Angewandte Mathematik und Optimierung Schriftenreihe
Applied Mathematics and Optimization Series
AMOS # 63(2017)

Leonie Marguerite Johannsmann

Optimized Spare Parts Inventory Management for Military Deployment

Herausgegeben von der
Professur für Angewandte Mathematik
Professor Dr. rer. nat. Armin Fügenschuh

Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg
Fachbereich Maschinenbau
Holstenhofweg 85
D-22043 Hamburg

Telefon: +49 (0)40 6541 3540
Fax: +49 (0)40 6541 3672

e-mail: appliedmath@hsu-hh.de
URL: http://www.hsu-hh.de/am

Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Print 2199-1928
Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Internet 2199-1936

HELMUT SCHMIDT UNIVERSITÄT
Universität der Bundeswehr Hamburg

Leonie Marguerite Johannsmann

Optimized Spare Parts Inventory Management for Military Deployment

Optimierte Ersatzteillagerhaltung für den militärischen Einsatz

Master Thesis

Department of Mechanical Engineering

Degree Course:	Industrial Engineering
Matr.-Nr.	866950
Submitted:	August 29, 2017
Auditor:	Prof. Dr. Armin Fügenschuh

Abstract

In participation with North Atlantic Treaty Organization (NATO) the German Armed Forces provide a German operation contingent to support the NATO Response Force (NRF). Therefore, a camp for accommodation, food supply, medical care, spare parts, and more has to be available. Such a camp is restricted in weight and size in order to be quickly movable in an upcoming deployment situation. To ensure an optimal use of such a restricted camp, this thesis introduces the optimization program OPtimizing software for Spare Parts INventory (tOPSPIN) to optimize the storage of a warehouse for an upcoming deployment situation.

Als Beitragspartner der North Atlantic Treaty Organization (NATO), stellt die deutsche Bundeswehr ein Kontingent zur Unterstüzung der NATO Response Force (NRF). Das benötigte Einsatzlager muss dabei Unterkünfte, Verpflegung, medizinische Versorgung, Ersatzteile und weiteres bereitstellen. Ein solches Lager is begrenzt in Gewicht und Größe, um schnell verlegt werden zu können bei einem möglichen Einsatz. Um sicherzustellen, dass dieses begrenzte Lager optimal genutzt wird, stellt die Masterarbeit das Optimierungsprogramm OPtimizing software for Spare Parts INventory (tOPSPIN) vor, dass das Einsatzlager für solche Einsätze optimiert.

Contents

List of Figures

Abbreviations

cur.Val.	current value
kg	kilogram
MILP	mixed-integer linear program
Model 1	"Fixed Parts to Systems Assignment"-Model
Model 2	"Flexible Parts to Systems Assignment"-Model
NATO	North Atlantic Treaty Organization
NRF	NATO Response Force
OR	Operations Research
SCM	Supply Chain Management
sec	second
Sim Data	simulated dataset
SOF	Special Operations Force
tOPSPIN	OPtimizing software for Spare Parts INventory
Real Data	reduced dataset

1 Introduction

In 2002, the North Atlantic Treaty Organization (NATO) launched the NATO Response Force (NRF) as a highly ready and technologically advanced, multinational force made up of land, air, maritime, and Special Operations Force (SOF) components that the Alliance can deploy quickly, wherever needed. The NRF is able to react in a very short time to the full range of security challenges from crisis management to collective defense [10].
The Federal Republic of Germany takes part in the NRF with their forces in 2017. Thus, the German Armed Forces stands by for upcoming deployments with their expertise, technology, and manpower. To provide a German operation contingent, many things has to be considered and planned to ensure a quick reaction in case of a deployment. The base of every deployment is an operation camp for accommodation, food supply, medical care, and mission planning. To insure a quick reaction, the capability of every part of this camp is highly limited in weight and size. Nevertheless, every part has to work smoothly and correctly to ensure a successful participation in the NRF. After the supply of all essential requirements, the remaining capacity is used for spare parts to be able to repair operational vehicles.
Therefore, the thesis is focused on the development of optimization for the storage of a warehouse in case of a deployment. By using the methods of Operations Research (OR) and the programming tool AIMMS, the software OPtimizing software for Spare Parts INventory (tOPSPIN) is developed to support the planning and preparation of the German Armed Forces' participation to the NRF or further deployments. The software tOPSPIN includes a pessimistic, one-staged model and optimistic two-staged model to optimize the storage. The following chapters describes the logistical and mathematical foundations, the mathematical models of tOPSPIN, declaration of the used data, comparisons and results of both models, and a conclusion.

2 Logistical Foundations

A well-working warehouse is important for every company, because Logistics is a part of each link of the supply chain. The supply chain describes the whole system of organizations, people, activities, information, and resources to move products or services from suppliers to customers. A warehouse serves as a buffer to store overproduction and to balance underproduction caused by variations in demand cite3. This chapter gives a introduction to the goals of logistics, storage tasks, and the impact of logistics in the military.

2.1 Supply Chain Management

This section presents information from [2]. The term logistics is also known under the term Supply Chain Management (SCM), which describes the tasks of logistics even better. The SCM pursues intelligent planning and control of all sections of the supply chain. Their primary goal is to ensure the right amount of a product or service in the right quantity and quality at the right time and place for every section of the supply chain. The goals supply chain management is the minimization of cost and maximization of service. On one hand, the SCM should provides a full service for every request and, on the other hand, the service cost rises if all possible materials, parts, and modules for the production are available every time. The SCM strives to balance both parts, high service, and low costs.
One section of SCM deals with optimal warehousing to fulfill the mentioned goals and balance the conflict of a high service under low costs. A warehouse stores and saves goods in a provided building or area. Furthermore, it is a desired interruption of the material flow to build buffer stocks. Typical stored goods include tools, products, aids, and means of production and transport. Warehousing performs three main functions. The secure function takes place if all information on demand and delivery time are unavailable. Storage bal-

ances seasonal fluctuations and supply bottlenecks of products, which must be available at any time. Storage also has a bridging function to store overproductions in times of lower demand or to store finished products until they get delivered. In times of high price fluctuations or especially low prices, storage has a speculation function to store larger procurement quantity, and furthermore, discounts of large order quantities can be used if the storage costs are lower than the savings.

2.2 Military Logistics

This paragraph presents information from [6]. In antiquity, armies had to move from place to place to secure supplies of fresh resources and operational capability. Eventually, they started to carry their board. Alexander the Great was the first to load ships with fresh resources and plan the route of his troops along the coast to ensure close access to the goods. The system made it possible to move large troops over a long time and distance. Up to the middle of the nineteenth century, the two options - obtain and carry - were the principal methods of supplying and sustaining the troops. The Industrial Revolution put forth a third option - send. Trains were employed for military uses and affected the way logistics were implemented. Supplies could now be sent fast from the rear area to the front over a long distance. Today the military uses a mix of all three options - obtain, carry, and send - to achieve an optimal operational capability.
In contrast to civilian companies, the military has its highest priority in operational capability instead of reducing costs. Storage in deployment is simple, less automated, and restricted in space. The main decision deals with the conflict of a high operational capability and the low storage capacity. The goal is to determine the best mix of resources within a certain area of logistic infrastructure. That includes an operational capability for a long time without supply to be independent in different cases, like being marooned. Transport routes and airports have to be developed or built. Furthermore, cooperation with foreign countries, their customs regulations, and laws must be taken into account.

2.3 Levels of Logistics

The military operates on three levels - strategic, operational, and tactical; logistics is separated into the same levels. They pursue different targets, but they are not necessarily disjointed. The following subsections are based on the references [6] and [4].

2.3.1 Strategic Logistics

The strategical level deals with major defense-related decisions that have long-lasting impacts. Decisions like investments in research and development, procurement and replenishment policies, and decision issues related to the physical infrastructure, result in significant and long range economic and operational implications. In particular, economic constraints affect logistic capabilities. Strategic logistics deals with building up and maintaining the national military and military-related infrastructure, like technology, industry, inventory, storage capacity, and transportation capabilities. It is characterized by standardization, uniformity, and relative predictability, because of its sheer size. Since German reunification, the German Armed Forces have been in a continuous process of adjustments and reforms. The mission of a purely national defense has changed to an operational mission to perform in a fast-packed security policy environment. New investments in training, equipment, and deployment with a limited budget leads to an economical view of the German Armed Forces. A measurable input, like the budget, contrasts with a difficult to measure output, like efficiency and benefits of armaments. Therefore, a new part of the conflict occurs. Besides the high operational capability with a small capacity in a deployment, also an decrease of costs have to take into account.

2.3.2 Operational Logistics

On a lower level, operational logistics describes active operations to seize, retain, and exploit the initiative to gain and maintain a position of relative advantage in sustained land operations. The supporting logistic is called operational logistics. Operational logistics deals with the economic and industrial base of a nation

and the combat units to set up the logistic system in the theater of operations. Furthermore, the missions forecast, analyze, and prioritize future demands for logistics assets according to the operational objectives. The implementation of these entities must be in accordance with the time and space parameters of the operation. Operational logistics supports theater-level activities and operational moves, and not directly combat units - like the tactical logistics level. Consequently, the objectives of operational logistics are specified by the operational goals rather than by military units. If the operational goals are determined in terms of time and space, operational logistics capabilities must also be specified by tactical parameters.

2.3.3 Tactical Logistics

The last level, tactical logistics, is used to affect the battle in progress and sustain the troops, providing them with production material and maintains their equipment. Tactical logistics is focused on three important characteristics - protection, mobility, and firepower - of tactical warfare. The main activities of tactical logistics are technical and apply directly to the combat units. The conditions to operate are disturbed by enemy actions, and may change its course of action abruptly and unexpectedly and leads to a complex and challenging logistical task. Tactical logistics is unpredictable and variable, and heavily depends on the random outcome of the tactical battle.

2.4 Summary

Military logistics pursue different targets than civilian companies, but the economical aspects gain in importance. Consequently, all level of the military logistics are impacted. All three levels of logistics are not strictly disjointed and may overlap between two adjacent levels. To find an optimal interaction of the logistics goals, mathematical optimization can be used. This thesis investigates the optimal interaction of the operational capability with limited capacities of space for an upcoming deployment, but not for a specific combat. Thus, the thesis moves on the level of operational logistics to support theater-level activities and operational moves.

3 Mathematical Foundations

Operations Research (OR) is a scientific discipline that deals with the formulation of mathematical models to describe real world optimization problems and the development of computational techniques (algorithms) to find solutions (i.e., to solve them). The mathematical subdomain, mathematical optimization, will be presented in this chapter and explains the mathematical foundations of linear, integer, mixed-integer, and stochastic optimization. Furthermore the chapter explains the simulation foundations and the current solving procedures to solve optimization models.

3.1 Operations Research

The following information is taken from [3]. In the twentieth century, progressive technical and economic development challenged companies to solve complex optimization questions with simple methods, which do not need a high number of calculation steps. It was necessary to find mathematical methods to solve problems such as reducing production costs with a high output of goods. During World War II, OR developed as a new mathematical method in the military sector. For example, it was used to calculate the optimal size of convoys or optimal distribution of tenders for submarine defense. Another example was the optimal work distribution for mine laying that resulted in a time saving of about 25 % for the German Army [5]. Today the methods of OR are essential for business and also military sections.
To solve problems, OR includes the three following characteristics:

- Decision preparation,
- striving for an optimal solution, and
- use of mathematical methods.

3.1.1 Linear Programming

The following information is taken from [**1**]. One tool of OR is linear programming. Linear optimization problems are one of the most thoroughly explored mathematical programming problems. They are well-known for economic and business applications and have been studied for years. The objective is to optimize a linear function of one or more variables. Therefore, the constraints are given by equations and inequations. A simplified model economizes the processing power for thousands of variables and constraints. Based on that, "non-linearities" are mostly replaced by linear approximations and binary variables [**1**]. Assuming a linear optimization problem, the model is formulate by, the following steps:

- Set and parameters from the real-world application as input data,
- Derived sets and parameter, based on the input data,
- specification of variables,
- formulation of the constraints as linear equations or inequalities, and
- specification of the objective function to minimize or maximize.

For example, consider a minimization problem which is given as:

$$\text{minimize} \sum_{j=1}^{m} c_j \cdot x_j, \tag{3.1}$$

$$\text{subject to} \sum_{j=1}^{m} a_{i,j} \cdot x_j \leq b_i \qquad (i = 1, ..., n), \tag{3.2}$$

$$x_j \geq 0 \qquad (j = 1, ..., m). \tag{3.3}$$

Given are the costs c_j for n products ($j = 1, ..., n$), the production factor $a_{i,j}$ for utilization of machine i to produce a unit of product j, and the capacity b_i of machine ($i = 1, ..., m$). The quantity unit of product j is described with decision variable x_j. A point x_j that satisfies the aforementioned constraint (3.2) is part of

the solution. Furthermore, if the point fulfills the non-negative constraint (3.3) it is part of the feasible solution. An optimal solution for a minimization problem is found if a feasible solution x^* has the lowest value of the objective function.

3.1.2 Simplex Algorithm

This subsection is based on the source [**1**]. The Simplex Algorithm is a classic solution procedure for linear optimization problems. The algorithm moves along a polyhedron, built by the constraints, to find an edge with an optimal solution. With assistance of a Simplex Tableau in Figure 3.1, simple linear optimization problems can be solved manually. First, the inequations are transformed to equations using slack variables. The slack variables get transfered into a Simplex-Tableau. Therefore, the last row shows the basic solution of the objective function. This row is called, in the following result F-row.
To find an optimal solution, the following three steps are sufficient:

Step 1: Choice of the pivot column s
If the F-row in the tableau contains only nonnegative values, the current basic solution, an edge of the polyhedron, is optimal. The algorithm can be terminated. Otherwise choose a row s with the smallest negative value in the F-row. The associated nonbasic variable x_s enters in the base and is called the pivot column.

Step 2: Choice of the pivot row z
If all $a'_{is} \leq 0$, there is no optimal solution for the current problem. The problem is unbounded. The algorithm can be canceled. Otherwise, choose a row z, given for:

$$\frac{b'_z}{a'_{zs}} = \min \left\{ \frac{b'_i}{a'_{is}} \qquad \text{i} = 1, ..., m \qquad with\ a'_{is} > 0 \right\} \tag{3.4}$$

The row z is called pivot row and the element a'_{zs} pivot element.

Step 3: Calculation of the new basic solution
To calculate the new basic solution, the previous basis variable of the pivot row z has to be replaced by the previous nonbasic variable x_s. By using linear transformation of the constraint-system and the new basic variable a unit vector $a'_{zs} = 1$

is created as following:

- Divide the pivot row by pivot element.
- Multiply for all pivot rows i (except s) the new pivot row with $-a_{is}$ and add it to the respective row i.

		non-basic variable			basic variable				
		x_1	...	x_{n-m}	x_{n-m+1}	...	x_n	F	b_i
basic variable	x_{n-m+1}	a_{11}	...	$a_{1,n-m}$	1	...	0	0	b_1
	.	.	...	.	.	...	.	.	.
	.	.	...	.	.	...	.	.	.
	x_n	a_{m1}	...	$a_{m,n-m}$	0	...	1	0	b_m
		$-c_1$	...	$-c_{n-m}$	0	...	0	1	*cur.Val.*

Figure 3.1: Exemplary Simplex Tableau.

3.1.3 Integer Programming

In linear programming, it is assumed that all variables are continuous. However, not every problem is writable with non-integer variables. In the real world, some variables can only assume integer values $x \in \mathbb{Z}^+$. Moreover, decisions about the state of a variable, for example accept or reject an order for production planning, binary variables $x \in \{0, 1\}$ are used:

$$x = \begin{cases} 1, & \text{order accepted,} \\ 0, & \text{else.} \end{cases} \tag{3.5}$$

3.1.4 Branch and Bound

The branch and bound method is one possible way of solving an integer optimization problem. The algorithm uses the concept of branching and bounding

the decision tree. The continuous branching of the primary problem in many subsets results in a tree structure, called the branch and bound tree [7]. A complete enumeration of every branch provides an opportunity to solve the optimization problem, but is only useful for small number of nodes n, because a large number of n causes long calculation times. For example, a set $P \in \mathcal{P}$ is given. If all binary values $x \in \{0, 1\}$ are successively generated and they belong to the set P the current value of the objective function get solved,and furthermore, the "best" value so far get saved. Finally, the last "best" value is the best solution for x^*. For integer programming with big n the implied enumeration is used. The method finds and eliminates successive subsets without any optimal solution. The algorithm generates a tree; its nodes include subsets of the requested set P. Every branch corresponds to a new subset of P. The upper and lower bound skips non-relevant subsets [7]. An example branch and bound tree is represented in Figure 3.2. x_0 is the first "best" value. The algorithm starts to investigate the first generated branch. x_1 results in no better value, so the branch is bounded. Because x_2 results in a new "best" value, the stage algorithm goes further to the next branched variables to find a better value of the objective function until all branches are bounded or all variables are investigated.

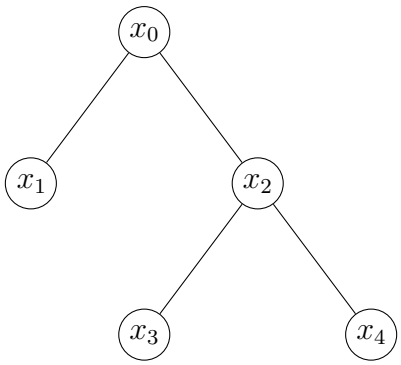

Figure 3.2: A Branch and Bound Tree.

3.2 Stochastics

"Stochastics" refers to the domains of probability theory and statistics. A stochastic method investigates the mathematical modeling of random events and is

used in nearly any empirical discipline; for example, strategies for gambling, climate research, quality controls, and quantum physics. The probability theory investigates random processes with known probabilities; in contrast, the statistic draws conclusions out of observed data for unknown probabilities. Therefore, the probability theory builds the base for theoretical investigation of statistical procedures.

3.2.1 Decision Tree

Nobody can predict on which side a falling coin will land: heads or tails. However, there is a probability for both sides. The decision tree shows the decision making process for the 50-50 chance. The final decision is a composition of all independent decisions of each single step. Based on a single root, all nodes of the first decision get drawn, for every follower the next nodes and so on. A final decision corresponds to the way from the root to a leaf (Head or Tail). The number of possibilities of any way from the root to the leafs are the product of all nodes. Figure 3.3 shows a example of a two staged decision tree and their probabilities of all occurrences. If the order of the occurrences is irrelevant, the probabilities of (Tail - Head) and (Head - Tail) can be summed to: $1/4 + 1/4 = 1/2$.

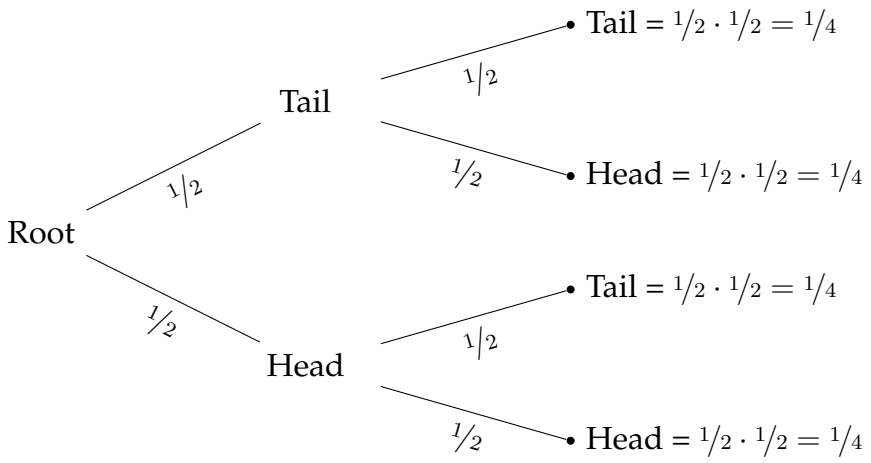

Figure 3.3: Decision Tree.

3.2.2 Random Experiment

A random experiment can be single- or multi-staged. For a single-staged random experiment the coin is thrown only one time. If a gambler rolls a four with a normal dice in the first try, he could assume that he will roll a four every time, because of that single observation. Such an observation is not mostly meaningful. Therefore, in a multi-staged random experiment the attempt repeats [9]. In a real experiment a gambler can get more tails or heads in the first rounds. However, if the number of repetitions rise the results get closer to the probability of 50 %. That leads to the law of large numbers; the law of large numbers states that the relative frequency of a random occurrence converges to the theoretical probability when the random experiment is repeated under the same conditions. If possibilities of a random experiment have the same probabilities to occur, the experiment is called a Laplace experiment [9].

3.2.3 Random Variable

A stochastic term sasy that, a random variable $X : \Omega \to \mathcal{E}$ is a measurable function from a set of possible outcomes Ω to a measurable space $\mathcal{E}$. If the mapped outcome X is real-valued, its called a random number. A random variable does not return a probability. The probability of a measurable set of outcomes is given by the probability measure P. The most used notation is $P(X = a)$ or $P(a \leq X \leq b)$ for the probabilities of outcomes "X maps to a" or "X maps to the interval $[a, b]$" [8].

3.3 Stochastic Programming

Stochastic programming is the part of OR that studies how to integrate uncertainty into decision problems. Some decisions lack important information to solve the problem until the decision is made. Problems like this are found in transportation, logistic, supply chain management, and other related fields. Some necessary figures, such as demand, cost, and quality, are unknown. The common practice of solving such an optimization problem is:

- Use statistics or experts to get information about the parameter.

- Solve the model with expected values.
- Observe whether an optimal solution $\tilde{x}$ depends on the used parameter.
- Use sensitivity analysis to determine whether $\tilde{x}$ depends on the parameters.

The sensitivity analysis estimates the degree to which parameter values can vary without changing the fundamental character of the solution. In practice, a uncertain parameter does not affect the stability of the solution related to the variations in the parameter. Otherwise, constant parameters cause an unstable solution. Overall the stability effecting parameters determine the usability of the solution. A scenario analysis generates a set of possible futures and is useful to explore different trendlines in fundamental parameters. All techniques investigate a variety of future states and generate a set of possible solutions for the next step of the decision process.
The following information is taken from [1]. The decision process has different information stages. An information stage is an important concept that distinguishes stochastic programming. A stage is a moment when decisions are made within a model. Stages can follow naturally from the problem setting or are modeling approximations. In the first stage the unknown parameters are set. The values of the unknown parameters can be determined by a simulation of random values. Therefore, the simulation follows a best suitable distribution to reflect the real world. The second stage shows the results of the decision and evaluates the worth of the placed parameters. The same decisions in the first stage can lead to different states in the next stage. One of the possible scenarios of parameters occurs and results in a state in the second stage. The number of different scenarios causes big models. To find an optimal solution for a model is almost impossible if the distribution of parameters depends on a large number of scenarios or follows a continuous distribution. For example, a two-stage

optimization problem with a penalty for overweight:

$$\text{maximize} \sum_{i=1}^{n} c_i \cdot x_i - d \sum_{s \in \mathcal{S}} p^s \cdot z^s, \tag{3.6}$$

$$\text{subject to} \sum_{i=1}^{n} w_i^s \cdot x_i - z^s \leq b, \qquad \forall s \in \mathcal{S}, \tag{3.7}$$

$$z^s \geq 0, \qquad \forall s \in \mathcal{S}, \tag{3.8}$$

$$x_i \in \{0, 1\}, \qquad i = 1, ..., n. \tag{3.9}$$

The parameter c_i is the value of item i, w_i its weight and b the capacity. Assuming that the weights are uncertain with a vector of random variables [w_1,...,w_n]. Let $\mathcal{S}$ be the set of scenarios describing the uncertainty of the weights. d is the unit penalty for overweight. The parameter p^s and z^s describes the penalty for every scenario. The actually weight is known, after the decision of the used item is made. For an exceed of the capacity a penalty has to be paid. The goal is to maximize the value of the items selected minus the excepted penalty for overweight.

3.4 Simulation

Simulation predicts the states of single parts of a whole system. Those states depend on the probability distribution. There are few types of probability distributions to describe various types of discrete and continuous random variables. In this thesis the triangular and geometric distributions are used. In the continuous triangular distribution all values of the random variable occur between a fixed minimum $min[x]$ and a fixed maximum $max[x]$, but with a most likely value, and its position β is not necessarily in the middle of the interval. It is quite common to use the triangular distribution for a little knowledge about an uncertain parameter except that its value has to lie anywhere within fixed bounds

and that there is a most likely value. The triangular distribution is defined as:

$$f_{(\beta,0,1)}(x) = \begin{cases} 2x/\beta, & \text{for } 0 \leq x \leq \beta. \\ 2(1-x)/(1-\beta), & \text{for } \beta \leq x \leq 1. \end{cases} \tag{3.10}$$

The discrete geometric distribution shows the probability to get a success after n repetitions of a Bernoulli experiment. The geometric distribution is defined as:

$$P(X = i) = (1-p)^i \cdot p \tag{3.11}$$

3.4.1 Monte Carlo Simulation

The eponym of Monte Carlo Simulation is the famous casino in the city of Monte Carlo. The game of roulette inspires the method to draw samples with a computer. The following two characteristics of roulette are important for Monte Carlo Simulation:

- The possibility that the ball lands on a specific number is equal for all numbers.
- The possibility that the ball lands on a specific number does not depend on the occurrence before.

Monte Carlo Simulation analyzes static and stochastical systems by means of sample experiments. Moreover, it is applicable for known probability distributions of the parameters and independently consecutive events [**1**].

3.4.2 Generation of Random Numbers

For a simulation, a lot of random numbers are necessary. To create these random numbers, picking real random numbers from an urn takes too much time. Instead the simulation uses pseudo-random numbers. Those numbers can be created in serval ways [**1**]:

- The sequence of numbers are arhythmically determined by a random-number-generators.

- By making use of irregular sequence of numbers in the decimal representation, like e, π and $\sqrt{2}$.
- By making use of a natural noise, like radios.

A random-number-generator has to meet the following requirements [**1**]:

- It represents a good approximation of the pseudo-random numbers distribution to the desired distribution function.
- The sequence of numbers has to be reproducible for repetitions under the same conditions.
- The sequence of numbers should have a long period length.
- The generation time should be short with low memory requirements.

The programming tool AIMMS generates random numbers by different distributions, like triangular-, geometric-, or normal distribution. The command "option seed" placed before the function gets called makes a reproduction of the random numbers possible.

4 Mathematical Models

In tOPSPIN the formulated optimization model is a mixed-integer linear program (MILP). The goal of the software is to fill a limited warehouse with spare parts for an optimal supply for a successful deployment. The software has two mathematical models. The first version, Model 1, is a one-staged problem and more pessimistic. The second version Model 2 is a two-stage problem and more optimistic. In this chapter the mathematical formulation, simulation, and the user interface of the model will be presented for both versions.

4.1 Model 1: "Fixed Parts to Systems Assignment"

Model 1 assumes that each part can only appear in each system once and is committed before it leaves to the deployment. Thus, it is a one stage problem and more pessimistic.

4.1.1 Sets and indices

The following sets and indices were defined for the problem:

s,	system, for $s \in \mathcal{S}$.
e,	part, for $e \in \mathcal{E}$.
c,	scenario, for $c \in \mathcal{C}$.
z,	state of system, for $z \in \mathcal{Z}$.
r,	priority of system, for $r \in \mathcal{R}$.
t,	state of part, for $t \in \mathcal{T}$.
l,	loop, for $l \in \mathcal{L}$.

4.1.2 Parameters

The following parameters describe the significant data about every system and contain the necessary information for the simulation and the optimization.

Simulation Parameters

TL_e,	lowest probability of success, for part e.
TH_e,	highest probability of success, for part e.
TB_e,	displacement factor, for part e.
$M_{c,s,e}$,	probability of success, for part e in system s in scenario c.
$G_{c,s,e}$,	random value, for part e in system s in scenario c.
$k_{l,c,s,e}$,	binary value, in loop l for part e in system s and scenario c with a range of t.

Simulation of the broken parts is based on a triangular and geometric function. For the triangular function the parameters $TL_e \in [0,1]$ and $TH_e \in [0,1]$ contain the lowest and highest probability of success for every single part. Furthermore, the parameter $TB_e \in [0,1]$ describes the displacement factor between the lowest and highest value. The parameter $M_{c,s,e} \in [0,1]$ stores the values from the triangular function. The parameter $G_{c,s,e} \in \mathbb{N}_0$ stores the resulting values of the geometric function. A further description for the functions are in chapter 4.1.6. The values of the simulation have a range of set t and are written in the element parameter $k_{l,c,s,e}$. The parameter shows the broken and unbroken parts e for every system s in different scenarios c over all loops l.

Optimization Parameters

$b_{s,e}$,	binary value for built-in state, for system s and part e.
$p_{s,z}$,	priority, for system s in state z with a range of r.
$q_{s,e}$,	priority, for system s of part e.
u_e,	unit size, for part e.
w_e,	weight, for part e.
v_e,	volume, for part e.
CW,	weight capacity.
CV,	volume capacity.
RW,	residual weight.
RV,	residual volume.
$DS_{s,z=0}$,	non-operational system s in state $z = 0$.
$RS_{s,z=1}$,	operational system s in state $z = 1$.
SG,	service level.
o_c,	probability of occurrence, for scenario c.

The following parameters are necessary for the optimization model and contain all relevant values. The binary parameter $b_{s,e} \in \{0, 1\}$ contains the information about the built-in parts for every system. If the part is build in the binary parameter takes the value $b_{s,e} = 1$, otherwise $b_{s,e} = 0$. Every system has a different priority in a deployment. The priorities of every system s are stored in the element parameter $p_{s,z} \in \mathbb{R}^+$ and can take a value from the range of set r from 4 (highest priority) to 1 (lowest priority). The binary parameter $q_{s,e} \in \{0, 1\}$ stores the priorities of the parts e for every system s. Some parts are necessary to run a system at all ($q_{s,e} = 1$) and some are just good to have, but does not effect the operational capability of a system ($q_{s,e} = 0$). The unit size of every part is stored in the parameter $u_e \in \mathbb{N}$. The parameter $w_e \in \mathbb{R}^+$ and $v_e \in \mathbb{R}^+$ contains the weight and volume of any part e. The entire capacity of weight and volume are stored in the parameters CW and CV and can be manually changed over the user interface to allows third parties a quick change of the restrictions. After the calculation the parameters RW and RV stores the residual weight and volume. To get an overview of the optimization, the parameters $DS_{c,z=0}$ and $RS_{c,z=1}$ show

the broken systems after the simulation and the operational systems after the optimization, respectively. The percentage of the operational systems relative to all systems is presented in the parameter SG. The parameter o_c gives a probability of occurrence for every scenario. It depends on the environmental influences and circumstances of the deployment.

4.1.3 Variables

$$x_{c,s,z} = \begin{cases} 1, & \text{if system } s \text{ in scenario } c \text{ has state } z \text{ after repairs are complete.} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

$$y_{s,e} = \begin{cases} 1, & \text{if part } e \text{ is designated to repair system } s. \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

The state z of every systems s in the different scenarios c is described in the binary variable $x_{c,s,z} \in \{0,1\}$ (4.1). The binary variable $y_{s,e} \in \{0,1\}$ (4.2) contains the storage of the optimal warehouse and shows which part e can be used to repair a system s. The total number of every part will be summed up in the variable $n_e \in \mathbb{R}_+$.

4.1.4 Constraints

The following constraints define a feasible assignment of the warehouse storage and the current state of every system.

System State

$$\sum_{z \in \mathcal{Z}} x_{c,s,z} = 1 \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}, \tag{4.3}$$

$$x_{c,s,1} \leq y_{s,e} \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}, e \in \mathcal{E}, l \in \mathcal{L} : q_{s,e} = 1, b_{s,e} = 1, k_{l,c,s,e} = 1, \tag{4.4}$$

$$1 - y_{s,e} \leq x_{c,s,0} \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}, e \in \mathcal{E}, l \in \mathcal{L} : q_{s,e} = 1, b_{s,e} = 1, k_{l,c,s,e} = 1. \tag{4.5}$$

In Equation (4.3), the system $s \in \mathcal{S}$ has only one current state for every scenario $c \in \mathcal{C}$. Equation (4.4) represents that the system $s \in \mathcal{S}$ has the current state 1 (operational system), if the part is available in the warehouse and if the following points meet the requirements: The part is built-in the system, the part is broken and the replacement is necessary to run the system. In Equation (4.5), the system $s \in \mathcal{S}$ has the current state 0 (non-operational system), if the part is not available in the warehouse and if the following points meet the requirements: The part is built-in the system, the part is broken, and the replacement is necessary to run the system.

Capacity

$$\sum_{e \in \mathcal{E}} w_e \cdot n_e \leq CW. \tag{4.6}$$

$$\sum_{e \in \mathcal{E}} v_e \cdot n_e \leq CV. \tag{4.7}$$

Equations (4.6) and (4.7) describe the capacities for the storage. The total weight of the spare parts e must not exceed the capacity of weight CW and the total volume of the spare parts e must not exceed the capacity of volume CV.

Demand

$$\sum_{s \in \mathcal{S}, e \in \mathcal{E}} y_{s,e} \cdot u_e = n_e. \tag{4.8}$$

Equation (4.8) calculates the total demand of all spare parts e.

4.1.5 Objective

$$\max \sum_{s \in \mathcal{S}, z \in \mathcal{Z}, e \in \mathcal{E}, c \in \mathcal{C}} o_c \cdot p_{s,z} \cdot x_{c,s,z} + 10^{-4} \cdot y_{s,e} - 10^{-6} \cdot n_e. \tag{4.9}$$

The primary goal of the objective function (4.9) is to maximize the number of high priority operational systems. On a subordinate point we attempt to fill the rest capacity with other spare parts, which are broken, but can not repair the whole system. Finally, we subtract a small influence to take the smaller storage in case of a solution with the same value of the objective function.

4.1.6 Simulation

The simulation generates different failure scenarios for the mathematical model. Based on a triangular function random numbers are generated and decide over a part's breakdown. The triangular function works with three parameters, $TL_e \in [0,1]$, $TH_e \in [0,1]$, and $TB_e \in [0,1]$ for every part. The parameters TL_e and TH_e contain the lowest and highest probability of success for every single part. Success means that the part is not broken. Furthermore, the parameter TB_e describes the displacement factor between the lowest and highest value. The triangular function issues a randomly picked probability of success to the parameter $M_{c,s,e} \in [0,1]$. The geometric function generates, based on the probability of success $M_{c,s,e}$, a random number. So far, the stored random numbers in parameter $G_{c,s,e} \in \mathbb{N}_0$ have higher values than 1 and does not fit to the set of $\mathcal{Z}$. A procedure changes every value > 1 to a 1 and passes them to the parameter $k_{l,c,s,e}$. Now, the parameter $k_{l,c,s,e}$ contains only elements of $\{0,1\}$ and decides if a part fails ($k_{l,c,s,e} = 1$) or not ($k_{l,c,s,e} = 0$). For example, consider following extract from the model:

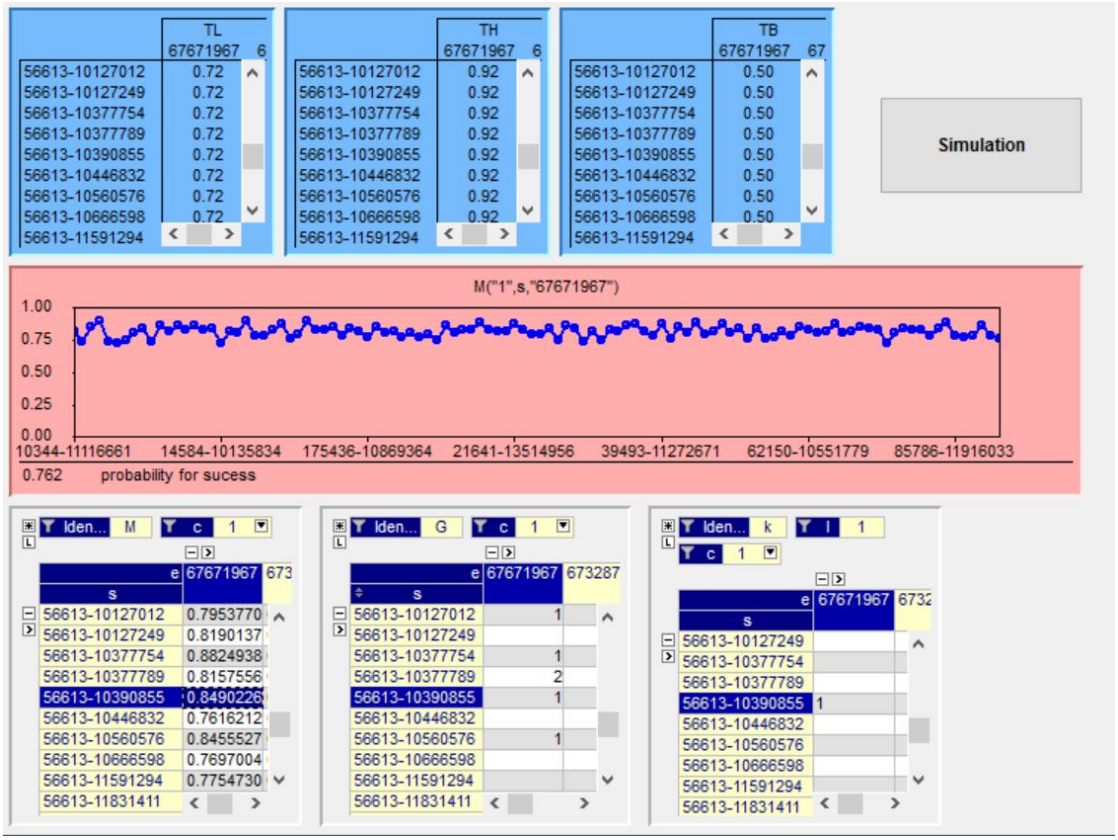

Figure 4.1: Example for a simulation with $l = 1$, $c = 1$, $e = 67671967$, and $s = 56613 - 1039855$.

The blue boxes show the values of the parameters TL_e, TH_e, TB_e. The value of the part e is equal for every system s. For example, the triangular function moves for the part $e = 67671967$ within limits of $TL_{67671967} = 72\,\%$ and $TL_{67671967} = 92\,\%$ with a displacement factor of $TB_{67671967} = 0.5$. The red box shows the graph of the triangular function for every system s for part $e = 67671967$. The accurate values of the triangular function are displayed in the first box on the lower left side. The value of success $M_{1,56613-1039855,67671967} = 84.9\,\%$ shows the probability for the survival of the part. The lower box in the middle displays the generated random numbers of geometric function based on parameter $k_{l,c,s,e}$. Unfilled fields have the value $k_{l,c,s,e} = 0$. The lower box on the right side shows the converted random numbers $k_{l,c,s,e} \in \{0, 1\}$. The values are only shown if the part e is built in the system s and if it is broken, such as $k_{1,1,56613-1039855,67671967} = 1$.

4.1.7 Procedure

In the optimization tool AIMMS different procedures have to be declared. The main procedures run the model to simulate the random values or to minimize or maximize the objective function. Other procedures can be created to automate calculation and analysis. For example, to initialize a high number of sets and parameters, AIMMS offers functions to read Excel sheets. Furthermore, the results can be written in Excel to use Excel tools for stochastic or other analyses. The simulation of the default parts is an upstream procedure to the main procedure to maximize the objective function.

4.1.8 User Interface

Within the optimization tool AIMMS offers the option to create a user interface for an easy handling for third parties. The interface allows a quick and easy initialization of the parameters, like CW and CV. Different buttons run all possible procedures. Therefore, third parties do not need any specific know-how of the model to change parameters or run procedures. Furthermore, it shows the important results of the optimization in overview. Figure 4.2 shows an example of the user interface after an optimization run.

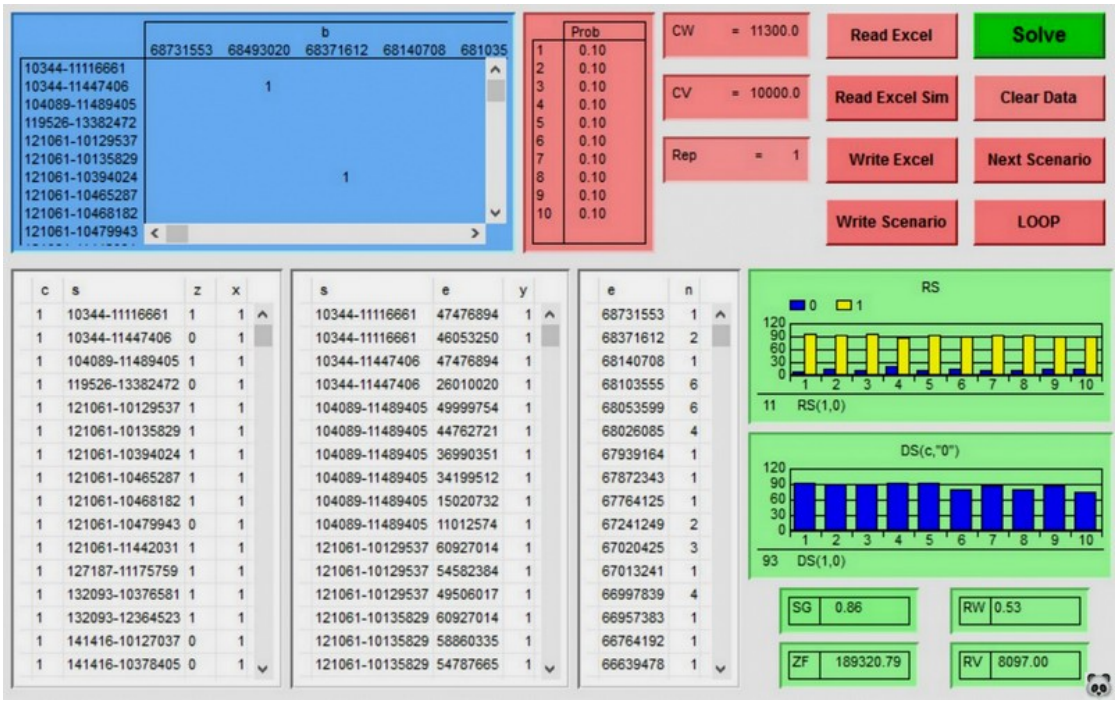

Figure 4.2: Example of the user interface "Calculation" with an optimum solution for the Real Data, with $CW = 11300$ and $CV = 10000$.

4.2 Model 2: "Flexible Parts to Systems Assignment"

The second version of the software tOPSPIN is a two-stage problem and more optimistic, based on the addition of index c to the binary variable $y_{c,s,e}$. In the first stage Model 2 decides which parts to include in the storage, given a collection of possible failure scenarios. In the second stage the actual failure scenario is revealed and the model optimally allocates the parts it chose in stage one. This subsection only describes the differences with respect to Model 1 in variables, constraints and objective.

4.2.1 Variables

$$x_{c,s,z} = \begin{cases} 1, & \text{if system } s \text{ in scenario } c \text{ has state } z \text{ after repairs are complete.} \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

$$y_{c,s,e} = \begin{cases} 1, & \text{if part } e \text{ is designated to repair system } s \text{ in scenario } c. \\ 0, & \text{otherwise.} \end{cases} \tag{4.11}$$

The state z of every system s in the different scenarios c is described in the binary variable $x_{c,s,z} \in \{0,1\}$ (4.10). The binary variable $y_{c,s,e} \in \{0,1\}$ (4.11) contains the storage of the optimal warehouse and shows which part e can be used to repair a system s in different scenarios c. The addition of index c allows the use of parts to repair the systems over the different scenarios. The total number of every part will be summed in the variable $n_e \in \mathbb{R}_+$.

4.2.2 Constraints

The following constraints define a feasible assignment of the warehouse storage and the current state of every system.

System State

$$\sum_{z \in \mathcal{Z}} x_{c,s,z} = 1 \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}, \tag{4.12}$$

$$x_{c,s,1} \leq y_{s,e} \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}, e \in \mathcal{E}, l \in \mathcal{L} : q_{s,e} = 1, b_{s,e} = 1, k_{l,c,s,e} = 1, \tag{4.13}$$

$$1 - y_{s,e} \leq x_{c,s,0} \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}, e \in \mathcal{E}, l \in \mathcal{L} : q_{s,e} = 1, b_{s,e} = 1, k_{l,c,s,e} = 1. \tag{4.14}$$

In Equation (4.12), the system $s \in \mathcal{S}$ has only one current state for every scenario $c \in \mathcal{C}$. Equation (4.13) represents that the system $s \in \mathcal{S}$ has the current state 1 (operational system), if the part is available in the warehouse and if the

following points meet the requirements: The part is built-in the system, the part is broken and the replacement is necessary to run the system. In Equation (4.14), the system $s \in \mathcal{S}$ has the current state 0 (non-operational system), if the part is not available in the warehouse and if the following points meet the requirements: The part is built-in the system, the part is broken, and the replacement is necessary to run the system.

Demand

$$\sum_{c \in \mathcal{C}, s \in \mathcal{S}, e \in \mathcal{E}} y_{c,s,e} \cdot u_e \leq n_e. \tag{4.15}$$

Equation (4.15) calculates the total demand ao all spare parts e. In contrast to the first version the total demand can be less or equal in case a scenario c results in a smaller demand compared to other scenarios.

4.2.3 Objective

$$\max \sum_{s \in \mathcal{S}, z \in \mathcal{Z}, e \in \mathcal{E}, c \in \mathcal{C}} o_c \cdot p_{s,z} \cdot x_{c,s,z} + 10^{-4} \cdot y_{c,s,e} - 10^{-6} \cdot n_e. \tag{4.16}$$

Also in Model 2, the primary goal of the objective function (4.16) is to maximize the number of high prioritized operational systems with an optimal storage of spare parts. On a subordinate point to fill the rest capacity with other spare parts, which are broken, but cannot repair the whole system. Finally, we subtract a small influence to take the smaller storage in case of a solution with the same value of the objective function.

5 Data

The following chapter explains the different data sets for the Real Data and Sim Data.

5.1 Real Data

The Real Data are based on records over past orders and storage movements in 2016 and 2017 of the logistic center of the German Armed Forces in Wilhelmshaven, Germany.

5.1.1 Sets

The set of systems s in the Real Data contains 108 systems. They are the result of a ranking of all 2643 systems and their replacements of parts based on the master data. The master data includes all systems and parts the are registered. The 108 systems have 7 to 36 built-in parts based on the last orders in 2016 and 2017. We have information only for parts which break down at least once. Information remaining all built-in parts, which possible can break down, are missing. The top 108 systems represent the systems in a possible deployment and are used for the basic analysis of the mathematical program. Because of secrecy towards unauthorized personal the data sets of systems and parts are encrypted. The identification number of a system is spit into two parts. The first digits describe the type of vehicle. The second digits describe the equipment of every system. The possible storable parts are based on the last orders of the top 108 systems, which corresponds to a total value of 708 parts. The whole logistic center holds up to 18062 different parts in weight categories up to 17600 kg. 8322 parts have no weight information on file. The leftover 9740 parts with stored weight information contains 87 % of parts with a weight of less than 3.8 kg, shown in Figure 5.1.

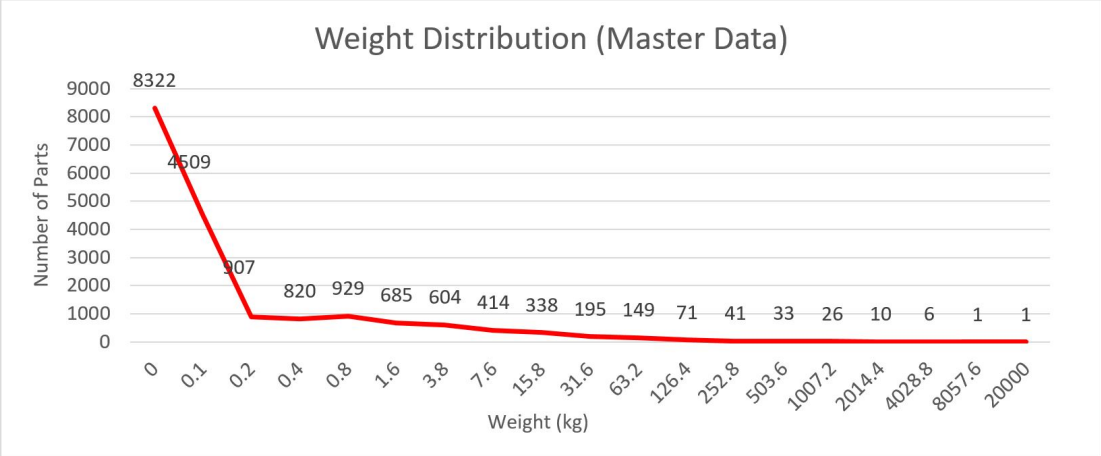

Figure 5.1: Weight distribution of Master Data.

Figure 5.2 shows that the 708 parts of the top 108 systems have a similar weight distribution to the master data, with a heaviest part of 3600 kg.

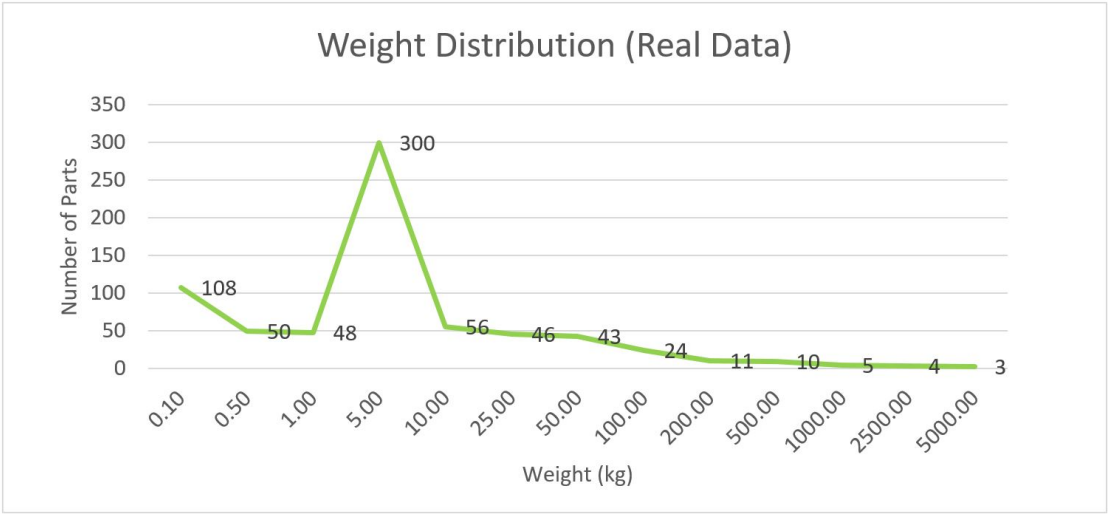

Figure 5.2: Weight distribution of Real Data.

5.1.2 Parameter

The binary parameter $b_{s,e} \in \{0, 1\}$ contains the information about the built-in parts for every system. A binary matrix over all 108 systems and 708 parts de-

clares a 1 for a built-in part e in system s and a 0 for non-built-in parts. The top 108 systems have different priorities in a deployment and are separated into combat and support vehicles. This information is stored in the parameter $p_{s,z}$ and can take values from 4 (highest priority) to 1 (lowest priority). The priorities of 4 and 3 are for combat vehicles and priorities 2 and 1 for support vehicles. If more systems of a type are on deployment, one half of the group get the higher priority to make sure that at least 50 % of the systems of any group are preferably repaired. The priorities can set for every state z of a system s. The parameter $q_{s,e}$ describes the necessity of every part for a system. Some parts are important to run a system at all, some are useful to have but not necessary. Currently all parts are important for their systems. Every system contains different quantities of a part. This information is stored in the $u_e \in \mathbb{N}$ and are based on the quantity of last orders for every system in the last two years. The parameter $w_e \in \mathbb{R}^+$ and $v_e \in \mathbb{R}^+$ contains the weight and volume of any part e. The information of both parameters are based on the master data for every part. Missing weight information is randomly picked using all master data information to generate a similar weight distribution. Caused by a higher amount of missing and misleading volume information, this parameter is set for all parts equal and does not affect the mathematical program. The weight and volume capacities are set in parameters CW and CV. The weight capacity for a possible deployment is 100000 kg. The volume capacity is set high enough to contain all parts as has no influence on the objective function. The parameter o_c gives a probability of occurrence for every scenario. Currently, no explicit scenarios are set. Therefore, the parameter o_c has the equal probability of occurrence $o_c = 1/c$ for every scenario. The scenarios are generated by a simulation. The parameters of the triangular function TL_e, TH_e and TB_e are based on the last orders of parts from the top 108 systems. If parts were ordered more frequently, the borders of the probability of success in the triangular function are lower valued than the borders for infrequently ordered parts. The calculated probabilities of success based on the data base correspond to the higher bound TH_e of the triangular function. The lower bound TL_e contains probabilities with 20 % less than the values of TH_e in order that the simulation results in more broken parts that primary were ordered. The probability of a part, to not break down, lies in a range of 63.3 %-91.7 %. The

displacement factor $TB_e = 0.5$ is equal for every part and generates a balanced movement between the borders. A comparison with the results of the simulation and the calculated probabilities of the data base shows, that a displacement factor of $TB_e = 0.5$ and a wide between the higher and lower bound of 20 %, leads to lower probabilities of success with a mean of 6.83 % lower success and a standard deviation of 3.15 %. Standard deviation is a measure that is used to quantify the amount of variation or dispersion of a set of data values. If the distance between the borders is 10 % the mean gets smaller by 3.24 % less success as well with a less standard deviation with 3.07 %. Overall, in the calculation more parts are broken with a mean of 6.83 % than the data base yield.

5.2 Sim Data

Because of the fact that only parts which break down once are accounted by the parameter $b_{s,e}$, the settings of Sim Data represent a more realistic number built-in parts per system. The Sim Data gives another point of view of the mathematical program and makes the results more comparable. The Sim Data are fictitious and are not based on any previous data.

5.2.1 Sets

The Sim Data settings contain an total number of 100 systems and 700 parts. The 100 systems are split in equal groups of 5 systems, like trucks, tanks, or airplanes. Every system has 53 built-in parts. Every part is built-in at least in one group of systems. The weight of the parts have a similar distribution with more low-weighted parts and a few heavy parts, simular to the Real Data. The heaviest part of the Sim Data has a weight of 5000 kg.

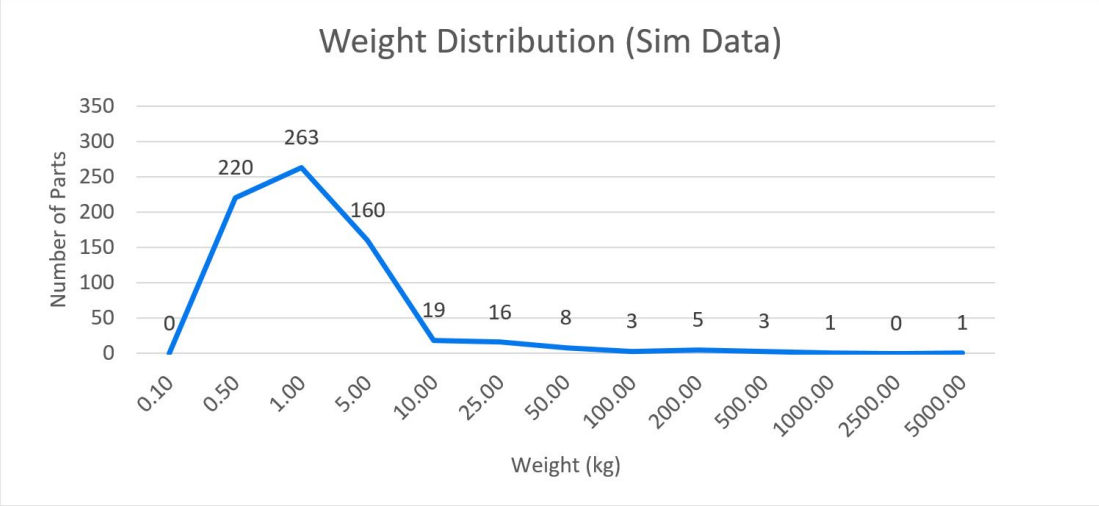

Figure 5.3: Weight distribution of Sim Data.

5.2.2 Parameter

The matrix of the parameter $b_{s,e}$ is more dense with a higher number of built-in parts per system and leads to more solving steps in the mathematical program. The groups of 5 systems are equally divided into combat and support vehicles. The parameter $p_{s,z}$ has the same possible values of priority, 4 (highest priority) to 1 (lowest priority), like the Real Data. The first 3 vehicles of a group get a higher priority than the last 2 vehicles, to make sure at least 50 % of a group have a value to the objective function to be repaired. The necessity of every part for a system in parameter is equal with $q_{s,e} = 1$. The information over every unit size per part in parameter u_e varies between 1 and 4 parts per unit and depends on the weight of the part. Mostly, a big and heavy part is built-in only once, in contrast to wheels or screws. The weight per part w_e has the same distribution, like the Real Data. The parameter w_e contains more low-weighted parts and a few heavy parts. The volume v_e is disregarded and set to a value of $v_e = 1$ for every part. The influence of the generated scenarios to the objective function are equal for every scenario with a setting of parameter $o_c = 1/c$. The parameters of the triangular function TL_e, TH_e and TB_e are not based on collected data. To keep the Sim Data simple, all parts have the same probability to success. Success

means to have an operational state after the simulation. The lower - TL_e- and upper - TH_e - bound have a range from 30 % to 70 % and an equal movement within the bounds of $TB_e = 0.5$.

6 Results

This chapter shows different analyses of both models of tOPSPIN and data sets to investigate the behavior of the optimal storage of a warehouse. The results give an assessment of the important parameters, which influence the simulation and the calculation of an optimal storage.

6.1 Level of Service

The parameter SG displays a level of service for every optimization run. It shows the percentage of parameter $RS_{s,z=1}$ over all systems s, reflecting the performance of an optimal warehouse. The first analysis tests the changes of the SG for different weight capacities for both data sets.
Figure 6.1 shows the service levels for Model 1 and Model 2 for different weight capacities based on the dataset Real Data. For both models, the service levels of the Real Data decreases really slow compared to the weight capacities. Even with a weight capacity of 2825 kg, 2.5 % of the total weight, both models can repair up to 71 % of the broken systems. The high level of service with a low weight capacity is caused by the high range of low-weighted parts in the dataset. For example, 71 % of all parts in Real Data have a weight up to 5 kg, therefore the models can store a lot of parts with a small weight capacity. In comparison of the models, the two-staged Model 2 can reach a higher service level for all weight distributions.

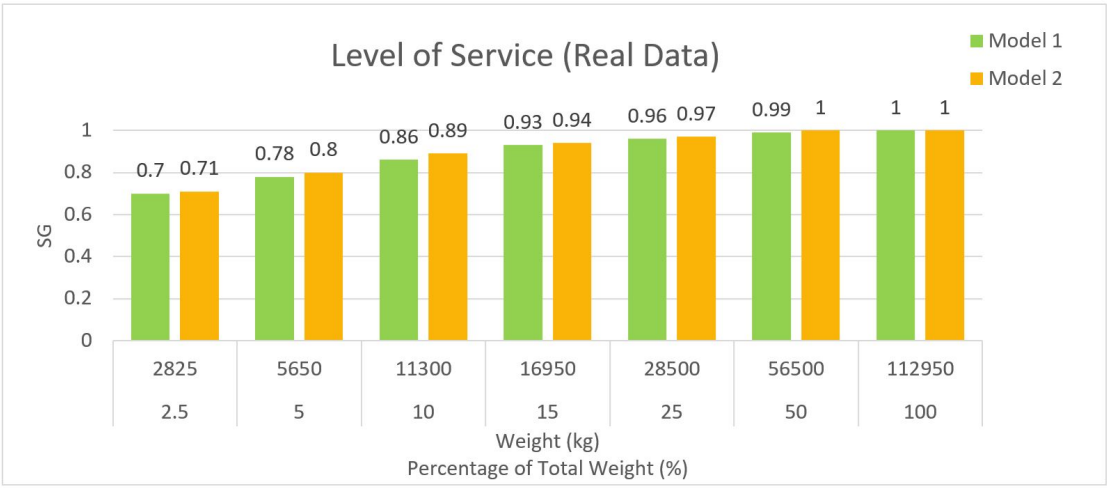

Figure 6.1: SG of Real Data for different weight capacities for Model 1 and Model 2.

The Sim Data, shown in Figure 6.2, decrease in their level of service faster than the Real Data. With a weight capacity of 3500 kg, 2.5 % of the total weight, Model 2 can repair only up to 30 % of the broken systems, and Model 1 even less with a service level of 22 %. To have a service level close to the lowest service level of the Real Data, it needs at least a weight capacity of 10 % of the total weight for Model 2 and at least 15 % for Model 1. This effect is caused by the higher amount of parts per system. After the simulation, more parts have to be replaced which leads to a higher demand of weight capacity.

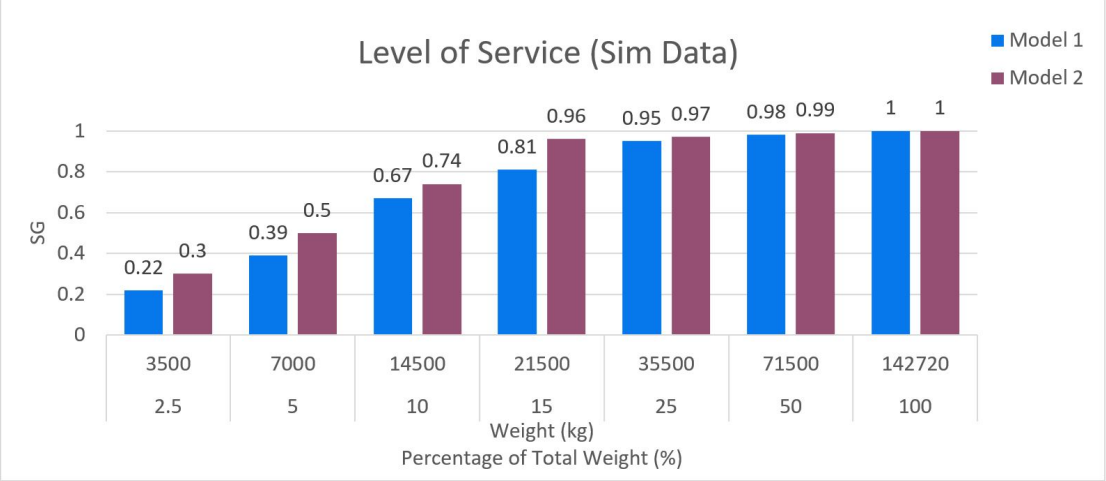

Figure 6.2: SG of Sim Data for different weight capacities for Model 1 and Model 2.

6.2 Number of Scenarios

The following analysis shows how many scenarios c are necessary to get a representative result for an optimal warehouse in an acceptable solving time based on Model 1. To show if the solution is representative, Model 1 solves an additional $c + 1$ scenario after the original optimization. If the storage of the warehouse is optimal, the next scenario is solved as good as the other scenarios before. The analysis, shown in Figure 6.3, was implemented over 10 repetitions for all different weight capacities CW and number of scenarios. The analysis for the Real Data shows that the number of necessary scenarios depends on the weight capacity. For a capacity of 2825 kg, 2.5 % of the total weight of all parts, the mean of $RS_{s,z=1}$ of the $c + 1$ scenario, for an optimal solution with 5 solved scenarios before, is under 40 $RS_{s,z=1}$ compared to the optimal solution with a mean of 65 $RS_{s,z=1}$. For the 2825 kg weight capacity it needs as least 50 scenarios for the calculation to get a representative result to solve the $c + 1$ scenario with a close number of $RS_{s,z=1}$ compared to the service level of the optimal solution. If the weight capacity is higher, fewer scenarios are necessary to get a representative result. For a weight capacity of 56500 kg, 50 % of the total weight, it only needs at least 15 scenarios to get a representative result of $RS_{s,z=1}$ around $RS_{s,z=1} = 107$.

With a higher weight capacity the warehouse stores more parts and can more easily solve new upcoming scenarios.

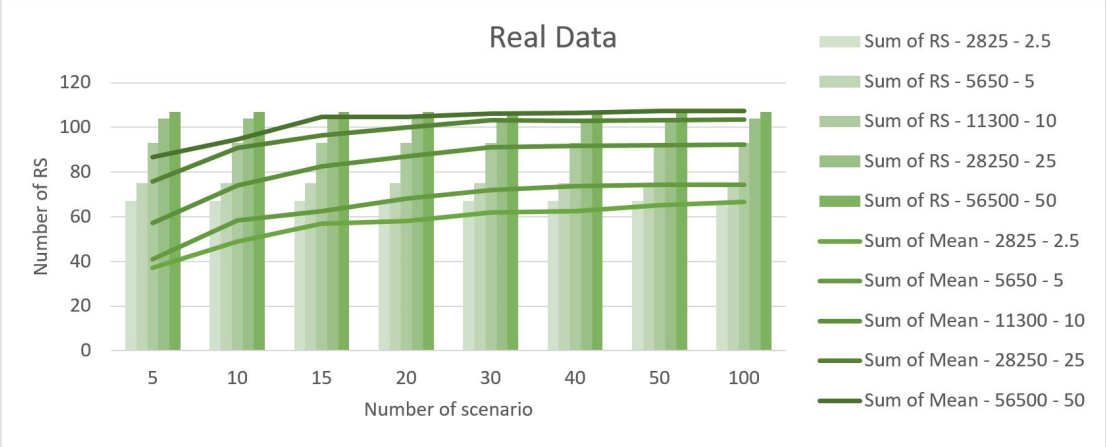

Figure 6.3: Analysis of necessary numbers of scenarios with weight capacity and total weight percentage for Model 1.

The analysis was implemented over 10 repetitions for each weight capacity CW and number of scenarios c. Figure 6.4 shows the standard deviation to the mean of $RS_{s,z=1}$ for the 10 repetitions for the different CW and c. The standard deviation stabilize with a higher number of scenarios and a higher CW.

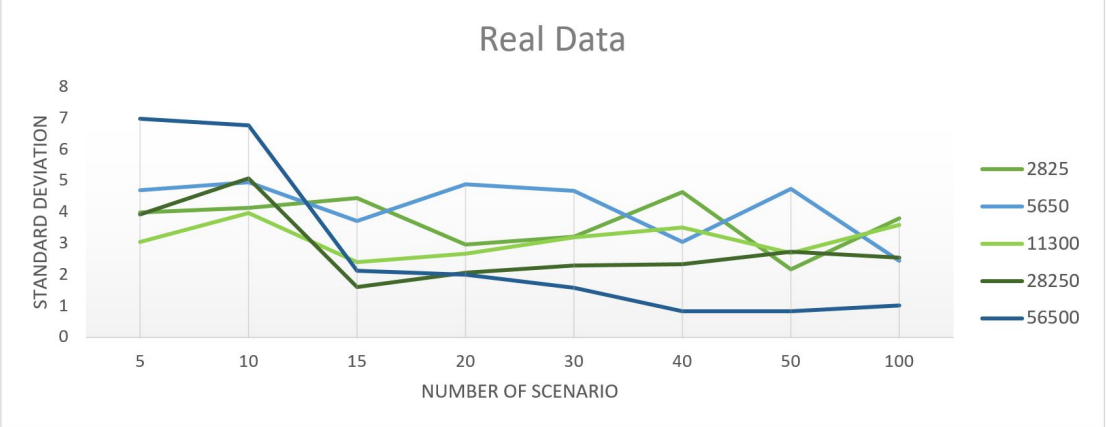

Figure 6.4: Standard deviation for 10 analysis repetitions over different CW - total weight percentage for Model 1.

Similar, the analysis of the Sim Data in Figure 6.5. The main difference between the Real Data and the Sim Data is the built-in parts per system and the distribution. The analysis shows that the equal distribution of the built-in parts have an impact on the number of necessary number of scenarios. For a $CW = 3500$, 2.5 % of the total weight, it only needs a number of 15 scenarios to get a representative result, compared to the need of 50 scenarios of the Real Data analysis. For a 50 % capacity of the total weight leads even only 5 scenarios to a representative result. The comparison shows, that the number of necessary scenarios depends on the weight capacity and furthermore on the distribution of the built-in parts.

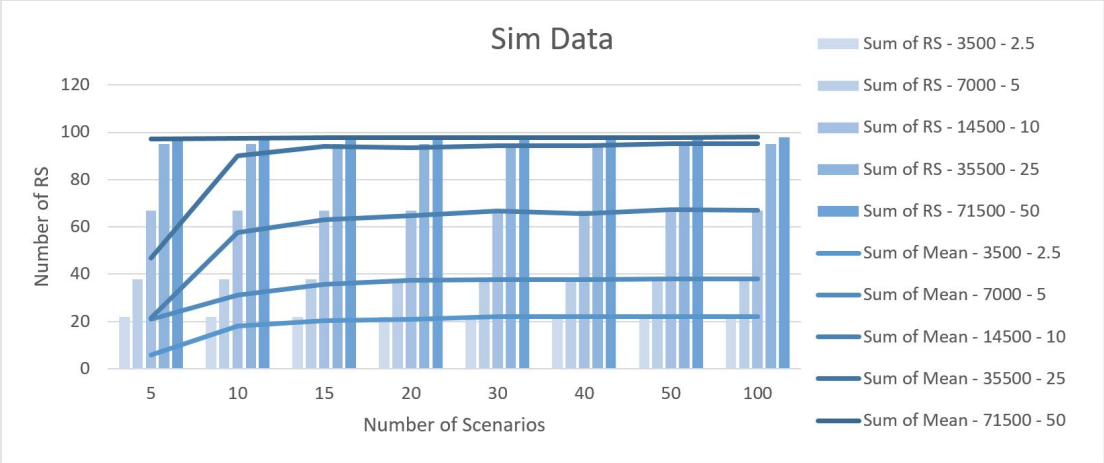

Figure 6.5: Analysis of necessary numbers of scenarios with weight capacity and total weight percentage for Model 1.

The analysis of the standard deviation (6.6) underlines the impact of the distribution of the built-in parts. With a equal distribution of all parts the standard deviation decreases fast to a low level for every CW.

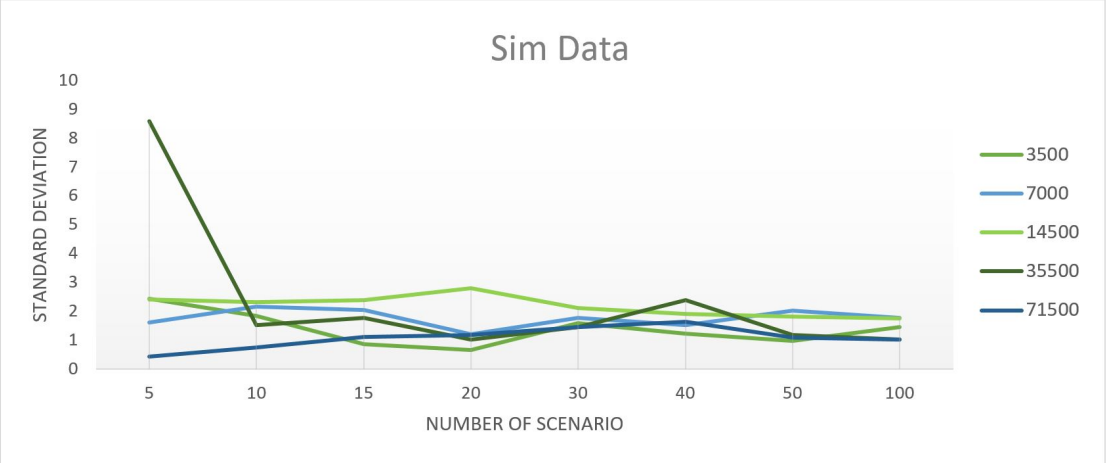

Figure 6.6: Standard deviation for 10 analysis repetitions over different CW - total weight percentage for Model 1.

6.3 Solving Time

Figure 6.7 shows the solving time for an optimal storage over different number of scenarios, datasets and models. The CW for both sets are set by 10 % of their total weights with $CW = 11300$ kg and $CW = 14500$ kg. The solving time raises for both datasets with a higher number of scenarios from 1 up to nearly 60 second (sec) for Model 1. In comparison with the necessary number of scenarios shown in Figures 6.3 and 6.5 the solving time acceptable with under 60 sec. The two-staged Model 2 takes more time to calculate the optimal storage, caused by a higher number of constraints and variables based on adding the index c to variable $y_{c,s,e}$. For example, the solving time of Model 2 for Sim Data with a weight capacity of 14500 kg and 50 scenarios took over 5 hours. To reduce the solving time, the calculates were terminated at a gap of 0.01 % of the optimal objective value. The stopping criteria decreases the solving time down to 17 minutes for $CW = 14500$ and $c = 200$ for an almost optimal solution.

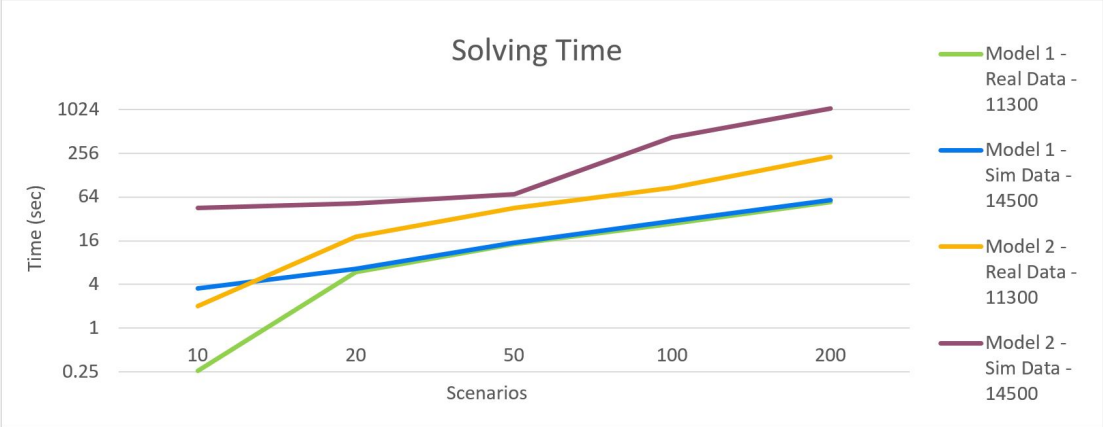

Figure 6.7: Solving time for Real Data and Sim Data with a weight capacity of 10 % over different numbers of scenarios and both models.

6.4 Home/Worst Case Analysis

The following analysis shows the solutions of the optimization Model 1 and the weight distribution of a "home"– and "worst-case"–point of view. The "home"–settings represent the conditions of national operations. Those conditions are predictable and well-known on basis of long-term views. The national territory and military bases have a good infrastructure and gentle weather conditions. Consequently, all parts break less. This factor can be taken into account over the parameter TB_e. The parameter describes the displacement factor between the lowest and highest value of the triangular function. A value of $TB_e = 0.99$ causes a shift to the higher value of the function and leads to a higher probabilities of success for the simulation. Therefore, less parts get broken during the simulation. In contrast, the "worst-case"–point of view simulates the conditions in a deployment with bad infrastructure and weather conditions and hostile attacks. A value of $TB_e = 0.01$ causes a shift to the lower bound of the triangular function to produce more broken parts in the simulation. Figure 6.8 shows the number of $DS_{s,z=0}$ after the simulation and the $RS_{s,z=1}$ after the optimization with $CW = 11300$, $TB_e = 0.99$ and $c = 50$. From the origin, 108 systems 70 to 90 systems are broken after the simulation. On average, the optimization can replace 80 % of the systems with a weight capacity of 11300 kg.

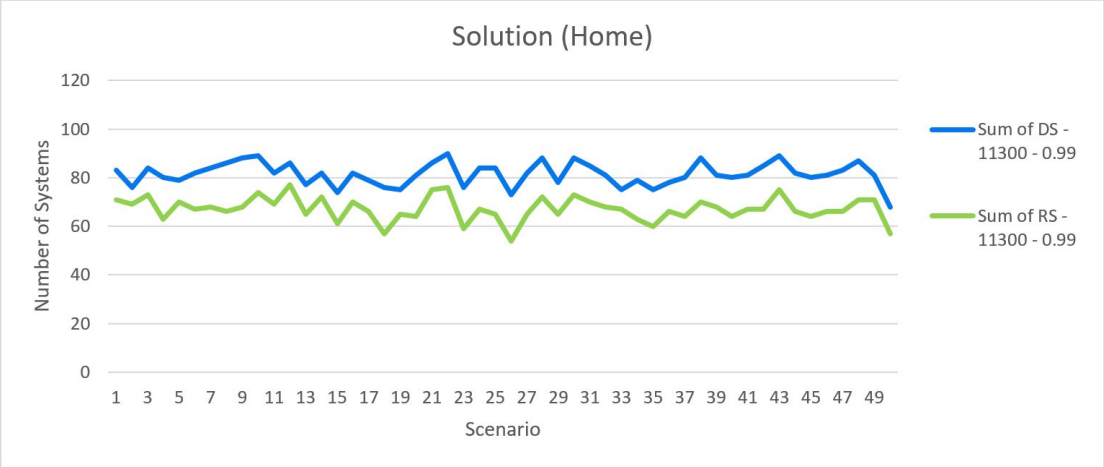

Figure 6.8: Optimal solution of Real Data with $CW = 11300$, $TB_e = 0.99$, $c = 50$ in Model 1.

Figure 6.9 shows the related weight distribution of all stored parts after the optimization and the ordered quantity of all weight groups. Compared to the weight distribution of all parts (Figure 5.2) less heavy parts are stored in the warehouse. In a case of a lower weight capacity the optimization stores fewer or no heavy parts up to 500 kg to "save" the capacity for broken systems with low-weighted parts.

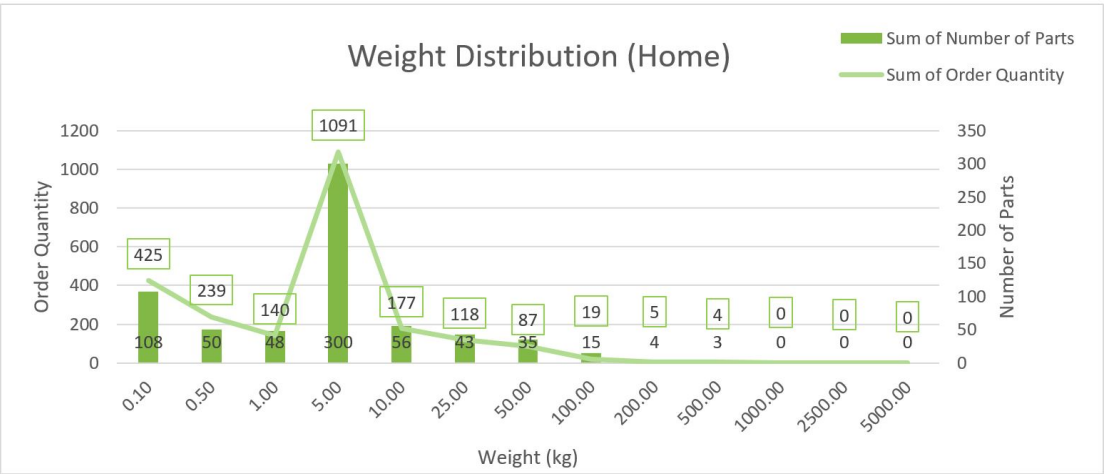

Figure 6.9: Weight distribution of "home"–solution for Real Data.

In the "worst case"–point of view, Figure 6.10 shows the expected, higher amount of $DS_{s,z=0}$ with 85 to 100 systems. On average, the optimization can replace 70 % of the systems with a weight capacity of 11300 kg.

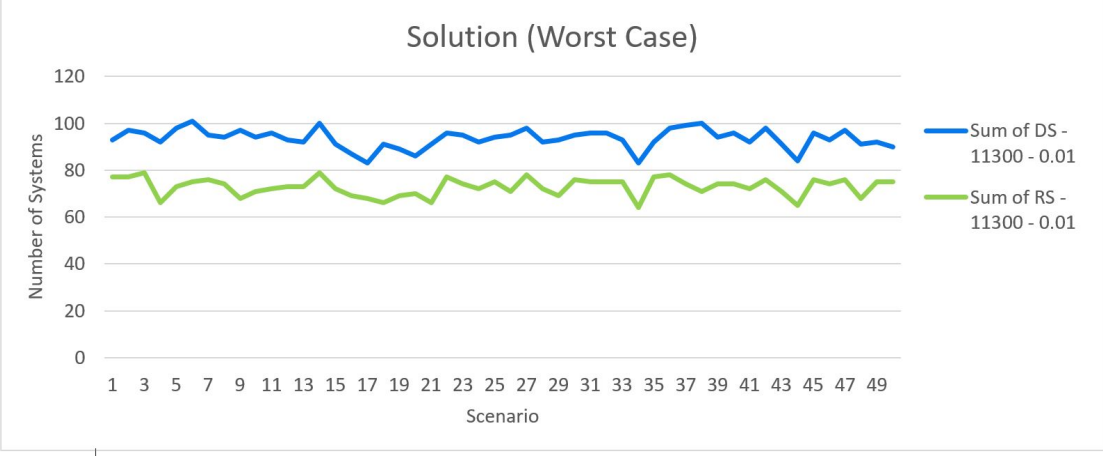

Figure 6.10: Optimal solution of Real Data with $CW = 11300$, $TB_e = 0.01$, $c = 50$ in Model 1.

The related weight distribution (Figure 6.11) shows that the stored parts in the "worst case"–scenario are even less heavy and contains only parts up to 200 kg. Instead, the "saved" weight capacity gets used for more parts up to 50 kg.

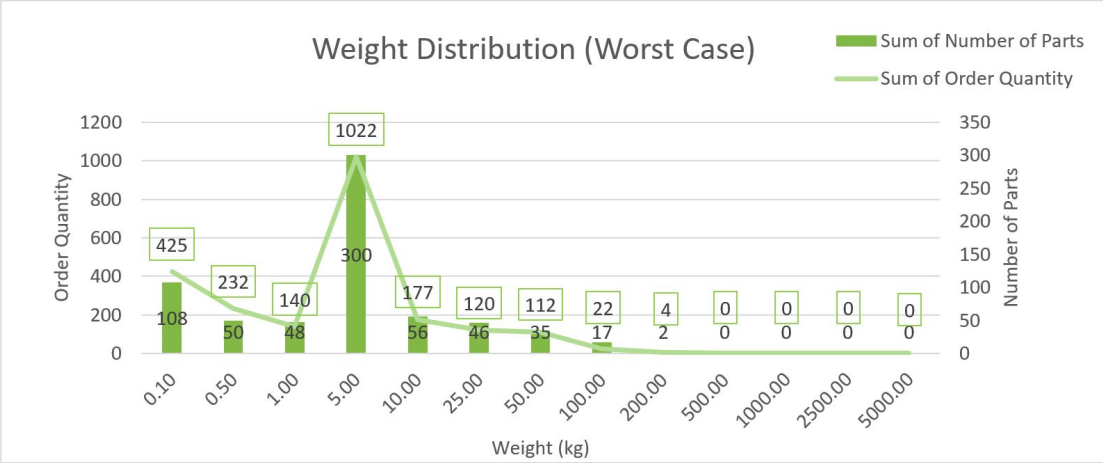

Figure 6.11: Weight distribution of "worst case"–solution for Real Data.

In contrast, we shall compare the "home" and "worst case" analysis of the Sim Data. The equal distribution of the built-in parts per systems causes the effect of total break-down of all systems in the "home"–scenario. With a number of 53 parts per system, even a shift to the higher bound of the triangular function and a resulting higher probability of success for not-breaking down, the probability for at least one broken part was too high. For that reason the analysis runs over two different bounds of the triangular function to show the difference between the "home"– and "worst case"–scenarios. The first run simulates with a $TL_e = 0.3$ and $TH_e = 0.7$ and the second run with $TL_e = 0.9$ and $TH_e = 1$. The first analysis shows a total break down, $DS_{s,z=0} = 100$, of all 100 systems. At least one built-in in every systems fails and has to be replaced. With the weight capacity of 14500 kg, 10 % of the total weight, 60 to 77 systems can be repaired in different scenarios.

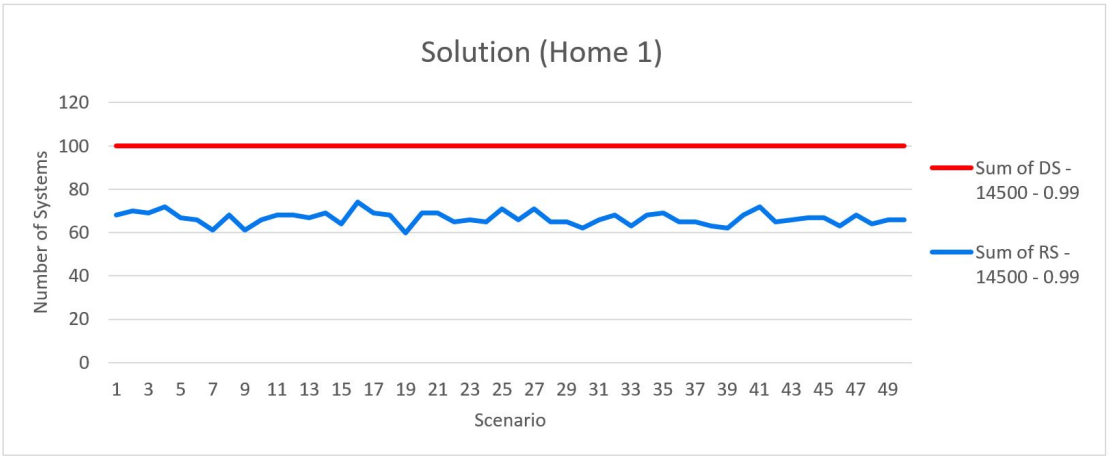

Figure 6.12: Optimal solution of Sim Data with $CW = 14500$, $TB_e = 0.99$, $TL_e = 0.3$, $TH_e = 0.7$, $c = 50$ in Model 1.

The weight distribution in Figure 6.13 shows compared to the total weight distribution in Figure 5.3 the same behavior like the Real Data. The optimized warehouse contains mostly light parts and no heavy parts over 200 kg.

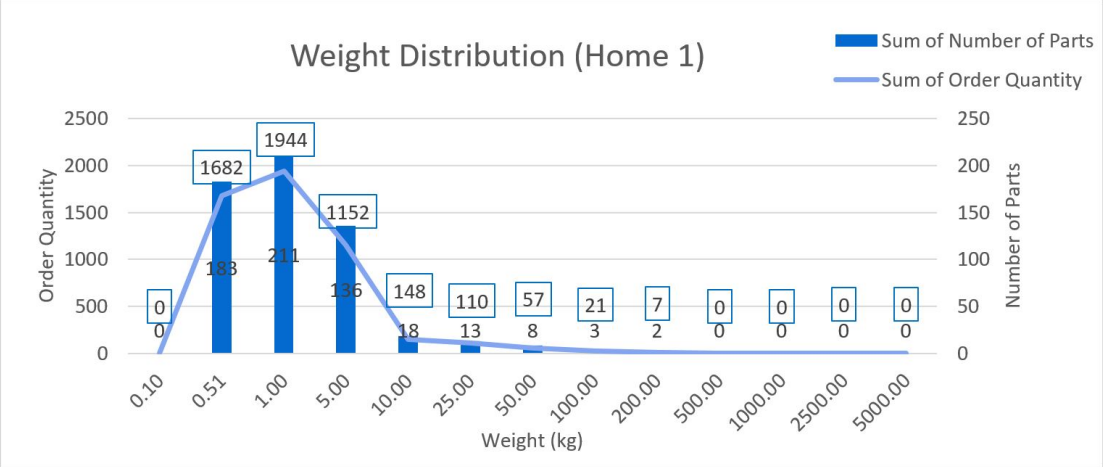

Figure 6.13: Weight distribution of "home 1"–solution for Sim Data.

The "worst case" analysis of the first run (Figure 6.14) illustrates the need for a second analysis with different borders of the triangular function. Likewise, all 100 systems break down after the simulation. However, the number of repaired systems is lower with a range of 58 to 68 systems, caused by the shift to the lower bound of the triangular function. Therefore, in total more broken parts per systems have to be replaced.

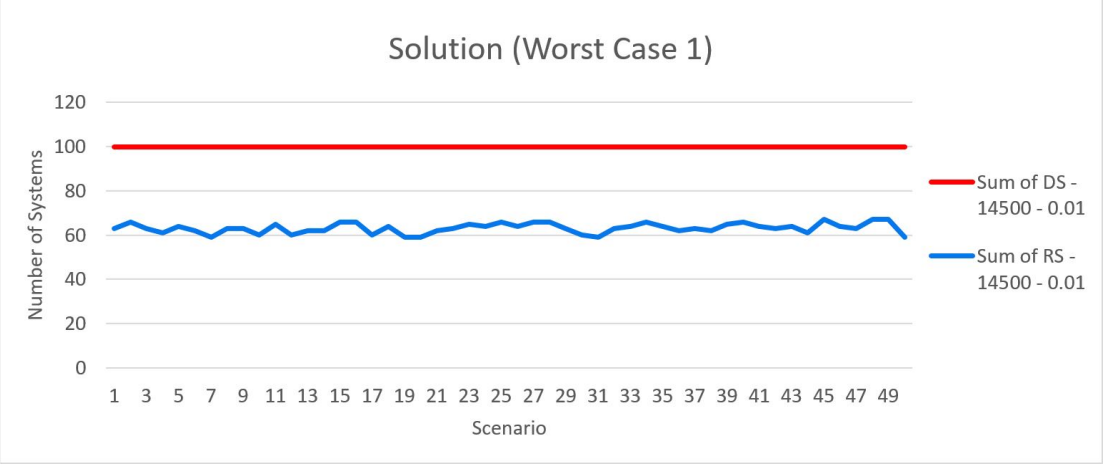

Figure 6.14: Optimal solution of Sim Data with $CW = 14500$, $TB_e = 0.01$, $TL_e = 0.3$, $TH_e = 0.7$, $c = 50$ in Model 1.

The weight distribution in Figure 6.15 changed to a smaller number of low-weighted parts and includes parts up to a weight of 500 kg. With a higher number of broken parts even more low-weighted parts were expected, but this depends on the broken parts in every simulation. If a heavy part of an important system gets broken in every scenario, it can increase the value of the objective function to store it in the warehouse.

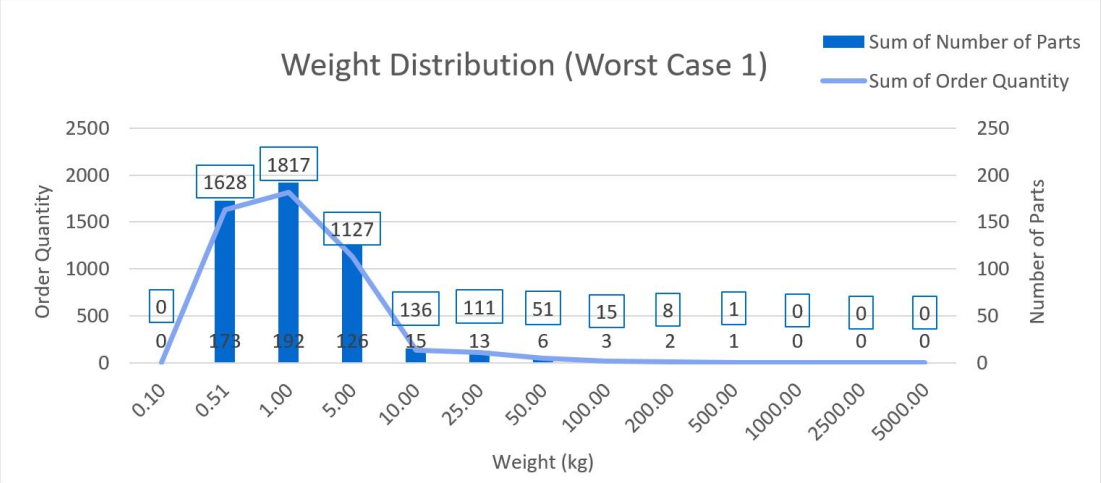

Figure 6.15: Weight distribution of "worst case 1"–solution for Sim Data.

Figure 6.16 shows the second run of the "home"–scenario. Because of the higher valued borders of the triangular function, fewer systems get broken after the simulation. The total number of broken parts is so low, that nearly every system can be repaired with an optimal storage.

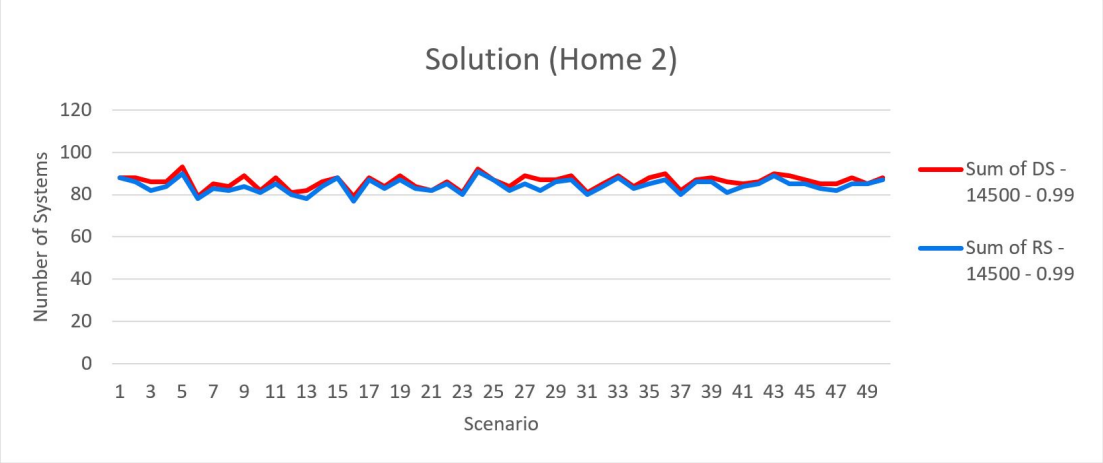

Figure 6.16: Optimal solution of Sim Data with $CW = 14500$, $TB_e = 0.99$, $TLe = 0.9$, $TH_e = 1$, $c = 50$ in Model 1.

Figure 6.17 shows, that a lower total number of broken parts allows two heavy parts in the 200 to 500 kg group in the storage. If all broken systems are repaired or weight capacity is left, the model fills up the warehouse with other parts. This leads to even a higher amount of small parts, to decrease the value of the objective function.

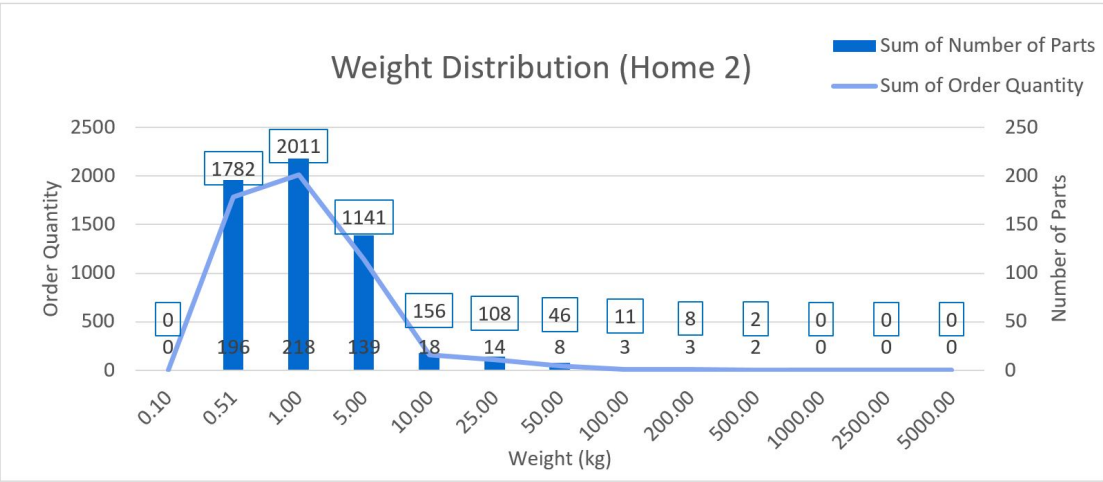

Figure 6.17: Weight distribution of "home 2"–solution for Sim Data.

The "worst case"–scenario, shown in Figure 6.18, of the second run of the analysis

has more broken systems after the simulation in contrast to the "home"–scenario (6.16), but still has some running systems compared to the first "worst case"-scenario with $TL_e = 0.3$ and $TH_e = 0.7$. With less broken parts per system the optimized storage can repair a lot of systems even after a simulation of a "worst case"–scenario.

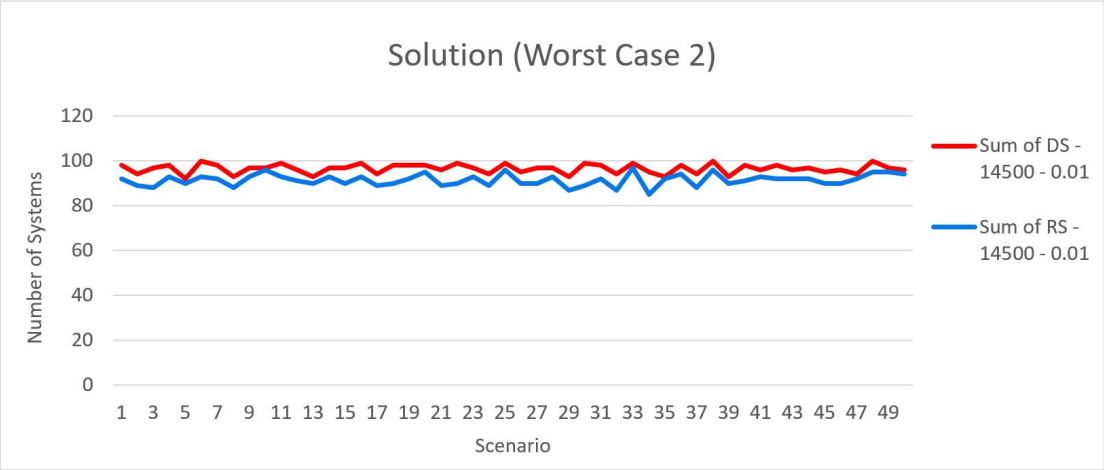

Figure 6.18: Optimal solution of Sim Data with $CW = 14500$, $TB_e = 0.01$, $TL_e = 0.9$, $TH_e = 1$, $c = 50$ in Model 1.

The weight distribution (Figure 6.19) clearly shows a decrease of low-weighted parts with up to 2384 parts in the 0.5 to 1 kg group and only one heavy part in the 100 to 200 kg group. The opposite to the weight distribution in Figure 6.15 happened and more different, low-weighted parts failed. The value of the objective function decrease for more small parts.

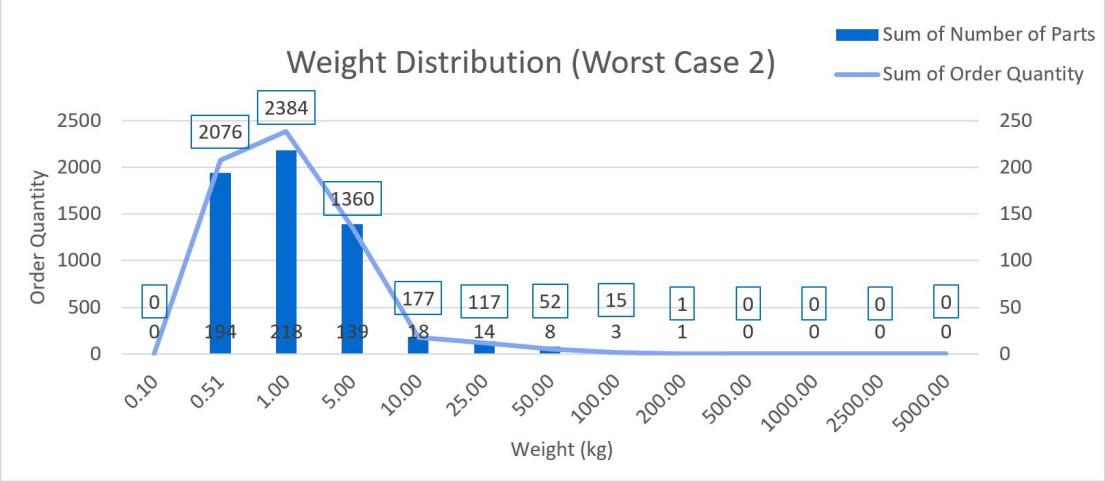

Figure 6.19: Weight distribution of "worst case 2"–solution for Sim Data in Model 1.

The analysis of the "home" and "worst case" point of view shows, that the number of $DS_{s,z=0}$ and $RS_{s,z=1}$ depends on the built-in parts distribution. All "worst case" analyses displays different weight distributions, depending on their distribution of broken parts after the simulation. Therefore, a large number of scenarios is even more important for a representative result.

6.5 Fixed Warehouse

The previous analyses show that more low-weighted parts are stored if the weight capacity increases. The next analysis shows what happens if only a general number of low-weighted parts were stored to save time and money for a mathematical optimization. Therefore, a number of scenarios were simulated and the results based on a fixed and a optimal storage were displayed. The first analysis runs with a fixed storage with a quantity of 3 units for every part up to 20 kg. The total weight for the fixed storage is 9210 kg. The total number of parts up to 20 kg is 600, 85 % of all parts. The model solves the optimization with the same weight capacity of 9210 kg. All runs are going over 100 scenarios. For the Real Data, Figure 6.20 shows the result of the analysis. After the simulation up to 99 systems are broken. With a optimized storage 80 % to 90 % of the broken

systems can be repaired. Only 50 % to 70 % of the broken systems can be repaired by the fixed storage. On average, 15 systems less were repaired.

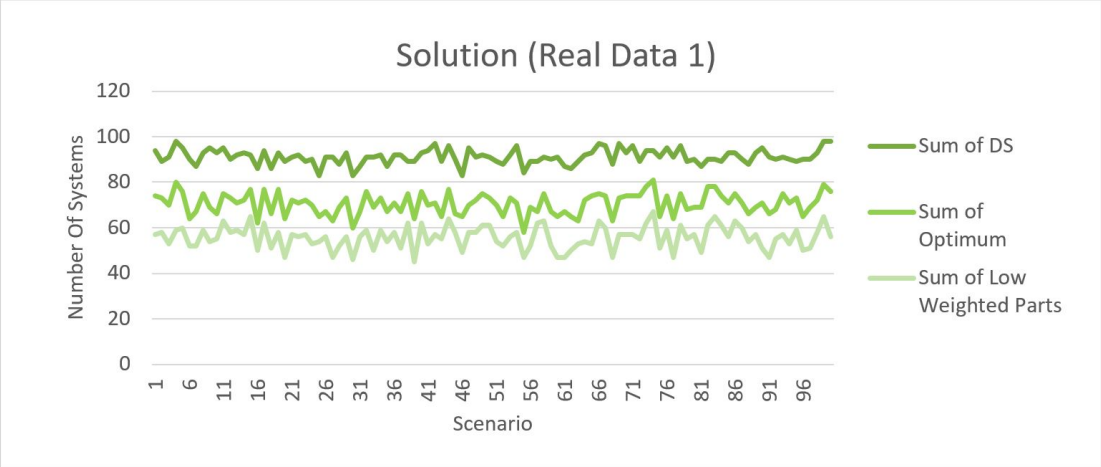

Figure 6.20: Performance comparison of optimal and fixed warehouse of Real Data 1 with low-weighted parts up to 20 kg in Model 1.

The second analysis, shown in Figure 6.21, verifying the behavior of a fixed storage with a weight up to 10 kg. Per part 5 unit are stored and arise to a total weight of 10052 kg. The optimal storage with a weight capacity of 10052 kg can repair 80 % to 90 % again. On average, the fixed storage can repair 20 systems less than the optimal storage. Therefore, a fixed storage up to 20 kg with less parts performs better than a fixed storage with more parts up to 10 kg.

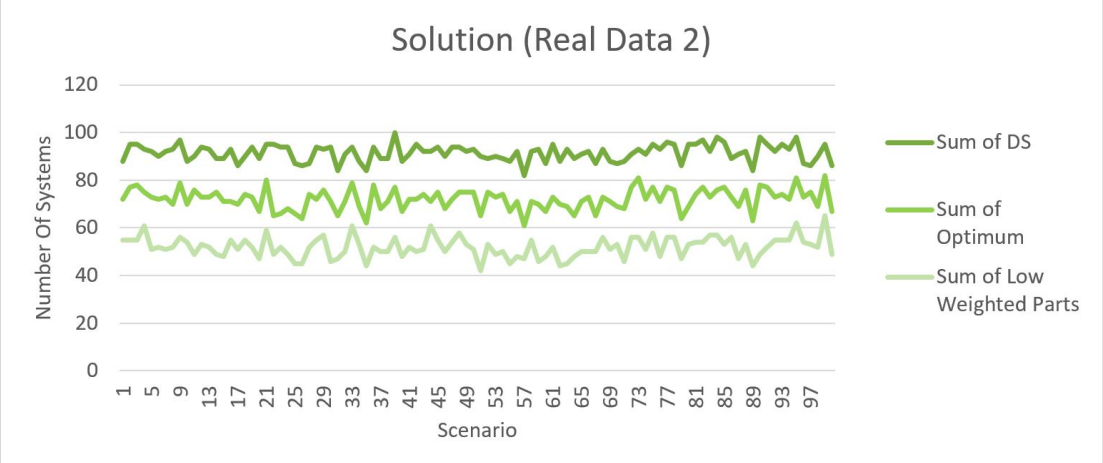

Figure 6.21: Performance comparison of optimal and fixed warehouse of Real Data 2 with low-weighted parts up to 10 kg in Model 1.

For the Sim Data, more units are stored in the fixed storage for both runs, caused by the higher number of parts per system. For the first analysis 6 units per parts up to 20 kg are stored with a total weight of 9210 kg. Parts with a weight up to 20 kg corresponds to 96 % of all parts. Figure 6.22 shows the following results: On average, 90 of 100 systems are broken after the simulation. An optimal storage can repair 75 % to 85 % of them. The fixed storage only reaches 40 % to 50 % repaired systems. That corresponds to an average of 29 less systems to the optimal solution.

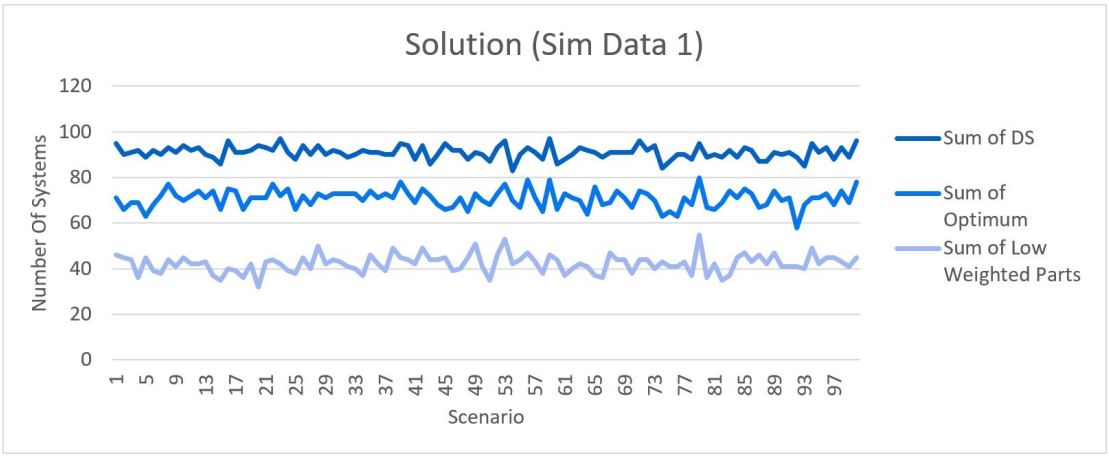

Figure 6.22: Performance comparison of optimal and fixed warehouse of Sim Data 1 with low-weighted parts up to 20 kg in Model 1.

The second run of the analysis for the Sim Data tested a fixed storage with 10 unit per part up to 10 kg. The total weight for the fixed storage and the capacity is 13280 kg. The results, shown in Figure 6.23, reveal the opposite behavior to the analysis of the Real Data. A fixed storage still leads to less repaired systems in contrast to the optimal solution, but the analysis shows that the deviation is smaller with the settings of the fixed storage. On average, 17 systems less were repaired compared to the optimal solution. However, the service level of the fixed storage is still lower with 63 % to 74 % of repaired systems compared to a service level 80 % to 90 % repaired systems with the optimal storage.

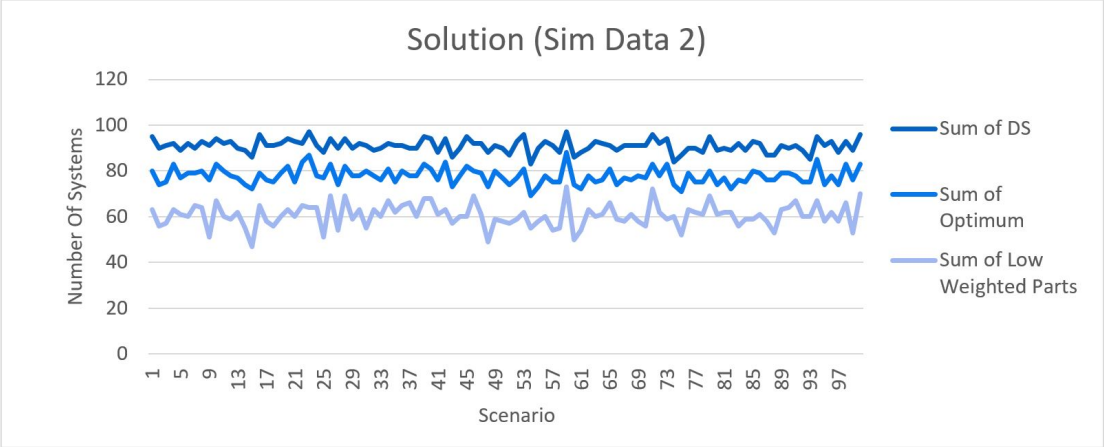

Figure 6.23: Performance comparison of optimal and fixed warehouse of Sim Data 1 with low-weighted parts up to 10 kg in Model 1.

Overall, a simple fixed storage with a general quantity of selected parts can reach service levels up to 74 %. The results changes with the settings of the fixed storage and the underlying set of data. However, all analyses still have high disadvantages to an optimized storage. In the end it is a decision made on priorities of time, money, and service level.

6.6 Comparison of Storage

Based on the different service level of both models, the question occurs: which parts are stored that increase the service level of Model 2? The analysis of this chapter shows the weight distribution for different weight capacities for both models and datasets to compare them.

6.6.1 Real Data

For a small weight capacity of 2825 kg, Figure 6.24 shows the weight distribution of the optimal storage calculated by Model 1. The storage includes mostly low-weighted parts up to 50 kg and peaks of the order quantity in the weight groups 0.1 to 0.5 kg and 1 to 5 kg. Moreover, 108 different parts are stored with a weight up to 0.1 kg. The service level of this storage is $SG = 0.70$.

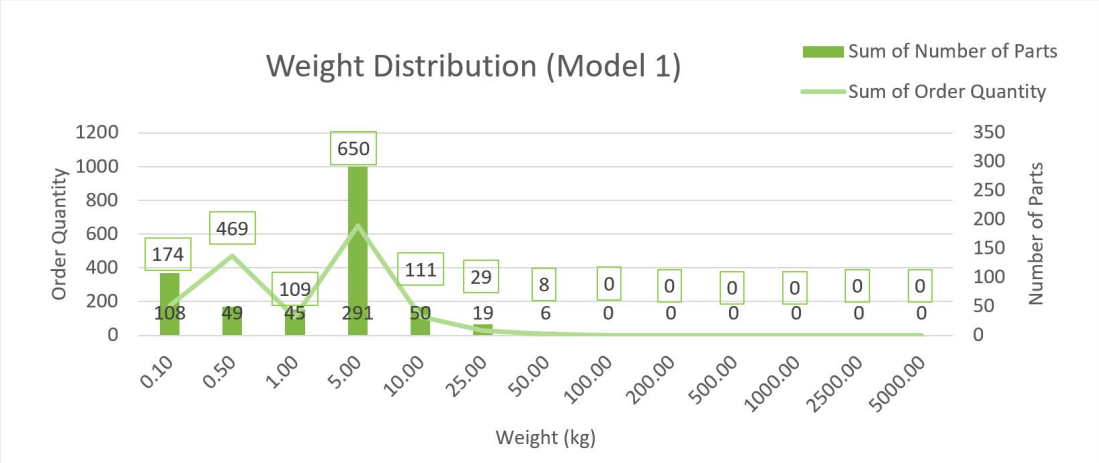

Figure 6.24: Weight distribution of Real Data with $CW = 2825$ by Model 1.

Figure 6.25 shows the weight distribution of the optimal storage calculated by Model 2. For the same weight capacity of 2825 kg, the optimal storage shows a different distribution with a single peak in the weight group 1 to 5 kg. Instead, more heavy parts are stored up to 100 kg. The service level of this storage is $SG = 0.71$.

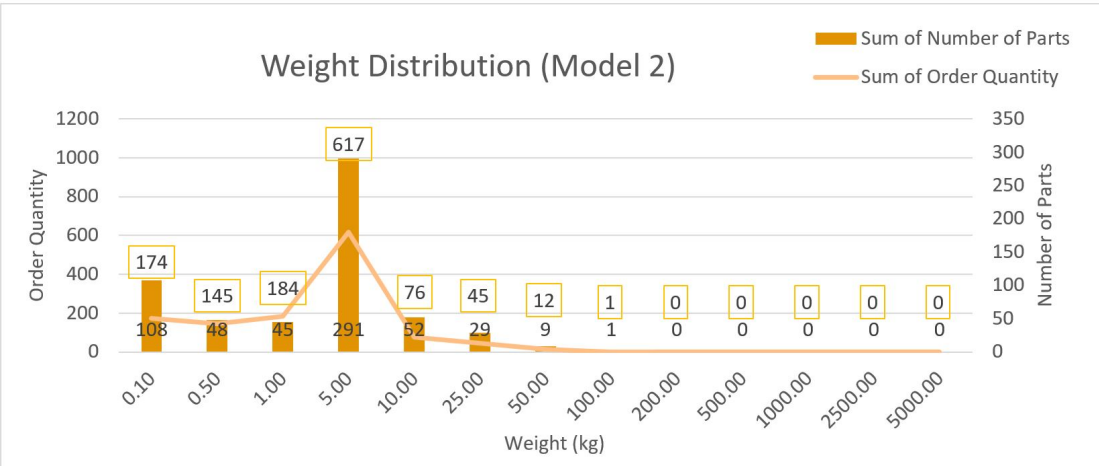

Figure 6.25: Weight distribution of Real Data with $CW = 2825$ by Model 2.

The weight distribution of the storage calculated by Model 1 based on $CW = 11300$ is shown in Figure 6.26. Like in Figure 6.24, there are peaks in the same

weight groups. The high weight capacity allows to store heavier parts up to 200 kg. The higher weight capacity causes a higher amount for all weight groups and equals in a service level of 86 %.

Figure 6.26: Weight distribution of Real Data with $CW = 11300$ by Model 1.

In comparison, the weight distribution of the storage by Model 2 is shown in Figure 6.27. Based on the higher weight capacity there is a increase of parts in the weight groups 0.5 to 1 kg and 1 to 5 kg. Moreover, there is a total increase of parts from 10 - 500 kg. Overall, the higher weight capacity is used to store more parts over 5 kg and equals in a service level of 89 %.

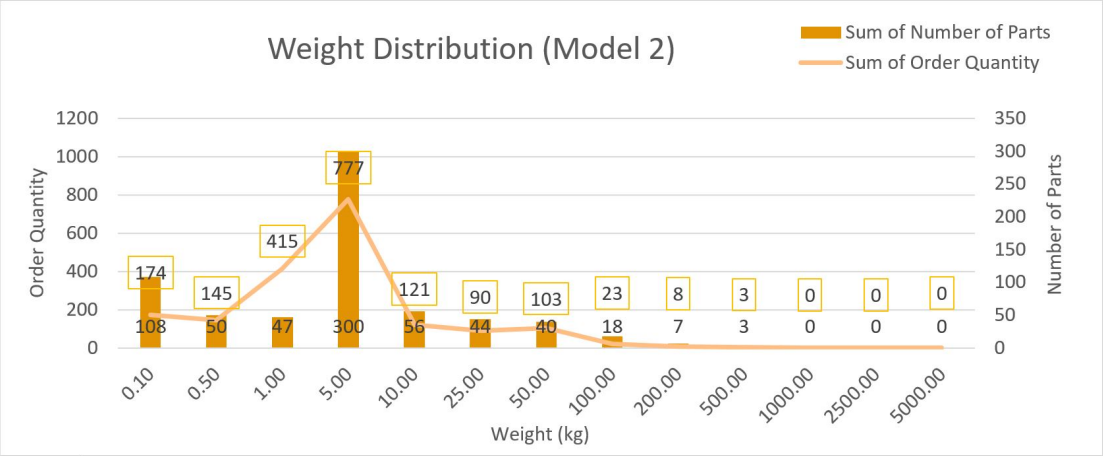

Figure 6.27: Weight distribution of Real Data with $CW = 11300$ by Model 2.

Figure 6.28 shows the weight distribution of the optimal storage by Model 1 for a weight capacity of 28500 kg. The distribution shows a peak in the weight group 1 to 5 kg and a total increase of parts from 10 to 2500 kg. The service level of the storage is 96 %.

Figure 6.28: Weight distribution of Real Data with $CW = 28500$ by Model 1.

The weight distribution of the optimal storage by Model 2, shown in Figure 6.29, has a high peak in the weight group 0.5 to 1 kg for the order quantity compared to Figure 6.28. Conversely, there is a high decrease of the order quantity for parts

in the weight group 1 to 5 kg. Overall there is a decrease of order quantities for parts from 50 to 2500 kg. The storage of the second model equals a service level of 97 %.

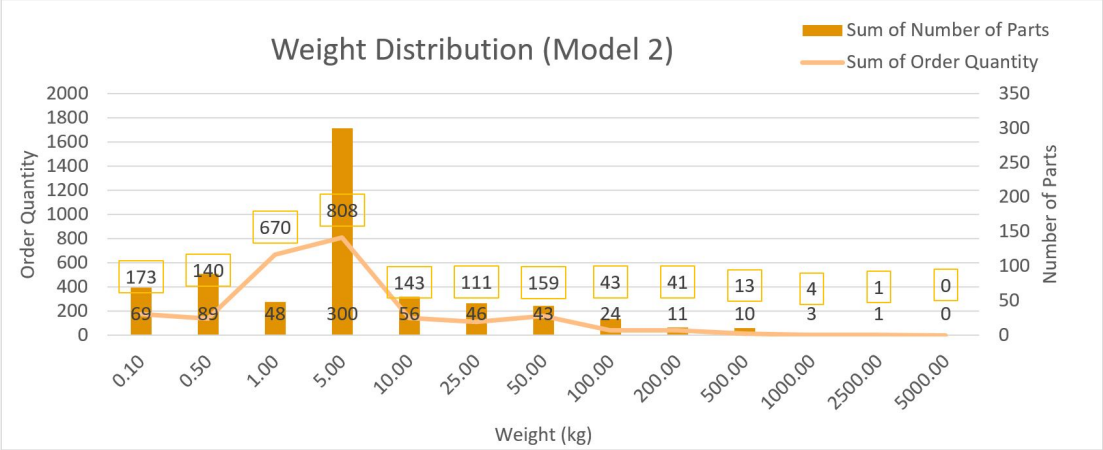

Figure 6.29: Weight distribution of Real Data with $CW = 28500$ by Model 2.

The previous weight distribution of both models shows different peaks in the weight groups and causes different service levels with a difference up to 3 % that corresponds to three more operational systems. Overall, the second model uses more capacity for heavy parts than the first one and can reach a higher service level.

6.6.2 Sim Data

For a small weight capacity of 3500 kg in the Sim Data, Figure 6.30 shows the weight distribution of the optimal storage calculated by Model 1. The storage includes mostly low-weighted parts up to 200 kg and peaks of the order quantity in the weight groups 0.1 to 0.5 kg and 1 to 5 kg. The service level of this storage is $SG = 0.22$.

Figure 6.30: Weight distribution of Sim Data with $CW = 3500$ by Model 1.

In comparison, the weight distribution of the storage by Model 2 is shown in Figure 6.31. The storage by Model 2 includes more different parts in the low weighted groups up to 5 kg, based on the "saved" with less heavy parts only up to 100 kg. The storage equals in a service level of 30 %.

Figure 6.31: Weight distribution of Sim Data with $CW = 3500$ by Model 2.

Figure 6.32 shows the weight distribution of the optimal storage by Model 1 for a weight capacity of 14500 kg. Over 80 % of the storage are parts of the weight groups from 0.5 to 5 kg. Only 2 % of the storage are heavy parts from 50 to 200 kg. The service level of the storage is 68 %.

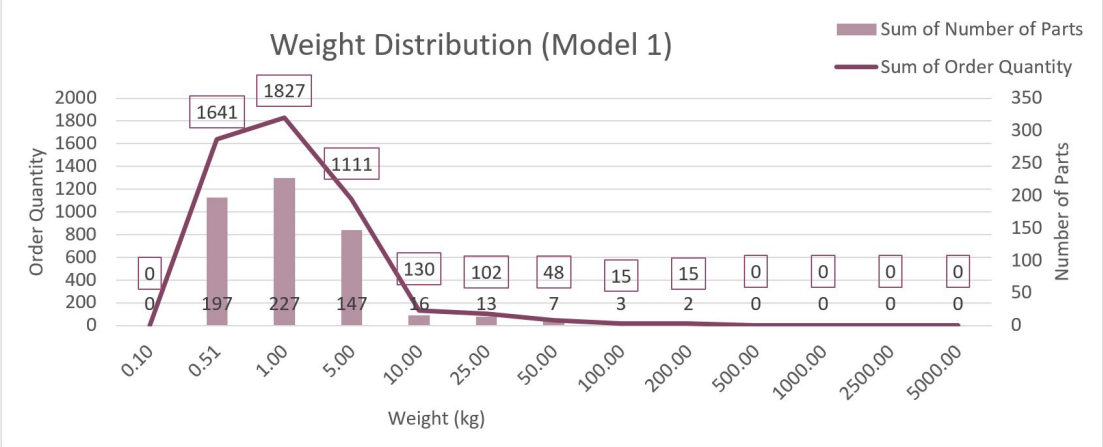

Figure 6.32: Weight distribution of Sim Data with $CW = 14500$ by Model 1.

Figure 6.33 shows the weight distribution of the storage by Model 2 with the same weight capacity. Compared to the weight distribution in Figure 6.32, there is a increase of parts in the weight groups from 0.5 to 1 kg and 200 to 500 kg based on a decrease of parts in the weight groups 5 to 50 kg.

Figure 6.33: Weight distribution of Sim Data with $CW = 14500$ by Model 2.

Based on a weight capacity of 35500 kg, Figure 6.34 shows the weight distribution of the optimal storage calculated by Model 1. Compared to Figure 6.32, the distribution shows a high increase of the order quantities with over 2000 parts

for the weight group 0.5 to 5 kg. Moreover, the weight capacity is used to store a higher amount of heavy parts up to 1000 kg.

Figure 6.34: Weight distribution of Sim Data with $CW = 35500$ by Model 1.

The weight distribution of the optimal storage by Model 2 is shown in Figure 6.35. The storage stores the same number of parts for each weight group, compared to Figure 6.34, but with different order quantities. In total the storage includes less parts in all weight groups up to 200 kg to store more parts up to 1000 kg.

Figure 6.35: Weight distribution of Sim Data with $CW = 35500$ by Model 2.

The comparison of both models for Sim Data shows more agreements of the number of parts and local peaks, but large differences in the order quantities. Overall, Model 2 stores always more heavy parts and less parts in total compared to Model 1.

7 Conclusion and Outlook

Currently, tOPSPIN is a tool to get a first impression of an optimal warehouse. Based on the current data, it shows that a high level of service is possible with a small weight capacity. That is an advantage for a successful deployment and contribution to the NRF. Model 1 is a one-staged model and more pessimistic by assuming that each part can only appear in each system once and being committed before it leaves to the deployment. Compared to the more optimistic, two-staged Model 2, which makes the decision for the optimal storage when the actual failure scenario is revealed, there are significant differences in the weight distribution of the storages under equal settings. Model 1 stores mostly a high amount of low weighted parts up to 5 kg, because the parts a preassigned to the systems. Therefore, more systems can be repaired with that storage. In contrast, Model 2 stores less low weighted parts and more heavy parts from 50 to 200 kg, because the parts are assigned to the systems, when they are failed. Consequently, equal parts in different systems can be used once and have not be stored for every system. Overall, there is a difference of service levels for both models, but for the Real Data, Model 1 is quite robust compared to the more optimistic Model 2. However, the current results are based on a small data base with some missing data, like all built-in parts. Furthermore, the simulation needs more data to generate more realistic scenarios. All analyses give an impression of the behavior of different parameters in both models. Moreover, the analyses shows which parameters influence the simulation and calculations and give advices for future users. Those have to take into account the distribution of the weight and built-in parts for a representative result. Moreover, the analyses show the value of an optimization, because a simple, pre-fixed warehouse shows a less satisfying level of service, than an optimized warehouse. To make the optimization even better and more realistic, more data has to be stored. The data about previous orders have to be collected over a longer time, especially in situations

of a deployment to make the simulation more realistic. Furthermore, all built-in parts, volume of parts and necessary systems data must be collected. Only then, optimal results can be realized. A complete dataset could be compared to the analyses of this thesis with a further analysis of the actual stored parts and not only over the weight distribution. With a complete dataset a clustering of the failed parts after the simulation could identify important parts for the storage to increase the service level. To make the simulation more realistic, one feature can be used in further calculations; the triangular function can be replaced by a Weibull function. The Weibull function describes the lifetime and default probability of a part based on previous data. Therefore, the function takes the history of a part into account. The function incorporates memory the aging of a part based on time and use. Thus, the Weibull function makes a simulation and their generated scenarios of default parts more realistic. Nevertheless, the Weibull function needs a big data base collected over long time. Another further option is the extension of the mathematical models. The calculations could include the use of parts from broken systems, where a total replacement is not possible. This extension could increase a number of operational systems with the same or even smaller weight capacity. Overall, the current software and results give a good impression how an optimal warehouse looks like. Furthermore, the thesis gives a tool and advices for further considerations and the important parameters and data, which have to be complemented with more collected data.

Bibliography

[1] Alan, J. and Stein, W. *Modeling with Stochastic Programming*. 1.Edition. Springer-Verlag, 2012.

[2] Bartmann, D. and Beckmann, M. *Lagerhaltung: Modelle und Methoden*. 1.Edition. Springer-Verlag, 1989.

[3] Ellinger, T. *Operations Research: Eine Einführung*. 3.Edition. Springer-Verlag, 1990.

[4] Essig, M. and Glas, A. *Performance Based Logistics*. 1.Edition. Springer-Verlag, 2014.

[5] Grunewald, M. *Planung von Milkruns in der Beschaffungslogistik der Automobilindustrie*. 1.Edition. Springer-Verlag, 2015.

[6] Kress, M. *Operational Logistics*. 2.Edition. Springer-Verlag, 2016.

[7] Lübbecke, M. *Branch-and-Bound Verfahren*. `http://wirtschaftslexikon.gabler.de/Definition/branch-and-bound-verfahren.html`. (accessed 06/20/2017).

[8] Mosler, K. and Schmid, F. *Wahrscheinlichkeitsrechnung und schließende Statistik*. 1.Edition. Springer-Verlag, 2011.

[9] Nahrstedt, H. *Die Monte-Carlo-Methode*. 1.Edition. Springer-Verlag, 2015.

[10] *NATO Response Force*. `http://www.nato.int/cps/pl/natohq/topics_49755.htm`. (accessed 01/16/2017).