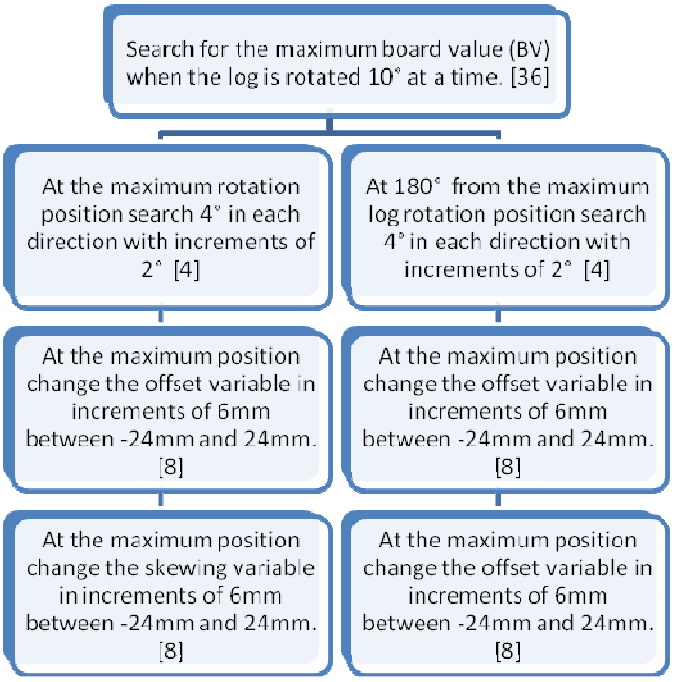


Alternative version 4

Step 1: A search was done for the maximum board value (BV) when the log was rotated 10° at a time. [36]

Did steps 2 – 4 with maximum angle obtained in step 1 as well as with angle 180° from maximum angle.

Step 2: At the maximum rotation position, 4° in each direction was searched with increments of 2° [4]



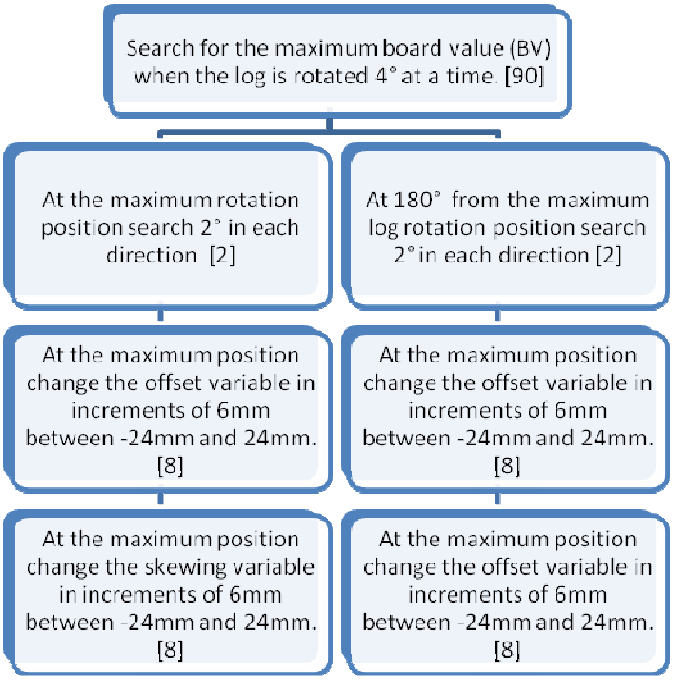
Followed same step 3 and 4 as in version 1.

Total positions evaluated = 76

Alternative version 5

Version 5 did the same as version 4, except for step one, where the log was rotated 4° at a time, and at step 2 only searched 2° to each direction.

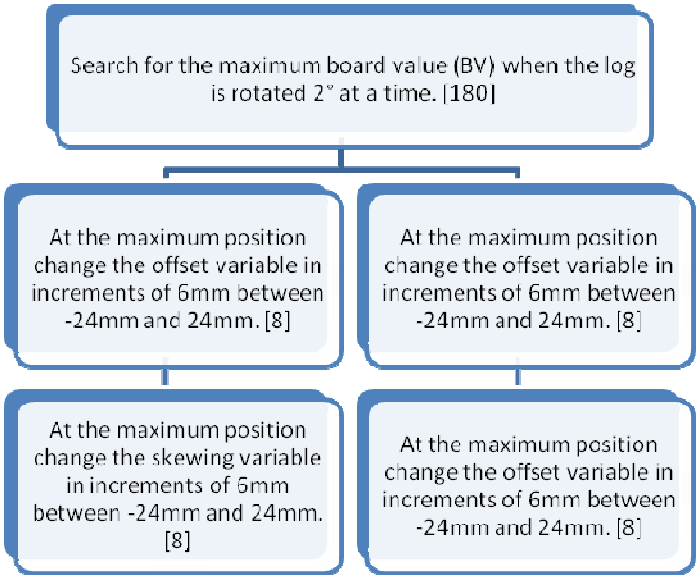
Total positions evaluated = 126



Alternative version 6

Version 6 did the same as version 5, except in step 1 the rotation increment was 2°.

Total positions evaluated = 212

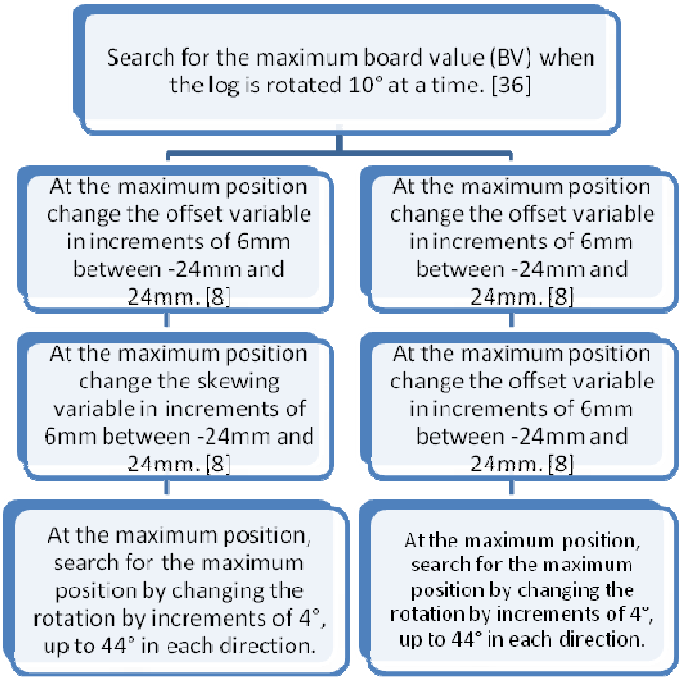


Alternative version 7

Step 1: A search was done for the maximum board value (BV) when the log was rotated 10° at a time. [36]

Step 2: At the maximum position, the offset variable was changed in increments of 3mm between -27mm and 27mm. [9]

Step 3: At the maximum position, the skewing variable was changed in increments of 3mm between -27mm and 27mm. [9]



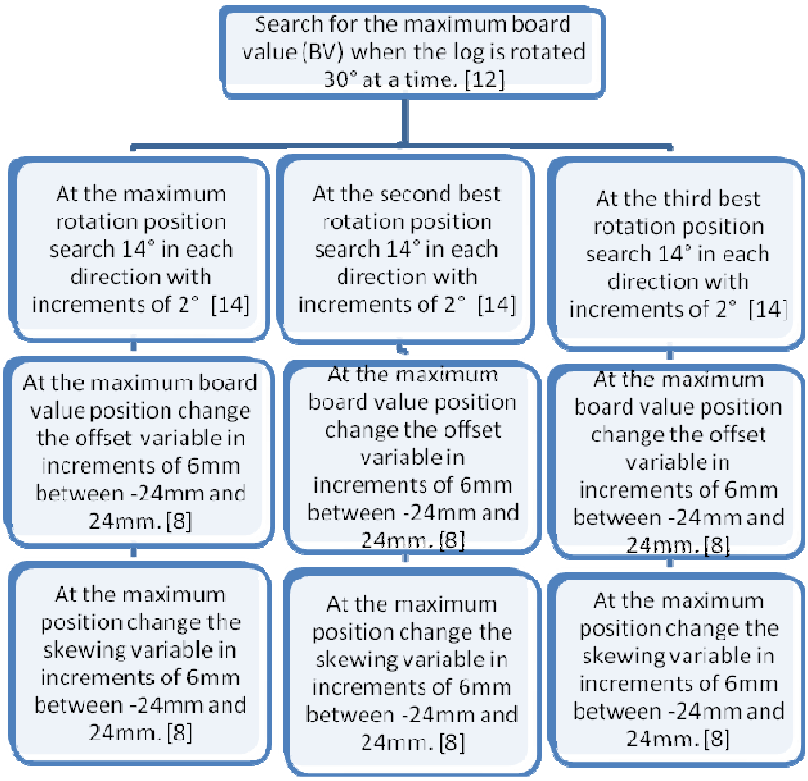
Step 4: At the maximum position in step 3, the maximum position was searched for by changing the rotation by increments of 4°, up to 44° in each direction.

Total positions evaluated = 94

Alternative version 8

Did the same as in version 1, but did steps 2 – 4 with the second and the third best positions obtained in step 1.

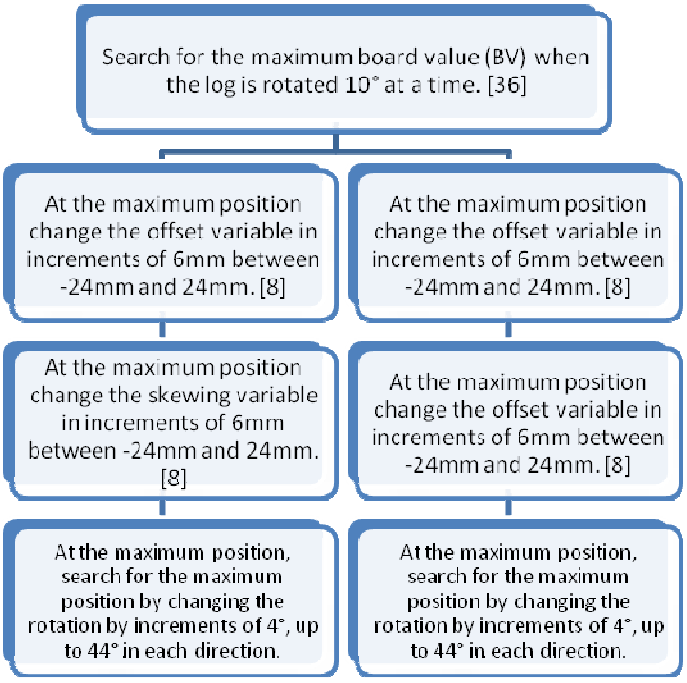
Total positions evaluated = 68



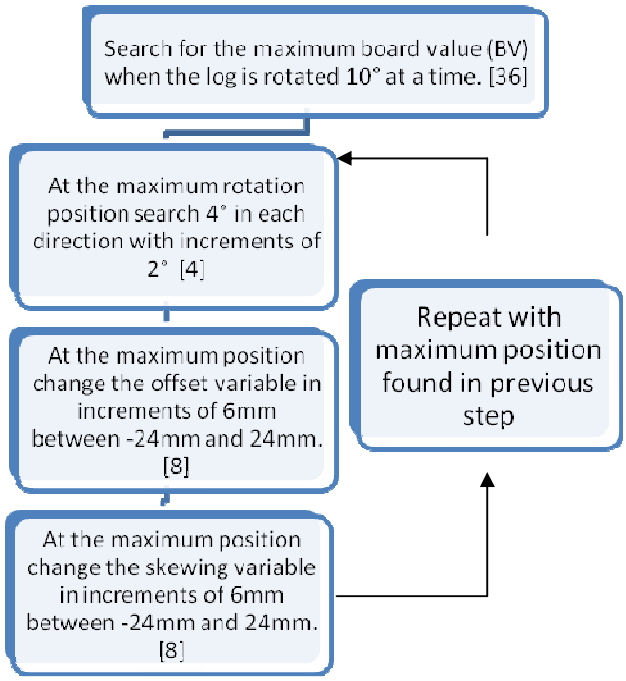
Alternative version 9

Version 9 did the same as version 7, but instead of doing steps 2 – 4 with the position 180° from the best position, it rather took the second best position to carry out steps 2 – 4.

Total positions evaluated = 94



Alternative version 10



Step 1: The maximum board value (BV) was searched for when the log was rotated 10° at a time. [36]

Step 2: At the maximum position, the offset variable was changed in increments of 3mm between -27mm and 27mm. [9]

Step 3: At the maximum position, the skewing variable was changed in increments of 3mm between -27mm and 27mm. [9]

Step 4: At the maximum position found in step 3, the rotation variable was changed with increments of 4°, up to 44° in each direction, the maximum board value found was used as the result for the search.

In versions 2 – 9 the optimizing steps were carried out 180° from the optimal position, except for 8, because as can be seen in Figure 27, the next position with the best probability where the optimal may lie, was 180° from the maximum obtained position.

4.2.7 Genetic Algorithm starting with random vs. intuitive initial population

The Genetic algorithm, Cross entropy algorithm and the PBIL algorithm's initial population were chosen randomly. To test if the initial population of an algorithm had an effect on the results of the algorithm, the Genetic algorithm was tested starting with an intuitive population versus starting with a random population. Since the PBIL and Cross entropy algorithms generated random populations throughout the running of the program, it will not make a difference to start the first population with an intuitive solution.

The intuitive starting population must have 20 individual solutions. All these solutions were chosen to have an offset and skewing value of zero, since the optimum board values are usually scattered around the zero offset and skewing positions. Since the population consists of 20 individuals, the rotation variable is distributed across the whole range of

possible rotation positions. The first solution has a rotation value of zero, and thereafter each individual's rotation value is 18° more than the previous.

H_0: An intuitive starting population will result in finding a higher board value than starting with a random solution with the Genetic algorithm.

Figure 55 shows the results for when the intuitive population is used as the initial population, compared to when a random population is used.

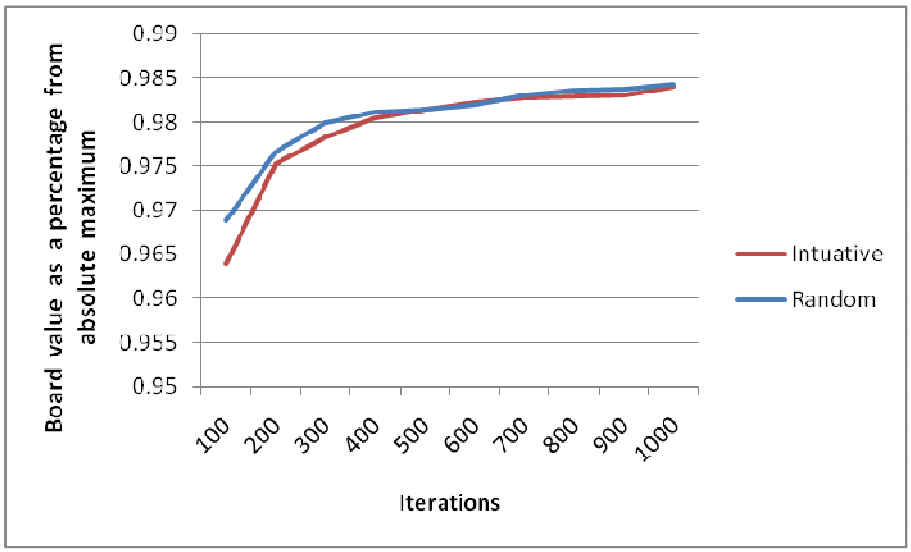


Figure 55: Results for starting the Genetic algorithm with an intuitive population compared to a random population

A data analysis was conducted on the data from the intuitive initial population compared to the random initial population. "The P-value is the probability that the test statistic will take on a value that is at least as extreme as the observed value of the statistic when the null hypothesis is true." (Montgomery, 2007) The P-value was determined for each log (Table 10), and the average for all logs was calculated.

Average from all 10 logs:

P-Value = 0,0000700526 < 0,05

Table 10: The P-values from data for starting the Genetic algorithm with intuitive population and starting with a random population

	Log1	Log2	Log3	Log4	Log5	Log6	Log7	Log8	Log9	Log10
P-value	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0005

The average P-value, as well as all the P-values for all the logs, is less than 0,05, which means H_0 is rejected. This indicates that it does not make a difference if the algorithm is started with an intuitive population or with a random population.

5 Results and discussion

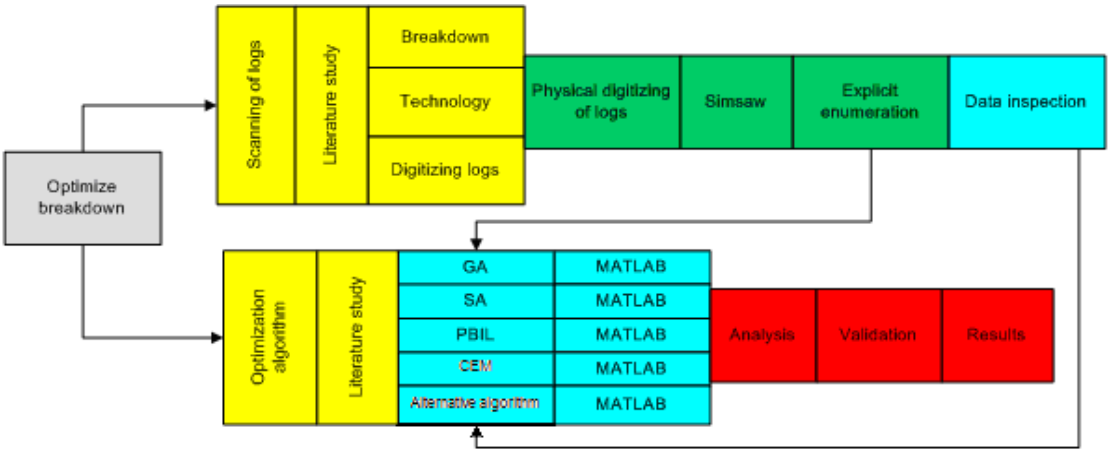


Figure 56: Thesis roadmap

5.1 Sawing with aid of scanning vs. current sawing technique

Currently there are only a few automatic log positioning systems at the headrig at sawmills in South Africa. The positioning of the log is usually done by an operator who uses his own judgement on how to position the log. The main factor influencing the decision the operator makes in positioning the log is the sweep of the log. If the log has no sweep, the operator simply tries to align the log with the middle of the saw. If the log has sweep, he tries to rotate the log so that it enters the saw in the horns-up or horns down position. The few automated log positioning systems in SA uses the same rules as described above. The logs used in this study did not have significant amounts of sweep. Because these logs had no sweep, the operator would have just positioned the logs in the middle of the saw, the rotation position would have been random. Thus the 0° log rotation position was taken as the conventional position in which we assume the operator would have positioned the log, because the 0° rotation position was also taken at random when the log's profile was measured. The offset and skewing variables was also taken at the 0mm position. The Rand (South African currency) value of sawing the log in the conventional log position compared to the optimal position after scanning and virtually sawing the log are given in Table 11 below.

Table 11: Comparing board value by comparing conventional log positioning to internally scanned log optimal position.

Log	1	2	3	4	5	6	7	8	9	10
Optimal position	392.2	480.4	453.8	421.3	441.6	241.6	264.9	328.9	298.1	549.9
Conventional	365.3	438	440.8	403.7	424	211.2	226.7	299.8	256.3	518.6
Percentage of improvement	6.8%	8.8%	2.9%	4.2%	4%	12.6%	14.4%	8.9%	14%	5.7%

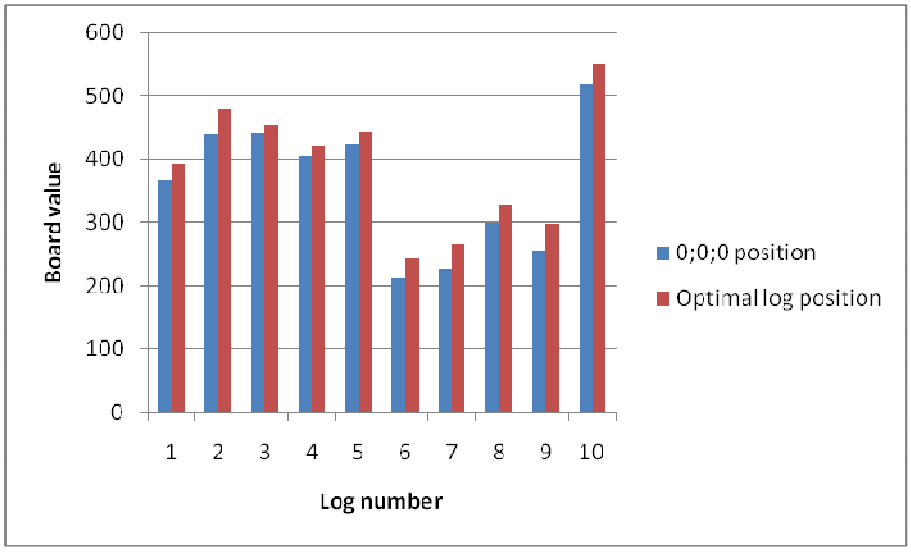


Figure 57: Board value for 0;0;0 (rotation;offset;skewing) position and global optimum

When the values for the optimal log position are compared to the conventional log position in Table 11, there is a substantial increase in log value recovery when a log is positioned in the optimal position in front of the headrig. The average increase in value is 8.23%.

5.2 Value yield vs. volume recovery

To determine the monetary value of the boards from a log, it is necessary to take the internal log information into account. Simsaw does these calculations by grading the boards into different board grades. Different board grades are worth different amounts.

If volume recovery from a log is maximized, it does not necessarily mean the optimum monetary value will be earned from the log. The grade of the boards that is sawn has to be

considered. Table 12 shows the Rand value from when a log was sawn in the optimal position for volume recovery compared to the optimal position for board value.

Table 12: Optimizing for log volume recovery compared to optimizing for board value

Log	1	2	3	4	5	6	7	8	9	10
Optimal board value position	392.2	480.4	453.8	421.3	441.6	241.6	264.9	328.9	298.1	549.9
Optimal volume recovery position	386.2	460.4	453.8	409.8	441.6	226.3	246	324.7	279.8	532.1
Increase from optimizing for value instead of volume	1.5%	4.2%	0%	2.7%	0%	6.4%	7.1%	1.3%	6.2%	3.2%
Increase from optimizing for volume recovery compared to conventional log position	5.4%	4.8%	2.9%	1.5%	4%	6.7%	7.8%	7.7%	8.4%	2.5%

The average increase from optimizing for value instead of increasing for log volume recovery is 3.26%. Todoroki (2001) also found an increase of 3% when finding the optimum log position in front of the headrig while considering internal log defects (optimizing value yield) compared to only considering external log information (optimizing volume recovery).

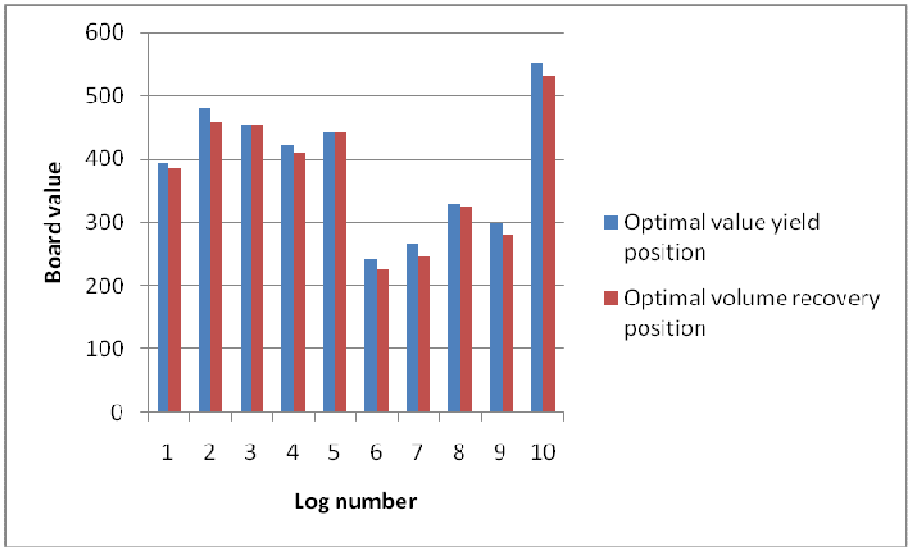


Figure 58: Comparing optimizing for value yield with optimizing for volume recovery

5.3 Comparing different optimization algorithms

The different optimizing algorithms to find the optimum log position in front of the headrig were compared in terms of the maximum value found, number of iterations needed to find a good solution, and the amount of deviation when optimum values were searched for.

There is a trade-off in terms of number of iterations. There is a direct relationship between number of iterations and calculation time. Since a limited time is available for optimization decisions it is important to consider after how many iterations the algorithm should be stopped. One iteration means that one log position from the explicit enumeration data is considered.

The results for the different algorithms after different number of iterations were evaluated is shown in Figure 59. Each algorithm was run 100 times for a certain number of iterations (e.g. 100, 200, 300, etc.). To draw the graphs, the average for the 100 runs of the specific algorithm and the specific number of iterations was taken to plot one data point. The x-axis on the graph is the number of iterations considered and the y-axis is the average board value from the ten logs when the board value is indicated as a percentage of the maximum possible board value.

The algorithms were only tested up to 1000 iterations. When, for instance, the Cross-Entropy algorithm is considered, it can be observed that its graph has not levelled off yet,

which means it could most probably achieve better results were more iterations considered. The reason why the algorithms were only tested up to 1000 iterations, is that it is not practically implementable to evaluate more than 1000 iterations. In current circumstances, 1000 iterations take on average 42 minutes to complete.

A random search was done with the data created with the explicit enumeration to compare with the other algorithms. By comparing the results of an algorithm with the random search, one can determine whether a specific algorithm searches more intelligently than a random search. With the random search, when for instance 100 iterations are allowed, 100 random log positions were selected and the maximum board value found was used as the outcome of the search. Just as with the other algorithms, this was done 100 times and the average was determined for each data point on the graph.

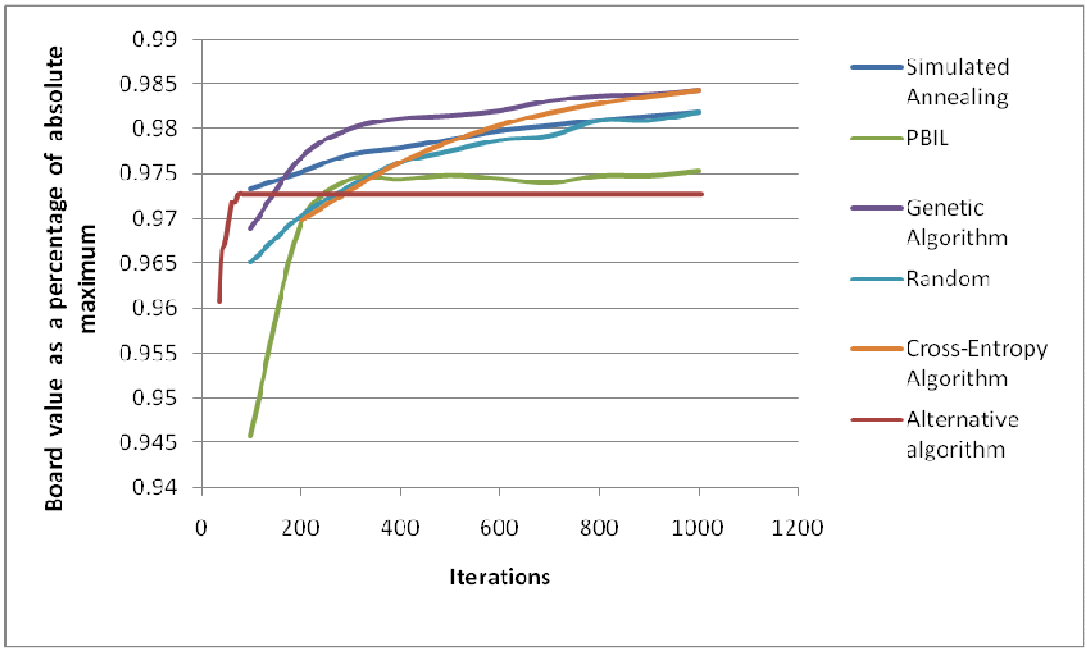


Figure 59: Results of all algorithms for different amount of iterations evaluated (average of all 10 logs)

The only two algorithms that outperformed the Genetic Algorithm at a specific point were the Simulated Annealing algorithm and the Alternative algorithm, which was only better up to about 150 iterations. From there onwards the Genetic algorithm outperformed all the algorithms. The Cross-Entropy algorithm gave equally good results as the Genetic algorithm after 1000 iterations. These were the only two algorithms that substantially outperformed the random search when more than 800 iterations were considered.

With the Genetic algorithm the board value showed a steep increase for the first 200 iterations. The increase continues, but at a lesser rate after 200 iterations. In this case, it seems as if little benefit will be realised by executing more than 400 iterations.

The PBIL algorithm showed a very steep increase in the board value until 300 iterations were reached, where after the solution did not increase, no matter how many iterations were evaluated. It can also be noted that only when 300 iterations were evaluated did it outperform the random search. This implies that, unless the algorithm is modified, it is definitely not a suitable algorithm for this application.

The Cross-Entropy algorithm gave more or less the same results as the random search up to 400 iterations, where after it gave slightly better results when more iterations were considered. It seems as if this algorithm may still show an increase if more than 1000 iterations are considered.

The Simulated Annealing algorithm showed the best results compared to the other algorithms up to 150 iterations, after which the Genetic algorithm performed better. After 800 iterations the random search gave better results than the Simulated Annealing algorithm.

Up to about 100 iterations the Alternative algorithm rendered the best results, but not if more iterations were evaluated. After less than 300 iterations the random search outperformed the alternative algorithm and after almost 350 iterations it performed the worst from all the algorithms.

The standard deviation of a sample is the square root of the variance. “The variance is a measure of the dispersion, or the variability in the distribution.” (Montgomery and Runger, 2007) Thus the standard deviation was calculated to measure how constant the outcome of each algorithm was when it was run a few times (in this case 100) under exactly the same circumstances and the same number of iterations. This tested whether the algorithm performed consistently. If an algorithm performed with a lot of variation it would not have been possible to detect this simply by looking at the average outcome from running the algorithm a certain number of times. The results for the standard deviation from all the algorithms are shown in Figure 60.

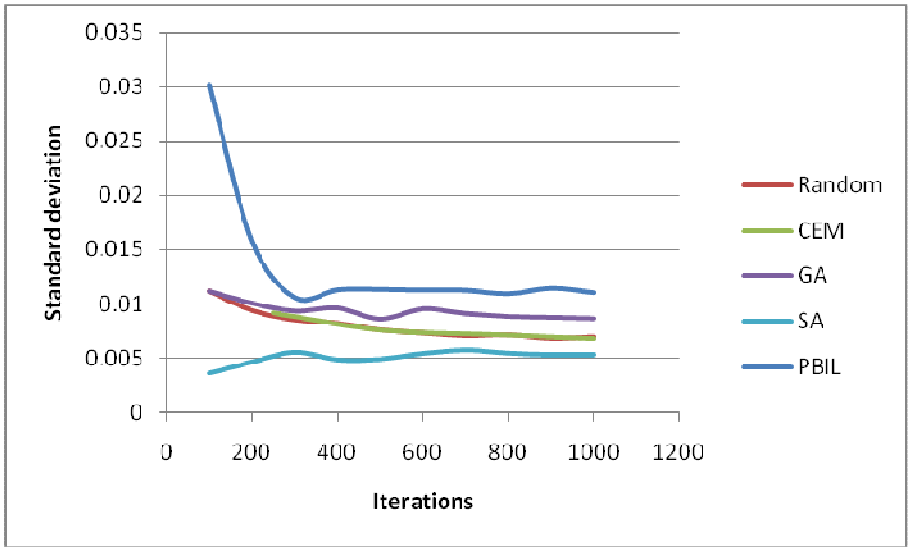


Figure 60: Standard deviation from all algorithms

The Simulated Annealing algorithm had the lowest standard deviation across any number of iterations. The main reason for this may be because it started at the same log position every time it was tested. The only algorithm that has a fairly high standard deviation is the PBIL algorithm, but only up to 300 iterations. As mentioned above, the results from the PBIL algorithm also did not improve after 300 iterations. All the other algorithms did not have a high standard deviation at any number of iterations. The standard deviation did not differ that much between the different algorithms that it will have an effect on the choice of the best algorithm.

The results for all logs for the different algorithms are shown in Figure 62 to Figure 67. The algorithms were discussed in detail in the previous section. Only aspects that stand out when all the logs' data are considered will be mentioned in the next section.

An interesting observation is that log 6 seems to have the lowest performance for all the algorithms. An investigation of the data showed that this was because log 6's global optimum board value was much higher than the second best solution. For instance, in the other logs, if all the solutions of the log were sorted in descending order, the 50 best solutions did not differ more than 1% from the global maximum. With log 6, however, the 50 best solutions differ up to 4% from the global maximum. This reduced the chances of finding a solution very near to the global optimum. This is illustrated in Figure 61. Log 6 had only one position closer than 2% away from the global optimum, which made the possibility to find a near optimum very low. The opposite counted for log 4, which had many more solutions near the global optimum. This was also why log 4 showed the best results for all the algorithms.

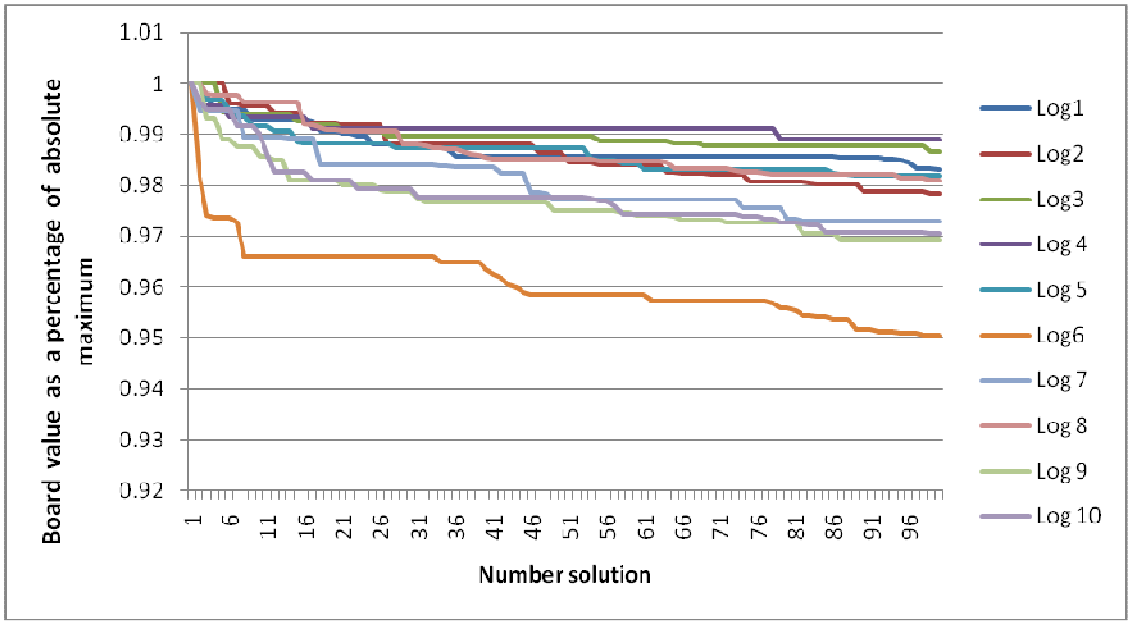


Figure 61: Best 100 solutions when all possible solutions from the explicit enumeration data are arranged in descending order.

5.3.1 Genetic algorithm

Figure 62 shows how the Genetic Algorithm performed for each log. This type of analysis was done to evaluate the different algorithms in terms of how constant each performed for the different logs. This algorithm shows the most deviation for the logs as more iterations were considered. Some of the logs showed worse results when more iterations were considered. When the average outcome of all logs was considered, the optimum amount of iterations to consider can be calculated. But it is not favourable to have a decrease in board value for some logs when more iterations were considered, as was the case with this algorithm.

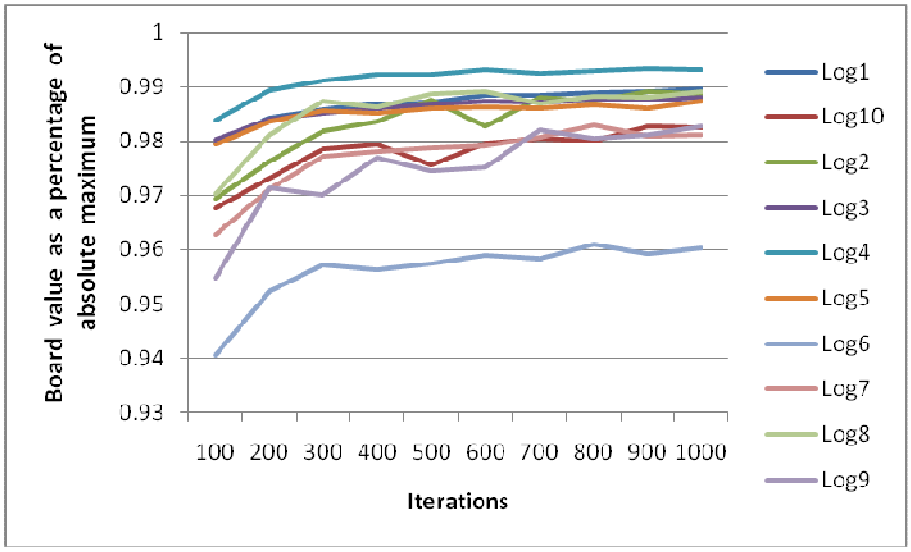


Figure 62: Board values obtained (Genetic Algorithm)

5.3.2 PBIL

Figure 63 shows that the PBIL algorithm seemed to perform almost the same on all logs. Practically all logs followed the same trend as the algorithm evaluated more iterations. As mentioned previously, the outcome of the algorithms stabilised after 300 iterations. Figure 63 shows that this was the case with all logs, indicating that there will in no case be a benefit to evaluate more than 300 iterations.

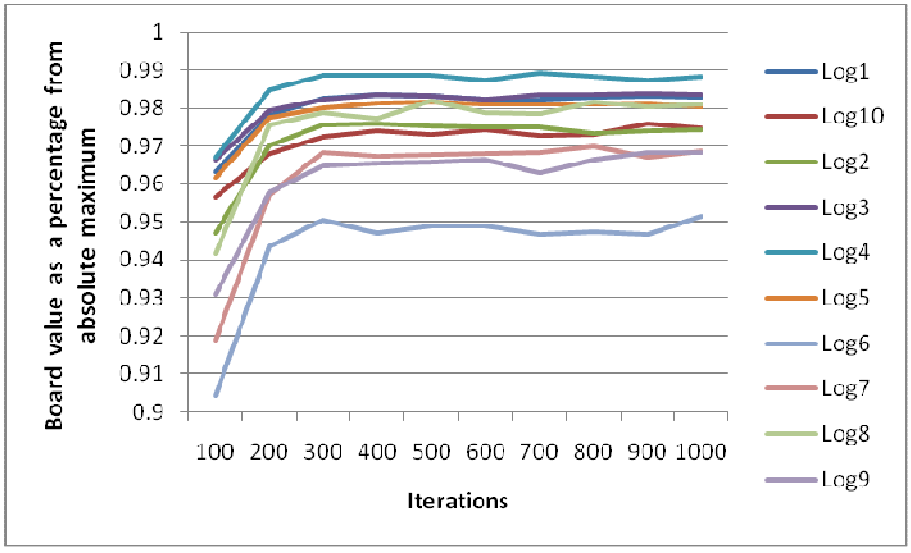


Figure 63: Board values obtained (PBIL)

5.3.3 Simulated Annealing

As seen in Figure 64, certain logs had a steeper increase than other logs. Log 4 had no increase no matter how many iterations were evaluated. One explanation for the difference in slope of the different graphs was that logs with a flatter slope started with a better solution in the first instance. Since the algorithm started with the position determined by the alternative algorithm's version 1, this can also act as a test for how good the solution was that the alternative algorithm produced.

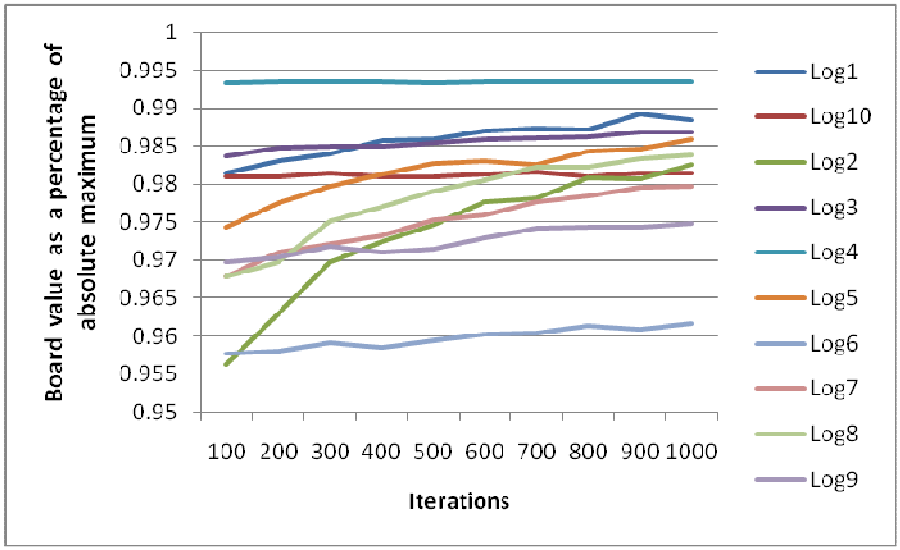


Figure 64: Board values obtained (Simulated Annealing)

5.3.4 Cross-Entropy Method

As shown in Figure 65, except for Log 6 that had a low outcome, there was no log that showed any significant deviation from the pattern that was followed by this algorithm. This was a reliable algorithm, in the sense that all logs continued to increase in value as the number of iterations increased.

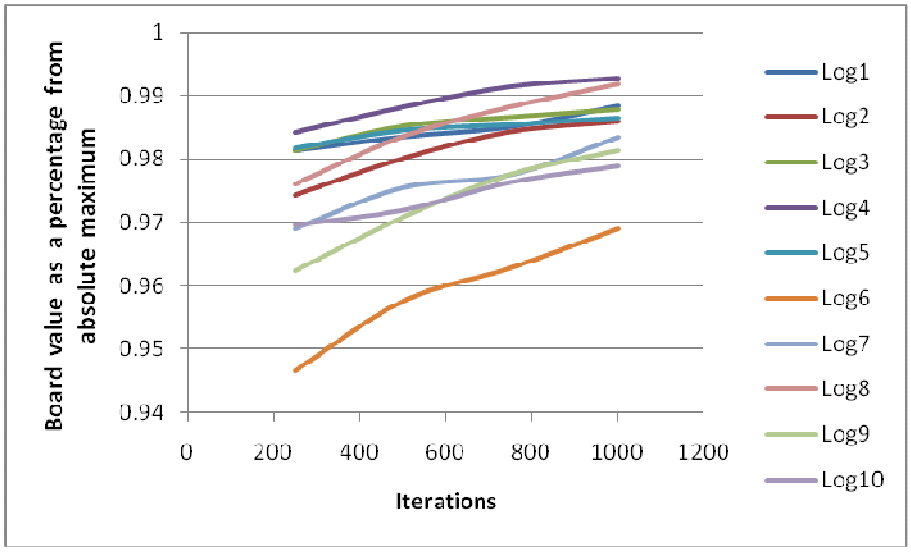


Figure 65: Board values obtained (Cross-Entropy Algorithm)

5.3.5 Alternative algorithm

Figure 66 indicates that there was no increase in any of the logs after 74 iterations. 74 iterations will take about three minutes using the current software and setup. Most of the logs had a very steep increase in value at the beginning, but then rapidly stopped increasing. This indicates that it was a very good algorithm when only a few iterations can be evaluated.

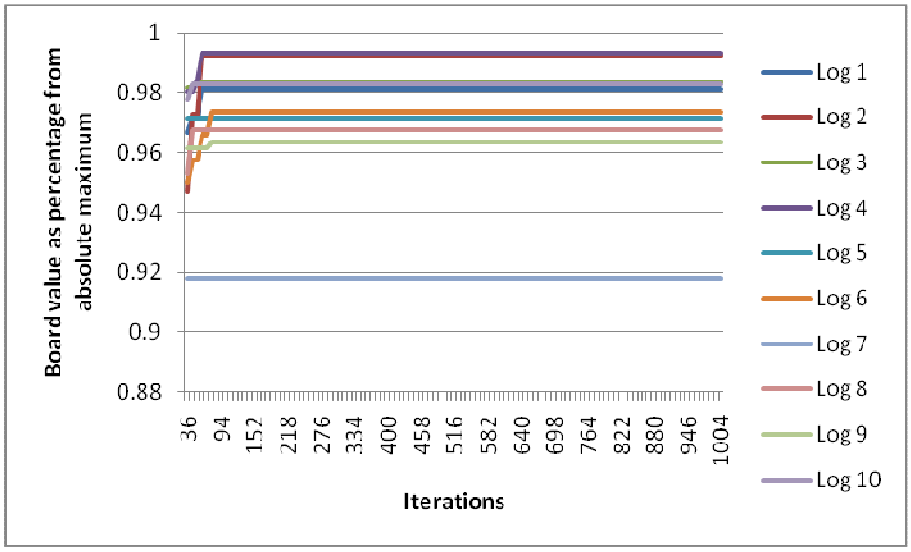


Figure 66: Board values obtained (Alternative algorithm)

5.3.6 Random

If enough simulations were done for each data point, all the lines in Figure 67 would follow the same trend and no lines would go down progressively as more iterations were evaluated. The reason why only 100 simulations were done to obtain each data point, is that only 100 points were taken for all other algorithms due to the time required to run all the algorithms.

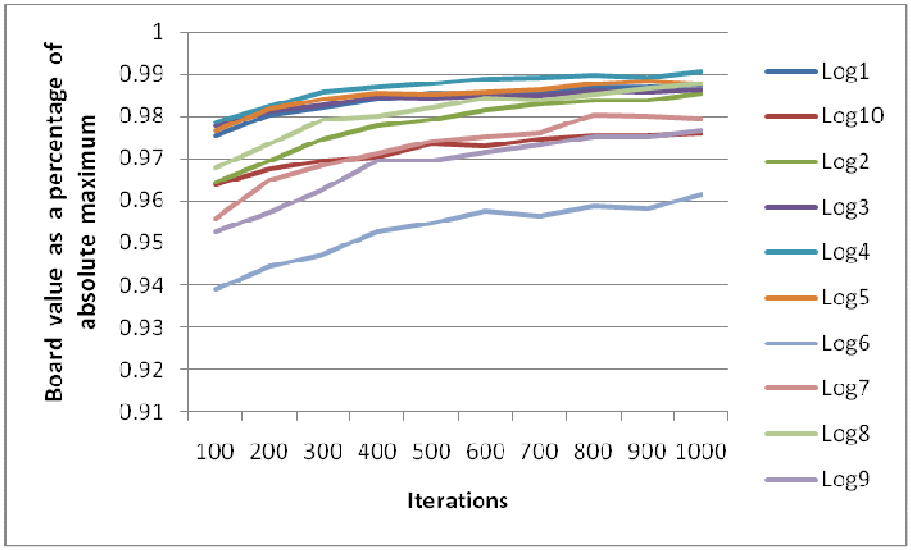


Figure 67: Board values obtained (Random search)

5.4 Conventional log position vs. optimal log position found by optimizing algorithms

The comparison between conventional log positioning and positioning the log in the optimal position found by the optimization algorithms, can be regarded as the actual improvement to be achieved in practice when an internal log scanning and positioning system is installed.

The number of iterations that can be evaluated in practice depends on many factors. The bottleneck in the scanning, optimizing and positioning system should be the sawing process at the headrig. The system should be set up and installed in such a way that it does not affect the throughput efficiency of the system. The scanners currently being developed to scan logs internally should be able to scan logs at a speed of 100m/min, which is much faster than the sawing speed of any headrig. (pers. comm. Martin Bacher, Microtec, 2010)

When calculations were done on how long it takes to simulate one log sawing position, a normal desktop computer with a 3 GHz processor and 2Gb RAM was used. In practice, a sawmill will install a computer with a much faster processor and more memory.

There are a few features that can be implemented to increase the number of iterations evaluated for each log:

Theoretically and depending on some practical constraints in sawmills, the logs can be scanned, tagged and stored in a log yard. The computer can then run the simulation to determine the optimum position of the log in front of the headrig whenever the time is available. The log position will be stored in the memory of the computer. When the log is then put on the conveyer to be sawn, the log will just be identified and the computer will indicate the way the log should be positioned. In this way, the computer can run 24 hour a day, even though the sawmill only operates eight or sixteen hours per day and also have a certain amount of downtime.

Another aspect that can be added to increase the number of iterations evaluated per log is to install more computers that run in parallel to compute the optimal position of the logs. If ten computers are installed, it will increase the number of iterations evaluated tenfold.

The following calculations were done to determine the number of iterations that can be evaluated for each log:

$$IT = \frac{TT}{ST} = \frac{86400}{2}.539 = 34029$$

$$LS = \frac{VO}{LV} = \frac{600}{0}.257 = 2334$$

$$IL = \frac{IT}{LS} = \frac{34029}{2334} = 14.58$$

Where

IT = Total iterations that can be evaluated per day

TT = Total time available per day (seconds)

ST = Average simulation time per log position (seconds)

LS = Logs sawn per day

VO = Volume input of logs per day for a large sawmill in South Africa (m^3)

LV = Average volume of a log (m^3)

IL = Iterations available per log

If the above mentioned aspects are implemented to increase the number of iterations that can be evaluated per log, a fairly accurate number of iterations that can be used for calculations is 200. The number of iterations available per log was calculated as 14, but when a larger number of computers and more powerful computers are used, 200 iterations is considered a fair estimate. In Table 13 the calculations are also done with 100 and 300 iterations to compare the expected number of iterations with a best and worst case scenario.

Table 13 indicates the Rand value for positioning each log in the conventional position as well as the average board value when an optimizing algorithm was run 100 times for each log and each number of iterations. The percentage increase from the conventional position to the position determined by the algorithm is also given. For the optimized log position with 100 iterations, the Simulated Annealing algorithm was used, since, as seen in Figure 59, at 100 iterations Simulated Annealing gave the best log position. For 200 and 300 iterations the Genetic algorithm was used because, when 200 and 300 iterations were evaluated, the Genetic algorithm gave the best results.

Table 13: Conventional log scanning compared based on board value with optimized log position by optimization algorithms for different amount of iterations evaluated

	Log1	Log2	Log3	Log4	Log5	Log6	Log7	Log8	Log9	Log10	Average
Conventional log position	365.3	438.0	440.8	403.7	424	211.2	226.7	299.8	256.3	518.6	
Optimizing algorithm position, 100 iterations	385	459.4	446.4	418.5	430.2	231.4	256.3	318.3	289.1	539.4	
% increase from 100 iterations	5.39	4.88	1.28	3.68	1.48	9.56	13.05	6.19	12.8	4.02	6.23%
Optimizing algorithm position, 200 iterations	387.3	472.5	446.9	415.7	435.6	229.1	255.5	322.9	285.9	533.5	
% increase from 200 iterations	6.03	7.87	1.38	2.97	2.75	8.45	12.67	7.72	11.56	2.89	6.43%
Optimizing algorithm position, 300 iterations	387	466.3	447.7	414.8	437.4	238.7	251.7	319.6	293.9	530.9	
% increase from 300 iterations	6.21	6.47	1.57	2.75	3.17	12.99	11	6.62	14.67	2.38	6.78%

To do the cost analysis in section **Error! Reference source not found.** for the practical implementation of an optimizing system, the average increase with the optimizing algorithm in Table 13 is used when 200 iterations are evaluated, which is 6,43%.

5.5 Cost analysis

The purpose of this project was to evaluate the optimization of the log breakdown process at the primary workstation by firstly considering both the shape and internal knots of the logs and secondly developing an optimization algorithm to assist the operator in deciding on the best log position. To accomplish this, one of the objectives that were set was to evaluate the economic feasibility of log breakdown through internal scanning.

The following hypothesis was set: a meta-heuristic method will result in an improvement in the economic feasibility from implementing an internal log scanning system.

It was proven that by internally scanning logs and finding a near optimal position in front of the headrig by means of optimization algorithms, an improvement in the value of the board sawn from a log can be made. The question still remains whether it is economically feasible to implement such a system and how long will it take for a company to earn back the initial investment made.

A cost analysis was done to show how an investment decision can be performed when considering a log positioning optimization system based on internal scanning data at a sawmill in South Africa. It was not possible to get accurate input data for some of the variables and this cost analysis should thus be seen as an example based on assumptions for a specific sawmill rather than a general evaluation of the economic feasibility of the technology.

Four different scenarios were considered during the cost analysis:

- Firstly the income was determined per m^3 of sawn timber when no optimization was implemented, in other words as it is currently done in most sawmills in South Africa today. This scenario was labeled *Pessimistic Scenari*o in Table 14. The breakeven time period was worked out accordingly.

- Secondly, a scenario was created when a log was optimized by only considering external log information (labeled *External Scanning* in Table 14). The percentage of value increase used was when the global optimum was found in every situation. When a meta-heuristic optimizing algorithm was used to determine the optimal position, it will be a bit lower, but this calculation was done just to see how it measured up compared to internal scanning of logs.

- A third scenario investigated was when the logs were internally scanned and a meta-heuristic optimization algorithm found a near optimum position. 6,43% was used as the average increase in value from the conventional position. This was calculated in Table 13 when 200 iterations were considered.

- The fourth scenario considered was optimistic. When the global optimum in the explicit enumeration data was used, an average increase of 8,23% can be achieved. For this calculation, 8% was used, to indicate the effect if a solution very near the global optimum can be reached most of the time.

Table 14: Cost analysis for implementing a internal log scanner at a large pine sawmill in South Africa

		Pessimistic Scenario	External Scanning	Practical Scenario	Optimistic Scenario
% improvement on value yield		0%	5.17%	6.43%	8%
Income per m^3	R 2610	R 2,610.00	R 2,744.94	R 2,777.82	R 2,818.80
Additional Profit with scanning tech/m^3		R -	R 134.94	R 167.82	R 208.80
Volume per day (m^3)	300	R -	R 40,481.10	R 50,346.90	R 62,640.00
Cost of scanner per day		R -	R 1,000.00	R 1,000.00	R 1,000.00
Contribution to profit per day		0	R 39,481.10	R 49,346.90	R 61,640.00
Interest Rate	12.5%				
			Days to breakeven		
Scanning Technology (optimistic)	-50 000 000			1244	950
Scanning Technology (neutral)	-75 000 000			2146	1574
Scanning Technology (pessimistic)	-100 000 000			3458	2368

The income for one cubic meter of timber sawn from a log in the conventional log position was determined by taking the average from all the logs used in this study. The value differs slightly from log to log since different grades of timber are worth different amounts. The price per cubic meter for the different scenarios investigated was calculated by adding the percentage increase investigated to the conventional log position's price per cubic meter.

The volume sawn timber produced per day was taken as $300m^3$. This is the amount of timber produced by a typical medium sized pine sawmill in South Africa.

It is difficult to estimate the operating cost of an internal log scanner. A rough estimate was made at R1000 per day for electricity and maintenance.

The interest rate for the loan made to purchase an internal log scanner was taken at 12,5%. There are many factors influencing the interest rate of the loan, for instance the credit history of the company and the default risk for the bank. The interest rate can differ between the repo rate, which is currently 9%, and repo plus 4%. In order to be conservative, the rate was taken as repo plus 3,5%.

The price of an internal log scanner is very difficult to estimate. Companies developing such systems do not want to give out such information. A laboratory version CT scanner that can scan short log sections was recently purchased at Stellenbosch University at a cost of R12.5 million rand. The price of commercial versions is thus estimated at R50 million rand.

The scenario where only external log information was considered was only compared in terms of the additional profit it can provide per day. This was because the initial investment amount required for such a piece of equipment was not known.

As mentioned earlier it is very difficult to estimate the real cost to implement such a system, since the first internal log scanner in a production environment will only be installed in 2011. Three different values were used as the initial investment made to install an internal log scanner. The number of working days to break even was calculated for each estimated initial investment value.

If the “practical scenario” is considered, and an initial investment of R100 million is considered, the breakeven point is 3458 working days. The average working days in a production environment are 250. This means the company will only start making a profit from their investment after almost 14 years.

6 Conclusion

When a log is positioned in front of the primary breakdown station, there is a characteristic or a combination of different characteristics of the log that makes that a near optimal log position is 180º from the actual optimal. The optimal log postion in term of skewing and offset, are most of the times positioned around the zero millimeter positions, but not necesseraly on the zero positions.

After optimization algorithms were developed to find a near optimal log position within reasonable time, it was found that from all algorithms tested, the Genetic algorithm is best suitable for this application.

During the analysis on the difference of optimizing the log volume recovery and value recovery, it was found that 5.17% more value can be earned when value is optimized. Since it was not possible to simulate all possible log positions to find the global optimum log position, optimization algorithms were developed. These algorithms find a near optimal log position. If volume recovery is optimized an external log scanner must be installed, and when value is optimized, an internal log scanner needs to be installed. The average increase found by these algorithms, considering a sensible number of iterations, was 6.43%. This shows that a significant increase in value can be achieved by installing an internal log scanner.

When volume recovery is optimized it still does not mean that the global optimum will be achieved every time. An optimization algorithm will also be required to find the optimum log position. When volume recovery is optimized and when optimization algorithms are used, optimistically speaking a value increase of 5% will be achieved. This makes the value increase for optimizing for value instead of volume and using optimization algorithms a mere 1.43%. For this reason it is at this stage not worthwhile to invest in an internal log scanner.

Because of statements made in the two above paragraphs, both hypothesis in section 1.2 can be confirmed as true.

6.1 Future research

The current study only used ten logs to test all optimization algorithms. Even though good trends could be seen when the results from the different logs after running the optimization algorithms were compared, it would be more accurate and feasible when more logs are

used. This study would also have been more accurate if three dimensional data from CT scanned logs have been used.

One section of this study that can be done more thoroughly in future is the analysis of the economic feasibility of implementing an internal log scanner and log positioning system. This whole section is based on assumptions. When such scanning devises are actually implemented in 2011, more information on the initial and running cost of such equipment would come available and more accurate calculations can be done.

References

Bacher, M. Personal communication, June, 2010

Barbour, R.J., Parry, D.L., Punches, J., Forsman, J. & Ross, R. 2003, "AUTOSAW simulations of lumber recovery for small-diameter Douglas-fir and ponderosa pine from southwestern Oregon.", *Research note PNW-RN-543, US Department of Agriculture*.

Bekker, J. & Groves, G. 2001, *Introduction to optimisation with genetic algorithms*. Report

Bekker, J. & Olivier, Y. 2008, "Using the Population-Based Incremental Learning algorithm with computer simulation: Some applications", *The South African Journal of Industrial Engineering,* vol. May.

Bekker, J. Personal communication, October, 2010

Crickmay and Associates. 2010, "Supply and demand study of softwood sawlog and sawn timber in South Africa". Report, www.crickmay.co.za.

De Boer, P.T., Kroese, D.P., Mannor, S. & Rubinstein, R.Y. 2005, "A tutorial on the cross-entropy method", *Annals of Operations Research,* vol. 134, no. 1, pp. 19-67.

Diwekar, U.M. 2008, *Introduction to applied optimization,* Springer Verlag, Heidelberg.

Du Plessis, J de V. 2010. Investigation into the use of meta-heuristics in the optimisation of log positioning during sawmill processing, MSc thesis, Stellenbosch University, South Africa.

Forestry Economics Services 2009, "Report on commercial timber resources and primary roundwood processing in South Africa". Report, www.forestry.co.za.

Giudiceandrea, F. 2010, *Advances in X-ray scanning and CT-scanning for logs*, Presentation edn.

Godsmark, R. 2006, "The South African Forestry and Forest products Industry 2005", [2010, September].

Houck, C.R., Joines, J. & Kay, M.G. 1995, "A genetic algorithm for function optimization: A MATLAB implementation", *NCSU-IE TR,* vol. 95, no. 09.

Luke, S. & Edition, Z. 2009, "Essentials of Metaheuristics", *Disponible en http://cs.gmu.edu/~ sean/book/metaheuristics,* .

Ma, J., Li, K. & Zhang, L. 2010, "The adaptive Parallel Simulated Annealing algorithm based on TBB", *Advanced Computer Control (ICACC), 2010 2nd International Conference on*IEEE, , pp. 611.

Microtec. 2011. http://www.microtec.eu/ProductView.aspx?Lang=en-ZA&Nid=10260,10328,10437 [2011, January]

Moins, S. 2002, "Implementation of a Simulated Annealing algorithm for MATLAB", *Linköping Institute of Technology,* .

Montgomery, D.C. & Runger, G.C. 2007, *APPLIED STATISTICS AND PROBABILITY FOR ENGINEERS, With CD,* Wiley-India.

Mouton, J. 2008, "How to succeed in your Master's and Doctoral studies", Van Schaik Publishers, Pretoria, South Africa

NeuroDimensions 2002, , *http://www.nd.com/products/genetic.htm* [2010, August] .

Obitko, M. 1998, , *http://www.obitko.com/tutorials/genetic-algorithms/index.php* [2010, August] .

Occeña, L., Schmoldt, D. & Thawornwong, S. 1997, "Using Internal Defect Information for Hardwood Log Breakdown", *Proceedings of the 7 th International Conference on Scanning Technology & Process Optimization for the Wood Products Industry*, pp. 63.

Perelman, L. & Ostfeld, A. 2007, "An adaptive heuristic cross-entropy algorithm for optimal design of water distribution systems", *Engineering Optimization,* vol. 39, no. 4, pp. 413-428.

Pinto, I., Pereira, H. & Usenius, A. 2002, "Sawing simulation of Pinus Pinaster", .

Rardin, R.L. 1997, *Optimization in operations research,* Prentice Hall Upper Saddle River, NJ.

Rinnhofer, A., Petutschnigg, A. & Andreu, J.P. 2003, "Internal log scanning for optimizing breakdown", *Computers and Electronics in Agriculture,* vol. 41, no. 1-3, pp. 7-21.

Schmoldt, D.L., Scheinman, E., Rinnhofer, A. & Occena, L.G. 2000, "Internal log scanning: Research to reality", *Proceedings of the Twenty-eighth Annual Hardwood Symposium: West Virginia Now–The Future For The Hardwood Industry. Memphis, TN*, pp. 103.

Smith, R.G. 2003, "A method enabling the reconstruction of internal features of logs from sawn lumber: The log end template", *Forest Products Journal,* vol. 53, no. 11, pp. 95-98.

Todoroki, C.L. 2001, "Volume and value variation with opening face position: an investigation with pruned softwood logs", *Forest Products Journal,* vol. 51, no. 1, pp. 36-42.

Todoroki, C.L. 1999, "Combined primary and secondary log breakdown optimisation", *The Journal of the Operational Research Society,* vol. 50, no. 3, pp. 219-229.

Turton, R., Bailie, R.C., Whiting, W. & Shaeiwitz, J. 2003, "Analysis, synthesis and design of chemical processes", *Upper Saddle River, New Jersey,* .

Wessels, C.B. 2009,Cant sawing log positioning optimization: A simulation study. Forest Prod. J. Vol. 59, No. 4..

Wessels, C.B., Price, C.S., Turner, P. & Dell, M.P. 2006, "INTEGRATING HARVESTING AND SAWMILL OPERATIONS USING AN OPTIMIZED SAWMILL PRODUCTION PLANNING SYSTEM", *Proceedings International Preceion Forestry Symposium, Stellenbosch, South Africa, IUFRO, ISBN 0-7972-1121-7, pp117-118.*

Winston, W.L. 2004, *Operations research : Applications and algorithms, .*

Wong, K.P. 1995, “Solving power system optimization problems using Simulated Annealing”, Contributing paper, The University of Western Australia, Western Australia

Zeng, Y. 1995, "Integration of an expert system and dynamic programming approach to optimize log breakdown using 3-dimensional log and internal defect shape information", PhD Thesis, Origon State University, .

Appendix A: Data analysis

Effect of log rotation

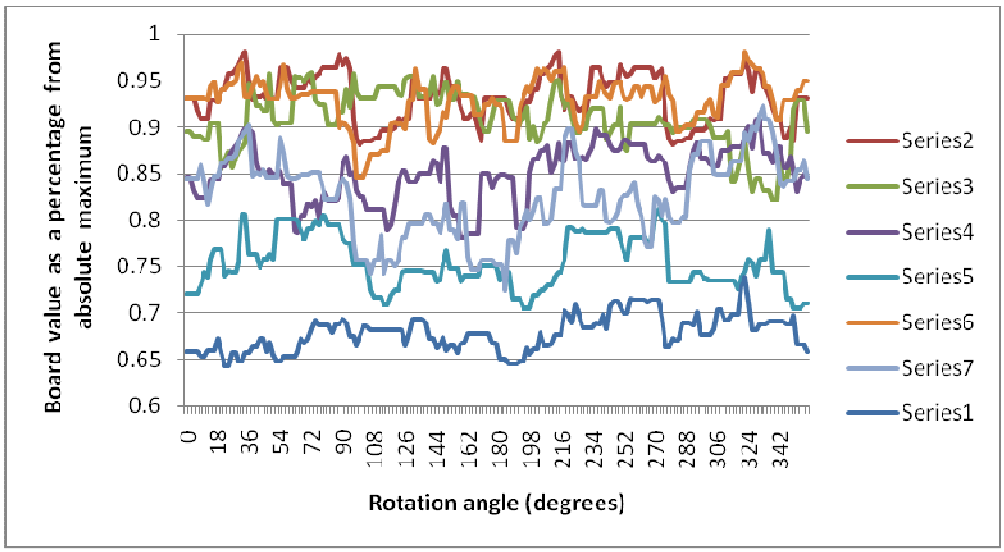


Figure 68: Log 1, Skewing and Offset = 0

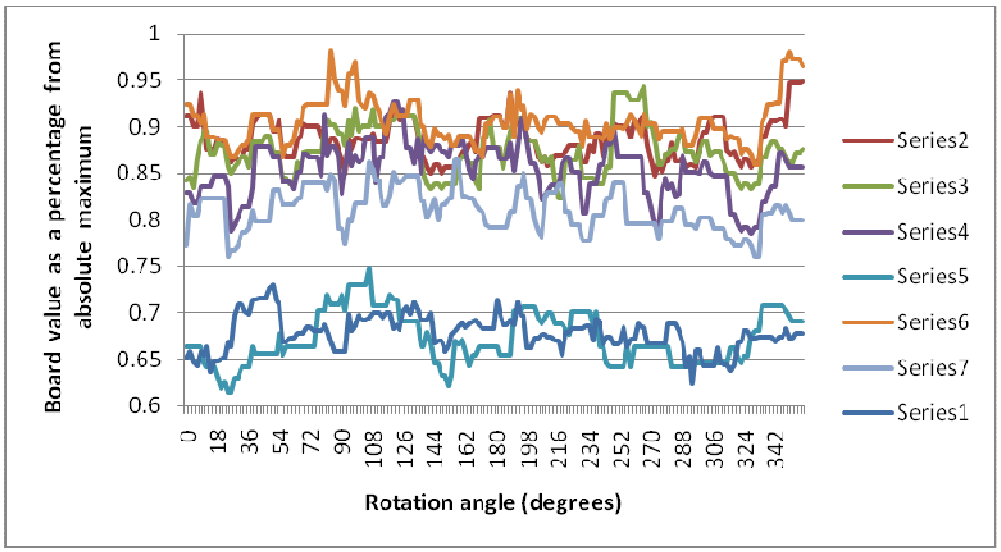


Figure 69: Log 2, Skewing and Offset = 0

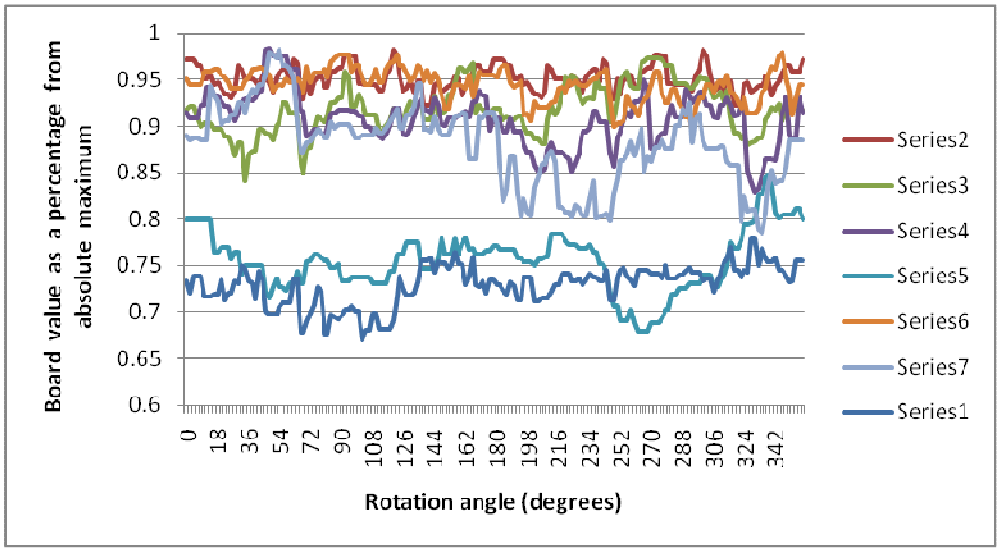


Figure 70: Log 3, Skewing and Offset = 0

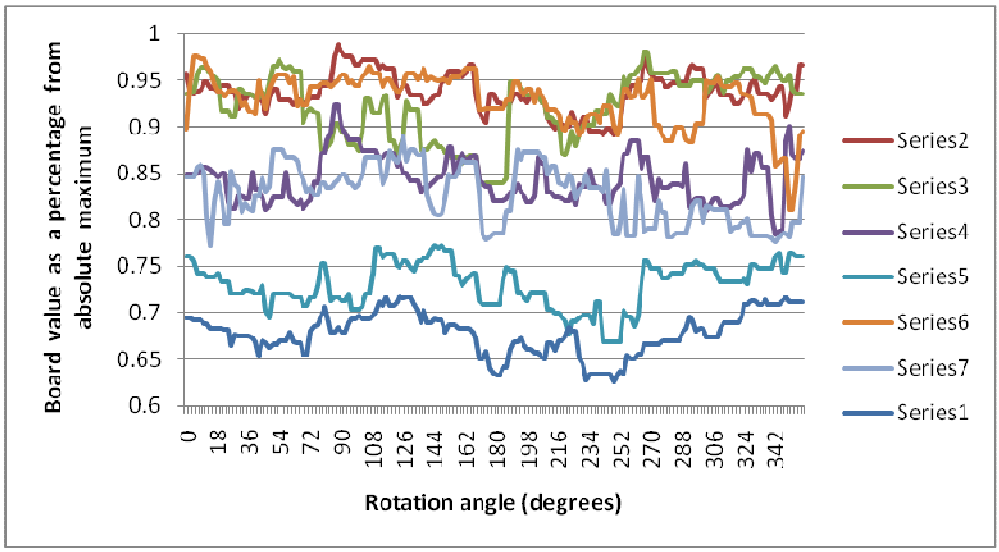


Figure 71: Log 4, Skewing and Offset = 0

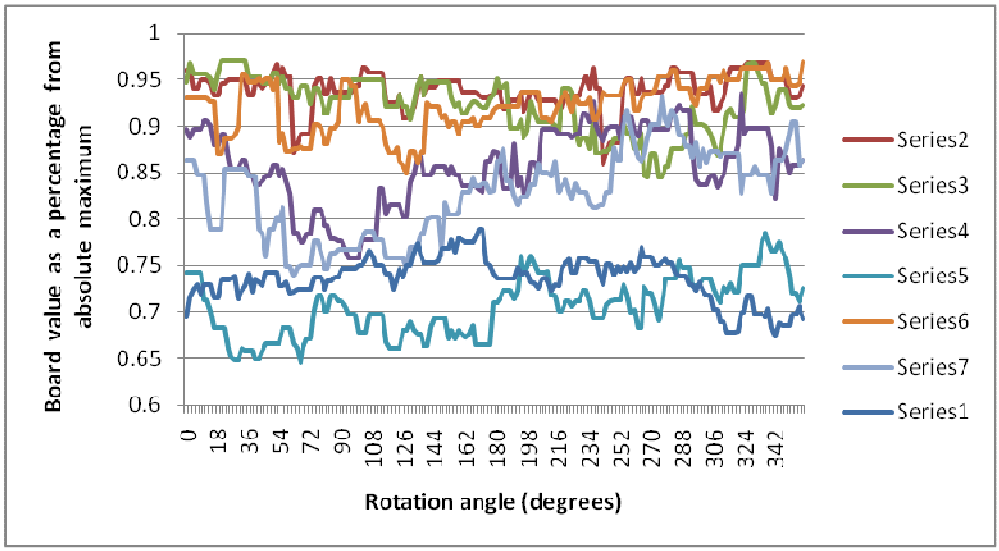


Figure 72: Log 5, Skewing and Offset = 0

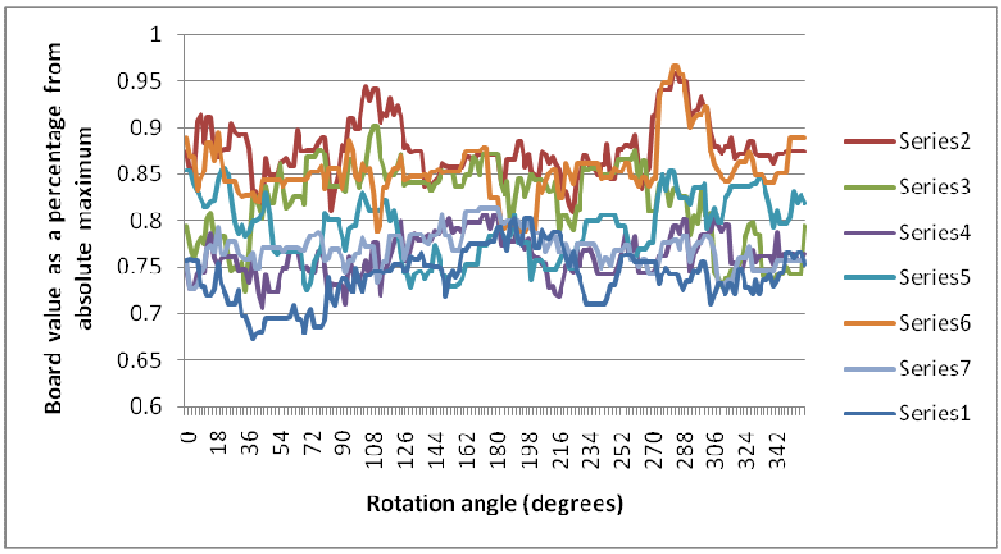


Figure 73: Log 6, Skewing and Offset = 0

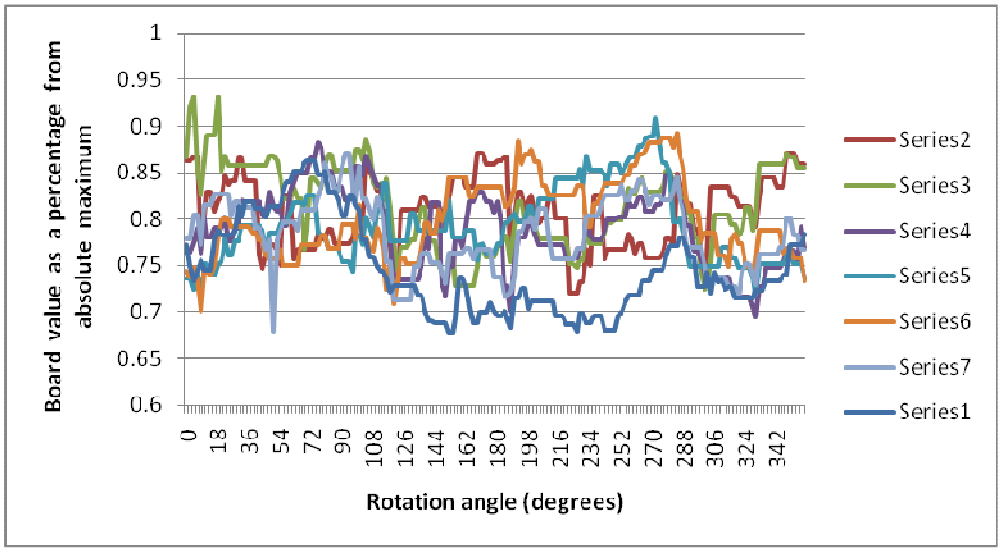


Figure 74: Log 7, Skewing and Offset = 0

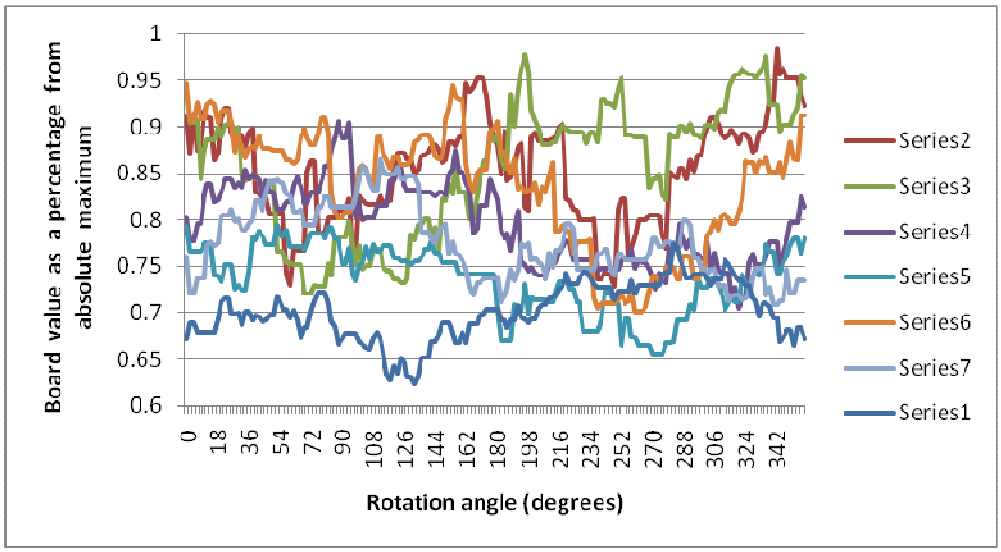


Figure 75: Log 8, Skewing and Offset = 0

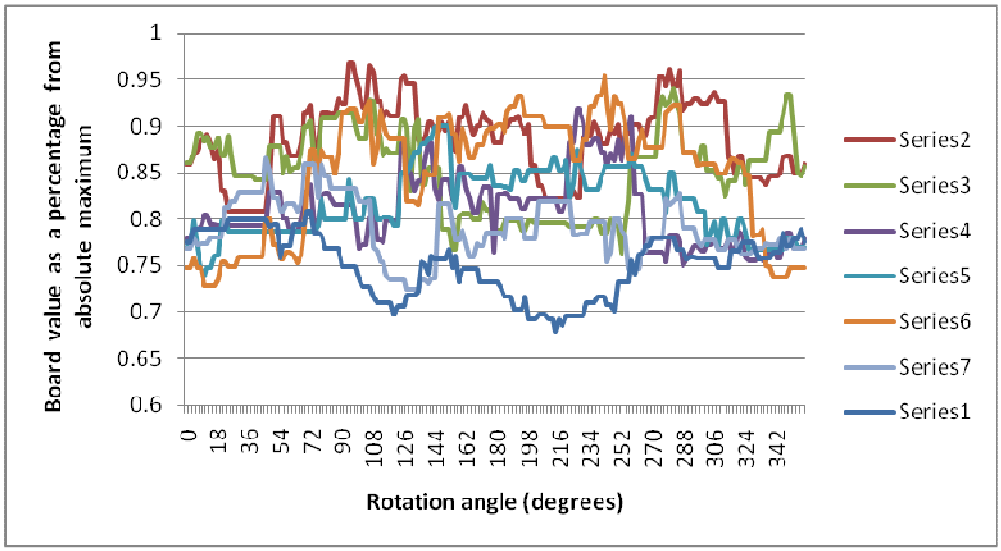


Figure 76: Log 9, Skewing and Offset = 0

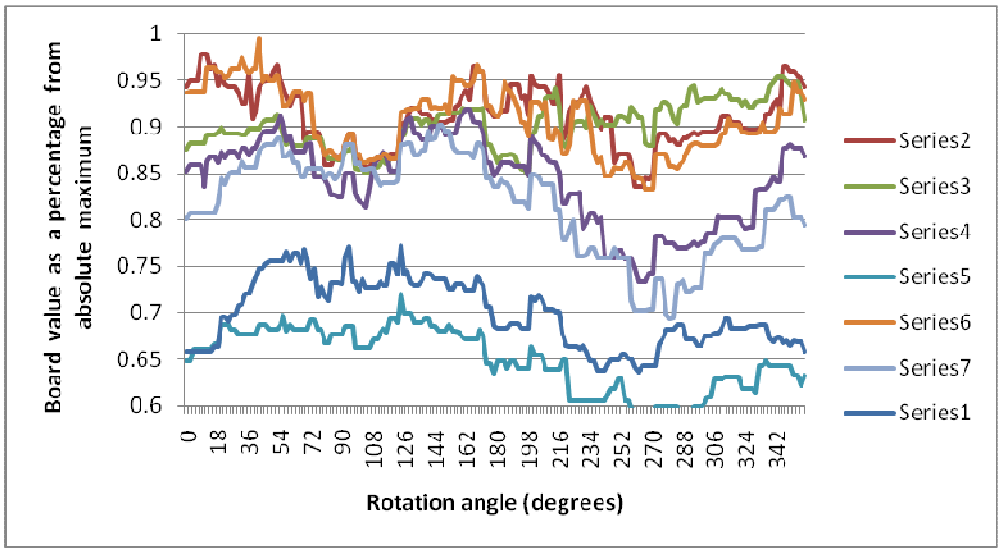


Figure 77: Log 10, Skewing and Offset = 0

Table 15: Outset from Simsaw (only a small part from log5)

Log5				
Log rotation	Log alignment	Log offset	Wet board volume	Board value
0	-27	-27	0.1748898	422.1058
0	-27	-24	0.1745982	421.177
0	-27	-21	0.1729458	416.5329
0	-27	-18	0.1726218	415.6041
0	-27	-15	0.1742418	420.2481
0	-27	-12	0.1739178	419.3193
0	-27	-9	0.1729458	416.5329
0	-27	-6	0.173043	416.5329
0	-27	-3	0.173043	416.5329
0	-27	0	0.1736505	418.3905
0	-27	3	0.1638225	390.5036
0	-27	6	0.1657422	396.0765
0	-27	9	0.1631178	370.8781
0	-27	12	0.1689498	383.7561
0	-27	15	0.1725138	392.0528
0	-27	18	0.1725138	398.2986
0	-27	21	0.1620018	374.3454
0	-27	24	0.1620018	374.3454
0	-27	27	0.1590498	367.746
0	-24	-27	0.173043	416.5329
0	-24	-24	0.1745982	421.177
0	-24	-21	0.1719738	413.7464
0	-24	-18	0.1739178	419.3193
0	-24	-15	0.1742418	420.2481
0	-24	-12	0.1726218	415.6041
0	-24	-9	0.1739178	419.3193
0	-24	-6	0.1710747	410.96
0	-24	-3	0.173043	416.5329
0	-24	0	0.1756674	423.9634
0	-24	3	0.1736505	418.3905
0	-24	6	0.1657422	396.0765
0	-24	9	0.1631178	370.8781
0	-24	12	0.1650618	376.451
0	-24	15	0.1708938	387.4087
0	-24	18	0.1725138	392.0528
0	-24	21	0.1649538	380.9448
0	-24	24	0.1620018	374.3454
0	-24	27	0.1620018	374.3454
0	-21	-27	0.1736505	418.3905
0	-21	-24	0.1736262	418.3905
0	-21	-21	0.1719738	413.7464
0	-21	-18	0.1739178	419.3193
0	-21	-15	0.1739178	419.3193
0	-21	-12	0.1742418	420.2481
0	-21	-9	0.1739178	419.3193
0	-21	-6	0.1798218	436.4741
0	-21	-3	0.1710747	410.96
0	-21	0	0.173043	416.5329
0	-21	3	0.1736505	418.3905

Table 16: Board grade prices

Thickness (mm)	Width (mm)	Length	Grade	Price per m3
25	76	All	Packaging	1566
25	76	All	Utility	2136
25	76	All	S7	3259
25	114	All	Packaging	1566
25	114	All	Utility	2136
25	114	All	S7	3259
25	152	All	Packaging	1566
25	152	All	Utility	2136
25	152	All	S7	3259
25	228	All	Packaging	1566
25	228	All	XXX	0
25	228	All	Utility	2136
25	228	All	S7	3259
38	76	All	XXX	1564
38	76	All	S5	2188
38	76	All	S7	3300
38	76	All	Clear	2445
38	114	All	XXX	1388
38	114	All	S5	2126
38	114	All	S7	3300
38	114	All	Clear	2376
38	152	All	XXX	1421
38	152	All	S5	2199
38	152	All	S7	3300
38	152	All	Clear	2457.5
38	228	All	XXX	1454
38	228	All	S5	2272
38	228	All	S7	3300
38	228	All	Clear	2539

Appendix B: MATLAB code for all optimizing algorithms

Genetic algorithm

```
PPL = 20;  %Present Population size
PC = 0.8;  %Probability value for Crossover
PM = 0.1;  %Probability value for Mutation
m = 0;
Convert(1:PPL,1:18) = 0;
array = xlsread('Log1_27');
   % PP is die Present Population
for i=1:PPL,
   PP(i,1:18) = round(rand(1,18));   %Creates array with PPL rows en
18 columns
end;  % i for loop
% Test each chromosome in the created array whether or not it is a
viable
  solution
for j =1:PPL,
   strSk2 = 20;
   strOf2 = 100;
   strRo2 = 400;
    while (strSk2>18)
          strOf2 = 100;
            while (strOf2>18)
               strRo2 = 400;
                  while (strRo2>179)
                     PP(j,1:18) = round(rand(1,18));
                     Convert = PP(j,1:18)
                     z = 1;
                     ConvToDecGA
                  end   % while
        end;   % while
  end;  % while

end  % j for loop

for big=1:50, %amount of times new offspring are produced
  for k = 1:PPL  %Determine board value for each chromosome in PP
      z = 1;
     Convert = PP(k,1:18);
     ConvToDecGA;
     PP(k,19) = array(nommer,4);
  end  % k for loop

  %%Create new population
  %Choose two parents with tournament selection method
  Crandom = ceil(rand(1,4) * PPL)  %Generate random numbers from one
to PPL
if PP(Crandom(1,1),19) > PP(Crandom(1,2),19)
```

```
    Parent1 = PP(Crandom(1,1),1:18)
else
    Parent1 = PP(Crandom(1,2),1:18);
end
if PP(Crandom(1,3),19) > PP(Crandom(1,4),19)
    Parent2 = PP(Crandom(1,3),1:18)
else
    Parent2 = PP(Crandom(1,4),1:18);
end

  %%Crossover
CrossWF = 0;
while CrossWF == 0

 Cif = rand;
 Cpos = 1;
 if Cif <= PC
    Cpos = ceil(rand * 17); %Choose place where crossover should be
performed
    Child1(1:Cpos) = Parent1(1:Cpos);
    Child1(Cpos+1:18) = Parent2(Cpos+1:18);
    Child2(1:Cpos) = Parent2(1:Cpos);
    Child2(Cpos+1:18) = Parent1(Cpos+1:18);
 else
    Child1 = Parent1;
    Child2 = Parent2;
 end
 % Test if Child1 gives viable solution
 Convert = Child1;
 ConvToDecGA;
    if strSk2 <= 18
        CrossWF = 1;
    end
    if strOf2 <= 18 && CrossWF == 1
        CrossWF = 1;
    else
        CrossWF = 0;
    end
    if strRo2 <= 179 && CrossWF == 1
        CrossWF = 1;
    else
        CrossWF = 0;
    end

% Test if Child2 gives viable solution
 Convert = Child2;
 ConvToDecGA;
    if strSk2 <= 18 && CrossWF == 1
        CrossWF = 1;
    else
        CrossWF = 0;
    end
    if strOf2 <= 18 && CrossWF == 1
        CrossWF = 1;
    else
```

```
        CrossWF = 0;
    end
    if strRo2 <= 179 && CrossWF == 1
        CrossWF = 1;
    else
        CrossWF = 0;
    end
end %While CrossWF

%Mutation for Child1
MutWF = 0;
Child1_2 = Child1
while MutWF == 0
   Child1_2 = Child1
for y=1:18
   RM = rand
   if RM <= PM
       if Child1(1,y) == 1
           Child1_2(1,y) = 0
       else
           Child1_2(1,y) = 1
       end
   end
end  % y loop
%Mutation Child1
 Convert = Child1_2;
 ConvToDecGA;
    if strSk2 <= 18
        MutWF = 1;
    end
    if strOf2 <= 18 && MutWF == 1
        MutWF = 1;
    else
        MutWF = 0;
    end
    if strRo2 <= 179 && MutWF == 1
        MutWF = 1;
    else
        MutWF = 0;
    end

end  % while MutWF for Child1

%Mutation Child2
MutWF = 0;
Child2_2 = Child2
while MutWF == 0
   Child2_2 = Child2
for y=1:18
   RM = rand
   if RM <= PM
       if Child2(1,y) == 1
           Child2_2(1,y) = 0;
       else
           Child2_2(1,y) = 1;
```

```
        end
    end
end%  of y loop
 Convert = Child2_2;
 ConvToDecGA;
    if strSk2 <= 18
        MutWF = 1;
    end
    if strOf2 <= 18 && MutWF == 1
        MutWF = 1;
    else
        MutWF = 0;
    end
    if strRo2 <= 179 && MutWF == 1
        MutWF = 1;
    else
        MutWF = 0;
    end
end % of while MutWF for Child2

z = 1;
%Test if new offspring should replace weakest gene in population

Convert = Child1_2(1,1:18)
ConvToDecGa;
Child1_2(1,19) = array(nommer,4)
BVC1 = Child1_2(1,19)
Convert = Child2_2(1,1:18)
ConvToDecGa;
Child2_2(1,19) = array(nommer,4)
BVC1 = Child2_2(1,19)
PP = sortrows(PP, -19)
C1 = Child1_2(1,19)
C2 = Child2_2(1,19)
PP9 = PP(9,19)
if Child1_2(1,19) > Child2_2(1,19)
    if Child1_2(1,19) > PP(9,19) && Child2_2(1,19) > PP(9,19)
       PP(PPL-1,1:19) = Child1_2(1,1:19)
       PP(PPL,1:19) = Child2_2(1,1:19)
    else
        if Child1_2(1,19) > PP(PPL,19)
            PP(PPL,1:19) = Child1_2(1,1:19)
        end
    end
else
    if Child1_2(1,19) > PP(9,19) && Child2_2(1,19) > PP(9,19)
       PP(PPL-1,1:19) = Child2_2(1,1:19)
       PP(PPL,1:19) = Child1_2(1,1:19)
    else
       if Child2_2(1,19) > PP
          PP(PPL-1,1:19) = Child1_2(1,1:19)
          PP(PPL,1:19) = Child2_2(1,1:19)
       else
          if Child2_2(1,19) > PP(PPL,19)
              PP(PPL,1:19) = Child2_2(1,1:19)
```

```
            end
        end
    end
end
PP = sortrows(PP, -19)

%%Write chromosome with best value to an array
Convert = PP(1,1:18);
ConvToDecGA;
BVmax = PP(1,19);
nommermax = nommer;
Romax = strRo2*2;
Ofmax = strOf2*3-27;
Skmax = strSk2*3-27;
m = m + 1;
finalarray(m,1) = Romax;
finalarray(m,2) = Ofmax;
finalarray(m,3) = Skmax;
finalarray(m,4) = BVmax;
finalarray(m,5) = nommermax;

end  %big for loop
```

"ConvToDecGa" function:

```
strSk2 = 0;
strOf2 = 0;
strRo2 = 0;
Convert
for a=5:-1:1
   strSk2 = strSk2 + Convert(z,a)*2^(5-a);
end
for a=10:-1:6
   strOf2 = strOf2 + Convert(z,a)*2^(10-a);
end
for a=18:-1:11
   strRo2 = strRo2 + Convert(z,a)*2^(18-a);
end
nommer = strRo2*361 + 19*(strOf2) + strSk2 + 1;
```

PBIL

```
Big = 100;
for final = 1:Big  %amount of times the algorithm is repeated to get
'final' amount of answers
LR = 0.3;   % Learning rate
b = 23;   % number of bits in chromosome
PV(1:18) = 0.5;
PV2(1:18) = 0.5;
```

```
SVL = 20;  %Number of rows in SV
z = 1;
count = 0;
for big=1:50,  %This determines how many iterations should be
evaluated
    for i=1:SVL,
      SV(i,1:18) = rand(1,18);   %Creates array with SVL rows and 18
columns
    end

%%
% Test each chromosome in the created array whether or not it is a
viable solution
for z=1:SVL,
   strSk2 = 20;
   strOf2 = 100;
   strRo2 = 400;
    while (strSk2>18)
          strOf2 = 100;
            while (strOf2>18)
               strRo2 = 400;
                  while (strRo2>179)
                     SV(z,1:18) = rand(1,18);
                     for y=1:18  % Tests if random number is smaller
than number in PV
                        if SV(z,y) < PV(1,y)
                            SV(z,y) = 1;
                        else
                            SV(z,y) = 0;
                        end
                     end
                     Convert = SV
                     ConvToDecPBIL2;
                  end;
        end;
  end;
     nommer = (strRo2)*361 + 19*(strOf2) + strSk2 + 1
     BV = array(nommer,4);
     board(1,z) = BV;
  end;  % z for loop
  %%
  maxry = 1;
  maxvalue = board(1,1);
  for y=1:(SVL-1)  % Search the row in SV with the greatest board
value
      if board(1,y+1) > maxvalue
          maxvalue = board(1,y+1);
          maxry = y + 1;
      end;
  end;
  count = count + 1
 values(count,1) = maxvalue;
    for x=1:18,
       if SV(maxry,x) == 1
          PV(1,x) = PV(1,x)*(1-LR) + SV(maxry,x)*LR;
```

```
          else
           PV(1,x) = PV(1,x)*(1-LR);
         end
      end;

  PV
end; % of big for loop


  %% convert PV's binary values to decimal values
  PV
  for x=1:18,
      if PV(1,x) > 0.5
          PV(1,x) = 1;
      else
          PV(1,x) = 0;
      end;
  end;
  Convert = PV;
  z = 1;
  ConvToDecPBIL2;
  nommer = strRo2*361 + 19*(strOf2) + strSk2 + 1;
strRo2 = strRo2*2;
strOf2 = strOf2*3-27;
strSk2 = strSk2*3-27;
     BV = array(nommer,4);
     finalarray(final,1) = strRo2;
     finalarray(final,2) = strOf2;
     finalarray(final,3) = strSk2;
     finalarray(final,4) = BV;
     finalarray(final,5) = nommer;

end  % big for loop
```

ConvToDecPBIL2 function:

```
strSk2 = 0;
strOf2 = 0;
strRo2 = 0;
for a=5:-1:1
   strSk2 = strSk2 + Convert(z,a)*2^(5-a)
end
for a=10:-1:6
   strOf2 = strOf2 + Convert(z,a)*2^(10-a)
end
for a=18:-1:11
   strRo2 = strRo2 + Convert(z,a)*2^(18-a)
end
```

Simulated annealing

```
clc
clear all
array = xlsread('Log1_27');
Fritz_algoritme_V1
Tmin = 0;
Tin = 5;
Sol = maxry4
IRo = (Sol-mod(Sol,361))/361  %Initial Rotation position , from 0 to
179
IOfvar = (Sol-IRo*361)-1;
IOf = (IOfvar-mod(IOfvar,19))/19   %from 0 to 18
ISk = Sol-IRo*361-IOf*19-1   % from 0 to 18

T = Tin;  %current temperature
m = 1;
BV = array(Sol,4)
val(m,1) = Sol;
val(m,2) = BV;
Sol = maxry4 %solution
Ro = IRo
Of = IOf
Sk = ISk
while T >= Tmin
    for p = 1:5
    RoTF = 0;
    while RoTF == 0
        ChRo = ceil((rand(1)-0.5)*2*45*(T/Tin));   %Change in
Rotation variable (kan 90 grade na elke kant toe verander)
        if Ro + ChRo < 0
            RoTF = 0;
        else
           if Ro + ChRo > 179
               RoTF = 0;
           else
               RoTF = 1;
               Ro = Ro + ChRo;
           end
        end
    end
    OfTF = 0;
    while OfTF == 0
        ChOf = ceil((rand(1)-0.5)*2*9*(T/Tin));   %Change in
Rotation variable
        if Of + ChOf < 0
            OfTF = 0;
        else
           if Of + ChOf > 18
               OfTF = 0;
           else
               OfTF = 1;
               Of = Of + ChOf;
           end
        end
    end
```

```
    SkTF = 0;
    while SkTF == 0
        ChSk = ceil((rand(1)-0.5)*2*9*(T/Tin));    %Change in
Rotation variable
        if Sk + ChSk < 0
            SkTF = 0;
        else
           if Sk + ChSk > 18
               SkTF = 0;
           else
               SkTF = 1;
               Sk = Sk + ChSk;
           end
        end
    end
    Ro
    Of
    Sk
    SolN = Ro*361 + Of*19 + Sk + 1  %New solution
    BVN = array(SolN,4)
 % for K = 1:10
    delta = BV - BVN;  % BVN = Board Value New
    if delta < 0
        Sol = SolN;  % SolN = Solution New
        BV = BVN;
        m = m + 1
        val(m,1) = Sol;
        val(m,2) = BVN;
    else
        Pe = rand(1)  % Pe = Probability to accept solution
        if Pe < exp(-delta/T)
            Sol = SolN;
            BV = BVN;
            m = m + 1
            val(m,1) = Sol;
            val(m,2) = BVN;
        else
        end
    end
  %end
  end
  T = T - 0.1
end
```

Cross-Entropy method

```
function SawWood

NumVars=3; PopSize = 50; varrho=0.75;
N(1) = 180; N(2) = 19; N(3) = 19;
Solutions = xlsread('Log1_27.xls');
```

```
    estimated_gamma = -1000000; Terminate = 0;
for i=1:NumVars
    Pr(1:N(i), i)=1/N(i);
end  %i
WorkArea(1:PopSize, 1:NumVars+1)=0;
tic
for z=1:50
for j=1:NumVars
    for i=1:PopSize
        TempR = rand;
        CumSum = 0;
        for k=1:N(j)
            CumSum = CumSum + Pr(k, j);
            if CumSum >= TempR, break, end
        end  %j
        WorkArea(i,j) = k;
    end
end  %i
%Calc indices:
WorkArea(:,1) = 2*(WorkArea(:,1) - 1);
WorkArea(:,2:3) = 3*(WorkArea(:,2:3) - 1) - 27;
%Lookup the objective associated with the three-combo of indices:
for i=1:PopSize
    Index = 361*WorkArea(i,1)/2 + 19*(WorkArea(i, 2)+27)/3 +
(WorkArea(i, 3)+27)/3 + 1;
    WorkArea(i, NumVars+1) = Solutions(Index, NumVars+1);
end  %i
%Sort for CEM:
WorkArea = sortrows(WorkArea, -(NumVars+1));
%Get proportions of each index in top varrho group:
N1 = round(PopSize*varrho);
SampleSize = PopSize - N1;
Sample=[];
Sample = WorkArea(1:PopSize-N1,:);

Last_estimated_gamma = estimated_gamma;
estimated_gamma = Sample(1, NumVars+1)
if Last_estimated_gamma == estimated_gamma, Terminate = Terminate +
1; end
if Terminate > 5, break, end
Stdev = std(Sample(:,NumVars+1));

for j=1:NumVars
    if j==1
       Bins(1:N(j)) = 2*((1:N(j)) - 1); %Maps onto 0 to 358 degrees
    else
       Bins = 3*((1:N(j))-1) - 27;  %Maps onto -27 to +27 in steps
of 3.
    end
    H = hist(Sample(:,j), Bins);
    for k=1:N(j)
        PrevPr = Pr(k, j);
        Pr(k, j) = Pr(k, j) + H(k)/SampleSize;
        Pr(k, j) = 0.75*Pr(k, j) + 0.25*PrevPr;
    end  %i
```

```
end

end %z

toc
z
Sample(1,:)
end  % SawWood
```

Alternative algorithm version 10

```
array = xlsread('Log1_27');
count = 0;
%Determine max when every 10degrees are tested
for z=1:36
   grade(1,z) = array(181+(z-1)*1805,4);
end
grade
  maxrycount = 1;
  maxvalue = grade(1,1)
  for y=1:35,
      if grade(1,y+1) > maxvalue
          maxvalue = grade(1,y+1);
          maxrycount = y + 1;
      end;
  end;
  maxry = (maxrycount-1)*1805+181
  maxvalue = array(maxry,4);  %stores monetary value of max
count = count + 1;
 values(count,1) = maxvalue;
 values(count,2) = maxry;
  maxry1 = maxry;


  %Determines max offset at maxry (degrees)
  for x=0:18,
     offset(1,x+1) = array(maxry+19*(x-9),4);
  end
  maxryOf = 1;
  maxvalue = offset(1,1);
  for y=1:18,
      if offset(1,y+1) > maxvalue
          maxvalue = offset(1,y+1);
          maxryOf = y + 1;
      end;
  end;
  offset;
  maxryOf = maxry + (maxryOf-10)*19;
 count = count + 1;
 values(count,1) = maxvalue;
 values(count,2) = maxryOf;
  %Determine max skewing at max offset
    for x=0:18,
     skew(1,x+1) = array(maxryOf+(x-9),4);
```

```
  end
  maxry4 = 1;
  maxvalue = skew(1,1);
  for y=1:18,
      if skew(1,y+1) > maxvalue
          maxvalue = skew(1,y+1);
          maxry4 = y + 1;
      end;
  end;
  skew;
  maxry4;
  maxry4 = maxryOf + (maxry4-10);
           count = count + 1;
 values(count,1) = maxvalue;
 values(count,2) = maxry4;

  %Determine max degrees at above offset en skewing, 44 degrees to
each side, with increments of 4degrees
for z=1:23
    ry(1,z) = maxry4+(z-1)*722-7942;
    if ry(1,z) < 0
        grade(1,z) = array(64981+ry(1,z),4);
    else
        if maxry4+(z-1)*722-7942 > 64980
            grade(1,z) = array(maxry4+(z-1)*722-7942-64980,4);
        else
            grade(1,z) = array(maxry4+(z-1)*722-7942,4);
        end
    end
end
  maxrycount = 1;
  maxvalue = grade(1,1);
  for y=1:22,
      if grade(1,y+1) > maxvalue
          maxvalue = grade(1,y+1);
          maxrycount = y + 1;
      end;
  end;
  grade;
  maxry = ry(1,maxrycount);
  maxvalue = array(maxry,4);
           count = count + 1;
 values(count,1) = maxvalue;
 values(count,2) = maxry;

  %Determine max offset at maxry2 (degrees)
  for x=0:18,
     offset_1(1,x+1) = array(maxry2+19*(x-9),4);
  end
  maxry3_1 = 1;
  maxvalue2 = offset_1(1,1);
  for y=1:18,
      if offset_1(1,y+1) > maxvalue2
          maxvalue2 = offset_1(1,y+1);
          maxry3_1 = y + 1;
```

```
      end;
  end;
  offset_1;
  maxry3_1 = maxry2 + (maxry3_1-10)*19;
  maxvalue2 = array(maxry3_1,4);
           count = count + 1;
 values(count,1) = maxvalue2;
 values(count,2) = maxry3_1;
  %Determine max skewing at max offset
    for x=0:18,
     skew_1(1,x+1) = array(maxry3_1+(x-9),4);
  end
  maxry4_1 = 1;
  maxvalue2 = skew_1(1,1);
  for y=1:18,
      if offset_1(1,y+1) > maxvalue2
          maxvalue2 = skew_1(1,y+1);
          maxry4_1 = y + 1;
      end;
  end;
  maxry4_1 = maxry3_1 + (maxry4_1-10);
  maxvalue2 = array(maxry4_1,4);
           count = count + 1;
 values(count,1) = maxvalue2;
 values(count,2) = maxry4_1;
  %Determine max degrees at above offset and skewing, 44degrees to
each side, with increments of 4 degrees
for z=1:23
    ry(1,z) = maxry4_1+(z-1)*722-7942;  %create array 'ry' which
stores the row value that needs to be evaluated
    if ry(1,z) < 0  %test of row values are negative or too big, and
chances it accordingly
        grade(1,z) = array(64981-ry(1,z));
    else
        if maxry4_1+(z-1)*722-7942 > 64980
            grade(1,z) = array(maxry4_1+(z-1)*722-7942-64980,4);
        else
            grade(1,z) = array(maxry4_1+(z-1)*722-7942,4);
        end
    end
end
  maxrycount = 1;
  maxvalue2 = grade(1,1);
  for y=1:22,
      if grade(1,y+1) > maxvalue2
          maxvalue2 = grade(1,y+1);
          maxrycount = y + 1;
      end;
  end;
  maxry_1 = ry(1,maxrycount);
  maxvalue2 = array(maxry_1,4);
           count = count + 1;
 values(count,1) = maxvalue2;
 values(count,2) = maxry_1;
```

```
 maxvalue
 if maxvalue > maxvalue2
     absmax = maxvalue
     absmaxry = maxry4;
 else
     absmax = maxvalue2
     absmaxry = maxry_1;
 end
    count = count + 1;
  values(count,1) = absmax;
values(count,2) = absmaxry;
```