Simulation-based Variable Neighborhood Search for Optimizing Skill Assignments and Priorities in Service Queues

Moustafa Abdelwanis

MSc. Thesis

December 2022

A thesis submitted to Khalifa University of Science and Technology in accordance with the requirements of the degree of Master of Science in Engineering Systems and Management in the Department of Industrial & Systems Engineering.
Simulation-based Variable Neighborhood Search for Optimizing Skill Assignments and Priorities of Service Queues

by

Moustafa Magdi Abdelwanis

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Engineering Systems and Management

at

Khalifa University

Thesis Committee

Dr. Andrei Sleptchenko (Main Advisor), Khalifa University
Dr. Mecit Simsekler (RSC Member 1), Khalifa University
Dr. Ahmad Mayyas (Co-Advisor), Khalifa University
Dr. Adriana Gabor (RSC Member 2), Khalifa University

December 2022
Abstract


This paper studies optimizing cross-training policies and static priorities in a multi-skill, multi-server repair facility with one inventory for ready-to-use spare parts. If available, the failed spare parts are immediately replaced with new ones from inventory. Otherwise, the spare parts are backordered with penalty costs. First, a Simulation-based Variable Neighborhood Search (VNS) framework is developed to optimize skill assignment with no static priorities. This study aims to minimize the repair facility’s total cost, including servers, training, holding, and backorder costs. The performance of our proposed framework is tested by comparing its results with optimal solutions for small-size instances obtained using brute-force optimization. Also, the study compares the performance of the proposed VNS algorithm to GA. The VNS-based framework obtains better results in 94.5% of the cases. The study also compares the convergence of both frameworks with the same number of iterations. The VNS has a better solution than GA in 74% of the cases with an average of 2.94%.

Second, This study optimizes the static priority rules while optimizing skill assignments. Two simulation-based frameworks are developed for optimizing skill assignments and priorities. First, a two-phase simulation-based Variable Neighborhood Search framework is developed. The first phase optimizes the skill assignments to dif-
ferent servers with no priorities, While the second stage optimizes the assignments of priorities to different tasks. The second framework is a one-phase simulation-based Variable Neighborhood Search approach for optimizing both priority classes and skill assignments of different tasks to servers. Both frameworks aim to optimize the system’s total cost, including server costs, skill costs, and holding and backorder costs. The cost savings of both frameworks are compared, and the results of the test experiments show that the one-phase framework produces better results in 86% of the test instances.

**Indexing Terms:** Variable Neighborhood Search, Simulation-based optimization, Multi-skilled repair servers, Cross-training, Static priorities
Acknowledgments

First, I would like to extend my gratitude to Prof. Nenad Mladenovic for his advice and mentorship. His contribution to the work is invaluable, and it was an honor for me to have the chance to work with him, even for a short time. Also, I would like to thank Dr. Andrei Sleptchenko for helping and supporting me throughout the duration of this project. His constructive comments and contributions to this project were indispensable for the completion of the work in this study.

I would like to thank my family and friends for their support and care. They always believed in me and encouraged me to be the best version of myself.
Declaration and Copyright

Declaration

I declare that the work in this thesis was carried out in accordance with the regulations of Khalifa University of Science and Technology. The work is entirely my own except where indicated by special reference in the text. Any views expressed in the thesis are those of the author and in no way represent those of Khalifa University of Science and Technology. No part of the thesis has been presented to any other university for any degree.

Author Name: Moustafa Magdi Abdelwanis

Author Signature:

Date: Friday 9th December, 2022

Copyright ©

No part of this thesis may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without prior written permission of the author. The thesis may be made available for consultation in Khalifa University of Science and Technology Library and for inter-library lending for use in another library and may be copied in full or in part for any bona fide library or research worker, on the understanding that users are made aware of their obligations under copyright, i.e. that no quotation and no information derived from it may be published without the author’s prior consent.
Abstract ................................................................. ii
Acknowledgments ....................................................... iv
Declaration and Copyright ........................................... v
Contents ................................................................. viii
List of Figures ......................................................... x
List of Tables ......................................................... xi
List of Abbreviations .................................................. xiii

1 Introduction ......................................................... 1
  1.1 Background ..................................................... 1
    1.1.1 Types and Degrees of Maintenance ...................... 1
    1.1.2 Spare Parts Inventory ..................................... 3
    1.1.3 Cross-training Servers ................................... 5
  1.2 Research Motivations .......................................... 6
  1.3 Problem Description and Contributions .................... 7
    1.3.1 Objectives and Contributions ............................ 7
    1.3.2 Methodology .............................................. 8
  1.4 Thesis Organization .......................................... 8
# Literature Review

2.1 Maintenance and Spare Parts Inventory ............................................. 9

2.2 Cross-training in Service Queues ...................................................... 11

2.3 Simulation-based Optimization ....................................................... 14

# Optimizing Skill-assignments with no Priorities

3.1 Problem and Framework Description ............................................... 18

3.1.1 Model Assumptions ................................................................. 18

3.1.2 Variable Definition ............................................................... 19

3.1.3 Mathematical Model .............................................................. 20

3.1.4 A Sim-based VNS Framework .................................................. 22

3.2 Computational Experiments ............................................................ 27

3.2.1 Tuning Experiments ............................................................... 27

3.2.2 Tuning Results ...................................................................... 28

3.2.3 Tuning Results Comparison to GA .......................................... 29

3.2.4 Large Cases Comparison between VNS and GA ...................... 31

3.2.5 Run-time of the Framework .................................................... 33

# Optimizing Skill-assignments with Priorities

4.1 Problem Description ................................................................. 35

4.1.1 Model Assumptions ............................................................... 35

4.1.2 Variable Definition ............................................................... 36

4.1.3 Mathematical Model ............................................................. 37

4.1.4 Simulation-based Frameworks ................................................ 39

4.2 A Two-phase Simulation-based Optimization Framework ............. 39

4.2.1 Local Search ...................................................................... 40

4.2.2 Shaking ........................................................................... 40

4.3 A One-phase Simulation-based Optimization Framework ............ 42

4.3.1 Local Search ...................................................................... 42

4.3.2 Shaking ........................................................................... 42
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Different Types of Maintenance Operations</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Different levels of cross-training.</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>Service repair shop architecture for two types of spare parts, three Servers with partial flexibility</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Local search illustration using Hamming distance.</td>
<td>24</td>
</tr>
<tr>
<td>3.3</td>
<td>Example of shaking when $k = 2$.</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>Simulation-based VNS framework for optimizing skill server assignment.</td>
<td>26</td>
</tr>
<tr>
<td>3.5</td>
<td>Relative error of different cases for each value of $K_{max}$ in logarithmic scale.</td>
<td>29</td>
</tr>
<tr>
<td>3.6</td>
<td>Number of iterations of different cases for each value of $K_{max}$.</td>
<td>29</td>
</tr>
<tr>
<td>3.7</td>
<td>Comparison of the convergences of different models using different initial solutions.</td>
<td>30</td>
</tr>
<tr>
<td>3.8</td>
<td>Relative error of VNS against GA for different configurations.</td>
<td>31</td>
</tr>
<tr>
<td>3.9</td>
<td>Factor-wise comparison between the optimized cost using the VNS framework and the GA framework.</td>
<td>33</td>
</tr>
<tr>
<td>3.10</td>
<td>Relative total cost savings using VNS compared to GA while increasing the number of iterations.</td>
<td>34</td>
</tr>
</tbody>
</table>
4.1 Service workshop architecture for two spare parts types, three Servers, three priority levels, and partial flexibility. .................................................. 36

4.2 Illustration of skill assignment matrix and priority vector in the two-phase framework. ................................................................. 40

4.3 Second stage local search illustration using two priority classes. ...... 40

4.4 Example of shaking in two neighborhoods with two priority classes. . 41

4.5 Simulation-based two-phase framework for optimizing skill assignment and priorities. ................................................................. 41

4.6 Combined skill assignment and priorities matrix with two priority classes. 42

4.7 Local search illustration for the one-phase procedure with two priority classes. ................................................................. 43

4.8 Shaking example in two neighborhoods with two priority classes. .... 43

4.9 Simulation-based one-phase framework for optimizing skill assignment and priorities. ................................................................. 44

4.10 Factor-wise comparison between the cost of two and no priority full flexible systems. ................................................................. 45

4.11 Cost savings of different classes compared to a fully flexible design with no priorities. ................................................................. 45

4.12 Factor-wise comparison between the cost of the two-phase framework with two priorities and optimized skill assignment with no priorities. .... 46

4.13 Factor-wise comparison between the cost of one phase framework with two priority classes and optimized skill assignment with no priorities. .... 47

4.14 Cost savings of different frameworks compared to a fully flexible design with no priorities. ................................................................. 48

4.15 Convergence comparison of the two-phase framework vs. the one-phase framework. ................................................................. 48
List of Tables

3.1  Number of feasible skill assignments for different numbers of skills and
    server in the small cases ........................................... 27
3.2  Design factors for tuning DOE ...................................... 27
3.3  Performance of VNS algorithm using different $K_{max}$ values .... 28
3.4  Comparison of the VNS framework to GA's ......................... 30
3.5  Design factors and levels for the experiments ....................... 32
3.6  Comparison of VNS and GA with their best configuration .......... 32
List of Abbreviations

CBM  Condition-Based Maintenance
DES  Discrete Event Simulation
EBO  Expected Backorder
FCFS First come, First Served
GA   Genetic Algorithm
KN   Kim and Nelson
LINV Least Number in Inventory
LNOR Least Number of Operations
LP   Linear Programming
OCBA Optimal Computing Budget Allocation
SAA  Sample Average Approximation
SKU  Stock Keeping Unit
SPT  Shortest Processing Time
SRETM  Shortest Remaining Processing Time

VNS  Variable Neighborhood Search
1.1 Background

Proper maintenance is essential in maintaining the system’s reliability or asset. It also aims to prevent the loss of value incurred to the systems during its lifetime [1]. The maintenance cost can be between 15 and 70% of the total production expenses [2]. For example, maintaining American naval equipment can cost 200 billion dollars every year [3]. The cost of maintenance operations can be divided into operations and spare parts. There is a trade-off between adding repair capacity and adding extra space in inventory [4]. Maintenance systems have become more complex over the last years because of the increased complexity of modern systems in terms of the increased number of components in the system and the way they interact.

1.1.1 Types and Degrees of Maintenance

Maintenance operations can be classified into three categories shown in Figure 1.1, preventative maintenance, corrective maintenance, and condition-based maintenance. Corrective maintenance happens when the asset breaks down unexpectedly. It is the sim-
plest and most common type of maintenance. Corrective maintenance has the highest associated cost and downtime among the other maintenance categories. It also depends on the availability of spare parts in the inventory.

Preventative maintenance is a scheduled type of maintenance where measures are taken to prevent the failure of a system or an asset. Preventative maintenance can be age-based or type-based, where parts are changed/repaired based on their age or type of machine. It can also be as simple as changing oil filters and lubricating a machine or changing a whole part. It is less expensive and not time-consuming than corrective maintenance. Due to the dynamic nature of operation schedules, preventative maintenance was modified into condition-based maintenance.

Condition-based maintenance (CBM) occurs when machines are continuously monitored through sensor readings or inspections. CBM takes advantage of continuous inspection, and action is taken once the failure reaches a certain level. The threshold of failure is defined based on historical data and managerial judgment. CBM has a lower cost of maintenance operations and downtime of the assets.

Van Horenbeek et al. [5] classified maintenance in terms of degrees between perfect and worse maintenance. Perfect maintenance implies that the asset is as good as new after the maintenance, while Minimal maintenance implies that the asset is as bad as once was before the maintenance. On the other hand, imperfect maintenance implies that the system is between perfect maintenance and minimal maintenance conditions. Worse and worst maintenance implies that the system is in a worse state than before the maintenance. The degree of maintenance is essential even in maintaining non-repairable components, as the replacement of components implies imperfection. Also, having a complex multi-component system will imply that replacing one part will not return the
system to as good as the new state.

One of the critical goals of good maintenance is to reduce the downtime of an asset under maintenance. Many factors can contribute to the downtime of an asset in maintenance. If the asset has many parts, it will take time to identify where the source of the error and which part should be removed or replaced. Also, another source of delay is assigning the job to the right worker if he is available. Workers with the assigned skill-set might not be available, which will increase the downtime of the asset. The availability of tools can also be a factor in increasing the downtime of the asset.

Van Horenbeek et al. [5] also identifies another major factor that can cause an increase in the downtime of an asset in maintenance: the availability of spare parts in the inventory of a workshop. The lead time of the spare parts, if they are not available, should be added to the downtime of the asset. Overall, the availability of resources and the delays induced by them should be a part of modeling maintenance optimization problems by assuming deterministic or stochastic lead time.

Another essential objective of maintenance operations is to reduce the total cost. Alrabghi et al. [1] divides the cost of maintenance into two categories: the cost of the maintenance operations and the inventory cost of spare parts. Operations costs can be divided into purchasing costs when buying spares, labor costs, and system downtime. On the other hand, inventory cost can be divided into holding cost, ordering cost, and shortage cost (back-order cost). Holding costs include the cost of keeping spare parts in inventory, the insurance cost of having expensive parts in stock, and the opportunity cost of investing money that cannot yield interest instead of investing it into other projects. Ordering costs are fixed with each order of spare parts, usually administrative costs. Since the lead time of ordering spare parts can vary, inventory can experience a shortage of spare parts which is the shortage cost that comes from delays to the customers.

1.1.2 Spare Parts Inventory

Maintenance operations and spare parts inventory are strongly related as the number of spare parts in an inventory depends on the demand of maintenance operations, whether
preventative or corrective maintenance \[5\]. Also, the availability of spare parts in inventory affects the downtime of assets under maintenance. Inventory policies can be divided into two categories which are continuous and periodic inventory management. Continuous inventory management is based on the inventory being checked continuously for reordering.

Two policies in continuous inventory management are commonly used: \((s, S)\) and \((s, Q)\). In the \((s, S)\) policy, an order will be made whenever the inventory level reaches \((s)\), and the order amount will make the inventory level reaches a level \((S)\). On the other hand, in the \((s, Q)\), an order of amount \((Q)\) will be ordered whenever the inventory level reaches a level \((s)\). In periodic inventory management, spare parts will be ordered on fixed periods based on the forecasted demand for the next period. Periodic inventory management aims to restore the inventory level to avoid stock-out between orders.

The high availability of spare parts is essential for tech systems, military applications, and medical equipment. These expensive spare parts can be repaired instead of being scrapped and replaced. Therefore, there is a trade-off between increasing the repair capacity and the size of the spare parts inventory. Another necessary trade-off in inventory management is whether the inventory has one part or multiple parts. Both trade-offs affect the size of inventory investment, the repair capacity, and the downtime of assets in the facility.

Another important inventory policy is the priority of repair parts entering the system. Multiple priority policies are adapted by management to schedule maintenance operations. Warren et al. \[6\] classifies priority rules into static, dynamic, or based on the current shop status. Examples of static priority policies are FCFS (first come, first served) which schedules jobs based on their arrival times into the shop, SPT (shortest processing time), which schedules jobs with the shortest operation time first; and RANDOM, which selects jobs randomly from the queue regardless of their arrival or operations time.

Dynamic priorities policies change the job priority within each operation. Dynamic priorities distribute tasks to queues based on their current utilization. Examples of dy-
namic policies are LNOR (least number of operations) which assigns the highest priorities to the jobs with the least number of operations remaining, and SRETM, which assigns the highest priority to the operation with the least processing remaining time.

Priority policies based on the shop status are a distinct type of dynamic policies which classify jobs based on the inventory levels of the shop. Examples of those types of policies are LINV which gives the highest priority to the parts with the least spares in the inventory, and RUNOUT, which assigns the highest priority to the job which needs spares that are the closest to stock out depending on their expected failure rates [6].

1.1.3 Cross-training Servers

Abrams et al. [7] defines cross-training as the strategy where employees are trained to do more than one task in the workplace. Cross-trained employees help improve the workplace’s productivity and agility in many conditions. Those conditions can be classified as internal and external. An example of internal conditions is to change the job description to adapt to modern business needs and reduce the effect of absences in the workforce. Agnihothri et al. [8] identifies cross-trained employees as better at handling new tasks and adapting new skills. Cross-trained employees also help the workplace become more agile in handling sudden surges on a specific task which can lead to departments being short-staffed while others are overstaffed.

On the other hand, external conditions where cross-training benefits the workplace are increased competition in the global economy, tough recessions, and technological advancements. During challenging economic crises, companies might downsize their business and let go of some of their employees. Cross-trained employees can help the business weather the storm and adapt to the new business structure [7].

However, Agnihothri et al. [8] identifies costs and drawbacks to cross-training employees. Additional to the psychological effects of cross-training on employers and employees, there is the cost of training employees for different tasks and skills. Another drawback is deciding which and how many employees need cross-training to get the most benefit of applying cross-training.
In a review of workforce flexibility models, Qin et al. [9] classifies cross-training into levels listed in Figure 1.2. Cross-training can be classified into no cross-training, chaining, pooling, and full cross-training. When there is no cross-training, each worker will have a dedicated task and nothing else. Chaining is when a worker can complete his assigned task and the task that follows him. Pooling is when there are clusters of similar jobs, and the worker of each cluster can perform the tasks of the cluster but cannot perform the tasks of other clusters. In full cross-training, workers can perform any task, which makes it the most flexible between different levels of cross-training.

1.2 Research Motivations

In today’s competitive world, managers must improve their operations’ cost and efficiency. Therefore, managers have to utilize the workforce and the facility’s inventory. Cross-training is where workers are trained to do more than their designated tasks in the work facility. Sleptchenko et al. [10] found optimizing cross-training has substantial cost savings for a repair facility that can reach more than 80%.

On the other hand, Adan et al. [11] found optimizing priorities has a 40% improvement in cost savings in a one server queue. Therefore, this research aims to find better cost savings using different optimization algorithms. Also, this work aims to study the effect of combining cross-training and priorities in service queues.
CHAPTER 1. INTRODUCTION

1.3 Problem Description and Contributions

This research uses a simulation-based variable neighborhood search framework to optimize skill assignments and priorities in service facilities. It aims to find the cost savings of using the framework and compare the cost savings to other frameworks presented in the literature. The study also aims to introduce a simulation-based VNS approach for optimizing priorities assignment to a full-flexible system and after optimizing skill assignments.

1.3.1 Objectives and Contributions

The main objectives of this research are listed below:

- Review the existing literature in the maintenance field, spare parts inventory, and cross-training.
- Identify the gaps in the current literature.
- Use discrete event simulation to simulate the model and feed the results to the optimization model.
- Use basic variable neighborhood search or a variant of VNS as an optimization heuristic and integrate it with the simulation model.
- Study the effect of adding different priority policies to service queues.

This study achieves its objectives by contributing to the maintenance, cross-training, and priorities research area. The main contributions of the study are listed below:

- Developed a simulation-based VNS approach for optimizing skill assignments with no priorities.
- Develop a two-phase sim-heuristic for optimizing priorities of a multi-server queues with optimized skill assignments.
• Develop a one-phase simulation-based optimization framework to optimize skill assignments and priorities.

• Do numerical analysis of the cost savings of using different frameworks.

1.3.2 Methodology

This research aims to answer the research questions presented above. First, the study will develop a mathematical model for the service queues to optimize the skill assignment matrix. The model’s objective function will aim to minimize the system’s total cost, including skill, server, holding, and backorder costs. This is presented in Section 3.1.3). Secondly, the study will introduce a simulation-based variable neighborhood search framework that will be used to optimize the skill assignments. The framework will be presented in Section 3.1.4, answering research questions three and four. Third, the study will provide a numerical analysis and comparison of our framework with other frameworks presented in the literature. Fourth, a mathematical model will be provided to optimize cross-training and priorities in service queues and present a simulation-based optimization model to minimize the system’s total cost. The study will compare the cost savings of using priorities with the model with no priorities. This will be presented in Section 4, and it will answer question five.

1.4 Thesis Organization

The rest of this study is organized as follows. Chapter 2 will present cross-training and simulation-based optimization models from the literature. Chapter 3 will introduce the mathematical model and the simulation-based VNS approach to optimize skill assignments to minimize the total cost of a repair shop. Chapter 4 will study the effect of optimizing priorities of a multi-skill, multi-server service queue with cross-training. The study will introduce a mathematical model and simulation-based framework to optimize priorities and skill assignments. Finally, the conclusion, limitations, and future work will be presented in chapter 5.
Since our project combines different topics, which are maintenance, inventory, and cross-training, the literature of past research is divided into three subsections: maintenance and spare parts inventory and cross-training in service queues.

2.1 Maintenance and Spare Parts Inventory

According to a review on maintenance models by Alrabghi et al. [1], most publications formulated their objective function in the problem as minimizing the total cost. As for the model’s decision variables, some articles considered the frequency of maintenance done to an asset as the more frequent the maintenance, the higher the cost. In some publications, the level of spare parts in inventory was considered the decision variable. The spare parts’ unavailability will increase the asset’s downtime, while the large spare parts inventory will increase the overall cost. In contrast, others attempted to study the effect of buffer size on maintenance operations in a production environment.

Alrabghi et al. [12] modeled preventative maintenance in a manufacturing environment using discrete event simulation to model the manufacturing environment and
continuous simulation to model the degradation of machines. The simulation model was combined with an optimization technique (simulated annealing, hill climb, and random solutions). Their objective was to minimize the total cost of the system, including maintenance cost (scheduled and corrective), spare parts cost (holding cost and ordering cost), and unavailability cost, which is a penalty cost. Simulated annealing produced the best results compared to the other methods; however, it has a long computing time.

Linnéusson et al. [13] used hybrid simulation techniques (system dynamics and discrete event simulation) combined with a multi-objective optimization model to model different strategic maintenance decisions between short-term and long-term maintenance plans. Rahmati et al. [14] also modeled a flexible job shop under ten different scenarios in terms of stochastic variables to model preventative and scheduled maintenance using simulation-based optimization. They used four different algorithms to see which would provide better results. They had a multi-objective function that included maintenance cost, complementation time, and system reliability.

Sharma et al. [15] used a simulation-based optimization model for army equipment. They used a genetic algorithm for the optimization engine. Their model aimed to forecast when the equipment will require maintenance to reduce the lead time in ordering spare parts. Their model also aimed to find what particular maintenance tasks need to be done in the available time for maintenance. They tested their model in different scenarios (the normal mode – training mode – mission with a warning – mission without warning), and they aimed to forecast the spare parts and select maintenance tasks that would reduce the cost of maintenance while maintaining the reliability of the mission and the equipment.

Waktu et al. [16] modeled maintenance of a power plant using discrete simulation with an optimization engine using tabu search and scatter search. Their objective was to minimize the total maintenance time assuming imperfect maintenance and failure can only happen to one subsystem each time.

Sleptchenko et al. [17] examined applying priorities to repairable items and their impact on reducing inventory investment. They proposed an algorithm for assigning
priorities to parts when they arrive based on iterations of assigning high priorities to items with a higher (cost/service time ratio) and then it proceeds to check if that would reduce the total cost. They represented the cost function as a sum of the expected number of items in a queue multiplied by their cost. The proposed algorithm was found to reduce the inventory investment by (10-20%).

Turan et al. [18] modeled a maintenance workshop to find which priority policy will minimize the cost, which includes the holding and backorder costs with the assumption of no cross-training. The model compared the results of four priority policies (first come, first served, based on the shortest service time, highest holding cost, and a combination between the service rates and the holding costs). The model used a simulation-based metaheuristic and variable neighborhood search as the optimization model.

As for modeling static priorities in queuing systems, there is some research done modeling the effect of priority classes on inventory levels. Adan et al. [11] modeled a single server queue in a repair shop and found that priority policies can reduce inventory costs by 40%, and having only two priority classes can lead to 90% of max savings. For multi-class, multi-server priority queues in repair shops, Sleptchenko et al. [17] found that assigning priority classes based on service times and costs can reduce inventory investment. However, there is a lack of studies optimizing priority policies in multi-skill, multi-server queues which is one of the main contributions of our study.

2.2 Cross-training in Service Queues

In the area of cross-training, several publications have been aiming to find the optimum cross-training policy in different fields, like healthcare, call centers, production, and maintenance. In health care, Paul et al. [19] modeled cross-training to solve the shortage of nurses. Their objective is to find the number of regular and cross-trained nurses that will meet the required quality and service levels at a minimum cost. They modeled the staffing cost (salaries) and the expected cost that will result in shortages: the cost
of hiring temporary staff to fill the shortage. They used an evolutionary algorithm and search algorithm for solving their model under the assumption that the demand for nurses will follow a Poisson distribution with a mean of either 5 or 10 to model high and low demand. They found that when using cross-training, the required number of nurses was less than when there was no cross-training.

Schober et al. [20] modeled the effect of cross-training and different qualification profiles on flow shops’ quality and service levels in a production environment with two production lines. They used a multi-objective genetic algorithm-based simulation to find the optimum solution of minimizing the sum of skills and maximizing the service levels of the flow shop. The level of cross-training can be identified depending on the service level required. Low and medium service levels can be identified with line-wise and stage-wise cross-training, while high service levels can only be achieved with full cross-training.

Altendorfer et al. [21] continued on the previous model and added the concept of stochastic absence of employees. They found that stochastic absence reduces the service level of the flow shop and cross-training was the way to reduce the effect of the stochastic absence of servers. Beham et al. [22] also changed the objective function to add the number of qualification profiles which are the different skill assignments among workers.

In the field of call centers, there have been various publications on applying cross-training in call centers or having multi-skilled operators. Munoz et al. [23] used discrete event simulation-based optimization to test different configurations of call centers whether their operators have only one skill, two, or multiple skills. They compared the different configurations based on their cost, including the cost of training and the operators’ salaries. They advise managers of call centers to train operators for multiple platforms instead of training them for only one platform. They discovered that bi-skill configuration was better economically than full-skill and single-skill configurations in terms of operator costs and training costs.

Also, in the field of call centers, Legros et al. [24] used simulation to compare
different policies of call centers. They compared having pooling with two skills per operator against having a chained architecture in the call center. They found that pooling allows operators to balance the work between different departments. Also, they found that pooling outperformed chained architecture in service levels.

In maintenance, Agnihotri et al. [8] modeled a simple service system with two job types and a fixed number of servers. They aimed to minimize the average service costs and customer delays per unit. They used a simulation model to investigate the different effects of the number of servers and their efficiency and utilization. They found that the fraction of cross-trained servers is inversely proportional to the number of servers. They also found that 20-40% of servers should be cross-trained if the cross-trained server efficiency is reduced by less than 10%. Simmons et al. [25] continued on the previously mentioned model but with a non-linear function of mean delay time. In his model, two dedicated servers and one multi-skilled server could work on different tasks. He found that the non-linearity of the delay cost will increase the number of cross-trained workers compared to other models in the literature, which assumed linear delay cost.

In the area of cross-training in the maintenance field, Sleptchenko et al. [10] modeled a multi-skill, multi-server repair shop with an inventory facility. They aimed to study how cross-training will affect the skill-server distribution and the optimized stocks in inventory. They used a simulation-based optimization approach to minimize the system’s total cost, including server costs (salaries), training costs and expected holding and backorder costs. They used Genetic Algorithm (GA) as their optimization heuristic. They found that the optimal cross-training policies will save, on average, 28% of the cost compared to a full-flexible design. They also found that the optimal design will minimize the holding and backorder costs.

Moreover, they found the optimal design to have less than 50% of skills for every server [10]. Different publications study the same model using other optimization heuristics and analyze the optimal design in [26, 27, 28]. Turan et al. [29] focused on optimizing a pooled repair shop design to minimize the total cost. They proposed a
sorting heuristic to group tasks and servers based on different criteria. They found that grouping tasks based on service times and holding costs achieved better results than the other criteria. It was also found that the pooled design has lower costs than different designs.

2.3 Simulation-based Optimization

According to past research on cross-training and spare parts inventory, classical optimization or simulation techniques will not capture the nature of the problem. Therefore, simulation-based optimization will capture the stochastic nature of large-scale problems. There are different techniques of simulation optimizations: Discrete optimization, Response surface methods, and gradient-based methods. Recent reviews of the different types of simulation optimization can be found in [30, 31].

Simheuristic algorithms have advantages over-optimization methods and simulation techniques. Conventional optimization problems need simplifying assumptions to deal with the uncertainty in the problems. Uncertainty in the problems reflects the actual real-life nature of the system. Therefore, simheuristic algorithms will give a better understanding of the system. On the other hand, simulation models perform well when modeling problems with uncertainty. However, classical simulation models do not have search algorithms to explore all feasible solutions [32].

According to chica et al. [32], simheuristic algorithms should be considered the first option when dealing with large-scale optimization problems with uncertain conditions, which can be found in enormous real-world applications. Real-life problems are often changing and stochastic, like change in customer demands, change in transportation time, machines wear with time, and environmental conditions. There are many applications for simheuristics like staffing in healthcare systems, logistics, transportation, and supply chain [33, 34]. For example, Juan et al. [35] modeled a simheuristic algorithm to solve vehicle routing problems with stochastic demand.

Using simheuristics will also allow optimization models to consider stochastic vari-
ables in the model’s objective function, like the expected time or cost under different conditions or in the model’s constraints. The simulation results, fed into the optimization model, can be either the best or a group of solutions. A group of solutions can benefit the risk analysis. Using simheuristics in modeling large-scale optimization problems has many advantages and disadvantages. The advantages of using simheuristics are dealing with the uncertainty of the model instead of making assumptions to simplify the actual model, and the risk analysis of a pool of solutions, which will give the management alternative options and scenarios. The limitations of using simheuristics are that the solution cannot be proven optimal, which means this method cannot guarantee optimality. More computational time and resources are needed than just a metaheuristic, or a simulation model [32].

Simheuristics algorithms consist of a simulation model with a metaheuristic. Metaheuristics are iteration-based algorithms that search the solution space to find better solutions, and they incorporate random changes to solutions to escape the local optimum [36]. Simulation models include Monte Carlo, Discrete Event Simulation, System Dynamics, and Agent-based Simulation. System dynamics models study time-dependent variables. Causal loops are used in System dynamics to model behaviors and effects between different variables. Agent-based models the effect of a new agent on the system, while discrete event simulations focus on specific events in the system. The other part of simheuristics is metaheuristics which can be classified into algorithms inspired by nature like the Ant Colony and Particle Swarm Optimization. Some algorithms are not inspired by nature, like Tabu search and variable neighborhood search [36].

Another essential part of simheuristics is the ranking techniques used to compare different designs using simulated data. There are different techniques when ranking different designs based on their simulation runs. The first type of ranking and selecting procedures in the methods uses the indifference zone, which is the range where the solution’s accuracy is not important. One of the oldest ways in this category is the two-stage Rinott procedure introduced by Rinott in [37]. All the designs are evaluated with \((n_o)\) simulation runs in this procedure. After that, each design’s metric mean and
variance are evaluated. Then a decision is to be made whether more replications are simulated or not using the following equation $N_i = \max(n_0, [S_i^2 h^2 / d^2])$ Where $S_i^2$ is the variance of design $i$, $d$ is the indifference zone, $h$ is a constant evaluated using $n_0$ and a confidence level $(1 - \alpha)$. According to Chen et al., the Rinott procedure is considered a conservative approach as it uses only the variance of the replications, while other methods utilize both the means and variances [38].

Another indifference zone method used for ranking and selection is the NSGS procedure introduced by Nelson et al. [39]. NSGS is also a two-step procedure, using an indifference zone ($\delta$) and initial sample size ($n_o$). After running ($n_o$) replications for each design, the mean and variance are evaluated for each design. After that, each replication is evaluated using the following equations and eliminating the weak designs. After simulating additional replications if needed, the design with the best sample mean is selected to be the best design.

Another indifference zone procedure is the KN procedure first introduced in [40]. KN procedure is also sequential, but it requires all the designs to run simultaneously and much switching between different designs. KN procedure will eliminate any inferior design from consideration as $r$ keep increasing and the screening procedure is recomputed until only one best system exists. According to Kim et al., a study was made to test the NSGS against KN against the Rinott procedure. The study found that KN is better than NSGS in terms of a smaller number of total simulation runs [41].

Other procedures do not require an indifference zone like the optimal computing budget allocation (OCBA) procedure. OCBA was first introduced in [42]. In this procedure, each replication’s simulation is run for ($N_o$), then evaluate each run’s mean and variance. The number of simulation runs is then allocated to each design according to the variance ratio. The simulation is run until the computational budget is exhausted, and the system with the best mean is considered the best design. Wu et al. [43] derived two algorithms (OBCA+, OCBAR) from OCBA to speed up the convergence of the process.

After comparing literature methodologies, a simheuristic model will be developed in
this project using variable neighborhood search as the optimization technique. Sample Average Approximation will be used as the ranking technique to compare different designs.
3.1 Problem and Framework Description

This chapter studies a multi-server, multi-skill service workshop with full cross-training and no priorities shown in Figure 3.1. The repair facility has an inventory place for different types of spare parts. When a failed part arrives at the repair shop, the part is immediately replaced with a new one from the inventory. If no spare parts are available, the part is backordered with a penalty. Figure 3.1 illustrates the repair facility with three servers and two types of spare parts. Server number three can only fix the second type of SKUs, while servers one and two can repair both types of SKUs. This model of a repair facility is similar to the models presented in [10, 26, 28].

3.1.1 Model Assumptions

The model of the service facility presented in Figure 3.1 follows certain assumptions presented below:

- The rate of arrivals for SKUs is modeled using a Poisson distribution. This as-
sumption is practical for stable bases according to models in literature [10].

• The repair time is independent of the working server, and they are modeled using an exponential distribution.

• The model does not support priority rules between different SKUs as first come, first served (FCFS) is the priority policy adopted in the model. This priority rule means that any server will work on the part with the longest time in the queue.

• The model’s holding costs have a linear relationship with the initial inventory level (initially acquired SKUs).

• Backorder costs are calculated when any failed SKU is not available, and they are calculated for every unavailable item per time unit.

• Skill costs are calculated when a working server has more than one skill.

• The expected backorder (EBO) costs are calculated using steady-state probabilities on an infinite time horizon.

3.1.2 Variable Definition

Index Sets:
$N$: number of distinct types of repairable parts (SKUs).

$M$: number of available servers in the repair facility.

**Decision Variables:**

$S_i$: initial number for SKU of type $i$ available in inventory, $(i = 1, \ldots, N)$,

$y_j$: binary variable denoting that server $j$ is a working server and has any skill, $(j = 1, \ldots, M)$,

$x_{ij}$: binary variable indicating that server $j$ has the skill to repair any part of type $i$.

$\alpha_{ij}$: Percentage of parts $i$ assigned to server $j$ ($i = 1, \ldots, N$ $j = 1, \ldots, M$)

**Problem Parameters:**

$\lambda_i$: failure rates of part of type $i$, $(i = 1, \ldots, N)$,

$\mu_i$: service rates of part of type $i$, $(i = 1, \ldots, N)$,

$h_i$: the holding cost for repairable parts of type $i$ per time unit $(i = 1, \ldots, N)$,

$b$: the backorder penalty cost per time unit per part (e.g., downtime costs due to a lack of spare parts),

$f$: the operational cost of a server per time unit (e.g., the annual salary for workers),

$c_i$: the cost of training servers or upgrading a machine to have the necessary skill to repair failed part $i$ per time unit, $(i = 1, \ldots, N)$.

### 3.1.3 Mathematical Model

This section provides the mathematical model that will optimize the skill assignment matrix to find the minimum total operational cost of the repair shop, including inventory holding costs, expected backorder costs, servers’ wages, and cross-training costs.
The objective function 3.1 considers the trade-off between different cost factors in the repair facility. Previous research shows a trade-off between upgrading servers to have additional repair capacity and adding an extra holding place in inventory [4, 5]. In other words, if upgrading a server to have more repair capacity is more expensive than the server’s wage, it is better to have dedicated servers in the system. Also, if upgrading a server is cheaper than holding an extra part in inventory, it is better to have a fully flexible design in our system [10].

The cost factors in the objective function 3.1 are evaluated in time units. The holding, skill, and server costs are linear in the decision variables $S_i$, $X$, and $y_j$, respectively. Holding costs are the extra part in inventory to be used, while the server costs are considered as the annual salaries of repair workers or the operational expenses of machines.

The cross-training costs are the cost of a server having an extra repair capacity to work on more than one part. In the case of repair machines, cross-training costs are considered as the software update of machines to have extra repair capacity. Backorder costs are considered penalty costs related to the number of parts not available in the time needed for repair. The expected backorder is the only non-linear term, and there is no analytical solution to evaluate the queue of a server given the skill assignment matrix [10]. Therefore a simulation-based optimization approach is used to optimize the skill assignment matrix.

To assure the stability of the model, the model is subjected to sets of constraints shown below:
Constraints 3.4 ensure that any working server can only work on the SKUs when they have the necessary skills. Constraints 3.3 ensure that the utilization of each server is not over the allowable limit, which is $(1 - \varepsilon)$. Constraints 3.2 ensure that all parts are assigned to any of the servers. The model does not allow priorities that control the assigning of parts to specific servers.

### 3.1.4 A Sim-based VNS Framework

This model has a high complexity meaning that the solution space increases as the number of servers and jobs increases. Also, the model is non-linear because of evaluating the EBO. Therefore it is not efficient or impossible to find the optimal skill assignment matrix using the traditional optimization techniques [10]. A sim-based VNS framework is developed to optimize the skill assignment matrix. A DES model is used to evaluate the expected backorder of the system under stochastic arrival and service times given the skill assignment matrix. Basic VNS which is used in this study is presented in Algorithm 1.
Algorithm 1: BASIC VNS ALGORITHM

**Result:** BasicVNS (Kmax, x, N)

\[ x \leftarrow InitialSolution; \]

\[ K \leftarrow 2; \]

**for** \( K \leq Kmax \) **do**

\[ y \leftarrow shake(x, a, b, k); \]

\[ y' \leftarrow localsearch(y, a, b); \]

*if* \( y' \) *is better than* \( x \) *then*

\[ x \leftarrow y'; k \leftarrow 2; \]

**end**

**end**

**return** \( x; \)

---

**Initial Solution**

Generating an initial solution is the first part of basic VNS. While generating a random feasible initial solution should be enough, in our model, two starting initial solutions are generated by solving LPs. the total costs of the two starting solutions are compared, and the one with the lowest cost is returned as our starting solution. The first initial solution is generated by solving an LP that minimizes the total number of servers while satisfying constraints (3.2-3.8). The second initial solution is generated by solving an LP that minimizes the total number of skills in the assignment matrix while also satisfying constraints (3.2-3.8).

\[
\min_x \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} \quad (3.10)
\]

\[
\min_x \sum_{j=1}^{M} y_{ij} \quad (3.11)
\]

**Local Search**

After the algorithm generates the initial solution, the algorithm does the local search, which is the improvement step in basic VNS. In local search, the algorithm searches the neighboring solution space of the incumbent solution. There are two local search techniques, either based on best or first improvement. The best improvement searches all the neighboring solutions and returns the one with the best objective function value.
On the other hand, the first improvement searches the neighboring solutions in order and returns the first solution with a better objective function than the incumbent solution. The best improvement technique is computationally expensive compared to the first improvement; however, the best improvement returns better solutions.

Our framework optimizes the skill assignment matrix, which is a binary matrix. Therefore, Hamming distance is used as the neighborhood structure. In other words, every element in the assignment matrix is changed from 0 to 1 and vice versa. Every neighboring solution is then evaluated, and the one with the best objective function value is returned. The algorithm then updates the incumbent solution and repeats until no other improvement is found. An illustrative example of a local search using Hamming distance is shown in Figure 3.2 and Algorithm 2.

![Figure 3.2: Local search illustration using Hamming distance.](image)

Algorithm 2: LOCAL SEARCH ALGORITHM FOR BINARY MATRIX OF SIZE (a, b)

```python
Algorithm 2: LOCAL SEARCH ALGORITHM FOR BINARY MATRIX OF SIZE (a, b)

Result: Function localsearch (x, a, b)

y ← x; i ← 0;
while improve = True do
    improve ← False;
    for i in range (a) do
        j ← 0;
        for j in range (b) do
            y(i, j) ← 1 − y(i, j);
            if f(y) < f(x) then
                improve ← True; i', j' ← i, j;
            end
            y(i, j) ← 1 − y(i, j);
        end
        if improve then
            x(i', j') ← 1 − x(i', j')
        end
    end
    return x;
```
CHAPTER 3. OPTIMIZING SKILL-ASSIGNMENTS WITH NO PRIORITIES

Evaluating Solutions

Every neighboring solution in the local search step is evaluated using a DES model. The solution’s feasibility is checked using an LP model to reduce the algorithm’s time. The model checks if any of the servers must be overutilized. Then, the solution is fed to the DES model. The simulation model is run for 35 replications, each with 100,000 arrivals. SAA is used to compare different designs based on simulated data. In other words, the design with the best average of the 35 replications is returned as the best solution.

As the DES model was the most computationally expensive part of the framework, a database is utilized to save time not running previously evaluated solutions. Figure 3.4 shows how the local search and the DES model are integrated with the use of LP to check the feasibility and database to reduce the time of the algorithm.

Shaking

One of the advantages of basic VNs is the shaking step which allows the algorithm to escape the local extrema. The shaking step is responsible for widening the algorithm’s search space by jumping from the local extrema (the incumbent solution) to a not necessarily better solution. The new solution resulting from shaking might have a different local extremum than the incumbent, making the algorithm explore different local extrema. The shaking step changes the incumbent solution randomly according to the neighborhood structure. The shaking step, in our case, changes 𝑘 elements in the assignment matrix randomly from 0 to 1 and vice versa. Figure 3.3 shows an illustration of the shaking step, and Algorithm 3 shows the shaking algorithm of a binary matrix.

Figure 3.4 shows a complete flowchart of the sim-based VNS framework to optimize
Algorithm 3: SHAKING ALGORITHM FOR BINARY MATRIX OF SIZE (a, b) IN K NEIGHBORHOODS

Result: Function shaking (x, a, b, K)

\[ i \leftarrow 0; \]

for \( i \) in range (K) do

\[ m \leftarrow \text{random number between} \ 0, a; \]
\[ n \leftarrow \text{random number between} \ 0, b; \]
\[ x(m, n) \leftarrow 1 - x(m, n) \]

end

return \( x \);

the skill assignment matrix. The flowchart shows the steps of basic VNS and how the solutions are evaluated using DES. The flowchart also shows how the database and feasibility check are integrated into the framework.

Figure 3.4: Simulation-based VNS framework for optimizing skill server assignment.
3.2 Computational Experiments

3.2.1 Tuning Experiments

Basic VNS only has one parameter to tune, which is $K_{\text{max}}$ (the maximum number of neighborhoods). The tuning experiments aim to find the best value of $K_{\text{max}}$. The performance of the VNS framework with different values of $K_{\text{max}}$ is measured against the optimal solution evaluated using the brute force technique. The data set in [44] is utilized for tuning the VNS algorithm. The data set has only small cases whose possible number of feasible solutions is limited to 3000. Table 3.1 shows the number of feasible skill assignments for the test cases.

Table 3.1: Number of feasible skill assignments for different numbers of skills and server in the small cases

<table>
<thead>
<tr>
<th>Number of servers</th>
<th>number of skills</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>46</td>
</tr>
</tbody>
</table>

To tune the proposed framework, the values of $K_{\text{max}}$ are changed between [6, 8, 10, 12] to find the best performing value of $K_{\text{max}}$. Some cost parameters are changed as well to test the algorithm’s performance with different cost parameters. The minimum holding cost is changed to be either 1 or 100. The cost of operating servers is either 10000 or 100000. The total experiments are calculated to be 16 (scenarios) $\times$ 840 (test cases) = 13440 experiments. Table 3.2 shows the design factors of the tuning experiments.

Table 3.2: Design factors for tuning DOE

<table>
<thead>
<tr>
<th>Factors</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine costs</td>
<td>[10000, 100000]</td>
</tr>
<tr>
<td>Minimum holding cost</td>
<td>[1, 100]</td>
</tr>
<tr>
<td>Maximum number of neighborhoods (K_{\text{max}})</td>
<td>[6, 8, 10, 12]</td>
</tr>
<tr>
<td>Total Experiments</td>
<td>$16 \times 840 = 13440$</td>
</tr>
</tbody>
</table>
3.2.2 Tuning Results

After running the 13340 experiments, the results of the VNS framework are measured against the optimal results using brute force techniques. The average error of all the experiments was 0.0107 %, while the average iteration per experiment was 303.4 iterations. Figure 3.5 shows the relative error of the VNS framework with different Kmax values. Figure 3.5 shows that the relative error of the VNS framework decreases with the increased value of Kmax. The figure also shows that only two cases with errors of more than 2.5%. Table 3.3 shows details about the performance of the VNS framework with different values of Kmax.

Table 3.3: Performance of VNS algorithm using different Kmax values

<table>
<thead>
<tr>
<th>Kmax</th>
<th>Average Error</th>
<th>Max. Error</th>
<th>Average iterations</th>
<th>Optimum was found</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.021%</td>
<td>2.69%</td>
<td>200.51</td>
<td>3142 (93.5 %)</td>
</tr>
<tr>
<td>8</td>
<td>0.01%</td>
<td>2.51%</td>
<td>269.39</td>
<td>3223 (95.9%)</td>
</tr>
<tr>
<td>10</td>
<td>4.52 × 10⁻³%</td>
<td>1.01%</td>
<td>338.39</td>
<td>3268 (97.2 %)</td>
</tr>
<tr>
<td>12</td>
<td>6.62 × 10⁻³%</td>
<td>2.2%</td>
<td>405.43</td>
<td>3278 (97.5 %)</td>
</tr>
</tbody>
</table>

This study compared the number of iterations of the VNS framework for different values of Kmax. Figure 3.6 shows box plots of the different values of Kmax. An increase in the number of iterations can be observed with the increasing number of Kmax. The effect of the initial solution was studied to realize how it affects the convergence of the algorithm. The study compared the convergence of the framework with different initial solutions and observed the convergence presented in Figure 3.7. Although the different models with different objective functions have similar relative errors and number of iterations, different initial solutions affect the convergence of the framework. Our initial solution has 75% of the experiments converge to an error of 1% in less than 100 iterations. Also, 75% of the experiments converged to an error of 0.1% in less than 200 iterations.
CHAPTER 3. OPTIMIZING SKILL-ASSIGNMENTS WITH NO PRIORITIES

3.2.3 Tuning Results Comparison to GA

This subsection compares the performance of the VNS framework to a GA framework presented in [45]. There are two sets of experiments with different tuning parameters for both frameworks. One comparison is when VNS’s $K_{max}$ is equal to 15 and GA has a
CHAPTER 3. OPTIMIZING SKILL-ASSIGNMENTS WITH NO PRIORITIES

Figure 3.7: Comparison of the convergences of different models using different initial solutions.

A mutation rate of 0.5, crossover probability of 0.7, gene mutation probability equals 0.3, a population size of 100, and the number of iterations is 25. The second comparison is when VNS’s $K_{\text{max}}$ is equal to 10 and GA has a mutation rate of 0.5, crossover probability of 0.7, gene mutation probability equals 0.3, a population size of 50, and the number of iterations is 25. A comparison of both frameworks with both scenarios is shown in Table 3.4. Figure 3.8 shows a histogram of the performance of both frameworks with both scenarios on a logarithmic scale. The histogram on the left shows that VNS has a more converged error, while the histogram on the right shows a similar performance, which shows that the VNS framework converged faster to the optimal solution than GA.

Table 3.4: Comparison of the VNS framework to GA’s

<table>
<thead>
<tr>
<th></th>
<th>VNS</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{\text{max}} = 10$</td>
<td>$K_{\text{max}} = 15$</td>
</tr>
<tr>
<td>Max. iterations</td>
<td>1188</td>
<td>2214</td>
</tr>
<tr>
<td>Avg. Error</td>
<td>$4.52 \times 10^{-3}%$</td>
<td>$2.52 \times 10^{-3}%$</td>
</tr>
<tr>
<td>Optimum was found</td>
<td>3278 (97.5 %)</td>
<td>3314 (98.63 %)</td>
</tr>
</tbody>
</table>
3.2.4 Large Cases Comparison between VNS and GA

Since VNS showed a better performance than GA in the tuning experiments, this section compares the cost savings of using VNS in optimizing skill assignments in large cases where the optimum solution cannot be found using brute force techniques. Optimizing skill assignments using the GA framework is presented in [10].

Test Experiments

In order to compare both frameworks, this study uses the same set of experiments used in [10]. The VNS framework used $K_{max} = 6$ for VNS, and the results of GA in [10] used a mutation rate of 0.5, crossover probability of 0.7, gene mutation probability equals 0.3, a population size of 100, the number of iterations is 25. Table 3.5 shows the seven design factors, each with two levels, that will result in 128 test cases. The number of SKUs has two levels which are ten and twenty. The number of available servers is either five or ten. The experiments also vary the cost parameters (holding cost, server costs, penalty costs, cross-training costs).

Cost Savings

After running the VNS framework for the 128 test experiments, the optimized objective function values of the VNS framework are compared to GA. The VNS framework produced a better result in 121 experiments (94.5%). The average improvement of the
Table 3.5: Design factors and levels for the experiments

<table>
<thead>
<tr>
<th>Factors</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of skills (SKUs) ( (N) )</td>
<td>([10, 20])</td>
</tr>
<tr>
<td>Number of servers ( (M) )</td>
<td>([5, 10])</td>
</tr>
<tr>
<td>Utilization rate ( (\rho) )</td>
<td>([0.65, 0.8])</td>
</tr>
<tr>
<td>Minimum holding cost ( (h_{min}) )</td>
<td>([1, 100])</td>
</tr>
<tr>
<td>Maximum holding cost ( (h_{max}) )</td>
<td>1000</td>
</tr>
<tr>
<td>Holding cost (workload relation)</td>
<td>([\text{IND}, \text{HPB}])</td>
</tr>
<tr>
<td>Server cost ( (f) )</td>
<td>([10h_{max}, 100h_{max}])</td>
</tr>
<tr>
<td>Cross-training cost ( (c_i) )</td>
<td>([0.01f, 0.10f])</td>
</tr>
<tr>
<td>Penalty cost</td>
<td>[50\sum_{i=1}^{N} \lambda_i h_i / \sum_{i=1}^{N} \lambda_i]</td>
</tr>
</tbody>
</table>

Table 3.6: Comparison of VNS and GA with their best configuration

<table>
<thead>
<tr>
<th></th>
<th>Average error to the best cost</th>
<th># of best cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNS</td>
<td>0.17%</td>
<td>121 (94.5%)</td>
</tr>
<tr>
<td>GA</td>
<td>9.89%</td>
<td>7 (0.5%)</td>
</tr>
</tbody>
</table>

The cost of VNS was 8.3%. Table 3.6 shows the average error of VNS and GA to the best of the two frameworks. The table also shows the number of experiments where each framework produced the best cost.

This study also compared the cost savings of both frameworks against having a full-flexible design. The cost improvement for both framework is calculated as follows

\[
\text{Percentage Improvement} = \frac{\text{TotalCost}_{\text{full flexibility}} - \text{TotalCost}_{\text{framework}}}{\text{TotalCost}_{\text{full flexibility}}}
\]

The average cost savings of using the VNS framework compared to the full-flexible design was 33.89%, while the average cost savings of using GA was 28.39%.

**Factor-wise Comparison**

This subsection does a factor-wise analysis of the cost of the VNS framework compared to the cost of the GA framework. Figure 3.9 shows the factor-wise comparison between the two frameworks. The graph shows that the median is higher than zero in all the cases. The VNS showed better cost savings when there is a larger system (the number of skills or servers is higher). Also, when server and training costs are higher, VNS shows better cost improvement than GA.


Convergence Comparison between VNS and GA

Since both frameworks had different stopping criteria due to the different natures of the algorithms, the GA framework in [10] used a population size of 100 and 25 GA iterations, meaning the framework explored 2500 solutions. On the other hand, The VNS framework did not have a specific number of iterations when the algorithm terminates. Therefore, the study compared the cost of the VNS framework at 2500 iterations to the cost of the GA framework. Figure 3.10 the relative cost savings of the VNS solution to the GA solution. The dotted red line shows where the GA framework stopped. At 2500 iterations, the VNS framework had better results than GA in 95 experiments (74%), while GA had a better solution in 33 experiments (26%). The mean and median cost improvements of the VNS framework against GA are 2.94% and 3.3%, respectively.

3.2.5 Run-time of the Framework

The proposed VNS framework has a faster convergence than GA. However, the lack of a limit on iterations makes the algorithm have an average of 22.33 hours when using $K_{max} = 6$ using a Xeon Processor E5-2650 v4. The long runtime of the algorithm is justified by the strategic nature of the problem, meaning that planning skill assignment is done in the strategic planning phase and not on an operational level. Therefore, higher cost savings are more important than the algorithm’s speed.
Figure 3.10: Relative total cost savings using VNS compared to GA while increasing the number of iterations.
4.1 Problem Description

This section will model the same repair shop in Section 3.1 with the addition of priority classes. Servers select items from the queue based on their priority, not arrival time. Our model allows preemptive priorities meaning that when an item arrives with higher priority than the part on service, it takes its place. The lower priority item resumes its service after the high priority item. Figure 4.1 shows an example of the repair shop with three servers and two types of SKUs types with three classes of priority.

4.1.1 Model Assumptions

The following assumptions are made to model the service facility with preemptive priorities:

- The model assumes that SKUs arrive according to a Poisson distribution.
- The service times are modeled using an exponential distribution.
• The model assumes preemptive priorities in the system. In other words, the service time of low-priority items is interrupted by any higher-priority items. The service time of the low priority item is either resumed or restarted due to the memory-less feature of the exponential distribution.

• Backorder costs occur when failed SKUs are not available and are calculated using steady-state probabilities on an infinite time planning horizon.

• Cross-training costs (training) only apply when a server has more than one skill.

• Holding costs are linear in the initial inventory level, and the inventory uses \((s - 1, s)\) replenishment policy.

### 4.1.2 Variable Definition

**Index Sets:**

- \(N\): number of SKU types.

- \(M\): number of available servers in the repair facility.

- \(K\): number of priority classes for SKUs (one is the highest and \(K\) is the lowest).

**Decision Variables:**
$S_i$: initial stock of inventory for SKU of type $i$, $(i = 1, \ldots, N)$,

$y_j$: binary variable meaning that server $j$ is operational with at least one skill, $(j = 1, \ldots, M)$,

$z_{ij}$: binary variable indicating that server $j$ can work on any SKU of type $i$.

$x_{ij}$: integer variable indicates the SKU’s priority class with type $i$ when worked on by server $j$.

$\alpha_{ij}$: Percentage of SKU with type $i$ assigned to server $j$ $(i = 1, \ldots, N \ j = 1, \ldots, M)$

**Problem Parameters:**

$\lambda_i$: arrival rates of SKUs of type $i$, $(i = 1, \ldots, N)$,

$\mu_i$: service rates of SKU of type $i$, $(i = 1, \ldots, N)$,

$h_i$: the holding cost for SKU of type $i$ for every time unit $(i = 1, \ldots, N)$,

$b$: the backorder cost per time unit per part.

$f$: the cost of one server per time unit (e.g., annual salary or operational cost of a machine),

$c_i$: the cost of training or upgrading servers to have the skill to repair SKU with type $i$ per time unit, $(i = 1, \ldots, N)$.

### 4.1.3 Mathematical Model

Our model optimizes the skill and priority assignment to minimize the system’s total cost. The objective function 4.1 considers different cost factors like training costs, server costs, and holding and backorder costs. There is a trade-off between adding a skill to a server or an extra place in inventory.

$$\min_{S_i, X, y_j} \left[ \sum_{i=1}^{N} h_i S_i + \sum_{j=1}^{M} f y_j + \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} z_{ij} + b \sum_{i=1}^{N} \mathbb{E} \{ EBO_i (S_i, X) \} \right] \quad (4.1)$$
The holding, server, and cross-training costs are linear in variables $S_i$, $z$, and $y_j$. The cost terms in the objective function are in time units. The upgrading costs are considered either training workers to have additional skills or upgrading a machine to have more repair capability.

The expected backorder cost is the only non-linear cost factor in the objective function, and there is no analytical solution to evaluate the expected backorder costs given the skill assignment and priority matrices. Therefore, a sim-based optimization framework is used to optimize the priority and assignment matrix.

The model will have constraints to ensure its stability:

\begin{align*}
\sum_{j=1}^{M} \alpha_{ij} &= 1, \quad i = 1, \ldots, N \tag{4.2} \\
\sum_{i=1}^{N} \alpha_{ij} \frac{\lambda_i}{\mu_i} &\leq (1 - \varepsilon), \quad j = 1, \ldots, M \tag{4.3} \\
\alpha_{ij} &\leq z_{ij}, \quad i = 1, \ldots, N, \quad j = 1, \ldots, M \tag{4.4} \\
x_{ij} &\leq K y_j, \quad i = 1, \ldots, N, \quad j = 1, \ldots, M \tag{4.5} \\
x_{ij} &\leq K z_{ij}, \quad i = 1, \ldots, N, \quad j = 1, \ldots, M \tag{4.6} \\
z_{ij} &\in \{0, 1\}, \quad i = 1, \ldots, N, \quad j = 1, \ldots, M \tag{4.7} \\
x_{ij} &\in \{0, \ldots, K\}, \quad i = 1, \ldots, N, \quad j = 1, \ldots, M \tag{4.8} \\
y_j &\in \{0, 1\}, \quad j = 1, \ldots, M \tag{4.9} \\
\alpha_{ij} &\geq 0, \quad i = 1, \ldots, N, \quad j = 1, \ldots, M \tag{4.10} \\
S_i &\in \mathbb{N}_0, \quad i = 1, \ldots, N \quad \text{(4.11)}
\end{align*}

Constraints 4.4 ensure that only servers with the necessary skill will work on certain tasks. Constraints 4.6 assign priority classes to parts if they have the necessary skills. Constraints 4.2 ensure that all parts are distributed to servers while Constraints 4.3 ensure that any servers are over-utilized. Constraints from 4.7 to 4.10 ensure the non-negativity and integrality of the decision variables.
4.1.4 Simulation-based Frameworks

This study proposes two frameworks to optimize skill and priority assignments in the following section. The first framework assumes tasks have a fixed priority across different servers, while the second assumes that tasks can have different priorities across different servers. The first framework is a two-stage simulation-based optimization framework, where the first stage is to optimize the skill assignment of servers, and in the second stage, the priority classes of spare parts are optimized. The second framework uses a simulation-based VNS one-stage framework to simultaneously optimize the assignment and priority matrix. Both techniques (one-phase and two-phase) use a DES model to evaluate the cost given the skill assignments and priority classes. The DES model is integrated with a basic VNS, and SAA was used to compare different designs.

4.2 A Two-phase Simulation-based Optimization Framework

This study develops a two-phase simulation-based framework to optimize skill assignments with no priorities in the first phase while optimizing priority assignments in the second phase. The assignment matrix and priority matrix are illustrated in Figure 4.2. The first phase uses the framework developed in Chapter 3 to optimize the skill assignment matrix with no priorities. The second stage optimizes the priority assignment using the optimized skill assignment matrix as our initial solution. This study uses a simulation-based framework using basic VNS illustrated in 1 for both phases. The following subsections will illustrate the different parts of the second phase of the framework.
CHAPTER 4. OPTIMIZING SKILL-ASSIGNMENTS WITH PRIORITIES

4.2.1 Local Search

In the local search step of basic VNS, the algorithm aims to find the best improvement (steepest descent) to our incumbent solution. Every element in our solution (priority matrix) is changed to find which element will result in the best value of our objective. The local search explores $K \times N$ solutions in each step. A DES is used to evaluate the objective function of all solutions. An illustration of iterations of local search to optimize the priority matrix is shown in Figure 4.3.

The DES model evaluates the local search step’s expected holding and backorder cost. The cost of servers and skills are fixed, as the framework only changes the priority matrix in the second stage while the assignment matrix remains unchanged. SAA was used to rank and compare different designs based on the simulation results. The simulation model was run for 35 replications with 100,000 arrivals each for every solution. A database is utilized to speed up the algorithm when evaluating previous solutions.

![Skill Assignment Matrix](image)

Figure 4.2: Illustration of skill assignment matrix and priority vector in the two-phase framework.

![Priority Vector](image)

4.2.2 Shaking

The shaking step is responsible for exploring different local extremas by changing the incumbent solution to a not necessarily better solution that has different local extrema than the incumbent. Exploring different local extrema increases the probability of finding a better local or global optimum. In the second stage of the two-stage framework,
shaking changes random elements in the priority matrix between 1 and \( K \) priority classes. Figure 4.4 shows an example of shaking in the two-phase framework. The new solution is fed into a local search step to find its local minimum, as illustrated in Algorithm 1.

\[
\begin{array}{cccc}
& 1 & 2 & 2 & 1 & 2 \\
\text{Before shaking} & (2,1) & & & \\
\text{After shaking} & 1 & 1 & 2 & 2 & 2 \\
\end{array}
\]

Figure 4.4: Example of shaking in two neighborhoods with two priority classes.

Figure 4.5 shows how the two-stage of the proposed framework are combined and how the DES model is integrated with both stages to optimize the skill assignment and priority matrix.

![Simulation-based two-phase framework for optimizing skill assignment and priorities.](image)

Figure 4.5: Simulation-based two-phase framework for optimizing skill assignment and priorities.
4.3  A One-phase Simulation-based Optimization Framework

The one-phase framework optimizes both skill assignments and priority assignments in one phase, as different parts can have different priorities across different servers. The decision matrix is the priority and assignment matrix shown in Figure 4.6. The elements in the matrix range from 0 to $K$. The server can not work on this part when the element is zero. If the element has a non-zero value (1 to $K$), the server will work on the part with a specific priority. This study uses a simulation-based optimization framework using basic VNS as the optimization heuristic.

Figure 4.6: Combined skill assignment and priorities matrix with two priority classes.

### 4.3.1 Local Search

In the one-phase framework, the local search will explore the neighboring solutions by changing elements in the skill-priority matrix between 0 to $K$ and evaluate the solutions through the DES model. The local search uses the steepest descent (best improvement) to find the best change to the incumbent solution. Figure 4.7 illustrates the local search in the skill and priority matrix. Sample Average Approximation is used to compare any two solutions using the DES results. The simulation model uses 35 replications, each with 100,000 arrivals.

### 4.3.2 Shaking

The shaking step changes the incumbent solution from the local search step to expand the search space. The shaking step changes an element randomly between 0 to $K$. The shaking step allows changing the incumbent to not necessarily a better solution.
CHAPTER 4. OPTIMIZING SKILL-ASSIGNMENTS WITH PRIORITIES

Then the VNS algorithm performs a local search on the new solution and updates the incumbent solution. Figure 4.8 illustrates the shaking step on the skill-priority matrix.

Figure 4.8: Shaking example in two neighborhoods with two priority classes.

Figure 4.9 shows the complete one-phase framework combining VNS with the DES model to optimize the system’s skill assignments and priority classes.

4.4 Numerical Experiments and Analysis

This section examines the effect and cost savings of optimizing priorities using the proposed frameworks. This study aims to compare the cost savings of two-phase and one-phase frameworks with the optimized skill assignments with no priorities.

4.4.1 Optimizing Priorities with no Cross-training

This subsection examines the effect of optimizing priorities of a full-flexible system (no cross-training). The study uses the same set of experiments in Table 3.5 to compare the cost savings of priorities and cross-training. The details of the experiments are detailed in Section 3.2.4. The experiments have one more set of test factors: priority levels between two and three.

After running the 256 experiments with two and three levels of priorities, the cost
savings from having two priorities are compared with having no priorities in a full-flexible design. This study calculates the cost savings as follows

\[
\text{Percentage Improvement} = \frac{\text{TotalCost}_{\text{No priorities}} - \text{TotalCost}_{\text{optimized priorities}}}{\text{TotalCost}_{\text{No priorities}}}
\]

The average cost savings from having optimized priorities with two levels was found to be 10.5%. The improvement in the cost savings was found to be 71.8%. This study created a factor-wise comparison of the different test factors to find which factors impact cost savings most. Figure 4.10 shows the conducted factor-wise analysis. The cost savings were found to be the highest when the machine cost was less. Also, the number of servers and SKUs in the system is inversely proportional to cost savings. In other words, the smaller the system, the highest the impact of the cost savings of using priorities.
Figure 4.10: Factor-wise comparison between the cost of two and no priority fully flexible systems.

The cost savings of using two priority levels are compared with three priority levels to examine the effect of the number of priority classes. Figure 4.11 shows a box plot to show the cost savings between having two and three levels of priorities. The graph shows that having three priority classes has a similar effect to having two priority classes.

Figure 4.11: Cost savings of different classes compared to a fully flexible design with no priorities.
4.4.2 One-phase vs. Two-phase

This subsection compares the cost savings of the two-phase framework with the one-phase framework. First, the study measures the cost savings of optimizing skill assignments and priorities using the two-phase framework. This study uses the same test experiments used in the previous sections in Table 3.5. After using the two-phase framework to optimize priorities with skill assignments, the average percentage improvement was found to be 3.5% while the maximum improvement was 28.5% to the cost of optimizing skill assignments without priorities. This study uses factor-wise analysis to see the effect of the test factors on optimizing priorities with skill assignments using the two-phase framework. Figure 4.12 shows the conducted factor-wise analysis. The factor-wise analysis shows that the machine cost, starting utilization rate, number of servers, and number of skills impact the cost savings. Also, the analysis shows that smaller systems (smaller number of servers and the number of skills) will better impact the cost savings.

Figure 4.12: Factor-wise comparison between the cost of the two-phase framework with two priorities and optimized skill assignment with no priorities.

Secondly, this study examines the effect of optimizing priorities and skill assignments using the one-phase framework. The optimized cost using the one-phase framework is compared to the cost of optimizing skill assignments with no priorities. The same test experiments listed in Table 3.5 are used for this study. The average cost savings of the one-phase framework was found to be 5.8% compared to optimizing skill
assignments with no priorities. The maximum cost savings were found to be 20.7%.
This study did a factor-wise analysis comparing the cost savings of the one-phase with
the different test factors presented in 4.13. The one-phase framework has the highest
impact under the same factors as the two-phase framework. The one phase was found
to have higher cost savings than the two-phase analysis presented in Figure 4.12. How-
ever, some cases have a negative cost improvement meaning the one-phase failed to find
a better cost than the cost of optimizing skill assignments with no priorities.

Figure 4.13: Factor-wise comparison between the cost of one phase framework with
two priority classes and optimized skill assignment with no priorities.

The third part of our analysis compares the two-phase framework with the one-
phase framework. The study compares the cost savings of the one-phase framework
to the two-phase framework. The one-phase framework had a better cost in 111 out
of 128 cases (86%) with average savings of 2.7% and maximum savings of 17.4%.
Figure 4.14 compares the performance of the one-phase, two-phase, optimizing skill
assignments without priorities, and optimizing priorities of a full-flexible design.

This study investigates the convergence of the one-phase framework and how fast it
surpasses the cost savings of the two-phase framework and optimized skill assignments
with no priorities. Figure 4.15 illustrates the convergence of four sample cases out of
the total 128 cases. The figures show that the one-phase framework achieves better cost
savings during the first phase of the two-phase framework. In one of the cases, the one-
phase did not reach a better cost saving than the two-phase framework because of the
more comprehensive search space of the one-phase framework.
Figure 4.14: Cost savings of different frameworks compared to a fully flexible design with no priorities.

Figure 4.15: Convergence comparison of the two-phase framework vs. the one-phase framework.
5.1 Final Outcomes

This study proposed simulation-based frameworks to optimize cross-training and priority policies of a multi-server, multi-skill service facility. Basic VNS was used as our optimization heuristic and DES as our simulation model. First, the study optimized the skill assignment matrix with no priorities and compared the outcomes with another GA simulation-based framework in the literature. Our proposed VNS framework was found to produce better results than GA in 74% of the test cases with an average of 2.94% when compared within the same iterations.

The study also proposed two frameworks to optimize priority classes with skill assignments. One framework assumes a fixed priority of parts across servers, and another assumes that parts will have different priorities depending on the server. The first framework is a two-phase simulation-based optimization framework, where skill assignments are optimized in the first phase, and in the second phase, priorities are optimized without changing the skill assignments. Using the two-phase framework resulted in an average cost savings of 3.5% compared to optimizing cross-training with no priorities.
The second framework is a one-phase simulation-based optimization, where skill assignments and priorities are optimized simultaneously. The one-phase framework resulted in better cost savings than the two-phase framework in 111 out of 128 test experiments (86%) with average cost savings of 2.7%.

5.2 Limitations

This study has some limitations. One of them can be running the test experiments. To run the test experiments of different frameworks, multiple computers were used for more than a month because of how expensive the computational time of the algorithms was. Another limitation of our study is how difficult it is to apply the one-phase framework in real life. Having different priorities for any part when being served by different servers is quite complex in real life.

Another limitation of the study is the small test instance to compare the different frameworks. Different cost parameters might affect the cost savings of the frameworks.

5.3 Future Work

For future studies, this work can be extended by studying the effect of using different heuristics than VNS, like GA or other variants of VNS. The effect of optimizing dynamic priority rules to minimize the system’s total cost can be studied, where the cost savings of optimizing dynamic priorities with the cost savings of optimizing static priorities are compared.

Another area of research could be using machine learning techniques to speed up the run-time of the algorithms for future runs. This research area will use past simulation runs to find correlations between cost factors and how the optimization model adds or removes skills in the local search step.

Another area of potential study is the application of the proposed frameworks to optimize skill assignments and priorities in different sectors like healthcare, banking, or other service facilities where cross-training can be applied.


