LUT
Lappeenranta
University of Technology

LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

School of Business and Management

Strategic Finance and Business Analytics (MSF)

Rudolf Sandström

Creating a profitability optimization tool for a multi-product portfolio – Case Meat Processing Industry

Master's thesis

2019

1st Supervisor: Post-Doctoral Researcher, D.Sc. (Econ. & BA) Jyrki Savolainen

2nd Supervisor: Professor, D.Sc. (Econ. & BA) Mikael Collan

ABSTRACT

Author: Rudolf Sandström

Title: Creating a profitability optimization tool for a multi-product portfolio – Case Meat Processing Industry

Faculty: LUT School of Business and Management

Master's Programme: Strategic Finance and Business Analytics (MSF)

Year: 2019

Master's thesis: 59 pages, 8 figures, 2 tables

Examiners: Post-Doc. Researcher, D.Sc. (Econ. & BA) Jyrki Savolainen,

Professor, D.Sc. (Econ. & BA) Mikael Collan

Keywords: product portfolio optimization, operations research, profitability optimization tool, meat processing industry

The objective of this thesis is to create a profitability optimization tool that can be used for strategic- and tactical product portfolio optimization in a case company operating in the meat processing industry. To achieve this objective, essential optimization theory is sought and presented and a literature review of the most potential optimization algorithms for the tool is made. The analysis suggests that the most potential algorithms for product portfolio optimization problems are: simplex, genetic algorithm, particle swarm optimization, simulated annealing, and tabu search.

The tool build for the case company is based on Excel Solver and it uses the simplex algorithm augmented with sensitivity analysis as its solving method. The tool includes slots for 200 decision variables, possibility to give maximum, minimum and equality constraints to each of the decision variables, slots for 30 availability (maximum) constraints for different raw materials and a possibility to give maximum constraints for 20 different product groups. Moreover, the tool can be used to perform different simulations of which a few examples are presented.

Results reveal that the tool can be built successfully by utilizing key concepts from the modeling and optimization theory. The potential for profitability increase by using the tool was calculated with historical data to be 1,68% on an annual level.

TIIVISTELMÄ

Tekijä: Rudolf Sandström

Tutkielman nimi: Kannattavuudenoptimointityökalun luominen monituoteportfoliolle – Case Lihanjalostus

Tiedekunta: LUT-Kauppakorkeakoulu

Maisteriohjelma: Strategic Finance and Business Analytics (MSF)

Vuosi: 2019

Pro Gradu -tutkielma: 59 sivua, 8 kuviota, 2 taulukkoa

Tarkastajat: Tutkijatohtori Jyrki Savolainen, Professori Mikael Collan

Avainsanat: tuoteportfolion optimointi, operaatiotutkimus, kannattavuudenoptimointityökalu, lihanjalostus

Tämän tutkimuksen tarkoituksena on luoda kannattavuudenoptimointityökalu, jota voidaan käyttää strategiseen ja taktiseen tuoteportfolion optimointiin lihateollisuuden case-yrityksessä. Työssä esitellään oleellisimmat teoriat liittyen kyseiseen optimoinnin alueeseen pohjustuksena työkalun luomiselle. Tämän lisäksi potentiaalisimmat optimointialgoritmit etsitään ja esitellään kirjallisuuskatsauksessa. Kirjallisuuskatsauksen tulokset osoittavat, että potentiaalisimmat optimointialgoritmit tuoteportfolion optimointiin ovat: simplex, genetic algorithm, particle swarm optimization, simulated annealing ja tabu search.

Case-yritykselle rakennettu työkalu pohjautuu Excelin Ratkaisimeen, ja se käyttää simplex algoritmia ja herkkyysanalyysiä optimointimetodina. Työkaluun voidaan syöttää 200 lopputuotetta, joille voidaan antaa vähimmäis-, yhtäsuuruus- ja enimmäismäärä rajoitteet. Lopputuotteille voidaan syöttää 30 raaka-ainetta, joille voidaan asettaa saatavuutta kuvaavat rajoitteet. Tuotteet voidaan myös ryhmitellä 20 eri tuoteryhmään, joille voidaan antaa enimmäismäärä rajoitteet. Tuoteportfolion optimoinnin lisäksi työkalua voidaan käyttää erilaisten päätöksentekoa tukevien simulaatioiden suorittamiseen, joita kuvataan tutkielmassa esimerkkien avulla.

Tutkielman tulokset osoittavat, että kannattavuudenoptimointityökalu voitiin rakentaa case-yritykselle onnistuneesti hyödyntämällä työssä esitettyjä optimointi- ja mallinnusteorioita. Hyödyntämällä historiallista dataa pystyttiin toteamaan, että työkalua käyttämällä case-yritys olisi pystynyt parantamaan kannattavuuttaan 1,68% vuosittaisella tasolla.

Acknowledgements

The writing of this thesis was a long and challenging process, and I feel happy to finally finish this project. First and foremost, I want to thank Post-Doc. Researcher Jyrki Savolainen and Professor Mikael Collan for good guidance and support with this thesis. I also want to thank the case-company for giving me the opportunity to put the theoretical knowledge acquired during my studies into practice with this project. Last but definitely not least, I want to thank my family, friends, and girlfriend for all the support they have given me under my studies. This thesis is dedicated to you.

The past years at LUT have been unforgettable. I have met many great people and made lifelong friends at LUT as well as during my two exchanges abroad. I want to thank you all for all the good memories we have made together, and I am looking forward to our future adventures!

In Stockholm, 8.9.2019

Rudolf Sandström

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Since the beginning of the industrial revolution, the size and complexity of companies have increased remarkably. Many of the small family-owned businesses of an earlier era have grown into billion class corporations. (Hillier & Liebermann 1990, 1) As a result, also business problems and decision-making situations have been growing in complexity. Many situations today are so complex in nature that it is very difficult for management to make rational fact-based decisions without the help of decision support. The function of decision support is to provide tools for the decision makers to prepare the decision and to analyze different actions to take (Eiselt & Sandblom 2012, 6). This thesis concerns decision support from a profitability management point of view by building a profitability optimization tool for a case company operating in the meat processing industry.

1.1 Scientific background of the study

The scientific background of this thesis is in the Operations Research (OR), also referred to as Management Science. The leading international association for professionals in operations research and analytics, INFORMS, describes OR in the following way: *"O.R is the application of scientific & mathematical methods to the study & analysis of problems involving complex systems."* In more practical terms OR is a scientific discipline that studies the applications of advanced analytical methods for enhanced decision making (Informs 2019a).

OR has attributes from several different scientific fields. Essentially it can be described as a subfield of applied mathematics, as it is utilizing many different mathematical and statistical techniques to solve complex real-world problems. The scientific field is also closely related to computer science as it uses extensively computers and algorithms to perform complex mathematical calculations. The historical development of operations research and an overview of its current state is presented in Informs (2019b).

1.2 Objective and methodology

The objective of this thesis is to create a profitability optimization tool utilizing mathematical optimization that can be used for strategic- and tactical product portfolio optimization in a case company operating in the meat processing industry. The problem can be formalized into the following question: which products should the case company produce from a limited amount of raw materials to maximize profits simultaneously considering all the constraints concerning the product portfolio? To achieve the objective of this thesis, the following research question is formed:

> *How to build and implement a simulation model for a product portfolio profitability optimization problem in a meat processing case company which has a fixed amount of raw materials and several constraints concerning its product portfolio?*

The main research question will be supplemented with the following sub-questions:

1. *Which are the most popular optimization algorithms for product mix optimization problems that could be applied to the problem of the case company?*

2. *Which optimization algorithm is most suitable for solving the problem of the case company?*

In order to seek answers to these questions, the research process is divided into two sections. The first section consists of a comprehensive review of previous literature concerning modeling and optimization. The second section includes an empirical case study where the theoretical concepts from the first section are applied for building a profitability optimization tool for the case company.

1.3 Structure of the study

This study is divided into six chapters. The second chapter contains basic theoretical concepts from the literature concerning modeling and different solving methods for optimization problems including linear programming. The third chapter includes a literature review where four most potential metaheuristic algorithms for the tool are sought and presented. In addition, a few examples of how these metaheuristics could be applied to product portfolio optimization problems are also presented in this chapter. The fourth chapter describes how the problem of the case company was modeled and how the profitability optimization tool was built. Next, the implementation of the tool will be described and the results of the optimization will be presented and analyzed in the fifth chapter. The sixth chapter concludes the study by presenting answers to the research questions and by suggesting how the study could be continued further.

2 Theoretical background

The goal of this chapter is to provide basic knowledge concerning modeling and mathematical optimization in order to give a good introduction to the theme before diving into more complicated topics in the literature review and empirical part. This chapter will begin by introducing the process of modeling in general followed by an introduction of modeling an optimization problem. These subchapters will be followed by a subchapter introducing different solution techniques that can be used to solve optimization models, followed by a subchapter of performance evaluation of approximate optimization algorithms. Lastly, in the final subchapter, linear programming and sensitivity analysis will be introduced.

2.1 Modeling in general

Modeling refers to the building process of a mathematical model of a given problem (Cottle & Thapa 2017, preface xxvi). It is reasonable to begin the modeling process by representing the problem first verbally and graphically. This tends to improve the understanding of the problem and hence eases the formulation of the mathematical model. Next, the objectives of the model should be clearly specified in order to understand what the model should be able to accomplish and then collect enough relevant data for the model. The collection of data requires resources which in turn causes costs, which means that the model contains a trade-off between costs and insights gained. If the problem at hand is very large and complex, it is reasonable to convert it first to a smaller subproblem and then gradually modeling larger subproblems ultimately modeling the whole problem. The advantage of this approach is that the errors are easier to determine and solve in the smaller submodels compared to the situation where a large comprehensive model is constructed straight from the beginning. Lastly, the goodness of the model should be evaluated. It should be determined whether the model describes the attributes of the problem in a proper manner or not, and if the solution obtained from the model is reasonable in the context of the problem (Eiselt & Sandblom 2012, 9-10).

Mathematical models have several advantages compared to non-mathematical models. These advantages include *clarity*, *precision*, *brevity*, meaning that a good mathematical model gets down to the essence of a real situation, and enables a rigorous way of defining the problem. Other advantages are *predictive power* and *economy*, denoting that mathematical models can be utilized to construct simulations for predicting and analyzing several different outputs/scenarios in an effective and affordable way compared to executing simulations in a real physical environment. (Cottle & Thapa 2017, preface xxvii)

The accuracy of a mathematical model can be evaluated from two different viewpoints: *form* and *data*. The form describes how well and accurate a model reflects the most important attributes of the real-world problem to be solved, and data refers to the quality of the data (parameter values) inside the model. (Cottle & Thapa 2017, preface xxvii) Moreover, it is crucial that all the assumptions and simplifications in the model are clearly stated in a way that a decision maker is able to decide whether to reject or accept the recommendations of actions that the model provides (Eiselt & Sandblom 2012, 4).

There are many different types of models that can be built, and the modeling process differs depending on the problem to be modeled. The process of modeling is somewhat art and there is no single golden formula that could be applied to every modeling process. However, there are some guidelines that can be followed, and the next subchapter will introduce the general building process of a mathematical model for solving an optimization problem.

2.1.1 Modeling an optimization problem

The three fundamental components in a mathematical optimization model are *decision variables, constraints,* and *objective function*. Objective function represents the function to be optimized and decision variables are all the single variables in the objective function and in the constraints. Constraints are requirements that decision variables must satisfy when optimizing the objective function, and in a mathematical optimization model, they are represented as relations, equations or inequalities involving mathematical functions. If an optimization problem is subject to constraints, it is considered as a *constrained optimization problem*. Conversely, when the problem is not subject to such constraints, it is considered as an *unconstrained optimization problem*. The goal of an optimization model is to find those values for decision variables that result in an optimum output (minimum, maximum or a predetermined optimal value) for the objective function. (Cottle & Thapa 2017, preface xxiii-xxvi)

Yang (2011) argues that the vast majority of the optimization problems can be written in general as the following generic mathematical form:

(1)

$$\text{Minimize } x \in \Re^n \quad f_i(x), \qquad (i = 1, 2, \ldots, M), \tag{1}$$

$$\text{Subject to} \quad h_j(x) = 0, \qquad (j = 1,2, \ldots, J), \tag{2}$$

$$g_k(x) \leq 0, \ (k = 1,2, \ldots, K), \tag{3}$$

Where $f_i(x), h_j(x)$ and $g_k(x)$ are functions of the design vector

$$x = (x_1, x_2, \ldots, x_n)^T \tag{4}$$

In Equation 1. the components x_i of x represent the decision variables that can be continuous, discrete or a mix of both types. The functions $f_i(x)$ where $i = 1, 2, \ldots, M$ represent the objective functions to be optimized. The equalities and inequalities, $h_j(x)$ and $g_k(x)$, form constraints for the optimization problem. $\Re^n$ is a symbol for *search space*, meaning all the possible values for the decision variables. *Solution space* refers to the space formed by the objective function values. This generic form is for a minimization problem with “less or equal than” constraints, but this form applies also to maximization problems and “greater or equal than” constraints. Yang (2011, 78) Moreover, optimization models can be roughly divided into two groups: *linear-* and *nonlinear optimization models*. Linear models require that all the functions used in the model are linear, whereas nonlinear models require at least one of these functions to be nonlinear (Cottle & Thapa 2017, preface xxvi).

2.2 Solving an optimization model

The techniques available for solving optimization models can be divided into two groups illustrated in Figure 1: *algorithms* and *closed-form solution techniques* (Eiselt & Sandblom 2012, 6). Closed-form solution techniques can be used only for very simple and very small problems, whereas algorithms can be applied also to very large and complex problems. Real-world optimization problems rarely are neither simple or small, and thus optimization algorithms are very popular for solving real-world problems. Optimization algorithms typically create a sequence of *iterations* (trial solutions) that *converge* towards an optimal solution iteration by iteration. The number of iterations an optimization algorithm executes before terminating itself depends on its type. Some algorithms might have a pre-determined number of iterations after which the algorithm terminates (*finite algorithms*), whereas other algorithms might generate new iterations as long as a certain condition is not fulfilled (*convergent algorithms*) (Cottle & Thapa 2017, preface xxvii). Finding the optimum can require thousands or millions of iterations if the problem is very large in scale. This explains the need for high computing power computers for solving large scale optimization problems (Eiselt & Sandblom 2012, 6).

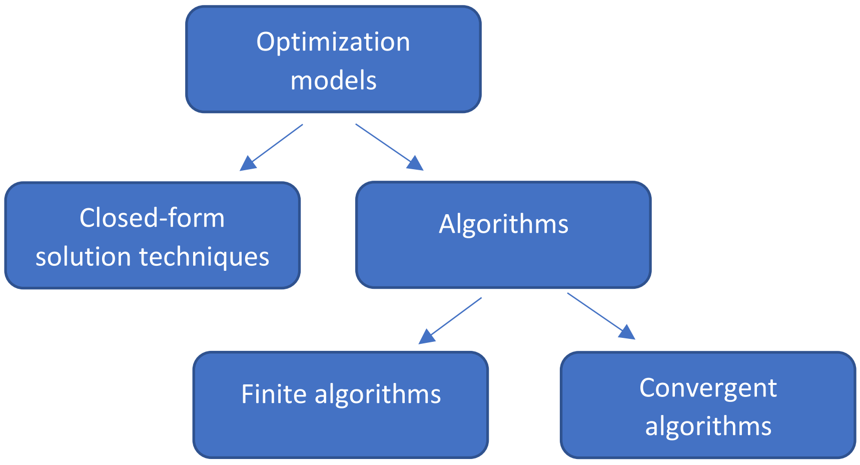


Figure 1. Solution techniques for optimization models.

The solutions that an algorithm can obtain on an optimization problem can be divided also into different groups. A solution that fulfills all the constraints the model includes is referred to as a feasible solution. (Srinivasan 2014, 11) On the contrary, if a solution violates one or more constraints, it is regarded as an infeasible solution (Taha 2007, 14). Moreover, optima can be divided into two categories: local- and global optima. These different types of optima can be easily understood with the help of a mountain range example. A mountain range can be thought of as the solution space of a maximization optimization problem, all mountain peaks representing optima. The peak of the highest mountain is the global optimum for the problem and all the remaining peaks represent local optima i.e highest points in restricted areas of the mountain range. Exact algorithms will find a global optimal solution for the model in bounded time. Approximate algorithms (also called heuristic- and metaheuristic algorithms) enable to find a relatively good solution with less effort, but these solutions are not necessarily always the global optima (Blum & Roli 2003, 269). Due to the enumerative nature of algorithms, exact algorithms are difficult to design with moderate computational effort as the complexity theory implies (Jiao, Zhang & Wang 2007, 177). If a model is remarkably large and complex, the exact algorithm might need a remarkably amount of time: weeks, months or even years, to find an optimal solution (Eiselt & Sandblom 2012, 417). Thus, if the answer to the problem is needed quickly, an approximate algorithm could be a more appropriate one for solving the problem.

2.3 Performance evaluation of approximate optimization algorithms

Because it can’t directly be determined how good a solution obtained from an approximate algorithm is, the performance of the given algorithm should be measured. This can be done for instance by utilizing simulation, which enables to specify an average or expected error bound of the approximate algorithm used. Another evaluation technique is to use theoretical error bounds, denoting the bounds that can’t be violated. For instance, some approximate algorithms might have a theoretical error bound which determines that the solution obtained from this algorithm can’t be more than 50% lower than the global optimum (Eiselt & Sandblom 2012, 423).

Wolpert & Macready (1997) developed an effective framework for identifying the best optimization algorithms for different optimization problems. They developed a theorem called *No Free Lunch* (NFL), which implied that if an algorithm has an increased performance on average over one class of optimization problems it will be offset by lower performance on average over the rest of the problem classes. (Wolpert & Macready 1997, 70) Also Yang (2011) points out that there is no superior algorithm for all optimization problems, but the choice of algorithm is very problem-specific. He argues that for large-scale nonlinear global optimization problems there are no guidelines that could determine what is the best-fit algorithm for a given problem. He even goes further arguing that it is not been verified that there would be an efficient algorithm available for very complex NP-hard (non-deterministic polynomial-time hard) problems (Yang 2011, 78).

2.4 Linear programming as an optimization technique

Linear programming (LP) is a widely used technique for solving optimization problems. The technique makes a foundation for many other optimization techniques and hence it is reasonable to examine this technique comprehensively before surveying more sophisticated techniques in the literature review. (Srinivasan 2014, 9)

Several people regard the development of linear programming as one of the most important scientific progression of the mid-twentieth century. LP is most commonly used to solve problems concerning the allocation of limited resources among competing activities in an optimal way. (Hillier & Lieberman 1990, 29) LP was the first optimization technique available, and it was later followed by non-linear programming, dynamic programming and several other techniques (Cottle & Thapa 2017, preface xxiv).

All equalities and inequalities in an LP model must be linear, and an LP-model must involve at least one inequality constraint. Moreover, LP-models have non-negativity restrictions meaning that the optimal solution can’t be derived from decision variables with negative values. (Cottle & Thapa 2017, 5; Srinivasan 2014, 9) The requirement of linearity implies that the LP model must fulfill four basic criterions: *proportionality, additivity, divisibility,* and *certainty*. Proportionality requires that the contribution of each single decision variable in constraints and objective function is directly proportional to the value of the variable. In other words: objective function and every constraint function must be linear. A linear function can be determined in following the way:

(2)

$$f(x_1, x_2, \dots x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

In Equation 2, c_1, c_2 and c_n represent constants and x_1, x_2 and x_n represent variables.

Additivity requires that the output of the constraints and objective function must be a direct sum of all the contributions of individual variables in the constraint functions and objective function. For instance, if there would be a function that expresses the relationship between sales amounts of two products and the total profits from these products and the total profits would equal the sum of profits from both products, the additivity requirement would be fulfilled. However, if the products would compete about the market share with each other, meaning that the increase in sales with product 1 would adversely affect the sales of product 2, the additivity criterion would not be fulfilled. (Taha 2007, 14-15)

The divisibility property denotes that all the variables in an LP model must be continuous. This requirement is often missing from real-world optimization problems (both linear and nonlinear), due to practical purposes. For instance, a car manufacturing company can't sell one and a half cars. Optimization problems that require at least one of the variables to have only integer values are denoted as *(mixed-) integer programs.* However, it is important to comprehend that also (mixed-) integer programming problems use LP as a solving technique. (Cottle & Thapa 2017, 16)

The requirement of certainty denotes that all the coefficients in the objective function and the constraints functions must be deterministic, meaning that they are known. However, often in real-world LP problems, the coefficients can't be determined as known constants because the data is uncertain. In these cases, the data is likely to be represented as probabilistic distributions, and the coefficients are average-value approximations derived from these distributions. Nevertheless, if the standard deviations in these distributions are not too large, the use of approximations is acceptable. (Taha 2007, 15)

2.4.1 Solving a linear programming problem

A linear program can be written down in several different ways. The representation of an LP model can range from representing it on an Excel spreadsheet to represent it with matrix theory and linear algebra. However, there is a standard form of an LP model, which is presented next.

(3)

$$\begin{gathered} Minimize \quad c_1x_1 + \cdots + c_nx_n \\ Subject\ to \\ a1_1x_1 + \cdots + a1_nx_n = b_1 \\ a2_1x_1 + \cdots + a2_nx_n = b_2 \\ \vdots \qquad\qquad \vdots \\ am_1x_1 + \cdots + am_nx_m = b_m \\ xj \ \geq 0, \qquad j = \ 1, \dots, n \end{gathered}$$

Equation 3 expresses a minimization LP problem, but the same form applies also to a maximization problem. This formulation is referred to as a standard form because all the variables are nonnegative and the constraints (equations beginning after the line “*Subject to*”) are represented as linear equations. An important aspect to consider is that LP models are not in this standard form in the beginning, because they have at least one linear inequality in the constraints. However, these types of models can be converted easily to the standard form. For instance, if the constraints include a “less or equal than” (≤) linear inequality, it can be converted to a linear equation by adding a nonnegative variable (referred to as a *slack variable*) to the left-hand side of the inequality. The slack variable shows how much unused resources there are left with the current solution, and it is required to be nonnegative. Moreover, it is important to notice that separate slack variables should be used for every single inequality in the constraints. (Cottle & Thapa 2017, 18-19) If an LP-model includes a “greater or equal than" (≥) constraint, a non-negative *surplus variable* must be subtracted from the left hand of this inequality in order to convert it to an equation (Taha 2007, 82-83).

The most famous algorithm for solving linear programming problems is the simplex algorithm which was initially introduced by George Dantzig in 1947 (Dantzig 1987,1-2). Before the simplex algorithm can be applied, the LP-problem must be converted to the standard form presented in Equation 3. If the standard form includes *m* equalities and *n* variables (including slack and surplus variables), the simplex algorithm iterates towards an optimum by setting at every iteration *n-m* variables equal to zero and then by solving the *m* equations for remaining *m* variables. The n-m variables which are set to zero are referred to as *non-basic variables* and the remaining variables with a value not equal to zero are called *basic variables*. A solution of basic variables obtained by solving the *m* equations is known as *the basic solution*. The algorithm usually begins by setting decision variables in the objective function as non-basic variables, thus obtaining the first basic solution. Then, the algorithm performs *a pivot,* an operation which exchanges the roles of basic- and non-basic variables. A pivot is performed at every iteration until an optimal solution has been found. (Taha 2007, 86-88)

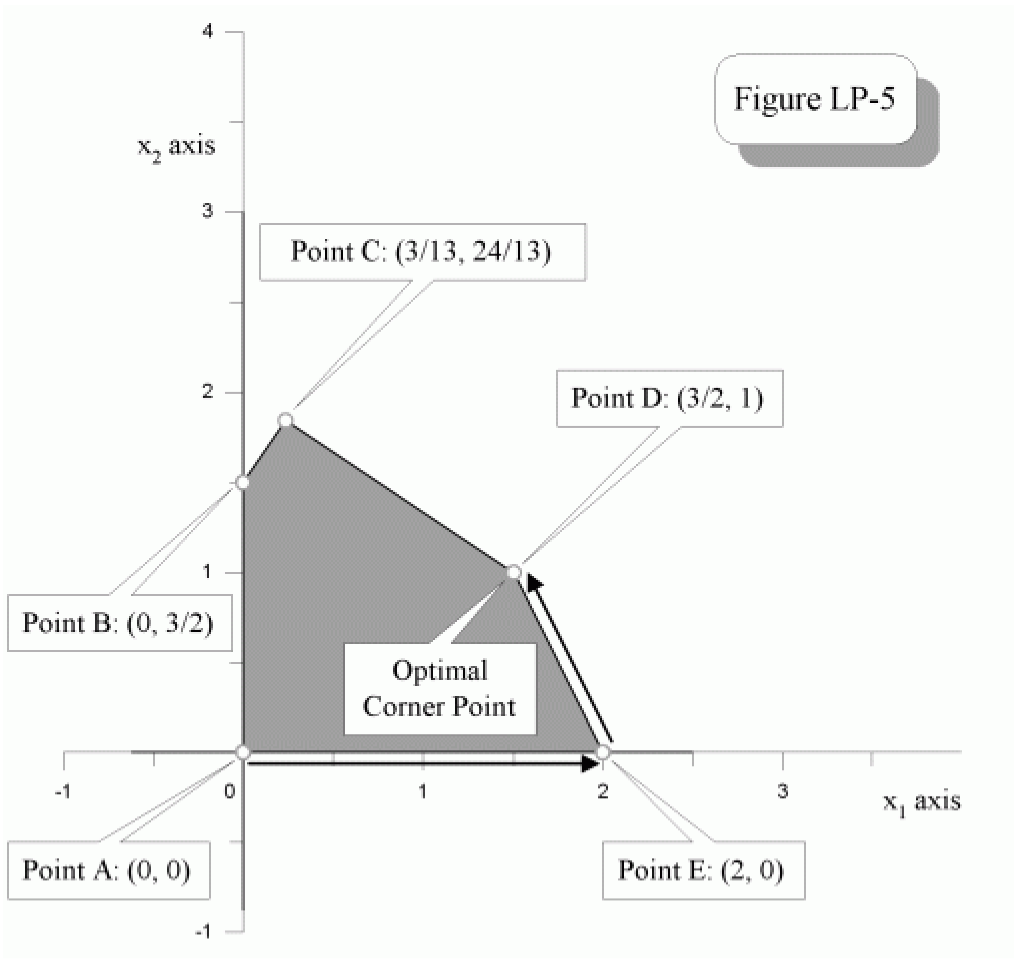


Figure 2. The behavior of the simplex. (The simplex method 2019)

A graphical representation of the behavior of the simplex can be found in Figure 2. Figure 2 exhibits an example where points A-E represent *corner point solutions*, the shaded area represents the *feasible region* and the arrows illustrate how the algorithm iterates from the initial basic solution (point A) to an optimum solution which is found from point D.

The feasible region for the problem will be obtained by plotting the constraints as lines to the search space. The plotted lines form borders for the feasible region, and every solution inside the borders are considered as a feasible solution for the given problem. The optimum will be found from one of the corner points from the feasible region. A corner point is a point where two constraint lines intersect with each other. At every pivot, the algorithm moves from a corner point solution to an adjacent corner point solution with the largest rate of improvement in the objective function value. The algorithm iterates from one corner point solution to its adjacent corner point solution with a better objective function value until there are no such points left, giving the algorithm a signal to terminate. This makes simplex very efficient computationally because it enumerates only selected corner points of the feasible region. An important thing worth noticing is that the simplex algorithm moves alongside the edges of the solution space, meaning that the algorithm can only move to adjacent corner points. (Taha 2007, 86-99)

2.4.2 Sensitivity analysis

The parameters of an LP model can be changed within certain limits without making any changes to the optimal solution obtained from the model. These limits can be determined by performing a *sensitivity analysis.* (Jansen, de Jong, Roos & Terlaky 1997, 15) The sensitivity analysis is a very useful tool for supplementing an LP-model because in many cases (especially with real-world problems) there is uncertainty associated with the input data, and the sensitivity analysis can be used to observe how sensitive the optimal solution is for these uncertainties. Sensitivity analysis can be also thought of as an evaluation of the robustness of the solution. As a rule of thumb, it can be considered that the more resilient the optimal solution is for the uncertainty, the more robust the solution is.

Sensitivity analysis for an LP model includes the following concepts: *shadow price, feasibility range, optimality range* and *reduced cost.* The shadow price is a sensitivity measure for constraints. This measure expresses how much the objective function value will change if the right-hand side of a constraint changes by one unit (increases or decreases). (Jansen et al. 1997, 16) Moreover, every constraint has an individual shadow price. The feasibility range specifies when the shadow price is valid meaning how much the right-hand side of a constraint can increase or decrease without changing the shadow price. The effect of simultaneous changes on several constraint right-hand sides must be calculated from the sum of all changes in proportion to their allowable changes. (Eiselt & Sandblom 2012, 79-84). The mathematical formulation of this will be presented next.

$$\sum_{k=1}^{n} \frac{\Delta b_k}{y_k} \tag{4}$$

Equation 4 presents the effect of simultaneous changes on n constraint right-hand sides. The change on a single right-hand side constraint is represented as Δb, and y represents the feasibility range. If the sum obtained from Equation 4 is greater than 100%, the shadow prices might change, however not necessarily. On the contrary, if the sum is less than 100%, the shadow prices will not change despite the several simultaneous changes. This is called the *100% rule*. (Eiselt & Sandblom 2012, 79-84)

The idea behind optimality ranges is very similar to one behind the feasibility ranges. Optimality range determines the interval of allowable increase and allowable decrease for the coefficients of every decision variable in the objective function for keeping the optimal solution valid. (Taha 2007, 140) Certainly, the objective function value will change if a decision variable coefficient changes, but the optimal solution will remain unchanged as long the change of a coefficient lies within the optimality range. Moreover, previously presented 100% rule applies also to optimality ranges (Eiselt & Sandblom 2012, 79-81).

Reduced cost specifies for a decision variable with a zero value in the optimal solution how much its coefficient should improve in order to get included in the optimal solution (Eiselt & Sandblom 2012, 84-85). Taking a product mix problem as an example: the reduced cost would show how much a product should improve its profitability (i.e. increase price or reduce costs ceteris paribus) per unit in order to get included in the optimal solution.

3 Literature review

The purpose of this chapter is to examine peer-reviewed articles concerning different optimization algorithms and their applications to problems similar to the one addressed in this thesis. Simplex algorithm for linear problems was presented in Chapter 2.4.2, and the goal of this literature review is to find other potential optimization algorithms that could be used for the profitability optimization tool that will be built in the empirical part. Either simplex or one of the algorithms presented in this chapter will be chosen for the tool. This chapter begins by introducing the methodology of the literature review. Next, the most promising optimization algorithms for the tool will be determined and theory and examples of applications for each of these algorithms will be presented.

3.1 Methodology of the literature review

All the searches for the literature were made in Scopus, which is the world's largest abstract and citation database of peer-reviewed literature including a collection of scientific journals, books and conference proceedings (Elsevier 2019). To achieve the goal of this literature review, a following four-step strategy was created and followed:

1. Create proper query strings for finding relevant articles from Scopus.

2. Collect every relevant article from the search queries and write down the algorithm used in the article and rank the relevance of the article with a "relevance-score" of 1-3 (1=min, 3=max).

3. Analyze which are the most popular algorithms and therefore the most potential for the optimization tool, and find high-quality articles that introduce the theory behind these algorithms.

4. Illustrate the applications of the most potential algorithms by presenting the most relevant papers where the algorithms have been utilized to solve optimization problems similar to the topic of this thesis.

The first search query used was: *TITLE-ABS-KEY (product AND portfolio AND planning AND problem AND optimization)*, which resulted in 54 papers. Every abstract of these 54 papers was examined. 17 papers of the total 54 were considered as relevant. A relevance-score was pointed to each paper and the optimization technique used was written down. The second search from Scopus was made with the search string *TITLE-ABS-KEY (product AND mix AND problem AND optimization)* which resulted in 317 articles. These articles were sorted based on their relevance. The abstracts of the first 100 articles were examined, and 21 of these were considered as relevant. The relevance scores and the optimization techniques used were marked in the same way that with the previous search string.

3.2 Results from the analysis of most potential algorithms

After a sufficient amount of relevant articles were collected, step 3 of the strategy was implemented. Results from the analysis concerning the frequency of different optimization algorithms and techniques used in the relevant articles are presented in Table 1.

Table 1. Frequency of different optimization algorithms and techniques used in the relevant articles.

Optimization technique/algorithm	**Frequency**
Genetic algorithm	13
Linear Programming (including variants)	13
Particle Swarm Optimization	3
Simulated Annealing	3
Tabu Search	2
Imperialist algorithm	1
Immune algorithm	1
Psycho-Clonal Algorithm	1
Markov decision process	1
Game Theory	1
Total	**39**

Table 1 suggests that the genetic algorithm was the most popular algorithm, and linear programming was the most used technique in the articles examined. The group linear programming includes also all the variants of basic linear programming, such as mixed-integer programming, fuzzy linear programming, and multi-objective linear programming. Moreover, a sharp-eyed reader might notice that the total number of different articles examined is 39 in the table, while the number of previously mentioned relevant articles in the result sheet was 38. This is so because one article used an algorithm which was a hybrid of simulated annealing and tabu search. Therefore, it was marked that the article used both of these algorithms, and this resulted in the total being 39 instead of 38 in the table. The most popular algorithms which were used multiple times in the relevant articles were: genetic algorithm, particle swarm optimization, simulated annealing, and tabu search.

The next step of the literature review strategy is to find high-quality papers which would explain the theories behind the most popular algorithms. For the particle swarm optimization, a paper written by Kennedy & Eberhart (1995) was found. An article written by Kirkpatrick, Gelatt & Vecchi (1983) was chosen for the simulated annealing. The article found for the tabu search was one written by Glover (1986), where he discussed the future paths for integer programming and links to artificial intelligence concerning the tabu search to be such area. For the genetic algorithm, a paper written by Forrest (1993) was found where he discussed the use of genetic algorithms for problem-solving (including optimization) and evolutionary systems modeling.

3.3 The most potential algorithms for the profitability optimization tool

This chapter will examine the four most potential optimization algorithms (in addition to the simplex) for the profitability optimization tool based on the analysis made in the previous chapter. These four algorithms are all nature-inspired *metaheuristics*, that have become increasingly popular in solving optimization problems. Thus, this chapter will begin with a short description of metaheuristics in general. After this introduction, the theories and examples of relevant applications will be presented for each algorithm.

3.3.1 Metaheuristics in general

The word *meta* means “beyond” or “higher-level” and according to Yang (2011) metaheuristics perform generally better on optimization problems compared to heuristics. All metaheuristic algorithms use a certain tradeoff between local search (referred to as exploitation) and global search (referred to as exploration), which enables them to perform efficiently on complex optimization problems. (Yang 2011, 79) Utilizing the previously presented mountain range example for global and local optima (can be found from Chapter 2.2) exploitation can be thought of as climbing a single mountain towards its peak, whereas exploration can be thought of as wandering in the mountain range and searching for promising high mountains. The right proportion of exploration and exploitation is crucial for a metaheuristic to perform efficiently on a global optimization problem. Furthermore, this proportion should not be fixed or only chancing in one direction (for instance a continuously increasing rate of exploitation), but it should rather be dynamical (Blum Roli 2003, 297).

If the overall level of exploration is too high compared to the level of exploitation, the algorithm is in danger of wandering aimlessly in the search space. On the other hand, if the level of exploitation is too high compared to the level of exploration, the algorithm easily converges quickly to a local optimum thus ignoring the rest of the search space and possibly better solutions. Moreover, a metaheuristic algorithm must have also an explicit criterion of how it selects the best solutions. The most common technique for this is the so-called *Survival of the fittest* which means that the algorithm updates the current best solution found so far in every iteration (Yang 2011, 84). Nevertheless, the distinction between heuristics and metaheuristics algorithms is still slightly fuzzy. There is no clear definition in the literature of what distinguishes a heuristic algorithm from a metaheuristic one, and some scholars use these both terms interchangeably. However, several researchers have categorized all stochastic algorithms with randomization and global exploration as metaheuristics (Yang 2011, 79). This definition will be also used in this thesis.

3.3.2 Genetic algorithm

The genetic algorithm (GA) is a metaheuristic which has been proven to be very effective in solving combinatorial optimization problems. The algorithm applies a probabilistic search technique which has got its inspiration from the Darwinian evolution and the principle of natural selection by the survival of the fittest. The genetic algorithm is a widely studied and used algorithm by both academics and practitioners. A vast majority of Fortune 500 companies are reported to use the genetic algorithms routinely to solve complex combinatorial optimization problems, and there is a vast amount of literature written on the topic including both scientific research articles and traditional books. The reasoning behind GA is that by combining good outcomes it is possible to find even better outcomes. (Yang 2011, 80)

The basic idea of GA is very simple. The optimization process of GA begins by creating an initial population of individuals (often randomly), which are stored as strings called *genotypes*. These genotypes are built from *bits*, which are in a binary number form in the simplest example. Each bit position in a genotype represents *a gene*. The algorithm begins to evolve this initial population towards an improved outcome by using the principles of *variation, selection,* and *inheritance,* which are inspired by the evolution in nature. (Forrest 1993, 872)

GA measures the “goodness” of the population by using a *fitness function*, a numerical value that determines how good a given solution is. Once the GA has created the initial population of individuals, it uses the fitness function to evaluate every individual in the population. The fitness value of the individuals will be used as the basis for the selection. Selection is a process where low fitness individuals are eliminated from the population. The selection will be followed by inheritance, which is implemented by making multiple copies of individuals with high fitness value. In addition to selection and inheritance, GAs strive to create a population of individuals with higher fitness values by applying genetic operators such as *mutation* and *crossover* to probabilistically selected individuals. The mutation is implemented by making changes to individual bits and the crossover is a process where two offspring are created by exchanging substrings of two individuals. (Forrest 1993, 873)

By using selection, inheritance, mutation, and crossover, the algorithm creates a new population of individuals that has a better than average chance to include good solutions for the given problem. When a GA has repeated this cycle of evolution over many generations, the overall fitness value of the population tends to improve, thus steering the algorithm towards the optimum iteration by iteration. (Forrest 1993, 873)

The above-presented description of the genetic algorithm is a general one, and there are still many details unspecified, meaning that the algorithm can be tailored differently to different problems. For instance, there are many ways the selection can be implemented. The algorithm could for example arbitrarily remove 40% of the population with the lowest fitness value individuals and then duplicate the remaining individuals or replicate the individuals proportionally to their fitness values. (Forrest 1993, 873)

Jiao et al. (2007) developed a genetic algorithm for solving a very complex product portfolio planning problem modeled as a mixed-integer program for a world-leading computer manufacturing company. They found out that the GA was an efficient tool for finding near-optimal solutions to this class of optimization problems. (Jiao et al. 2007, 1780-1797) Also Onwubolu & Muting (2001) applied a genetic algorithm to solve a complex product mix problem which was modeled as a linear program. Even though the problem was formed as an LP-model, it could not be solved with traditional LP-algorithms in a reasonable computing time due to its complexity. They found out that the GA applied was able to find very high-quality solutions for these types of problems within a reasonable short computing time (Onwubolu & Muting 2001, 1908).

3.3.3 Particle swarm optimization

Kennedy & Eberhart (1995) introduced Particle Swarm Optimization (PSO) in 1995 as a method for optimizing continuous nonlinear functions. They applied PSO to the training of artificial neural networks weights and evaluated the performance of the algorithm by applying it to a genetic algorithm test function called Schaffer's f6, which is an extremely nonlinear function that is very difficult to optimize. The PSO algorithm performed excellently on the performance evaluation and the application to the artificial neural networks.

The algorithm was discovered from a simulation project where the social behavior of a flying bird flock was simulated. The search technique of the algorithm mimics the navigation and foraging of a flock of birds and schools of fish. The main insight of PSO is that a single member (a bird or a fish) benefits from the discoveries and previous experiences of all the other members of the school or the flock when the group is executing some common mission such as searching for food. This means that working as a group is a better alternative compared to working individually for some animals and that social sharing of information among members might offer an evolutionary advantage to the whole group. These insights form the base of PSO's search strategy for finding the optimum. (Kennedy & Eberhart 1995, 1942-1944)

To explain the logic behind the algorithms search strategy, a few terms must be explained. Global best (*gbest*) refers to the point in the search space that has the best objective function value of all the points the *swarm* (the algorithm) has found so far. Moreover, every *particle* in the swarm has a point in the search space called personal best (*pbest*). This point is the point that has given the best objective function value of all the points a single particle has visited in the search space. When the swarm is moving in the search space, each particle "knows" values of three variables, which are pbest, gbest and current position in the search space. These three variables mixed with a random variable *rand()* determines how the swarm moves in the search space. The variable rand() forces the swarm to perform also some stochastic search (exploration) in the search space, making PSO a (partly) stochastic optimization algorithm. (Kennedy & Eberhart 1995, 1943-1946)

The mathematical definition of PSO is simple. The algorithm begins by randomly initializing the starting position of the swarm in the search space. After the initial positions of all particles are defined, the particles begin to move in the search space trying to find the global optimum. The direction and intensity of the movement of the particles in the search space are defined by a vector called *velocity*. The updated position of the particle can be mathematically determined for every iteration in the following way:

$$vx[\,][\,] = vx1\,[\,][\,] + 2 * rand(\,) * (pbestx[\,][\,] - presentx[\,][\,]) + 2 * rand(\,) * (pbestx[\,][gbest] - presentx\,[\,][\,]) \tag{5}$$

In Equation 5, vx1[][] is the current velocity vector of the particle, rand() is a random number in the interval of 0 and 1, presentx[][] is the current position of the particle in the search space, pbestx[][] is the position of the personal best of the particle in the search space and pbestx[][gbest] is the position of the best solution the whole swarm has found so far from the search space. (Kennedy & Eberhart 1995, 1943-1946)

PSO has become a very popular algorithm for solving complex optimization problems. After this first publication by Kennedy & Eberhart (1995), more than 20 variants of the original PSO have been developed, indicating that the basic idea behind the algorithm is very usable for different optimization problems (Yang 2011, 81).

Rezaie, Nazari-Shirkouhi & Ghodsi (2010) applied a PSO algorithm to a product mix optimization problem and compared its performance against other optimization methods including TOC heuristic, revised TOC heuristic, integer linear programming, tabu search and hybrid tabu–simulated annealing algorithm, of which tabu search and simulated annealing will be introduced in the upcoming chapters. The results revealed that the proposed PSO performed slightly better compared to these other algorithms. (Rezaie et al. 2010, 6483-6490) Also Zheng & Yu (2013) utilized PSO in solving a product mix decision problem. They applied a revised multi-layer embedded PSO metaheuristic to the problem. Lastly, this algorithm was compared against a genetic algorithm, and the results showed better performance for the PSO on this specific problem (Zheng & Yu 2013, 782).

3.3.4 Simulated annealing

The idea behind simulated annealing was presented first time in Kirkpatrick, Gelatt & Vecchi (1983). The algorithm is inspired by statistical mechanics, which is the central discipline of condensed matter physics. Kirkpatrick et al. (1983) used central constructs from this field and applied these concepts to combinatorial optimization problems. Annealing (which is one target of research in this field) is a technique used to make strong metal items by first melting the metal at very high temperature, and then by cooling the temperature slowly down, and keeping the temperature a long time close to the freezing point. Kirkpatrick et al. (1983) applied a simple algorithm created by Metropolis, Rosenbluth, Rosenbluth, Teller & Teller (1953) for solving combinatorial optimization problems. The algorithm in question is related to annealing and it is used to simulate the thermal motion of atoms in thermal contact with a heat bath at temperature T.

Key concepts of simulated annealing are *energy, temperature,* and *annealing schedule* which will be explained next. Energy is the value of the objective function, describing the goodness of the solution. Temperature determines how much exploring and how much exploiting the algorithm performs. The temperature is a function of time, and its value decreases iteration by iteration all the way down to zero which is a signal to the algorithm to terminate. When an algorithm performs exploring, it must be able to accept also a solution that is worse than the current solution found. Without this attribute, the algorithm would just perform “hill-climbing” (only accepting solutions better than the current best one) in one region of the search space. When the temperature is high, simulated annealing will accept more frequently solutions that are worse than the current solution found, thus performing more exploring. As the temperature decreases the algorithm will accept less frequently solutions that are worse than the current best solution, thus performing more exploiting and converging towards an optimum. The annealing schedule determines how many times the algorithm should iterate before lowering the value of the temperature variable. (Kirkpatrick et al. 1983, 672-673)

As with the genetic algorithms and particle swarm optimization, the search process of simulated annealing is relatively simple, and it is based on just a few rules. The algorithm follows the steps and rules presented in Figure 3 when searching for an optimum for a given minimization problem.

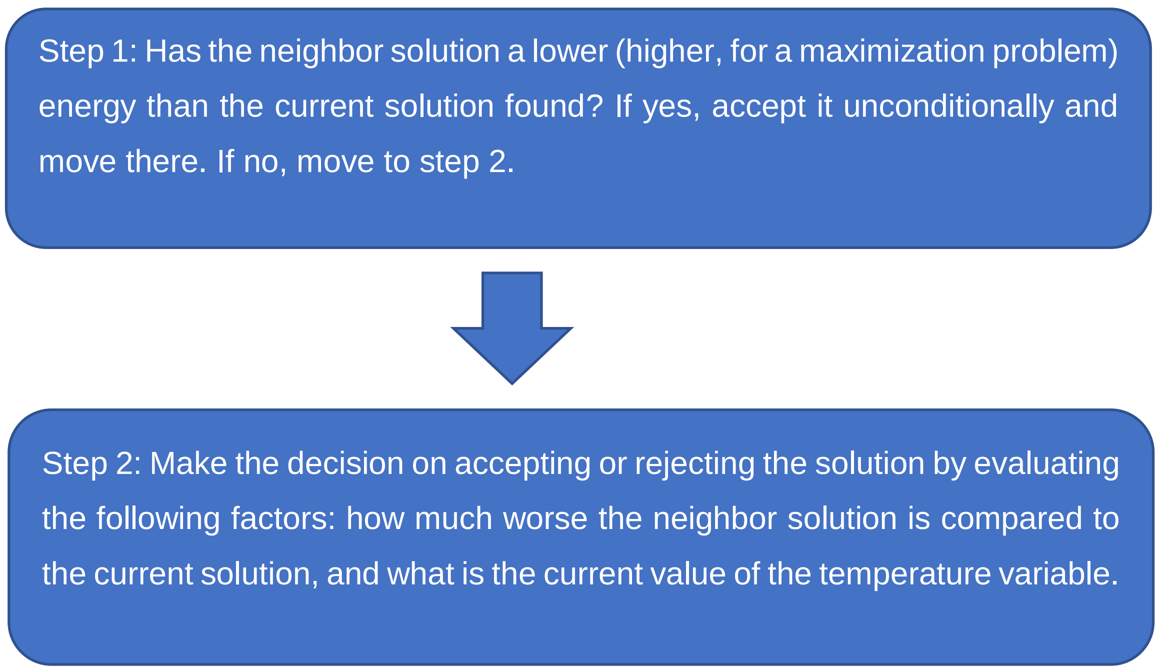

Step 1: Has the neighbor solution a lower (higher, for a maximization problem) energy than the current solution found? If yes, accept it unconditionally and move there. If no, move to step 2.
Step 2: Make the decision on accepting or rejecting the solution by evaluating the following factors: how much worse the neighbor solution is compared to the current solution, and what is the current value of the temperature variable.

Figure 3. The search process of simulated annealing.

In step 2, the decision on whether to accept or reject the neighbor solution is treated probabilistically. Higher the temperature, the higher the chance of the solution getting accepted. Lower the difference of energy between the neighbor solution and the current solution, the higher the chance of the solution getting accepted. The probability that the neighbor solution is accepted in step 2 is can be written in mathematical terms as follows:

$$P(\Delta E) = \exp(-\frac{\Delta E}{k_B T}) \tag{6}$$

In Equation 6, ΔE represents the difference between the energy of the current solution and the energy of the neighbor solution, k_B is the Boltzmann factor and T is the temperature. $P(\Delta E)$ is compared against a random number that is uniformly distributed in the interval of 0 and 1. If $P(\Delta E)$ is greater than the random number, the neighbor solution is selected and if $P(\Delta E)$ is less than the random number, the neighbor solution is rejected. The algorithm iterates this way towards the final solution and lowers the temperature according to the annealing schedule until the temperature reaches zero and the algorithm terminates. (Kirkpatrick et al. 1983, 672-673)

Chaharsooghi & Jafari (2007) applied the simulated annealing algorithm on multiple problems of different sizes to identify and determine optimal mixes of products and their quantities according to available resources. The performance of SA on large scale problems was compared against the performance of the genetic algorithm and tabu search. SA outperformed these two metaheuristics on 5 problems out of 6. (Chaharsooghi & Jafari 2007, 233-235) These results suggest that SA is a very potential metaheuristic for large scale product mix problems.

Sadeghi, Alem-Tabriz & Zandieh (2011) used the simulated annealing algorithm to determine an optimal product portfolio that would maximize the market share of the company while minimizing the engineering costs associated with the product portfolio. They stated that the diversity of a product portfolio should be on a level where its engineering costs do not exceed the advantages derived from an increased market share (Sadeghi et al. 2011, 2327-2328). The article included a case study where the SA algorithm was applied to a real-world notebook computer planning problem for a world-leading computer manufacturing company, similar to the problem addressed in Jiao et al. (2007) presented in Chapter 3.2.2. Sadeghi et al. (2011) compared the performance of their SA against the performance of the GA presented in Jiao et al. (2007). Both algorithms were applied to problems of different sizes derived from the above-mentioned notebook computer product mix problem. SA outperformed GA in all the problems examined in the paper (Sadeghi et al. 2011, 2338-2347).

3.3.5 Tabu search

Tabu search (TS) is one of the most cited and used metaheuristics in both industry and academia for solving combinatorial optimization problems. The idea behind the algorithm was initially introduced in an article written by Glover (1986), which was inspired in turn by his earlier ideas in Glover (1977). (Blum & Roli 2003, 275)

TS handles the problem of local optimum solutions by using a strategy that forbids the selection of certain solutions by marking these solutions as *tabus*. The key idea with the TS is that the only case when a worse solution should be accepted is when there are no other undiscovered paths nearby in the search space. In more simple terms; it is better to select a bad solution than to select an already visited solution if there is a possibility that this bad solution might lead to a path of better solutions in the coming iterations. (Glover 1986, 541)

Important concepts of the TS that will be introduced next are *tabu list, tabu tenure, solution attributes, tabu conditions, allowed set,* and *aspiration criteria.* TS in its simplest form utilizes a best improvement local search as the base of its search strategy. The algorithm uses tabu lists as its short-term memory to avoid getting stuck in local minima. Tabu lists register the most recently visited solutions and prohibits the algorithm to re-visit these solutions. The lengths of the tabu lists are referred to as tabu tenures and they control the relationship of exploration and exploitation of the algorithm. If the tabu tenures are relatively small, the algorithm will concentrate on the small areas of the search space thus performing more exploiting. On the contrary, relatively large tabu tenures force the algorithm to perform more exploration and to search from larger areas of the search space. Tabu tenures do not have to be fixed, and they can change as the algorithm iterates towards an optimum. It would be inefficient to manage a tabu list containing complete solutions, and hence only solution attributes are stored in the tabu lists. Solution attributes can refer for instance to components of solutions, moves, or difference between two solutions. Moreover, each attribute has its own tabu list, which forms the tabu conditions for the algorithm. Tabu conditions are used to create the allowable set, which determines the solutions that can be selected. However, storing attributes instead of complete solutions to tabu lists is a double-edged sword, because one attribute can be associated with many solutions. This might cause a situation where unvisited solutions of high quality would not be included in the allowed set, which of course is an undesirable outcome. Hence, aspiration criteria are defined to solve this problem. Aspiration criteria allow the acceptance of a solution to the allowed set even though it would be forbidden by tabu conditions. The most popular aspiration criterion allows the algorithm to accept a solution forbidden by the tabu conditions if it is better than the current best solution found. (Blum & Roli 2003, 275-276; Glover 1986, 541-544) A summary of the key concepts is presented in Table 2.

Table 2. A summary of key concepts of tabu search.

Concept	**Content**	**Purpose**
Tabu list	Short term memory of visited solutions	To avoid local optimum solutions
Solution attributes	Attributes associated with certain solutions	To enhance management of the tabu lists
Tabu conditions	Tabu lists of different solution attributes	To create the allowed set
Tabu tenure	Length of the tabu lists forming the tabu conditions	To tune the proportion between exploitation and exploration
Allowed set	Solutions not included in the tabu conditions	To avoid local optimum solutions
Aspiration criteria	Rules for accepting certain solutions forbidden by tabu conditions	To prevent the exclusion of unvisited high-quality solutions from the allowed set

TS iterates towards an optimum by selecting the best solution from the allowed set at each iteration. Always after a movement to a new solution, the tabu list will be updated by adding this solution to the tabu list and by deleting one of the previous solutions from the list (usually following a FIFO order). TS moves in this way in the search space and terminates when a termination condition is met or when the allowed set is empty. (Blum & Roli 2003, 275-276; Glover 1986, 541-544)

A good illustration of how the tabu search has been applied to a product mix problem is presented in Onwubolu (2001). He introduced a TS based theory of constraints (TOC) heuristic algorithm that can be used for solving product mix problems of different sizes encountered in industrial companies. TOC is a management philosophy that strives to maximize profits in a manufacturing company that has a demonstrated bottleneck such as limited resources. He developed this algorithm because the original TOC heuristic was considered to yield unrealizable solutions in a situation where a manufacturing plant has several resource constraints. The TS based algorithm developed was successfully applied to small-, medium- and large size product mix problems typical in the manufacturing industry. One thing worth noticing is that the algorithm managed to successfully solve large size problems for which appropriate methods had not been developed in terms of feasibility in computational times. This implies that TS based algorithms are very potential alternatives for solving large size product mix problems. (Onwubolu 2001, 2065-2066, 2073-2075)

Another interesting example of how the tabu search has been utilized in solving product mix decision problems is when Mishra, Prakash, Tiwari, Shankar & Chan (2005) developed a metaheuristic by combining best parts from tabu search and simulated annealing. They applied this algorithm to a multi constraint theory of constraint (TOC) product mix problem. The problem in question is computationally a complex one and it has a very large search space. Moreover, the hybrid tabu-SA algorithm was compared against the original tabu search, original simulated annealing, integer linear programming, TOC heuristic, and Revised-TOC heuristic. All the algorithms were applied to the same problem, and the results revealed that the tabu-SA hybrid was superior compared to the other algorithms. (Mishra et al. 2005, 446-454)

3.4 Conclusions from the literature review

The goal of this literature review was to find the most potential optimization algorithms (in addition to simplex) for the profitability optimization tool and to present the theory and examples of relevant applications of these algorithms to product portfolio optimization problems. The level of potential of a given algorithm was assessed by the frequency it was used in relevant articles. The most potential algorithms that were used multiple times were: genetic algorithm, particle swarm optimization, simulated annealing, and tabu search. The genetic algorithm was used in 13 articles thus being the most popular. Both particle swarm optimization and simulated annealing were used in three articles and tabu search was used in two articles. Either simplex or one of these four algorithms will be chosen for the profitability optimization tool that will be built in the next chapter. This literature review also revealed that there is a research gap concerning the topic of this thesis as none of the relevant articles concerned product portfolio optimization problems in the meat processing industry.

Many of the articles presenting the applications of the algorithms made performance comparisons between different algorithms. The results from these comparisons supported the viewpoints of Wolpert & Macready (1997) and Yang (2011) meaning that there is no superior optimization algorithm available that would outperform all other algorithms on all problems. This means that all the five potential algorithms must be mirrored carefully against the problem of the case company, and the decision on the best-fit algorithm for the tool must be done based on this analysis.

4 Modeling the problem and creating the profitability optimization tool

The purpose of this chapter is to build a profitability optimization tool for a case company operating in the meat processing industry. The tool will be build by utilizing the theoretical concepts introduced in the previous chapters. The chapter will begin by presenting the company and the problem to be solved. This will be continued by describing the modeling process and illustrating how one of the subproblems was modeled. The next subchapter presents how the optimization algorithm for the illustrated submodel was selected. Then, the submodel will be solved and scaled up by adding more attributes of the problem to it. The next subchapter discusses model applications with examples of how the tool can be used for performing different simulations. Lastly, the model interface will be described.

4.1 Background

The case company is a large meat processing company that produces meat products for both B2B and B2C customers. The company purchases the largest proportion of its meat raw materials as whole animals which must fulfill certain standards. The availability of these types of animals is strictly limited and the company purchases currently all the available animals which fulfill the standards. The company has currently a wide product portfolio including over 200 end products.

A single part of the animal can be used for multiple end products, and many of the end products can be produced from several different combinations of different single parts, as long as certain criteria concerning factors such as fat content, type of meat, the proportion of saturated fat and connective tissue protein content are fulfilled. As an imaginary example: a certain minced meat product could be produced from any combination of different raw materials as long as following main criteria are fulfilled: the end product must contain 40% beef and 60% pork, the overall fat content must be 20% and the proportion of saturated fat can’t be higher than 20%. All end products have a limited demand in the current market situation. Moreover, some products have minimum quantities that must be produced according to contracts made with the retailers, according to strategic decisions made or in order to create stability in the market. This means that the portfolio must have a certain level of product diversity to consume all the raw materials purchased and to not violate the minimum quantities. Moreover, as the company purchases whole animals, a certain level of product portfolio diversity must be naturally maintained to use all the available raw materials.

4.2 Problem description

Due to previously mentioned factors, reasoning an optimal profit-maximizing product portfolio becomes challenging. Hence, the company management decided that a profitability optimization tool should be created. The tool should be able to support both tactical-, and strategic product portfolio planning by optimizing a portfolio from a given amount of raw materials simultaneously taking the market demands and other constraints into consideration. Moreover, the tool should be able to simulate different scenarios, such as the effects of changes in market demands or the effect of different strategic decisions on the optimal product portfolio. Technical requirements from the company for the optimization tool were the following:

1. The tool must be easy to understand and use (ideally usable directly from Excel).
2. Inputs, calculations, and results should be in separate sections.
3. The tool should be able to optimize a product portfolio including 200 products and 30 raw materials.
4. The user should have a possibility to give minimum, maximum and equality constraints for each product.
5. The tool must be compatible with the current BI-software enabling quick and easy price- and cost updates for the products.
6. The results must be clearly presented and easily saved after each run.

4.3 Modeling process

The modeling process began by modeling a small subproblem which will be presented next. The logic behind this sub-model was used as a basis for the larger model build for the tool. All the numbers in this descriptive example are manipulated, but the logic behind the modeling is the same as in the real submodel build.

For illustration one of the subproblems, shown in Figure 4, included three B2C end products: a sausage, sparerib and pork neck. The sausage requires three raw materials which are P20 (pork with 20% fat), P35 (pork with 35% fat) and B10 (beef with 10% fat). P20, P35, and B10 can be produced from several parts of the animal as long as the main criteria concerning the type of meat (pork or beef) and fat content are fulfilled. Whole meat products will be the same as their raw materials (e.g. pork neck requires only pork neck and sparerib only sparerib as raw material). Pork neck and sparerib can be also processed to P20 and P35. Moreover, P20, P35, and B10 can be also sold as raw materials to other companies. Thus, all end products except sausage are simultaneously raw materials.

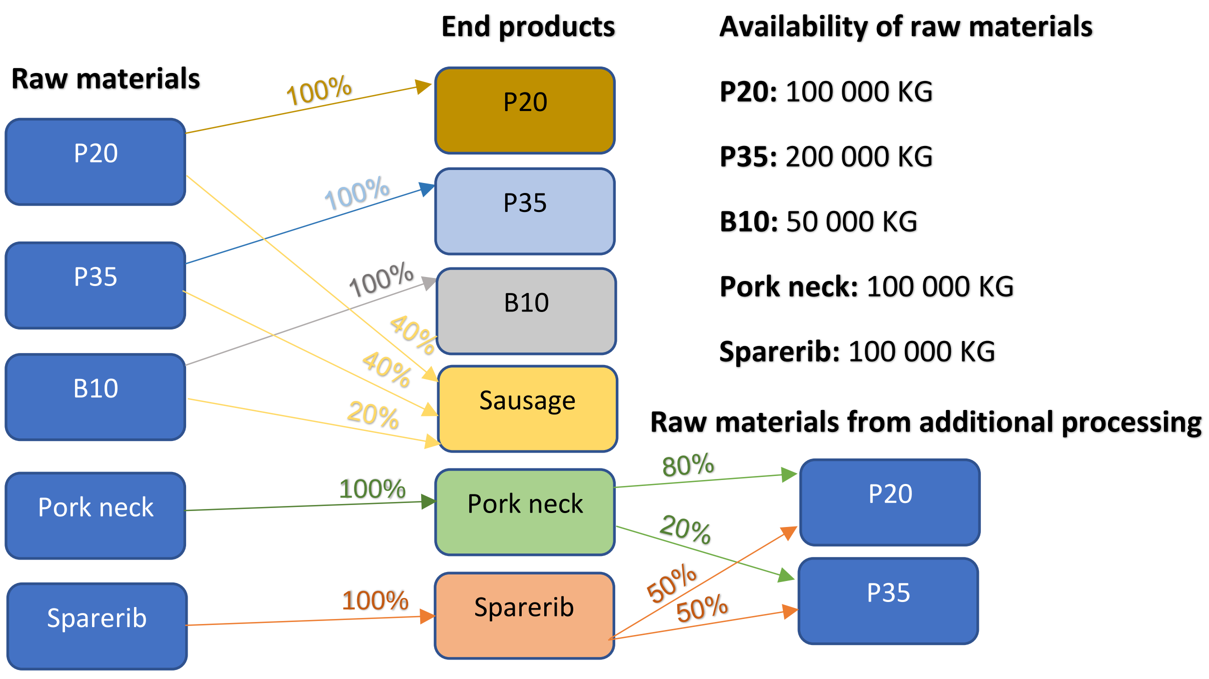
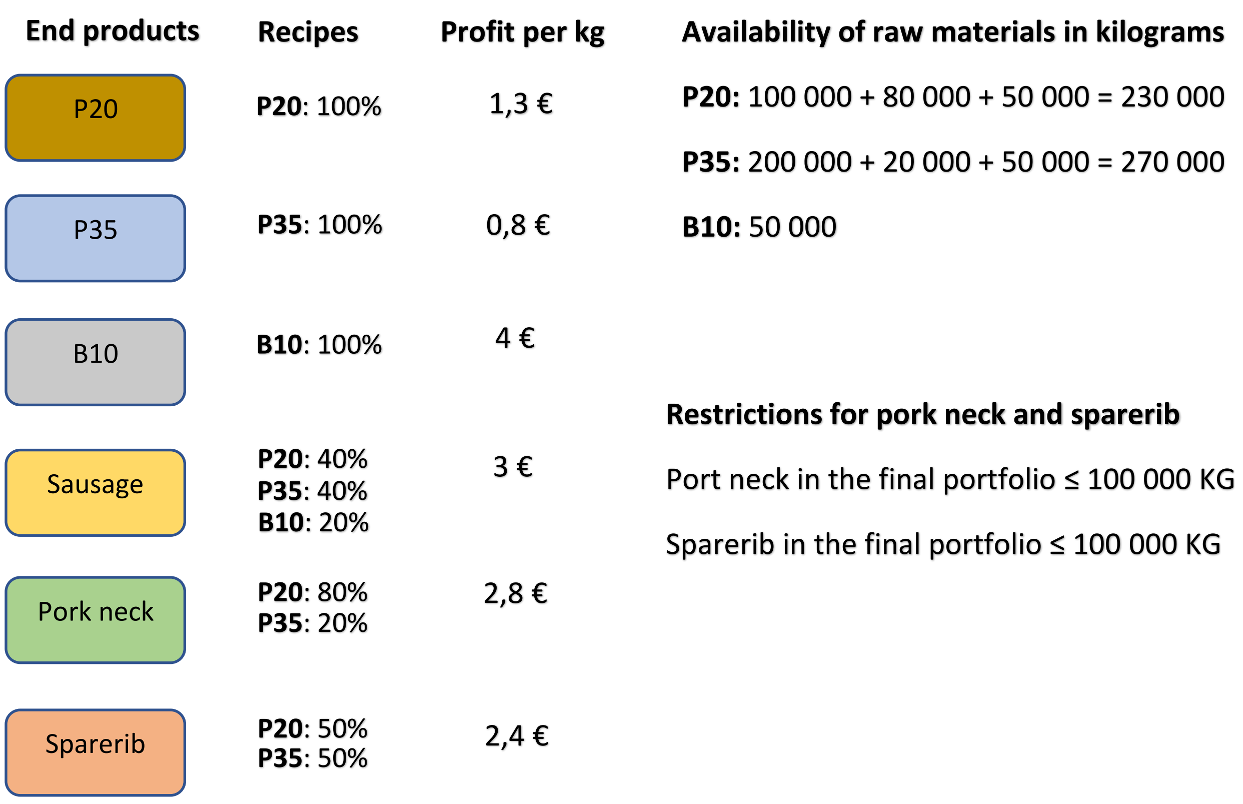


Figure 4. Logic between raw materials and end products in the subproblem.

To model the possibility that pork neck and sparerib can be processed to P20 and P35 three things were done. Firstly, the recipes of pork neck and sparerib were converted to consist of P20 and P35. Secondly, the availabilities of P20 and P35 were increased with the amount equal to processing all the pork neck and sparerib to P20 and P35. Thirdly, the amount of pork neck and sparerib in the final portfolio were both limited to 100 000 kilograms. Figure 5 illustrates graphically the submodel build.



End products	Recipes	Profit per kg
P20	**P20**: 100%	1,3 €
P35	**P35**: 100%	0,8 €
B10	**B10**: 100%	4 €
Sausage	**P20**: 40% **P35**: 40% **B10**: 20%	3 €
Pork neck	**P20**: 80% **P35**: 20%	2,8 €
Sparerib	**P20**: 50% **P35**: 50%	2,4 €

Availability of raw materials in kilograms

P20: 100 000 + 80 000 + 50 000 = 230 000

P35: 200 000 + 20 000 + 50 000 = 270 000

B10: 50 000

Restrictions for pork neck and sparerib

Port neck in the final portfolio ≤ 100 000 KG

Sparerib in the final portfolio ≤ 100 000 KG

Figure 5. A graphical presentation of the submodel.

The next step was to convert this model into a mathematical form. The following linear program was created:

$$
\begin{aligned}
Max\ Z = \ & 1{,}3x_1 + 0{,}8x_2 + 4x_3 + 3x_4 + 2{,}8x_5 + 2{,}4x_6 \\
& Subject\ to \\
& x_1 + 0{,}4\,x_4 + 0{,}8x_5 + 0{,}5x_6 \ \le 230\,000 \\
& x_2 + \ 0{,}4x_4 + 0{,}2x_5 + 0{,}5x_6 \ \le 270\,000 \\
& x_3 + \ 0{,}2x_4 \ \le 50\,000 \\
& x_5 \ \le 100\,000 \\
& x_6 \ \le 100\,000 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \ge 0
\end{aligned} \tag{7}
$$

Variables x_1, x_2, x_3, x_4, x_5 and x_6 in Equation 7 are continuous and they represent the amounts of P20, P35, B10, sausage, pork neck and sparerib produced in kilograms respectively.

4.4 Selecting the algorithm

After converting the submodel into a mathematical form, the next step was to apply an optimization algorithm to solve it. As the company required that the tool should run in Excel if possible, the first try to solve this submodel was done with Excel. The basic version of Excel Solver includes three optimization algorithms which are: simplex, evolutionary and GRG nonlinear. Simplex was presented in chapter 2.4.1 and evolutionary is based on the genetic algorithm presented in chapter 3.2.2. Hence, simplex and evolutionary were potential alternatives. As the problem is modeled as an LP-model, both of the algorithms could be applied to it. When opening Excel Solver, it suggests choosing simplex for linear problems and evolutionary for nonlinear problems that are non-smooth.

A solution obtained from the simplex algorithm is always a global optimum, whereas evolutionary can only guarantee a “good” solution (Frontline Systems 2019b). Hence, simplex would be applied first to the problem. If simplex would not be able to find a solution in a reasonable computational time, evolutionary would be applied as the second attempt to solve the problem with Excel Solver. If evolutionary would fail too, the next step would be to attempt to solve the problem by using tabu search, simulated annealing or particle swarm optimization with a suitable programming language such as R or Matlab.

4.5 Scaling up the preliminary model

Simplex found immediately a globally optimal solution for the subproblem with a normal corporate laptop. The solution was found after 5 iterations. The solution for the submodel was the following:

$$\begin{aligned} Z &= 1\,350\,000 \\ x_1 &= 0 \\ x_2 &= 100\,000 \\ x_3 &= 0 \\ x_4 &= 250\,000 \\ x_5 &= 100\,000 \\ x_6 &= 100\,000 \end{aligned} \tag{8}$$

Variables x_1, x_2, x_3, x_4, x_5 and x_6 in Equation 8 represent the amounts of P20, P35, B10, sausage, pork neck and sparerib produced in kilograms respectively. Variable Z represents the objective function value (total profits form the product portfolio).

After solving the simple submodel presented, the next step was to start expanding it by adding more decision variables, constraints, and attributes of the problem to it. Price elasticity was one main attribute that was not included in the submodel. The selling prices are not fixed, and often the case company must lower the selling price if it wants to increase sales on a certain product. The selling prices do not change linearly or with constant intervals, but the price elasticity is more depended on the individual negotiations with the retailers, and thus it is impossible to model mathematically in an accurate way. This problem was solved by utilizing the sensitivity analysis presented in Chapter 2.4.2. The optimality ranges of the sensitivity analysis provide vital information concerning how much the prices can change without making changes in the optimal portfolio. Thus, after optimizing the product portfolio, the sensitivity analysis should be presented to the sales managers so that they would know the price ranges within they can operate when negotiating contracts with the retailers. If they could not stay within these price ranges in the negotiations, a new optimization with the new suggested prices should be done to observe how they affect the optimal portfolio.

The shadow price for a raw material constraint provides additional useful information as it reveals how much total profits would increase (decrease) if the availability of the raw material would increase (decrease) by one kilogram, and the feasibility range specifies when the shadow price is valid. As an example: a feasibility range with an allowable increase of 15 and an allowable decrease of 5 would mean that the shadow price would change if the raw material availability would increase more than 15 kilograms or decrease more than 5 kilograms. The effect of simultaneous changes on several raw material availabilities must be calculated from the sum of all changes in proportion to their allowable changes according to Equation 4. introduced in Chapter 2.4.2. Moreover, the reduced cost illustrates how much the products not included in the optimal portfolio should improve their profitability to get included.

As the final model is a tool, the number of decision variables and constraints are not fixed but they change according to the portfolio to be optimized. The final model includes slots for 200 decision variables (different products), possibility to give maximum, minimum and equality constraints to each of the decision variables, slots for 30 availability (maximum) constraints for different raw materials and a possibility to give maximum constraints for 20 different product groups.

In order to model that several products can be either sold as they are or be processed to raw materials for other products, the recipes were modeled in the same way as with the subproblem presented previously. This resulted in 7 different raw materials that form all the recipes. Each of the seven raw materials has two attributes: overall fat content and the type of meat (pork or beef). These are the most crucial attributes of the raw materials that had to be incorporated into the model. This is a simplification of reality, as there are also other factors that define a single raw material as described previously. However, this simplification had to be made in order to model this attribute of the problem effectively. Moreover, the experts in the company considered that the results obtained from the tool are valid and applicable to decision making despite this simplification made. In this way, the complex system of thousands of possibilities of processing one product to another was modeled in a simple way.

4.6 Model applications

The tool can be also used for performing simulations. The simulations can be conducted by manually chancing selling prices- or costs of the products, by adding new products to the tool that do not exist yet or by giving maximum, minimum and equal to constraints for different products and product groups.

As an example: if the case company is considering launching a new product, the tool can be utilized to observe how the new product would affect the portfolio and the overall profitability. Moreover, the tool could be used for pricing a new product, as it would illustrate how high the price should be to incorporate the new product into the optimal portfolio. The tool could be also used to simulate the effects of different strategic decisions. For instance, if the company would consider a strategy focusing on increasing the sales of sausages, different outcome scenarios of this strategy could be inputted to the tool, and the effects of each scenario on overall profitability and the optimal portfolio could be observed. These different outcome scenarios could differ for instance by selling prices and/or market demands. As the last example, manual price change simulations could be utilized to determine the selling prices when a product (with an option to process it to raw materials for other products) should be sold directly to customers, and when it should be processed to raw materials.

4.7 Model interface description

Next, a solid interface for the model was built. The tool was divided into seven different Excel sheets. The tool is operated by importing BI-software data (copy-paste) into the interface, defining variable values and then clicking the “Optimization”-button which runs the optimization algorithm. The sheets were the following:

1. *Inputs for products, raw materials, and constraints.* By inputting product names, product codes and product group names the tool searches automatically selling prices and costs associated with each product from the sheets containing data from the BI-software. Product-, raw material- and product group constraints will be inputted to this sheet. Moreover, this sheet provides the possibility to modify the selling prices for simulation purposes.

2. *Selling price data from the BI-software.* The selling prices can be updated by downloading the respective data from the BI-software and by copy-pasting it to this sheet.

3. *Cost data from the BI-software.* The costs of the products can be updated by downloading the respective data from the BI-software and by copy-pasting it to this sheet. Moreover, this sheet provides the possibility to modify the costs for simulation purposes.

4. *Recipes for all products.* This sheet includes recipes for all products with a possibility to modify the recipes for simulation purposes.

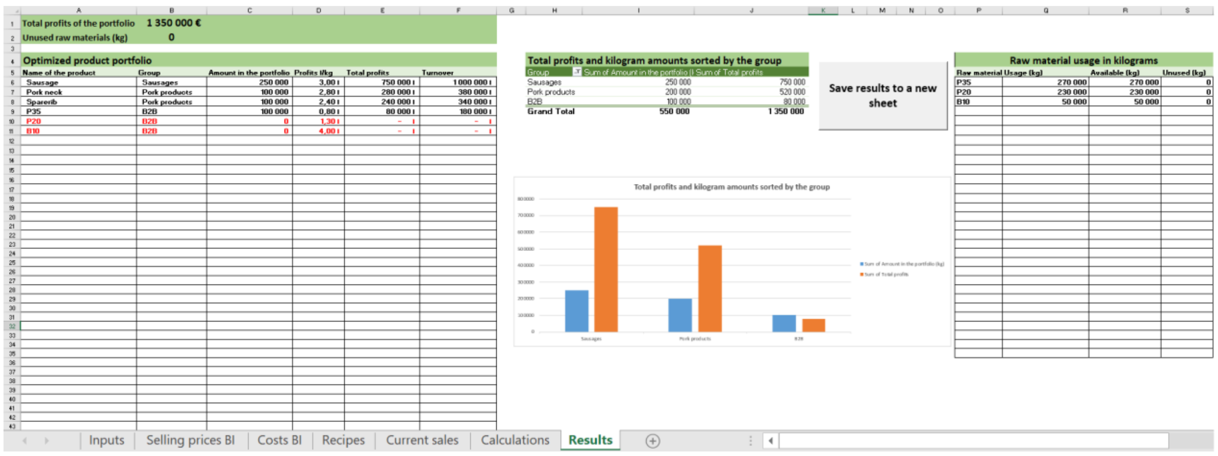
5. *Year to date sales of each product in kilograms.* The historical sales data can be inputted to this sheet by downloading respective data from the BI-software and by copy-pasting it to this sheet. The tool will connect automatically the data

to each product in a way that that the user can compare the historical sales volumes of different products with the optimized portfolio.

6. *Calculations.* This sheet includes all the calculations that the tool performs when optimizing a portfolio.

7. *Results.* This sheet includes results concerning the optimized portfolio, profitability of different product groups and raw material usage.

Sheet 7 for results is illustrated in the following figures. Figure 6 shows the whole results sheet. Figure 7 presents the left part of the sheet including results concerning the optimized portfolio. Figure 8 illustrates the right part of the sheet including the results by product group, raw material usage and the macro which saves the result.



Total profits of the portfolio	1 350 000 €
Unused raw materials (kg)	0

Optimized product portfolio

Name of the product	Group	Amount in the portfolio	Profits €/kg	Total profits	Turnover
Sausage	Sausages	250 000	3,00 €	750 000 €	1 000 000 €
Pork neck	Pork products	100 000	2,80 €	280 000 €	380 000 €
Sparerib	Pork products	100 000	2,40 €	240 000 €	340 000 €
P35	B2B	100 000	0,80 €	80 000 €	180 000 €
P20	B2B	0	1,30 €	- €	- €
B10	B2B	0	4,00 €	- €	- €

Total profits and kilogram amounts sorted by the group

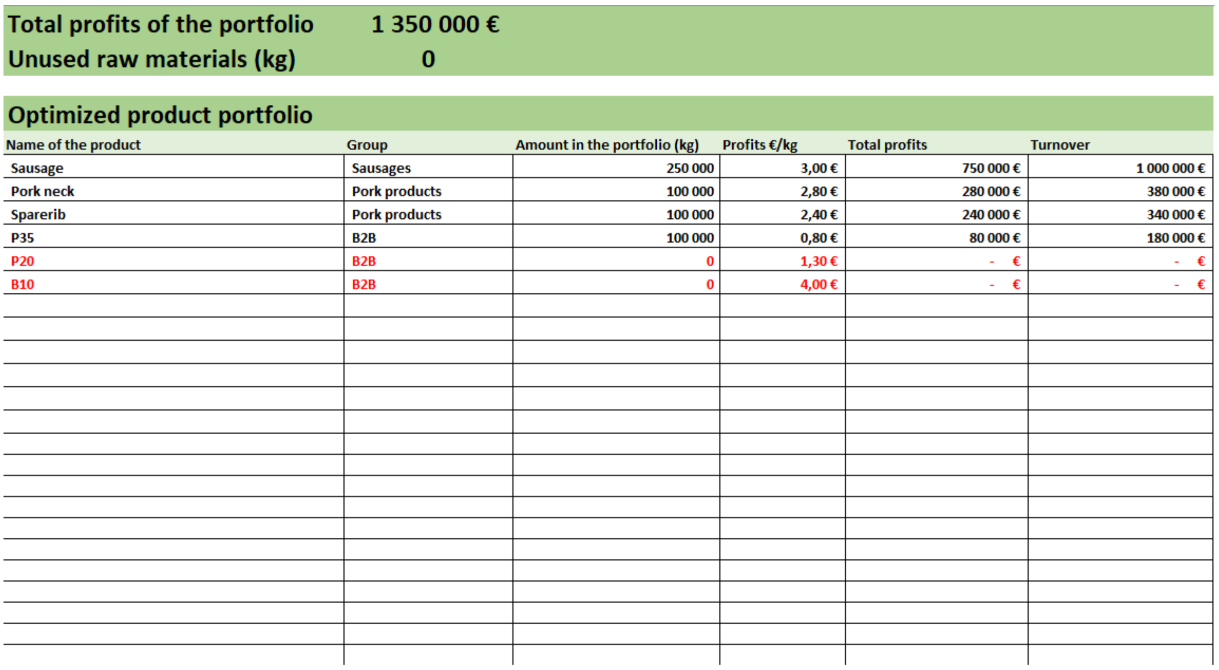
Group	Sum of Amount in the portfolio (I	Sum of Total profits
Sausages	250 000	750 000
Pork products	200 000	520 000
B2B	100 000	80 000
Grand Total	550 000	1 350 000

Save results to a new sheet

Raw material usage in kilograms

Raw material	Usage (kg)	Available (kg)	Unused (kg)
P35	270 000	270 000	0
P20	230 000	230 000	0
B10	50 000	50 000	0

Figure 6. Overview of sheet 7.

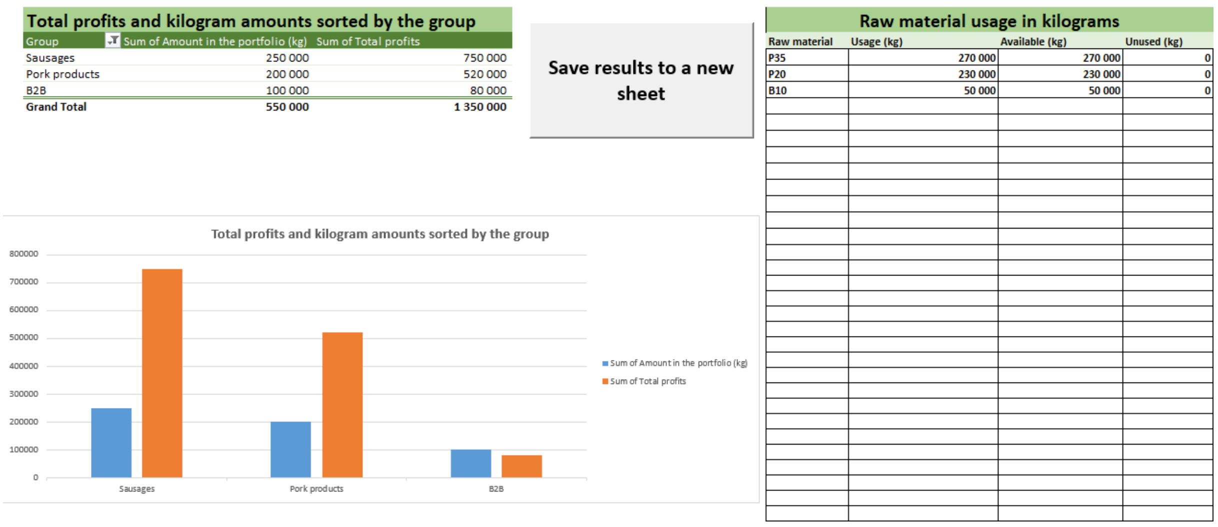


Total profits of the portfolio	1 350 000 €
Unused raw materials (kg)	0

Optimized product portfolio

Name of the product	Group	Amount in the portfolio (kg)	Profits €/kg	Total profits	Turnover
Sausage	Sausages	250 000	3,00 €	750 000 €	1 000 000 €
Pork neck	Pork products	100 000	2,80 €	280 000 €	380 000 €
Sparerib	Pork products	100 000	2,40 €	240 000 €	340 000 €
P35	B2B	100 000	0,80 €	80 000 €	180 000 €
P20	B2B	0	1,30 €	- €	- €
B10	B2B	0	4,00 €	- €	- €

Figure 7. Results of optimized product portfolio on sheet 7.



Total profits and kilogram amounts sorted by the group

Group	Sum of Amount in the portfolio (kg)	Sum of Total profits
Sausages	250 000	750 000
Pork products	200 000	520 000
B2B	100 000	80 000
Grand Total	**550 000**	**1 350 000**

Raw material usage in kilograms

Raw material	Usage (kg)	Available (kg)	Unused (kg)
P35	270 000	270 000	0
P20	230 000	230 000	0
B10	50 000	50 000	0

Save results to a new sheet
Total profits and kilogram amounts sorted by the group
800000
700000
600000
500000
400000
300000
200000
100000
0
Sausages
Pork products
B2B
Sum of Amount in the portfolio (kg)
Sum of Total profits

Figure 8. Results by product group, raw material usage and the saving macro on sheet 7.

To ensure that the tool functions as it should and actually finds an optimal portfolio, it was tested with a simple test problem to which the correct answer was known. The problem was similar to the subproblem presented in Chapter 4.3, but it will not be presented more closely as it contains sensitive information about the company. The tool found the correct answer to the problem, and thus was claimed to function as it should.

5 Implementing the profitability optimization tool

This chapter begins by illustrating how the optimization tool was implemented in the case company by optimizing a product portfolio of one business area. After this, the results will be presented by calculating the potential of profitability improvement by using historical data. Next, the results will be analyzed and summarized. The last subchapter discusses the reliability and validity of the empirical work done.

5.1 Implementation process

The tool was implemented by presenting it for sales managers and for a category & strategy director in a workshop where a product portfolio of one business area was optimized. The portfolio included 89 products using the seven different raw materials mentioned previously. The workshop began by introducing the basics of the tool – what it does, how it does it and how to use it.

5.2 Testing procedure

The first optimization was executed and only a few loose constraints were given to some products. This resulted in a portfolio that included only a few products of the possible 89. However, this portfolio was unrealistic as the quantities of the products in the portfolio greatly exceeded the current market demands. Hence, a maximum constraint was set to every product that was included in the portfolio and a new optimization was executed. After this, there were more products included in the portfolio, but still, several products had quantities that were unrealistic. As previously, more maximum constraints were placed, and the portfolio was optimized again. This pattern of optimizing and adding more constraints was repeated until the sales managers and category & strategy director concluded that the optimized portfolio was realistic considering the market situation and marketing resources. The reason why all the constraints were not placed in the beginning was that by making several optimizations the participants got a deeper understanding of how the tool function as well as illustrating how the optimal portfolio would look in different market situations.

The computing time with the final optimization was 9,43 seconds and simplex required 70 iterations to find the global optimum. The optimization was done with the same computer which was used to solve the submodel in Chapter 4.5.

5.3 Results of optimization utilizing historical data

The potential of profitability improvement was calculated by comparing one-year profits from a realized portfolio against profits that could have been obtained from the same raw materials by using the tool. The portfolio in question was the same portfolio of 89 products which was used in the previously described workshop, and the sales data was from 2018. The amounts of raw materials from the realized portfolio were inputted to the tool, and a sales expert estimated minimum- and maximum constraints to each of the 89 products according to the market situation-, marketing resources-, strategic decisions- and contracts with retailers in 2018. The optimized portfolio generated profits 1,68% higher than the realized portfolio, meaning that the tool has the potential to improve the profitability of the case company.

5.4. Result analysis

The key insight from the empirical work done within this thesis was that LP-modeling augmented with the simplex algorithm and the sensitivity analysis forms an effective combination for solving product portfolio optimization problems and improving the profitability of the case company. Several products in the meat processing industry have the option to be either sold as end products or alternatively processed to raw materials that can be used for other products. We were able to show how all these complex connections between different products and between products and raw materials can be modeled in a simple way by creating artificial recipes for all products consisting of raw materials that have only two attributes: type of meat (animal) and fat content. This includes a simplification of reality as the real raw materials have more attributes than only fat content and type of meat. However, experts from the company considered the results as valid and applicable despite the simplification made.

5.5 Reliability and validity

Reliability refers to the consistency of the results, meaning how consistent would the results be if the study would be executed multiple times by different people. Validity refers to the extent that the study measures what it should measure. (Saunders, Lewis & Thornhill 2009, 156-157) This thesis was a case study, and hence it is not guaranteed that the results would be directly applicable to all the companies operating in the meat processing industry. Despite the same industry, meat processing companies are different and thus are also the models build for their profitability optimization tools.

However, this study can provide guidelines and fruitful ideas for a meat processing company that is considering to build a profitability optimization tool, especially if the company desires to keep the optimization model linear and simple with high reliability of finding the global optimum. For instance, this study shows how the possibility of producing end-products to raw materials for other end-products can be modeled easily by making simplifications of reality yet without making the results inapplicable. Moreover, the price-elasticity of the products can be difficult to model mathematically in large meat processing companies, and this thesis presents how this attribute of the model can be incorporated in the model by utilizing the sensitivity analysis. In addition, this research introduces several potential optimization algorithms that can be used for an optimization model build for a meat processing company. Lastly, this study provides examples of how meat processing companies can utilize optimization models for performing different simulations to support decision making.

6 Conclusions and discussion

This thesis presented how mathematical optimization can be applied for building a profitability optimization tool for a case company operating in the meat processing industry. This chapter concludes this study by presenting answers to the research questions and by discussing how this research could be continued further.

6.1 Answering the research questions

The objective of this thesis was to create a profitability optimization tool that can be used for strategic- and tactical product portfolio optimization in a case company operating in the meat processing industry. In addition to the case company's need for the tool, there was a research gap in the literature concerning the topic. The main research question was the following:

> *How to build and implement a simulation model for a product portfolio profitability optimization problem in a meat processing case company which has a fixed amount of raw materials and several constraints concerning its product portfolio?*

The building process began by creating a sub-model of the problem. The sub-model was modeled first verbally and graphically. Next, the sub-model was converted into a mathematical form. After this, the simplex algorithm was applied to solve the submodel by using Excel Solver. After solving the first submodel, the next step was to expand the submodel gradually by adding more attributes of the problem to it until the whole problem was modeled. The price elasticity of the products was considered with the sensitivity analysis in the final model constructed.

One special attribute in the meat processing industry is that several products have an option to be either sold as end products or alternatively processed to raw materials that can be used for other products. A large product portfolio including several such products can form a complex system of connections between different products and raw materials which can be challenging to model and computationally complex to solve. To avoid these problems in this study, this system of connections was modeled by creating artificial recipes for all products including raw materials that have only two attributes: type of meat (animal) and fat content. This is a simplification of reality, but it was considered not to affect remarkably to the validity of the results.

The main research question was supplemented with the following sub research questions:

1. *Which are the most popular optimization algorithms for product mix optimization problems that could be applied to the problem of the case company?*

2. *Which optimization algorithm is most suitable for solving the problem of the case company?*

Answer to the first sub research question is that according to the analysis made in this study, the most popular optimization algorithms for this area of optimization are: simplex, genetic algorithm, particle swarm optimization, simulated annealing, and tabu search. Answer to the second sub research question is that simplex supplemented with sensitivity analysis is the most suitable algorithm for solving the problem of the case company.

6.2 Suggestions for further research

The profitability optimization tool build in this thesis is able to optimize a product portfolio including 200 end products and 30 different raw materials which is a sufficient amount for optimizing portfolios of single business areas of the case company. However, as the total number of end products is larger than 200, the research could be continued by expanding the model to include all the end products, thus optimizing all the business areas simultaneously. Ideally, the number of different types of animals bought and the amount of additional raw materials available could be inputted to the expanded model and then an optimization could be run to optimize the whole product portfolio. Moreover, future research could aim to model the possibility of selling an end product or processing it to raw materials for other end products without simplifying the reality as much as in the model built in this thesis.

References

Blum, C. & Roli, A. (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, *ACM Computing Surveys*, 35, 3, 268-308.

Chaharsooghi, S. K. & Jafari, N. (2007) A Simulated Annealing Approach for Product Mix Decisions, *Scientia Iraniaca*, 14, 3, 230-235.

Cottle, R. & Thapa, M. (2017) Linear and nonlinear optimization, New York, USA, Springer.

Dantzig, G.B. (1987) Origins of the simplex method, Stanford University technical report sol 87-5.

Eiselt, H. A., Sandblom, C. (2012) Operations research: a model based approach, 2nd ed., New York, USA, Springer.

Elsevier (2019) [www document] [Accessed 5.6.2019] Available: https://www.elsevier.com/solutions/scopus

Forrest, S. (1993) Genetic algorithms: Principles of natural selection applied to computation, *Science*, 261, 5123, 872-878.

Frontline Systems (2019a) Excel solver - algorithms and methods used [www document] [Accessed 20.7.2019] Available: https://www.solver.com/excel-solver-algorithms-and-methods-used

Frontline Systems (2019b) Excel solver - what solver can and cannot do [www document] [Accessed 20.7.2019] Available: https://www.solver.com/excel-solver-what-solver-can-and-cannot-do

Glover, F. (1977) HEURISTICS FOR INTEGER PROGRAMMING USING SURROGATE CONSTRAINTS, *Decision Sciences, 8, 1, 156-166.*

Glover, F. (1986) Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research,* 13, 5, 533-549.

Hillier, F. S. & Lieberman, G. J. (1990) Introduction to operations research, 5th ed., New York, USA, McGraw-Hill.

Informs (2019a) Operations research & analytics [www document] [Accessed

5.4.2019] Available: https://www.informs.org/Explore/Operations-Research-Analytics

Informs (2019b) What is O.R.? [www document] [Accessed 5.4.2019] Available: https://www.informs.org/Explore/What-is-O.R.-Analytics/What-is-O.R.

Jansen, B., De Jong, J.J., Roos, C. & Terlaky, T. (1997) Sensitivity analysis in linear programming: Just be careful!, *European Journal of Operational Research,* 101, 1, 15-28.

Jiao, J., Zhang, Y. & Wang, Y. (2007) A heuristic genetic algorithm for product portfolio planning, *Computers and Operations Research*, 34, 6, 1777-1799.

Kennedy, J. & Eberhart, R. (1995) Particle swarm optimization, *Proceedings of the 1995 IEEE International Conference on Neural Networks. Part 1 (of 6),* 27.11.-1.12.1995 Perth, Australia, 1942-1948.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983) Optimization by simulated annealing, *Science,* 220, 4598, 671-680.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. & Teller, E. (1953) Equation of state calculations by fast computing machines, *The Journal of Chemical Physics*, 21, 6, 1087-1092.

Mishra, N., Prakash, Tiwari, M.K., Shankar, R. & Chan, F.T.S. (2005) Hybrid tabu-simulated annealing based approach to solve multi-constraint product mix decision problem, *Expert Systems with Applications*, 29, 2, 446-454.

Onwubolu, G. C. (2001) Tabu search-based algorithm for the TOC product mix decision, *International Journal of Production Research*, 39, 10, 2065-2076.

Onwubolu, G. C. & Muting, M. (2001) Optimizing the multiple constrained resources product mix problem using genetic algorithms, *International Journal of Production Research*, 39, 9, 1897-1910.

Rezaie, K., Nazari-shirkouhi, S. & Ghodsi, R. (2010) Theory of Constraints and Particle Swarm Optimization Approaches for Product Mix Problem Decision, *Australian Journal of Basic and Applied Sciences*, 4, 12, 6483-6491.

Sadeghi, A., Alem-Tabriz, A. & Zandieh, M. (2011) Product portfolio planning: A metaheuristic-based simulated annealing algorithm, *International Journal of*

Production Research, 49, 8, 2327-2350.

Saunders, M., Lewis, P. & Thornhill, A. (2009) Research methods for business students, 5th ed., Harlow, England, Pearson Education Limited.

Srinivasan, R. (2014) Strategic business decisions: a quantitative approach, New York, USA, Springer.

Taha, H. (2007) Operations research: an introduction, 8 ed., Upper Saddle River, New Jersey, USA, Pearson Prentice Hall.

The Simplex Method (2019) [www document] [Accessed 15.7.2019] Available: https://www.utdallas.edu/~scniu/OPRE-6201/documents/LP4-Simplex.html

Wolpert, D. H. & Macready, W. G. (1997) No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, 1, 1, 67-82.

Yang, X.-S. (2011) Review of Metaheuristics and Generalized Evolutionary Walk Algorithm, *Int. J. Bio-Inspired Computation,* 3, 2, 77-84.

Zheng, Y.-Q. & Yu, S.-J. (2013) Two-level product mix decision problem based on regenerated particle swarm optimization, *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS,* 19, 4, 782-789.