Brandenburg
University of Technology
Cottbus - Senftenberg

Stochastic Mixed-Integer Programming for a Spare Parts Inventory Management Problem

Leonie M. Johannsmann
Emily M. Craparo
Thor L. Dieken
Armin R. Fügenschuh
Björn O. Seitner

Cottbus Mathematical Preprints
COMP # 11(2020)

Stochastic Mixed-Integer Programming for a Spare Parts Inventory Management Problem

Leonie M. Johannsmann, Emily M. Craparo, Thor L. Dieken,
Armin R. Fügenschuh, Björn O. Seitner

December 29, 2019

Abstract

The German Armed Forces provide an operation contingent to support the North Atlantic Treaty Organization (NATO) Response Force (NRF). For this purpose, a "warehouse" containing accommodations, food supplies, medical supplies, and spare parts for the systems has to be available. Such a warehouse is restricted in weight, in order to be quickly movable in an upcoming deployment situation. It should be able to supply the NRF troops for a certain amount of time (e.g., one month) without re-supply from the outside. To ensure optimal use of such a restricted warehouse, we developed the computer program "The OPtimization of a Spare Parts INventory" (TOPSPIN) to find an optimal mix of spare parts to restore a set of systems to functionality. Each system is composed of several parts, and it can only be used again in the mission if all broken parts are replaced. The failure rate of the individual parts follows a given random distribution, and during deployment it is expected to be higher than in the homeland. Due to the stochastic nature of the problem, we generate scenarios that simulate the actual failure of the parts. The backbone of TOPSPIN is a mixed-integer linear program that determines an optimal, scenario-robust mix of spare parts and is solved using standard state-of-the-art numerical solvers. Using input data provided by the Logistikzentrum, we analyze how many scenarios need to be generated in order to determine reliable solutions. Moreover, we analyze the composition of the warehouse over a variety of different weight restrictions, and we calculate the number of repairable systems as a function of this bound.

Keywords: Logistics, Warehouse Management, Uncertainty, Scenario Generation, Mixed-Integer Programming, Two-Stage Stochastic Optimization, Operations Research.

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 forces (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 [NATO, 2017].

The Federal Republic of Germany takes part in the NRF with their forces since 2006. Thus, the German Armed Forces stands by for upcoming deployments with their expertise, technology, and manpower.

To provide a German operation contingent, many things have 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 living accommodations, food supply, medical care, and mission planning. To ensure a quick reaction, the equipment necessary to provide these capabilities is highly limited in weight. Nevertheless, every part has to work smoothly and correctly to

ensure successful participation in the NRF. After the supply of all essential requirements, any remaining weight capacity is used for spare parts to be able to repair the camp's systems (which are mostly vehicles). A similar problem in a maritime context is the selection of a vessel's board inventory, since ships only have a very limited storage space to carry spare and exchange parts.

In our research we consider the problem of optimally filling a "warehouse" with spare parts to be used in case of a deployment. By using the methods of Operations Research and the programming tool AIMMS, we develop the software package TOPSPIN to support the planning and preparation of the German Armed Forces' participation to the NRF or further deployments. TOPSPIN is based on a two-stage stochastic programming model to optimally assign parts to the warehouse. The first stage is the decision of which parts to include in the warehouse, and in the second stage the parts are assigned to failed systems in order to repair them.

The remainder of this article is organized as follows. In Section 2 we survey relevant literature. The mathematical model behind TOPSPIN is given in Section 3. The input data generation is presented in Section 4. Computational results are discussed in Section 5. In Section 6 we present our conclusions and thoughts on future research.

2 Survey of the Literature

Inventory optimization problems have appeared in practical applications for decades. Although analytical solutions often exist for problems involving a single item, the problem becomes more complicated when multiple items share a single budget. Dynamic programming has been applied with some success, but the "curse of dimensionality" prevents its widespread application in problems of practical scale [Bertsimas and Thiele, 2006]. Most inventory optimization models seek to optimize aggregate, long-term figures of merit such as the fill rate (i.e., the probability that an item will be available when a demand arrives and will not need to be backordered). In contrast, we seek to select a set of spare parts that will perform well over the relatively short time window of a single mission. Thus, we adopt a two-stage robust optimization approach and focus the remainder of our literature review on that area.

[Ahmed, 2010] provides a brief overview of two-stage stochastic integer programs, highlighting opportunities for efficient solution using decomposition methods and cuts. [Küçükyavuz and Sen, 2017] provides a broader overview the discusses the implications of uncertainty in various problem parameters, as well as the stage in which the discrete decision variables occur (first or second).

In the field of inventory optimization and logistics, various researchers have proposed robust models to solve particular problems. For instance, [Yu and Li, 2000] considers a general production planning problem, while [Santoso et al., 2005] utilizes stochastic programming to design a supply chain network. The work most closely related to ours is that of [Kemper, 2014], which develops an integer programming model to choose the quantity of spare parts to keep on hand for repairable components in a multi-echelon system. Although the integer linear program in [Kemper, 2014] successfully models part failure uncertainty, repair lead times, and a multi-echelon supply structure, its computation times are too long for practical use.

Two-stage stochastic optimization models have also found application in areas other than inventory optimization. A prime example is disaster planning, in which the first stage decisions often involve procurement or prepositioning of assets, and the second stage involves deployment and utilization of these assets once the disaster has occurred. Works in this area include [Rawls and Turnquist, 2010], [Salmerón and Apte, 2010], [Bayram and Yaman, 2017], and [Hoyos et al., 2015].

Energy systems represent another common application area for two-stage stochastic models. Demand for energy is notoriously uncertain, and with the growing utilization of renewable sources, supply is becoming increasingly uncertain. The two-stage unit commitment problem and its variants have been addressed by [Schwarz et al., 2018], [Carøe and Schultz, 1999], and [Craparo et al., 2017], while [Baker et al., 2014] considers optimal storage sizing. Using a longer planning horizon, [Ulmer, 2014] considers a first-stage capital planning model in which microgrid components are purchased in the first stage and utilized in the second stage.

3 A Two-Stage Mixed-Integer Optimization Model

Our goal is to optimally fill a capacity-limited warehouse with spare parts in such a way as to maximize the overall availability of systems during a mission. Our model is a two-stage stochastic mixed-integer optimization problem. In the first stage it is decided which parts to include in the warehouse, given a probability distribution for the (independent) failure of each part. We represent these stochastic part failures by a finite collection of possible failure scenarios, generated with respect to the individual parts' probability distributions. Due to the short duration of the mission, we assume that each individual part may fail at most once. In the second stage the actual failure scenario is revealed and the model optimally allocates the spare parts it chose in stage one. We do not consider the possibility of repairing broken parts, only replacing them with functional parts from the warehouse.

We now present our mathematical formulation and the scenario generation process. A survey of all symbols used for sets, indices, parameters, and variables is given in Tables 1-3.

index $\in$ set	set of ...
$s \in \mathcal{S}$	systems
$e \in \mathcal{E}$	parts
$c \in \mathcal{C}$	scenarios
$z \in \mathcal{Z}$	states (of a system)

Table 1: Sets and indices.

parameter	description
$p_{s,z}$	priority of system s for being in state z
L_e	lower bound of triangular failure probability for part e
U_e	upper bound of triangular failure probability for part e
α_e	skewness of triangular failure probability factor for part e
$r_{c,s,e}$	failure probability for part e in system s in scenario c
$k_{c,s,e}$	is part e in system s broken in scenario c ?
w_e	weight of part e
W	warehouse weight capacity
o_c	probability that scenario c occurs

Table 2: Parameters.

variable	description
$x_{c,s,z}$	indicates whether system s is in state state z in scenario c after repairs
$y_{c,s,e}$	indicates whether part e is used to repair system s in scenario c
n_e	quantity of part e in storage

Table 3: Decision variables.

3.1 Sets and Indices

The following sets and indices define an instance of the problem (c.f. Table 1 for a survey): Given is a set of systems $\mathcal{S}$, where the term "system" stands for a vehicle, weapon, or any other technical object. Each system $s \in \mathcal{S}$ is composed of various parts $e \in \mathcal{E}$ that can fail and be replaced individually.

The set of potential states of a system is denoted by $\mathcal{Z}$. In our experimental studies we assume that $\mathcal{Z} = \{0, 1\}$ is a binary set; that is, a system is either fully functional (working, 0) or in a fail state (broken, 1). Our constraints (see below) state that a system is working if and only if all its parts are working. However, a more general model formulation could also cover the case of partly functional systems and partly working parts. Due to a lack of real-world data, this feature is not further described here.

3.2 Parameters

The parameters that are used to describe an instance are surveyed in Table 2. Some parameters are directly used as input to the optimization model, while others are used to generate part failure scenarios via simulation; these scenarios then serve as input to the optimization model.

Every system s has a different priority for the mission in which it is deployed. The priority (reward) of system s in state z is stored in the parameter $p_{s,z} \in \mathbb{R}_+$. The higher the value, the more important it is to have system s in state z.

The simulation of failed parts is based on a triangular probability distribution. For this the parameters $L_e \in [0, 1]$ and $U_e \in [0, 1]$ with $L_e \leq H_e$ contain a lower and an upper bound on the failure probability for part $e \in \mathcal{E}$. Furthermore, the parameter $\alpha_e \in [0, 1]$ describes the skewness of the triangle. A family of random values $r_{c,s,e} \in [0, 1]$ is computed according to this distribution for each scenario c, system s, and part e (in case part e is built in system s). This value represents the actual failure probability. Then a Bernoulli (binomial) trial determines whether the part fails for each system in each scenario. This results in a binary value which is stored in $k_{c,s,e} \in \{0, 1\}$, where $k_{c,s,e} = 1$ represents a failure.

The parameter $w_e \in \mathbb{R}^+$ contains the weight of any part e. The weight capacity of the entire warehouse is stored in the parameters W.

Finally, the parameter o_c gives an overall probability of occurrence for each scenario c.

3.3 Decision Variables

The total number of part e in the warehouse is given by the variable $n_e \in \mathbb{Z}_+$. Note that this is a stage 1 decision, hence it is independent of the scenarios.

The end state of a system in a given scenario is described in the binary variable $x_{c,s,z} \in \{0, 1\}$, where $x_{c,s,z} = 1$ if and only if system s in scenario c has state z after repairs are completed, i.e., at the end of stage 2.

The binary variable $y_{c,s,e} \in \{0, 1\}$ represents the assignment of parts to systems per scenario, and we have that $y_{c,s,e} = 1$ if part e is used for the repair of system s in scenario c.

3.4 Constraints

The following constraints appear in our model:

3.4.1 System Status

Each system $s \in \mathcal{S}$ has exactly one current state for every scenario $c \in \mathcal{C}$ (after a potential repair):

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

If part $e \in \mathcal{E}$ is broken in scenario $c \in \mathcal{C}$ in system $s \in \mathcal{S}$, then system s cannot be repaired to full functionality unless part e is in the storage and assigned to system s during scenario c (recall that state $z = 0$ denotes a working system):

$$x_{c,s,0} \leq y_{c,s,e}, \qquad \forall s \in \mathcal{S}, c \in \mathcal{C}, e \in \mathcal{E} : k_{c,s,e} = 1. \tag{2}$$

3.4.2 Demand

The total number of part $e \in \mathcal{E}$ used for repairs in any scenario $c \in \mathcal{C}$ cannot exceed the number of part e that was included in the warehouse in the first stage.

$$\sum_{s \in \mathcal{S}} y_{c,s,e} \leq n_e, \qquad \forall c \in \mathcal{C}, e \in \mathcal{E}. \tag{3}$$

3.4.3 Capacity

The total weight of the spare parts must not exceed the warehouse's weight capacity W:

$$\sum_{e \in \mathcal{E}} w_e \cdot n_e \leq W. \tag{4}$$

3.5 Objective

The primary goal of the objective function is to maximize the number of high-priority operational systems. A subordinate goal is to fill the residual warehouse capacity with other spare parts that can replace broken parts in some systems, but cannot repair the whole system. Although these parts do not increase the number of operational systems in the given set of scenarios, they might become helpful in other scenarios outside the optimization run. Finally, we subtract a small term to express our preference of a lighter-weight package of parts:

$$\max \sum_{s \in \mathcal{S}, z \in \mathcal{Z}, c \in \mathcal{C}} o_c \cdot p_{s,z} \cdot x_{c,s,z} + \sum_{s \in \mathcal{S}, e \in \mathcal{E}, c \in \mathcal{C}} \varepsilon_1 \cdot y_{c,s,e} - \sum_{e \in \mathcal{E}} \varepsilon_2 \cdot n_e. \tag{5}$$

Here $\varepsilon_1, \varepsilon_2$ are parameters that reflect a lexicographical ordering among the three terms in the objective function. Maximizing the total priority of the fully operational systems is the most important goal. If there are multiple optimal solutions with respect to this goal, then additional spare parts are added to the warehouse that can fix some broken parts (but not entire systems). If there are also multiple optimal solutions with respect to this goal, then we wish to choose the solution with the lightest weight. To accomplish this lexicographical ordering, we set $\varepsilon_1 := 10^{-4}$ and $\varepsilon_2 := 10^{-6}$ in our experiments.

4 Input Data

We utilize three data sets: `RawData`, `RealData`, and `SimData`. The `RawData` set is based on records of past orders and inventory transfers for the Logistikzentrum (Logistic Center) of the German Armed Forces in Wilhelmshaven, Germany. From this we derive the `RealData` set, which corrects errors and missing entries in the `RawData` set. Finally, the `SimData` set mimics the properties of the `RealData` set, but is of larger size and thus provides a greater challenge to the numerical solver.

4.1 The `RawData` Set

The `RawData` set contains 2,643 systems and 18,062 parts with weights up to 17,600 kg. The set consists of all fulfilled orders in 2016 and 2017. An order for a part also includes the system it is used for, and hence the assignment of parts to systems is known. However, not all systems needed parts replaced in 2016 and 2017, and not all parts needed to be replaced within a given system. Thus, our data does not describe the total collection of parts that are contained in each system; this is a limitation of our study. According to the `RawData`, each system thus consists of 0 to 36 parts. Additionally, the `RawData` contains no weight information for 8,322 out of the 18,062 parts. Among the remaining 9,740 parts with weight information, 87 % of the parts weigh less than 3.8 kg (see Figure 1).

Due to the omissions and discrepancies present in `RawData`, we perform a data cleanup step to obtain the `RealData` set.

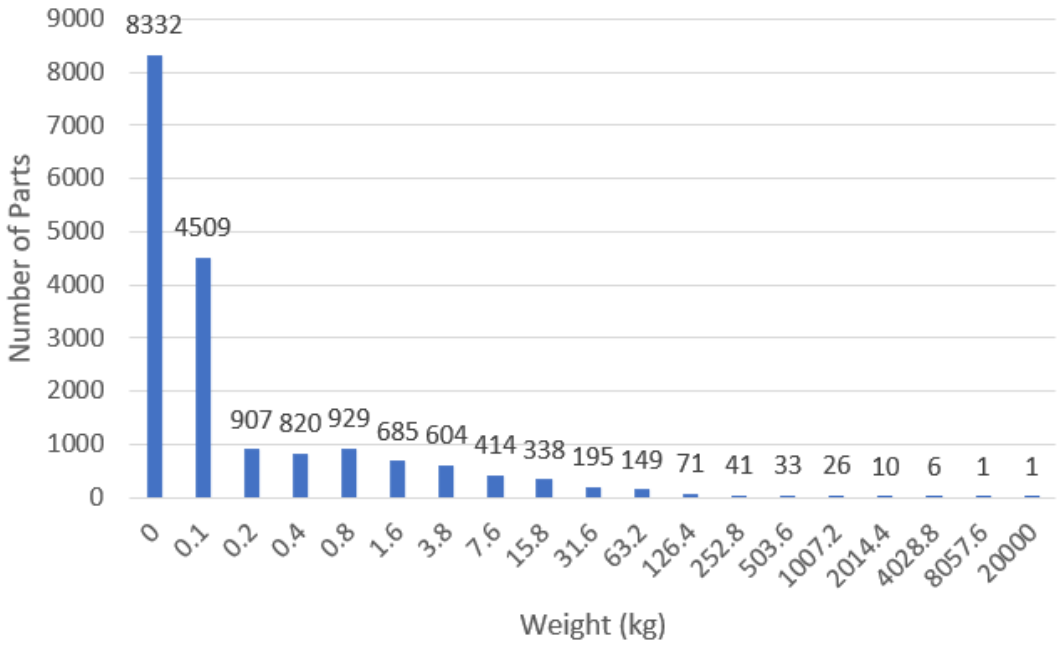

Figure 1: Weight distribution in `RawData`. Parts with no weight information are set to 0 kg.

4.2 The `RealData` Set

We now construct the `RealData` set by cleaning and condensing the `RawData` set. A typical NRF utilizes about 100 main systems; thus, we consider the 108 systems with the most replaced parts in the `RawData` set. Each of these systems contains 7 to 36 parts, and a total of 708 different types of parts are used. About 77.5 % (549) of the parts are used in exactly one system, 12.6 % (89) in two systems, and 4.4% (31) in three, and 5.5 % (39) in four or more systems (see Figure 2). The binary parameter $b_{s,e}$ reflects parts contained in every system, where $b_{s,e} = 1$ if system s contains part e, and $b_{s,e} = 0$ otherwise.

We fill in missing weight data with random picks from the list of parts with weight data. Hence, `RealData` has (in expectation) the same weight distribution as `RawData`. Figure 3 shows the weight distribution resulting from a particular random draw of weight data. A typical weight capacity W for a possible deployment is approximately 100,000 kg, but we vary this capacity limit in order to demonstrate the dependency of the solution on this parameter.

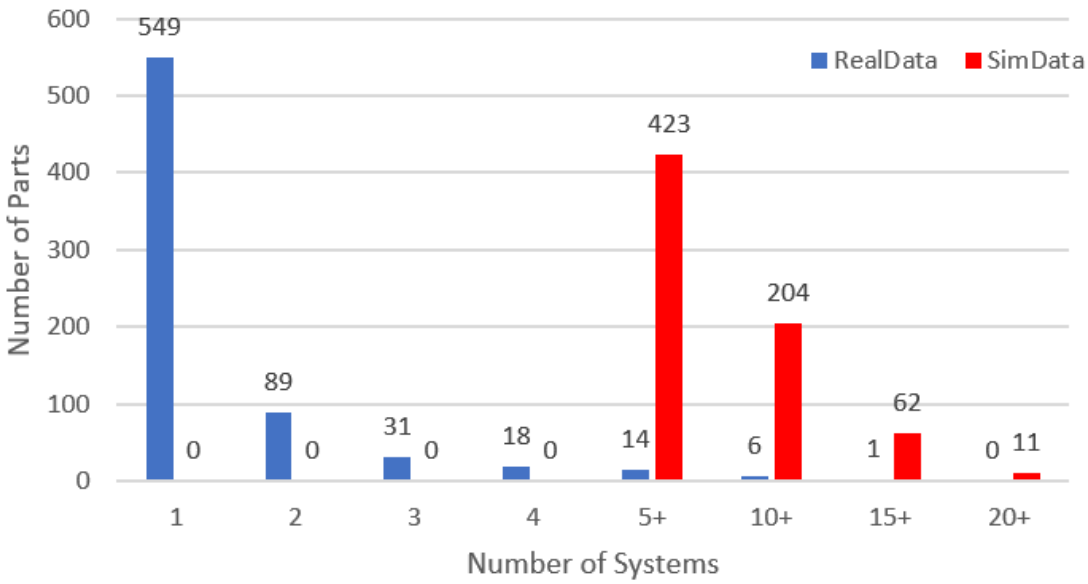

Figure 2: Number of parts that are used in one or more systems in `RealData` and `SimData`.

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 takes values from 4 (highest priority) to 1 (lowest priority) for $z = 0$, i.e., for functional systems. The priority for a broken system ($z = 1$) is 0 for all systems s. Here the priority values 4 and 2 are for combat vehicles, and the priority values 3 and 1 are for support vehicles. If multiple systems of the same type are on deployment, half of the group gets a higher priority to incentivize the repair of at least 50 % of the systems of any type.

We use simulation to generate part failure scenarios based on a triangular probability distribution. The parameters of the triangular function L_e, U_e, and α_e are based on the orders of parts in `RawData` for the top 108 systems. If parts were ordered more frequently (hence they break down more often), the lower and upper bounds of the failure probability in the triangular function L_e and U_e have higher values than the bounds for parts were seldom ordered. The relative frequency of the ordering of a part is taken as the the lower bound L_e of the triangular function. This value is in the range of 8.3 % to 36.7 %. The upper bound is set to 20 percentage points more than the value of L_e, in order to simulate scenarios with even more broken parts: $U_e := L_e + 0.2$. The displacement factor is set to $\alpha_e = 0.5$ for all parts $e \in \mathcal{E}$, and generates a balanced distribution between the lower and the upper bound. Hence the expected value is $\frac{1}{2}(L_e + U_e)$. A comparison of the number of failed parts in a scenario with that in `RawData` shows that these settings for L_e, U_e, and α_e lead to an average failure rate of 6.83 % more failed parts on average, with a standard deviation of 3.15 % over all scenarios. Hence, our scenarios are slightly more pessimistic than the original data.

The parameter o_c gives a probability of occurrence for every scenario. It is set to $o_c := \frac{1}{|\mathcal{C}|}$ for every scenario c, so that all scenarios have an equal probability.

4.3 The `SimData` Set

In order to have another test case with more controllable properties, we generate the `SimData` set. This data set is inspired by `RawData` and `RealData` but is not directly based on either.

`SimData` contains a total of 100 systems and 700 parts. The 100 systems are split

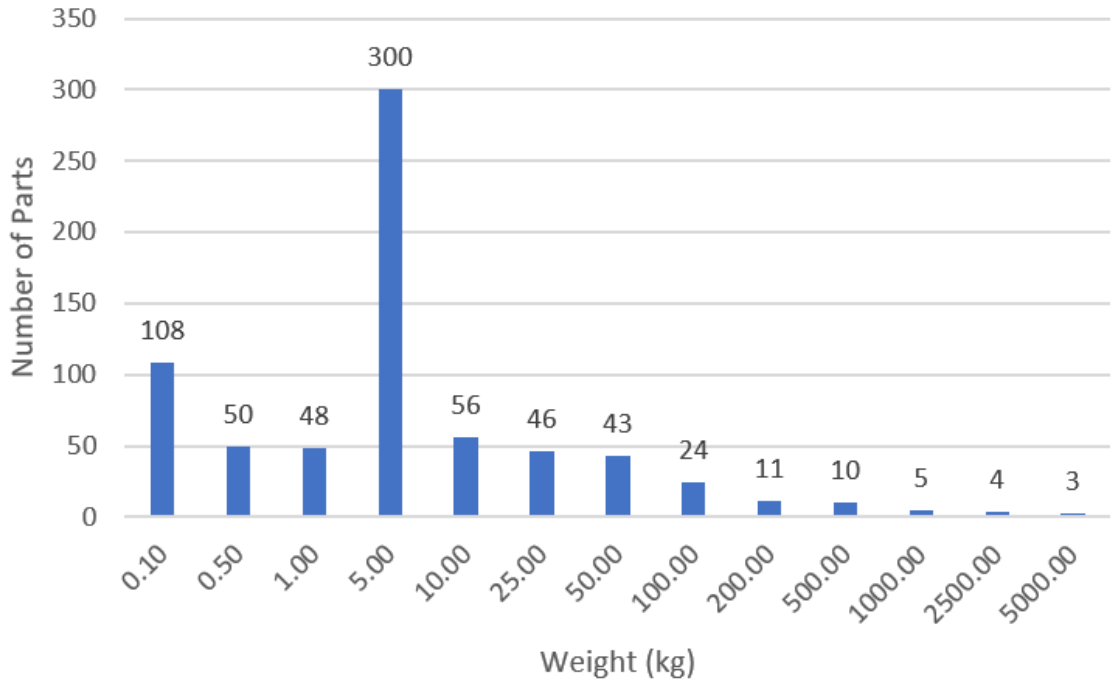

Figure 3: Weight distribution in `RealData`.

into 20 equal groups (e.g., trucks, tanks, or helicopters) of 5 systems each. Every system contains 53 parts. Compared to the `RealData` set, the parts are more interchangeable: about 60.4 % (422) of the parts are used in 5-9 systems, 29.2 % (204) in 10-14 systems, 8.9 % (62) in 15-19 systems, and 1.6 % (11) in 20 or more systems (see Figure 2). The weight distribution of the parts is similar to that of `RealData`, with more low-weight parts and a few heavy parts. The heaviest part has a weight of 5000 kg, compared to 3,600 kg in `RealData` (see Figure 4).

Because the systems in the `SimData` set contain more parts than those in the `RealData` set, the matrix representation of the parameter $b_{s,e}$ is denser, resulting in a more difficult optimization problem. The 20 groups of systems are equally divided into 10 combat and 10 support systems. The parameter $p_{s,z}$ has the same possible values for the priority: 4 (highest priority) to 1 (lowest priority). The first three vehicles of each group receive a higher priority, and the remaining two vehicles receive a lower priority.

The parameters of the triangular function L_e, U_e, and α_e are selected so that all parts have the same failure probability. We set $L_e := 0.3, U_e := 0.7$, and $\alpha_e := 0.5$. Hence, the failure rate in `SimData` is significantly higher than that in `RealData`.

Again, the probability of each generated scenario is equal, and thus we set $o_c := \frac{1}{|\mathcal{C}|}$.

5 Computational Results

We now exercise our model on the `RealData` and `SimData` sets to investigate the behavior of an optimal selection of parts in a warehouse. Our results give an assessment of the importance of certain parameters that influence both the optimal solution and the computation time required to achieve such a solution.

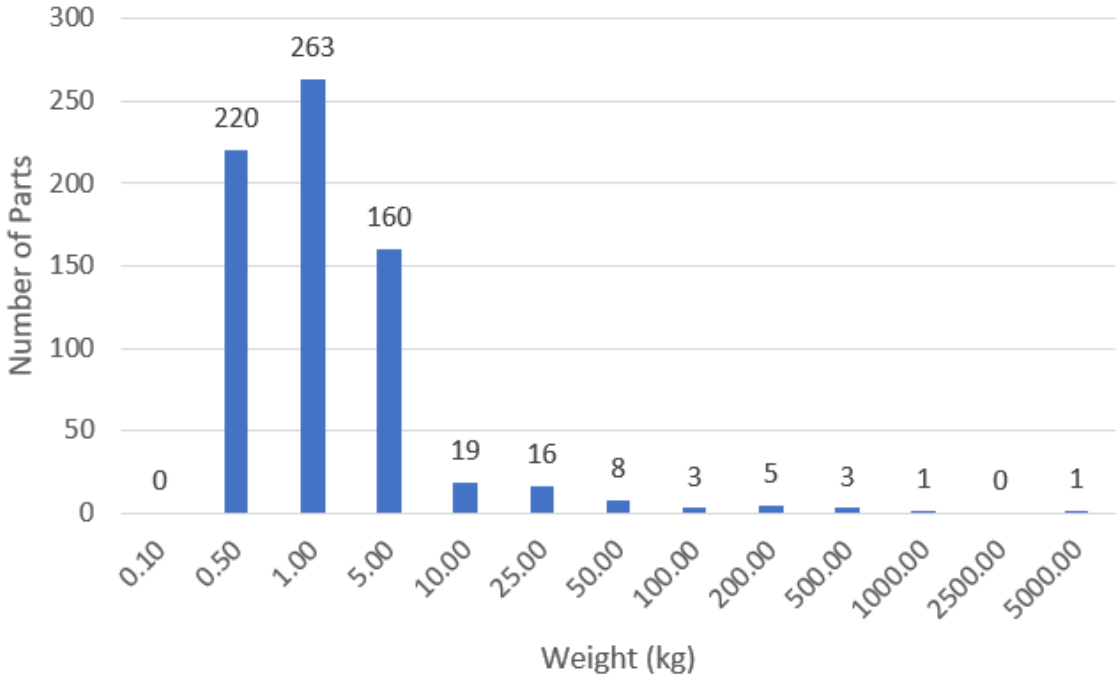

Figure 4: Weight distribution of `SimData`.

5.1 Computation Time

We implemented the model using the modeling language and graphical user interface framework AIMMS Version 4.31(64-bit). Our computing hardware was a standard laptop with Windows 10, an Intel Core i5-7200U CPU running at 2.50 GHz clock speed, and 8 GByte RAM. The mixed-integer linear optimization problems were solved to a 0.01 % optimality gap with IBM ILOG CPLEX Version 12.7, using the default settings of the solver.

Figure 5 shows the solution time for different numbers of generated scenarios using each of the two data sets. The weight capacity limit W is set to 10 % of the total weight of the available parts in each data set, which corresponds to $W = 11,300$ kg for `RealData` and $W = 14,500$ kg for `SimData`. Depending on the number of scenarios in $\mathcal{C}$, the solution time rises from a few seconds for $|\mathcal{C}| = 10$ up to approximately 250 seconds for $|\mathcal{C}| = 200$ using `RealData` and approximately 1000 seconds for $|\mathcal{C}| = 200$ using `SimData`. The instances using `SimData` are more difficult to solve compared to those using `RealData` because the parts are used in more systems, hence more combinatorial possibilities need to be evaluated during the solution process. Overall, however, the problem remains tractable for both data sets even for a large number of scenarios. Section 5.3 investigates the impact of the number of scenarios on the solution quality.

5.2 Service Level

As the "service level" (or "level of service") for a solution we calculate the expected percentage of working systems after repairs, averaged over all scenarios. Figure 6 shows the service level for different weight capacities based on the data set `RealData`. Here, a service level of 1 (100 %) corresponds to a warehouse that contains every broken part for every system in every generated scenario. With $W = 112,950$ kg, it is possible to include every component of every system in the warehouse, clearly achieving a service level of 100 %. As W decreases, we expect the service level to decrease. However, even with a

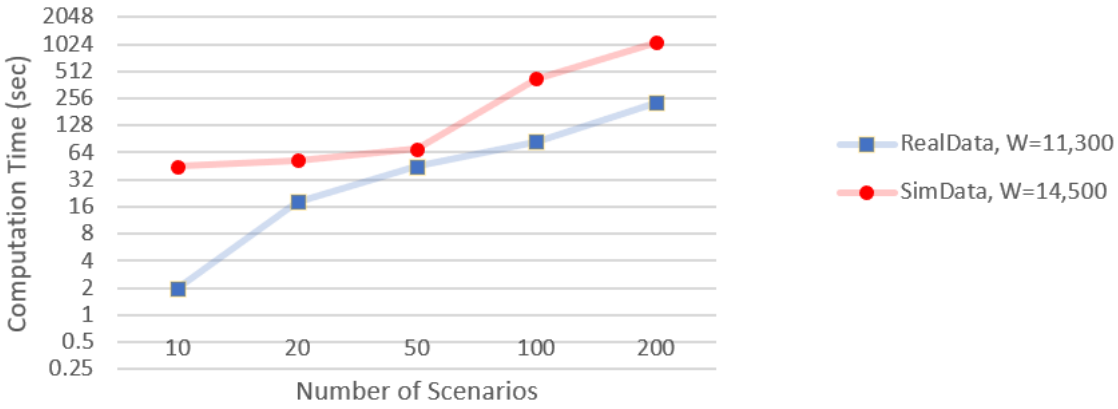

Figure 5: Solving time for `RealData` and `SimData` with a weight capacity of 10 % over different numbers of scenarios.

weight capacity of 2,825 kg (2.5 % of the total weight of all parts), it is still possible to repair up to 71 % of the broken systems with an optimal selection of parts. A possible reason for this might be the high number of lightweight parts. For example, 71 % of all parts in the `RealData` set have a weight of less than 5 kg, therefore one can store a lot of these parts even in a warehouse with only a low weight capacity W.

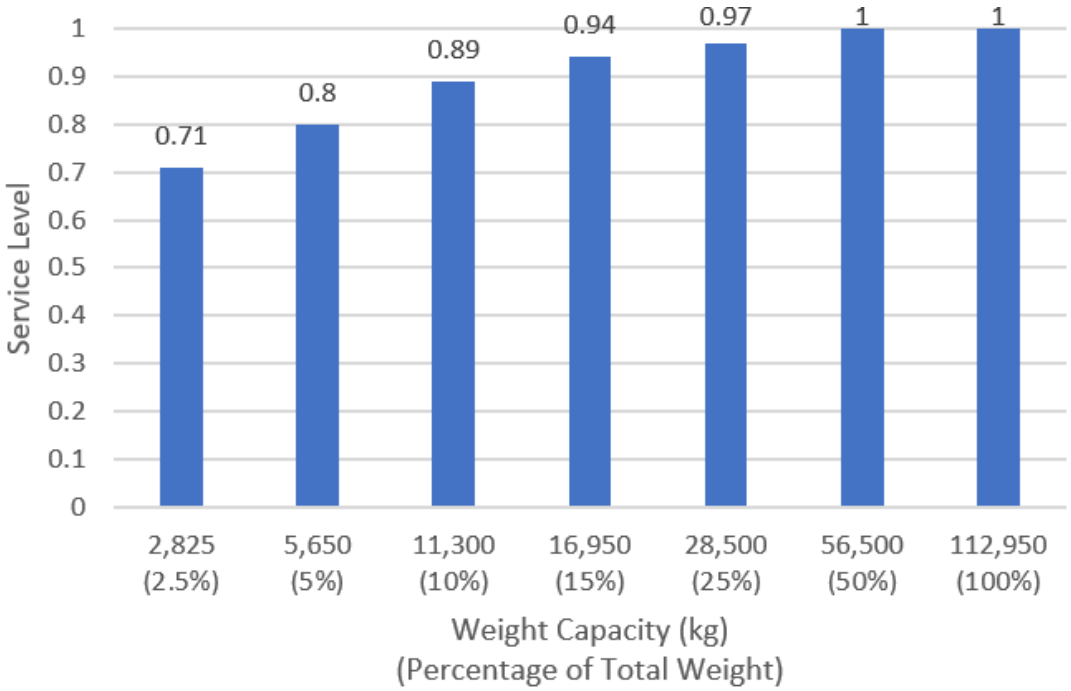

Figure 6: Service level for an optimal part selection using `RealData` for various weight capacities W.

Figure 7 shows the same results for the `SimData` set. Although we observe the same dependency of the service level on W, the service level now decreases more quickly with decreasing W. Here the total weight of all parts included in systems is 142,720 kg. With a weight capacity limit of 3500 kg, i.e., 2.5 % of the total weight, only up to 30 % of the broken systems can be repaired on average. To have a service level close to the lowest

service level achieved with the `RealData` set, one needs at least a weight capacity of 10 % of the total weight. This effect is due to the higher number of parts per system.

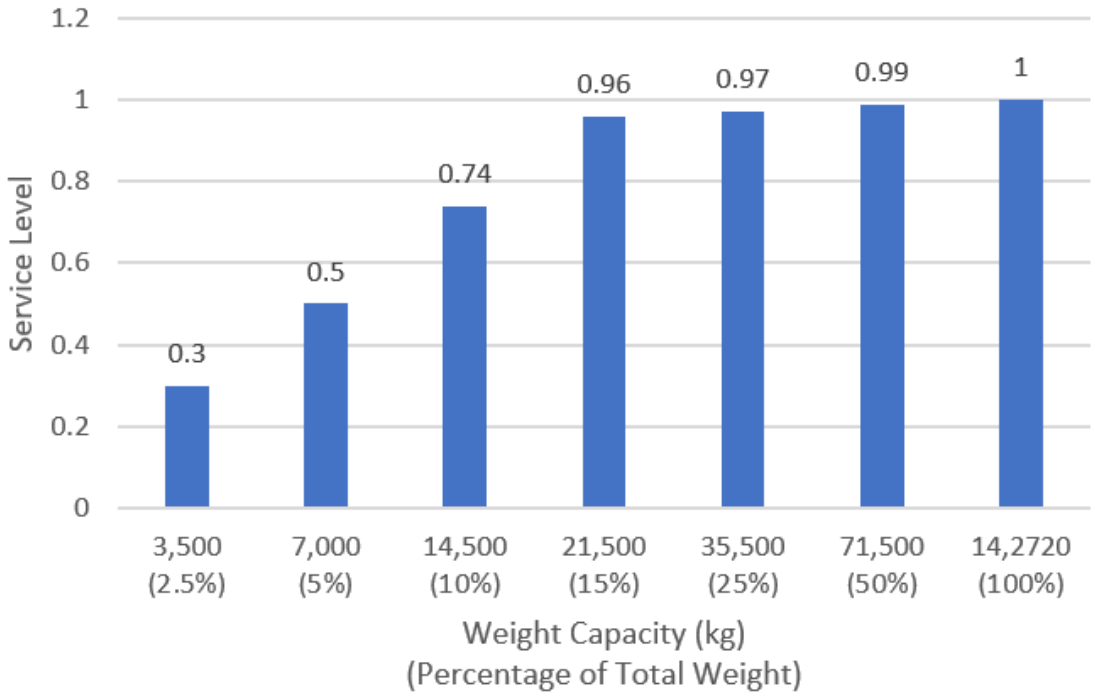

Figure 7: Service level for an optimal part selection using `SimData` for various weight capacities W.

5.3 Number of Scenarios

We now consider the following question: how many scenarios in $\mathcal{C}$ are necessary to obtain a representative solution that will perform well outside the considered scenarios? In order to answer this question, we generate one additional scenario and use it to test how well the optimal solution performs. Ideally, the new scenario has a service level that is similar to that of the other scenarios; if so, we consider the solution to be "representative". The outcome of this analysis for `RealData` is shown in Figure 8. Here we vary the number of scenarios $|\mathcal{C}|$ between 5 and 100, and the weight capacity limit W between 2.5 % and 50 % of the total weight of all parts (i.e., between 2,825 kg and 56,500 kg). We repeat each run 10 times (generating a different "new scenario" each time) and calculate the average number of repaired systems. Our results indicate that the number of scenarios required to generate a representative solution depends on the weight capacity W. For $W = 2,825$ kg, one needs as least 50 scenarios in the optimization run in order for the new scenario to achieve a similar service level as those in the optimization run. If the weight capacity limit is higher, fewer scenarios are necessary to get a similar representative result. For example, if $W = 56,500$ kg (50 % of the total weight), only 15 scenarios are needed to get a solution that performs well in a new scenario. With a higher weight capacity limit the warehouse stores more and more parts, and thus it become more and more likely to have the "right" parts also for a new unknown scenario.

Figure 9 shows the same results for the `SimData` set. These two data sets, `RealData` and `SimData`, mainly differ in the number of parts per system and their distribution. Our results indicate that these differences do impact the number of scenarios required. For a weight capacity limit of $W = 3,500$ kg (which corresponds to 2.5 % of the total weight

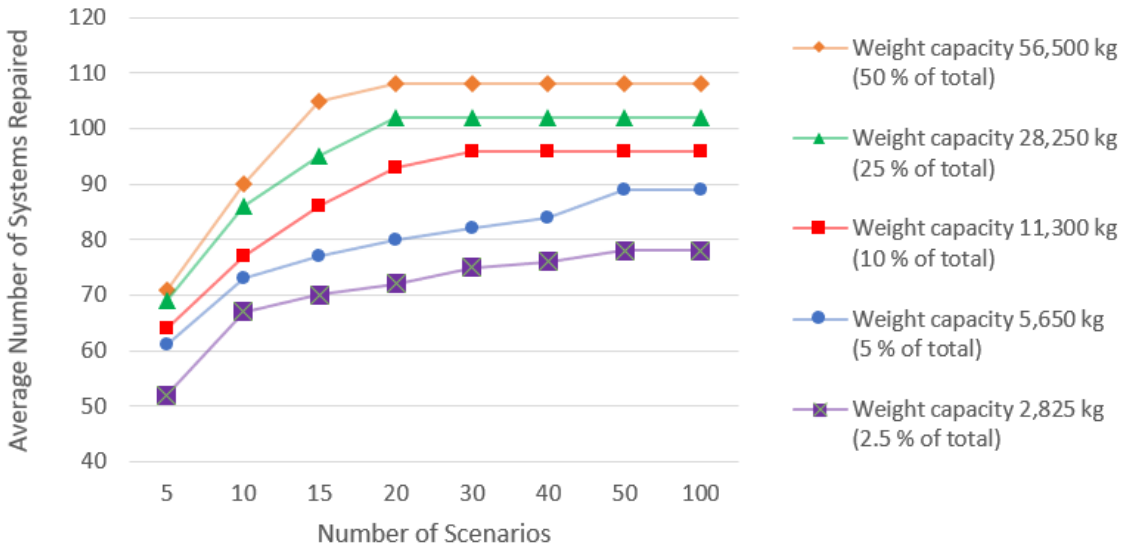

Figure 8: Analysis of the impact of number of scenarios on solution quality for the `RealData` set. For each number of scenarios $|\mathcal{C}|$, we first run our optimization model to obtain an optimal set of spare parts n_e. We then examine the ability of those parts to repair vehicles in a new scenario that was not included in the original $|\mathcal{C}|$ scenarios. Each data point above represents the average number of vehicles repaired in ten new scenarios. Note that the number of scenarios necessary to achieve top performance decreases as the weight capacity of the warehouse increases.

of all parts in `SimData`), we only need $|\mathcal{C}| = 15$ scenarios to get a representative result. This is significantly less than the 50 scenarios to obtain a similar result for `RealData`. For a 50 % weight capacity limit, only 5 scenarios are necessary to obtain a representative selection of parts.

5.4 Home Case vs. Mission Case

We now compare the spare parts' weight distribution in an optimal solution under two different operational conditions: a "home case" and a "mission case". The "home case" settings represent normal operational conditions. These conditions are predictable and well-known on the basis of long-term available data, and as a result, vehicle maintenance schedules are well-calibrated. The national territory and military bases have good infrastructure (e.g., decent roads, shelters for vehicle parking) and gentle weather conditions (moderate winters and summers). Consequently, all parts have a moderate failure rate. We represent this failure rate is represented using the parameter α_e, which describes the displacement factor between the lowest and highest value of the triangular function. A value of $\alpha_e = 0.01$ skews the overall distribution toward the lower value of the failure probability function and leads to lower failure probabilities in the simulated scenarios. In contrast, the "mission case" settings mimic the conditions in an active deployment with bad infrastructure, extreme weather conditions, and hostile attacks on the systems. A value of $\alpha_e = 0.99$ skews the triangular distribution toward its upper bound to produce more failed parts in the simulation.

Figure 10 shows the average number of failed systems in each scenario and the number of functional systems following repairs, for $W = 11,300, \alpha_e = 0.01$, and $|\mathcal{C}| = 50$. Between 70 and 90 systems (out of 108) are broken in each scenario. On average, an optimal allocation of parts is capable of repairing 80 % of the systems with these settings.

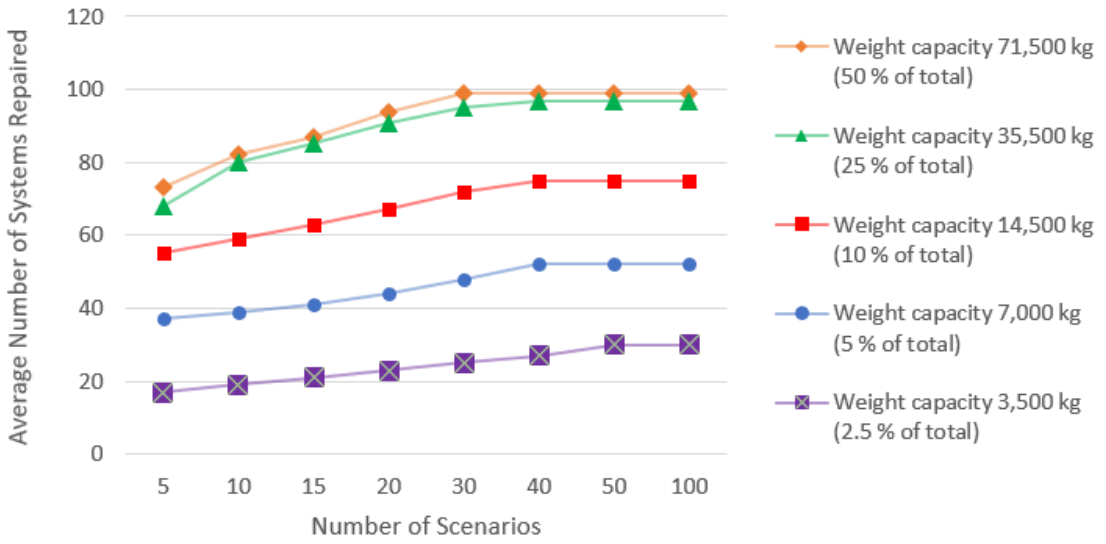

Figure 9: Analysis of the impact of number of scenarios on solution quality for the `SimData` set. For each number of scenarios $|\mathcal{C}|$, we first run our optimization model to obtain an optimal set of spare parts n_e. We then examine the ability of those parts to repair vehicles in a new scenario that was not included in the original $|\mathcal{C}|$ scenarios. Each data point above represents the average number of vehicles repaired in ten new scenarios. Note that the number of scenarios necessary to achieve top performance decreases as the weight capacity of the warehouse increases.

Figure 11 shows corresponding results for the "mission case". Here a higher number of system break; around 85 to 100 systems per scenario. On average, the optimization can fully repair around 70 % of the systems.

5.5 A Simple Rule-Based Doctrine

When writing a military doctrine, one is in general interested in clear and simple rules that can be used as good guidelines for the practitioner. From this perspective it could be tempting to formulate a simple rule to decide which parts to include into the warehouse, and how many of each to include. We now formulate such a rule and, using our optimization model, demonstrate its effect. We express our simple rule as: "For each item weighing at most m kg, pack n of that item. If an item weighs more than m kg, do not pack the item." We choose this rule for two reasons. First, we observe in our optimal solutions that most of the selected items have low weight. Second, this rule is unambiguous, and a warehouse configuration is easily specified simply by providing two numbers. For given values of m and n, we denote the resulting package of parts as an (m, n)-storage. Using our optimization model, we can easily show how much optimization potential is sacrificed for the sake of having a fool-proof rule. We perform this analysis now, for a warehouse with a weight limit of $W = 10,000$ kg and two settings for m and n.

We first set $n = 3$ units for every part up to $m = 20$ kg. The total weight of the selected parts is 9,210 kg for the `RealData` set. The total number of parts selected is 600, or 85 % of all parts. Figure 12a shows the resulting number of operational systems, compared to that of an optimal selection of parts. Up to 99 systems are broken in each scenario considered. Only 50 % to 70 % of the broken systems can be repaired with the parts from the $(20, 3)$-storage. With an optimal selection of parts 80 % to 90 % of the broken systems can be repaired. On average 15 more systems are repaired per scenario using an optimal selection of parts.

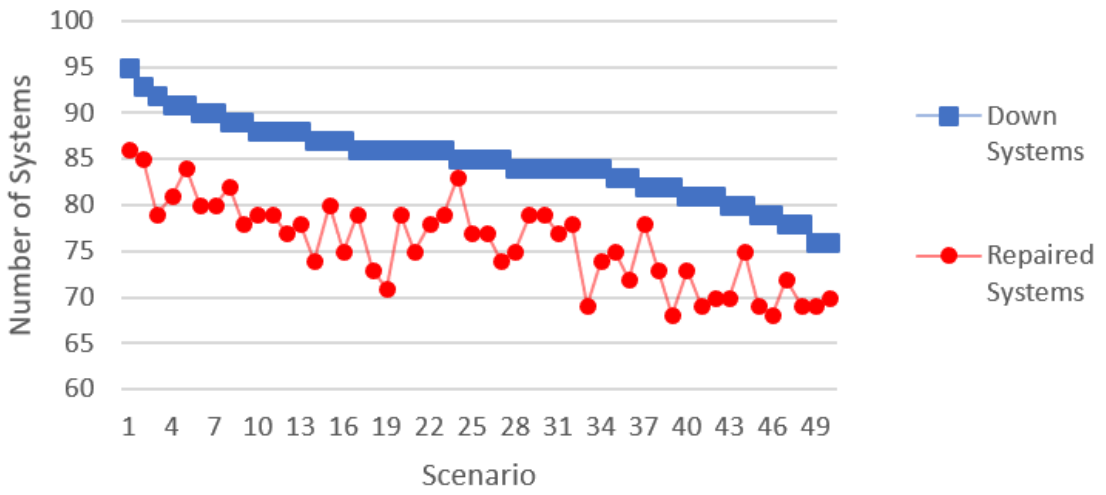

Figure 10: Optimal solution of `RealData` with $W = 11,300, \alpha_e = 0.01$, and $|\mathcal{C}| = 50$. Shown are the initial number of down systems and the number of repaired systems for each case.

Figure 12b shows the same results for a $(10, 5)$-storage, which has a total weight of 10,052 kg. An optimal selection of parts with $W = 10,052$ kg can repair 80 % to 90 % of the systems, while on average, the $(10, 5)$-storage can repair 20 systems less than the optimal selection. Thus, the performance of the $(10, 5)$-storage is somewhat inferior to that of the $(20, 3)$-storage.

For the `SimData` set, we first consider $n = 6$ units for every part up to $m = 20$ kg, resulting in a total weight of 9,210 kg. 96 % of all parts have a weight less than 20 kg in this data set. Figure 13s shows that on average, 90 of 100 systems are broken in each scenario. An optimal selection of parts can repair 75 % to 85 % of them, while $(20, 6)$-storage only repairs 40 % to 50 % of the systems. That corresponds to 29 less repaired systems on average.

The second run of the analysis for `SimData` we consider $n = 10$ units per part up to $m = 10$ kg, for a total weight of 13,280 kg. These results appear in Figure 13b. Although the $(10, 10)$-storage also repairs fewer systems compared to the optimal part selection, the deviation is smaller than that of the $(20, 6)$-storage. On average, 17 fewer systems were repaired compared to the optimal solution. However, the service level of the $(10, 10)$-storage is still lower, with 63 % to 74 % of repaired systems compared to a service level 80 % to 90 % repaired systems for an optimal part selection. Overall, our simple rule-based selection of parts is able to reach service levels up to 74 %. Of course, this result varies depending on the settings of m and n and the underlying set of data.

6 Conclusions and Future Work

We developed TOPSPIN, an optimization tool based on a two-stage stochastic mixed-integer linear program. TOPSPIN computes an optimal selection of spare parts for a warehouse, given a finite set of part failure scenarios. We have applied to real-world and simulated data sets. Our results show that a relatively high service level is possible with even a small weight capacity. This is a positive message for a light-weight NRF that has a short readiness due date and therefore must be highly mobile. From a computational perspective it was rather surprising that a moderate number of scenarios leads to stable solutions that also perform well in a failure scenario that was not included in the optimization. Rule-based selection of parts is a possible alternative, with the advantage of

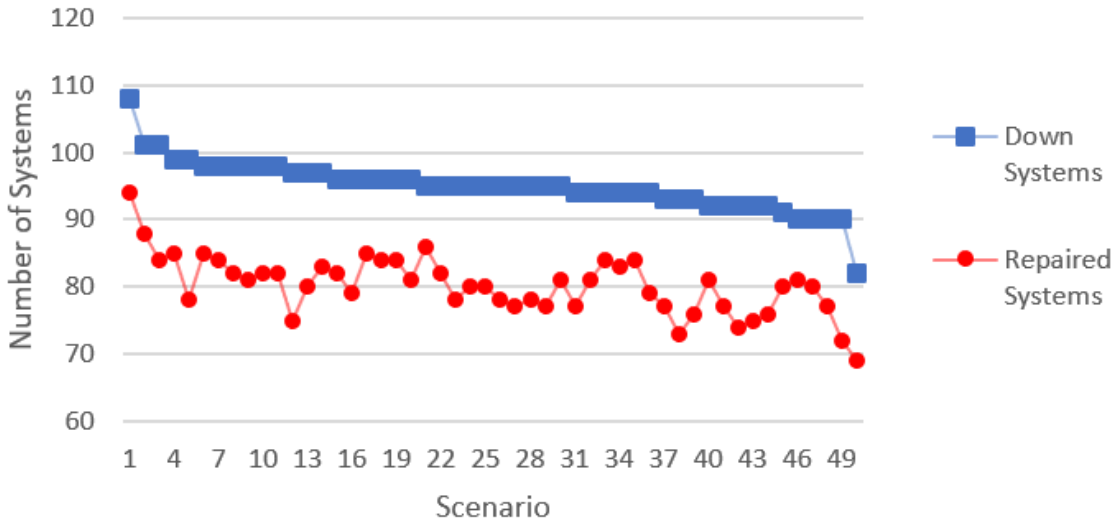

Figure 11: Optimal solution of `RealData` with $W = 11,300, \alpha_e = 0.99$, and $|\mathcal{C}| = 50$. Shown are the initial number of down systems and the number of repaired systems for each case.

avoiding the collection of data for all parts, but it results in a significantly lower service level. Hence, in the future, relevant data should be assessed with a high degree of accuracy. Future work may consider other stochastic models for the simulation of part and system failures, for example, the Weibull distribution. Another possible direction for the model would be an inclusion of "cannibalism", meaning that functional parts can be extracted from systems in order to repair other (more valuable) systems.

References

[Ahmed, 2010] Ahmed, S. (2010). *Two-Stage Stochastic Integer Programming: A Brief Introduction*. John Wiley & Sons, Inc.

[Baker et al., 2014] Baker, K., Hug, G., and Li, X. (2014). Optimal storage sizing using two-stage stochastic optimization for intra-hourly dispatch. In *2014 North American Power Symposium (NAPS)*, pages 1–6.

[Bayram and Yaman, 2017] Bayram, V. and Yaman, H. (2017). Shelter Location and Evacuation Route Assignment Under Uncertainty: A Benders Decomposition Approach. *Transportation Science*.

[Bertsimas and Thiele, 2006] Bertsimas, D. and Thiele, A. (2006). A Robust Optimization Approach to Inventory Theory. *Operations Research*, 54(1):150–168.

[Carøe and Schultz, 1999] Carøe, C. and Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37–45.

[Craparo et al., 2017] Craparo, E., Karatas, M., and Singham, D. I. (2017). A robust optimization approach to hybrid microgrid operation using ensemble weather forecasts. *Applied Energy*, 201(C):135–147.

[Hoyos et al., 2015] Hoyos, M. C., Morales, R. S., and Akhavan-Tabatabaei, R. (2015). OR models with stochastic components in disaster operations management: A literature survey. *Computers & Industrial Engineering*, 82:183–197.

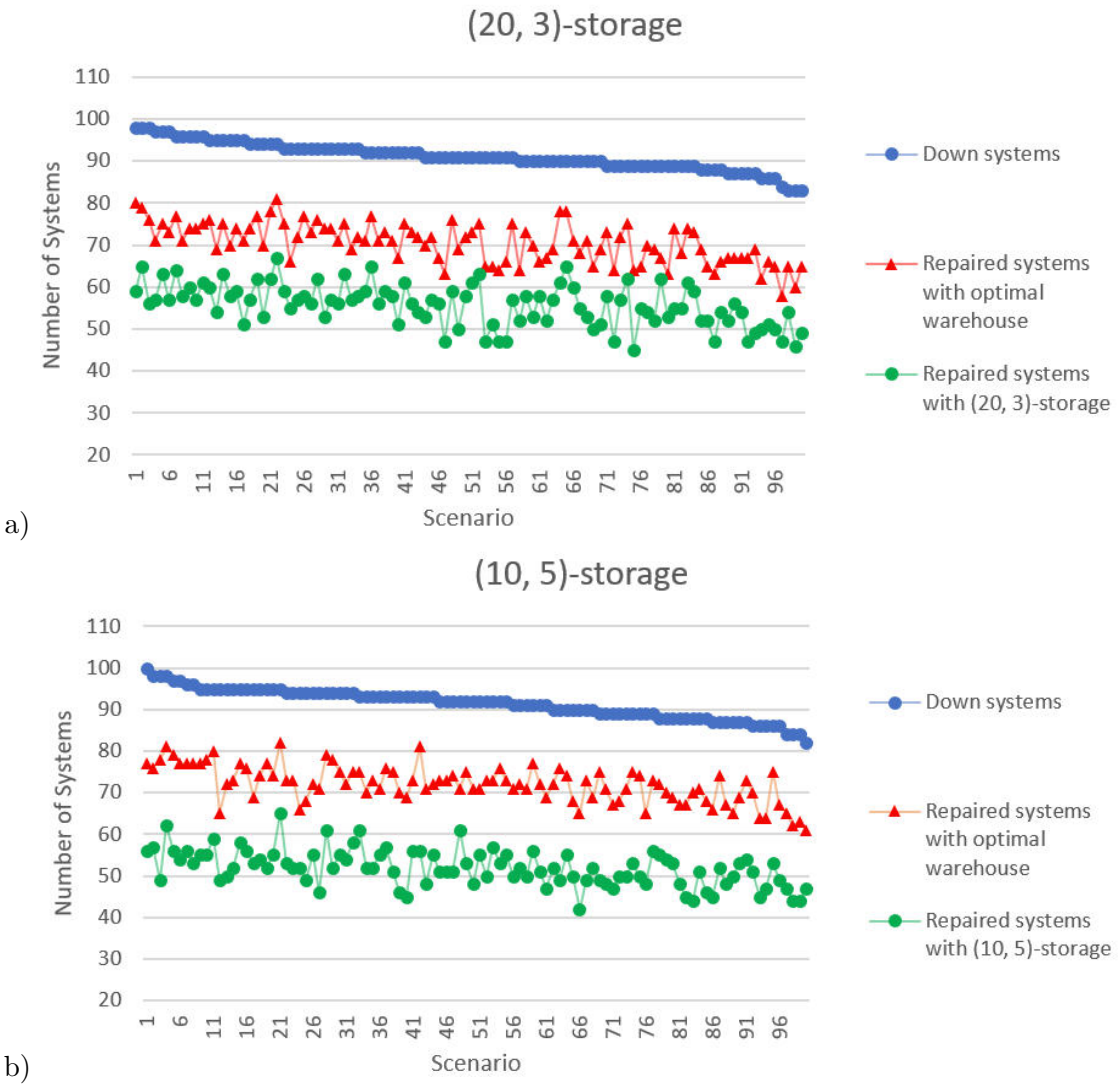

Figure 12: Performance comparison of optimal and fixed warehouse of `RealData` with a) 3 low-weighted parts up to 20 kg and b) 5 parts up to 10 kg. Shown are the initial number of down systems and the number of repaired systems for each case.

[Kemper, 2014] Kemper, B. J. (2014). Maximizing weapon system availability with a multi-echelon supply network. Master's thesis, Naval Postgraduate School, Monterey, CA, USA.

[Küçükyavuz and Sen, 2017] Küçükyavuz, S. and Sen, S. (2017). An Introduction to Two-Stage Stochastic Mixed-Integer Programming. In *TutORials in Operations Research – Leading Developments from INFORMS Communities*, chapter 1, pages 1–27. The Institute for Operations Research and the Management Sciences (INFORMS), Cantonsville, MD.

[NATO, 2017] NATO (2017). NATO Response Force. http://www.nato.int/cps/pl/natohq/topics˙49755.htm.

[Rawls and Turnquist, 2010] Rawls, C. R. and Turnquist, M. A. (2010). Pre-positioning of emergency supplies for disaster response. *Transportation Research Part B*, 44:521–534.

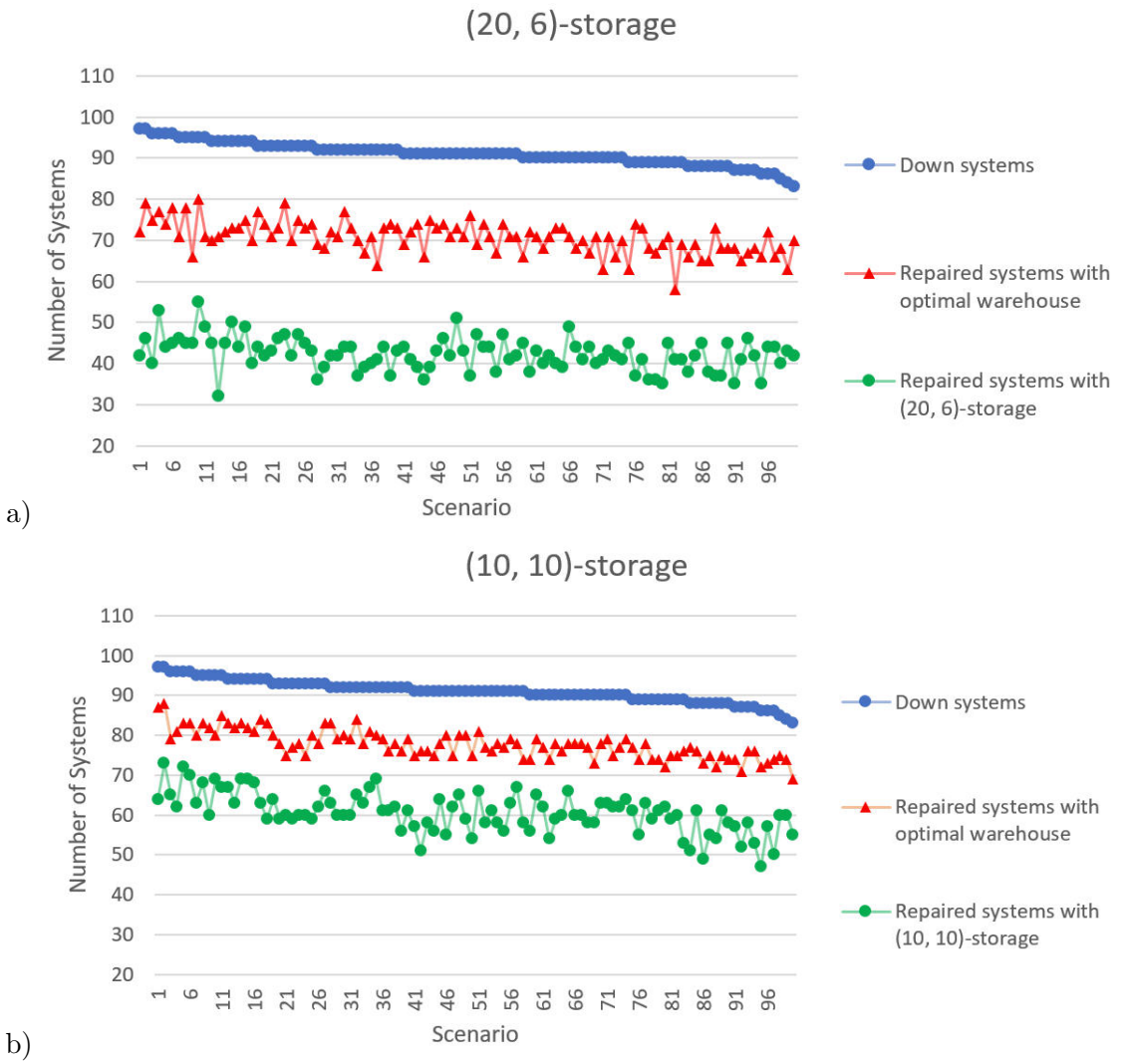

Figure 13: Performance comparison of optimal and fixed warehouse of `SimData` with a) 6 low-weighted parts up to 20 kg and b) 10 parts up to 10 kg. Shown are the initial number of down systems and the number of repaired systems for each case.

[Salmerón and Apte, 2010] Salmerón, J. and Apte, A. (2010). Stochastic Optimization for Natural Disaster Asset Prepositioning. *Production and Operations Management*, 19(5):561–574.

[Santoso et al., 2005] Santoso, T., Ahmed, S., Goetschalckx, M., and Shapiro, A. (2005). A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96 – 115.

[Schwarz et al., 2018] Schwarz, H., Bertsch, V., and Fichtner, W. (2018). Two-stage stochastic, large-scale optimization of a decentralized energy system: a case study focusing on solar PV, heat pumps and storage in a residential quarter. *OR Spectrum*, 40(1):265–310.

[Ulmer, 2014] Ulmer, N. (2014). Optimizing microgrid architecture on Department of Defense installations. Master's thesis, Naval Postgraduate School, Monterey, CA, USA.

[Yu and Li, 2000] Yu, C.-S. and Li, H.-L. (2000). A robust optimization model for stochastic logistic problems. *Int. J. Production Economics*, 64:385–397.

IMPRESSUM

Brandenburgische Technische Universität Cottbus-Senftenberg
Fakultät 1 | MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik
Institut für Mathematik
Platz der Deutschen Einheit 1
D-03046 Cottbus

Professur für Ingenieurmathematik und Numerik der Optimierung
Professor Dr. rer. nat. Armin Fügenschuh

E fuegenschuh@b-tu.de
T +49 (0)355 69 3127
F +49 (0)355 69 2307

Cottbus Mathematical Preprints (COMP), ISSN (Print) 2627-4019
Cottbus Mathematical Preprints (COMP), ISSN (Online) 2627-6100

www.b-tu.de/cottbus-mathematical-preprints
cottbus-mathematical-preprints@b-tu.de
doi.org/10.26127/btuopen-5080