



Faculty of Engineering  
Department of Industrial Engineering

IE 402 FINAL REPORT  
Fall 2024

OPTIMIZATION OF DATA TRANSFER BETWEEN DATA  
STORAGE FACILITIES

Submitted by		
İlayda Surkultay S021433	Ceren Gülkokan S020748	Onat Öcal S020946
Selin Dinar S024949	Duru Demirbağ S021739	Faizan Tariq S019423

Supervisor  
Prof. Burcu Balçık

# **APPROVAL PAGE**

**Faculty of Engineering  
Department of Industrial Engineering**

## **IE 402 FINAL REPORT**

**Spring 2024**

**İlayda Surkultay  
Ceren Gülkokan  
Faizan Tariq  
Onat Öcal  
Selin Dinar  
Duru Demirbağ**

### **OPTIMIZATION OF DATA TRANSFER BETWEEN DATA STORAGE FACILITIES**

**Jury Members:**

**Supervisor : Prof. Burcu Balçık** \_\_\_\_\_

**Jury Member 1 : Prof. Burcu Balçık** \_\_\_\_\_

**Jury Member 2 : Asst. Prof. Dilek Günneç Danış** \_\_\_\_\_

# Declaration of Own Work Statement/ (Plagiarism Statement)

Hereby, I confirm that all this work is original and my own. I have clearly referenced/listed all sources as appropriate and given the sources of all pictures, data etc. that are not my own. I have not made any use of the essay(s) or other work of any other student(s) either past or present, at this or any other educational institution. I also declare that this project has not previously been submitted for assessment in any other course, degree or qualification at this or any other educational institution.

Onat Öcal	İlayda Surkultay	Ceren Gülkokan	Selin Dinar	Duru Demirbağ	Faizan Tariq
					
İstanbul, April 20, 2024	İstanbul, April 20, 2024	İstanbul, April 20, 2024	İstanbul, April 20, 2024	İstanbul, April 20, 2024	İstanbul, April 20, 2024

# Abstract

The project aims to minimize the delay in data backup between Fibabanka's data storage facilities in Istanbul and Ankara for disaster recovery purposes, addressing limited network bandwidth and the cause of the delay. This project ensures effective server and network bandwidth usage by optimizing data transfer based on priority (application) levels (U1, U2, U3) and Recovery Point Objective (RPO) criteria. Outputs include individual server transfer times and total data transferred, with performance measured by capacity allocation and overall transfer duration. Constraints involve Recovery Point Objective (RPO) times based on the priority levels and network capacity. After formulating the problem statement, a mathematical model was constructed. Two heuristic approaches, one of which was enhanced with a genetic algorithm, were developed alongside an exact approach using Microsoft Excel's Solver tool. Moreover, A simulation approach was utilized employing Arena Simulation Software, followed by the construction of a Python code based on the simulation model's logic. Subsequently, outputs were scrutinized, and comparisons were conducted between the approaches. Finally, insights regarding the implementation of these outputs were discussed. The report presents a concise analysis of key components, objectives, and constraints to ensure reliable business operations.

## Acknowledgments

We extend our heartfelt gratitude to Kerem Akkaya, Fibabanka's Enterprise Architecture CoE Manager, whose invaluable assistance significantly contributed to the success of our project.

We also wish to express our deepest appreciation to our advisor, Prof. Burcu Balçık, for her unwavering guidance and expertise throughout the project. Both Prof. Balçık and Mr. Akkaya's insightful counsel and dedication were instrumental in steering us in the right direction.

In addition, while we have made every effort to ensure accuracy and thoroughness in our work, any errors or omissions that persist are solely ours. We also acknowledge the sources and materials

utilized in this project, ensuring proper permissions and citations were observed wherever applicable.

# Contents

	Page number
<b>APPROVAL PAGE</b> .....	<b>2</b>
DECLARATION OF OWN WORK STATEMENT/ (PLAGIARISM STATEMENT) .....	3
ABSTRACT .....	4
ACKNOWLEDGMENTS .....	4
<i>Chapter 1 Introduction</i> .....	6
<i>Chapter 2 Company or Public/Nonprofit Organization</i> .....	7
<i>Chapter 3 Current Situation and the Problem</i> .....	8
<i>Chapter 4 Problem Description</i> .....	9
<i>Chapter 5 Design (or Solution) Approach</i> .....	13
<i>Chapter 6 Computational Results</i> .....	26
<i>Chapter 7 Implementation</i> .....	31
<i>Chapter 8 Conclusion</i> .....	32
REFERENCES.....	33
APPENDICES .....	34

# Chapter 1 Introduction

The report focuses on enhancing Fibabanka's data transfer efficiency, primarily addressing the challenge of timely data transfer from Istanbul to Ankara via physical cables, crucial for minimizing data loss during disasters. It outlines the identified problem, the bank's expectations, and various approaches taken to solve the issue. Starting with system analysis to understand its workings and requirements, the report details the creation of different small-scale samples for testing optimization methods, aiming to ensure consistency and efficiency. By considering system constraints and the nature of data transfer, multiple approaches were applied, successfully achieving timely data transfer. The report highlights that these improvements are influential in protecting Fibabanka's data backup reputation, ultimately providing a viable solution to optimize data transfer and maintain the bank's credibility post-disaster.

The project structure, outlined across eight chapters, is initiated with Chapter 1, Introduction, setting the context by defining the problem and objectives. Chapters 2 and 3 provide an in-depth examination of Fibabanka's data transfer system within the organizational framework, emphasizing the specific obstacles faced. Chapter 4 carefully details the constraints within the system and the complexities of data transfer, laying the groundwork for the subsequent solution-oriented chapters. Chapter 5 presents various strategies designed to optimize data transfer and reduce potential loss. The thorough testing procedures are used by applying different scenarios described in Chapter 6. Chapter 7 delves into the implementation phase, consisting of comparisons and decision-making based on test outcomes to reach the best decision for Fibabanka. The results are finally summarized in Chapter 8, which demonstrates not only the effective optimization of Fibabanka's data transmission but also the strategic reinforcement of the bank's credibility against future catastrophes.

## **Chapter 2 Company or Public/Nonprofit Organization**

Fibabanka, a subsidiary of the Fiba Group, established itself in 2010 with a strong emphasis on personalized customer service, rapidly expanding to 1,979 employees and 44 branches by 2022. Originating from Sitebank A.Ş., it underwent rebranding through acquisitions, ultimately becoming Fibabanka A.Ş. in 2011 under the Fiba Group. The bank's vision is to be "the most beloved bank in Türkiye" [1], aiming to attract 10 million customers within three years, prioritizing digital channels and innovative financial solutions. Operating from data facilities in Istanbul and Ankara, Fibabanka focuses on customer satisfaction, staff engagement, and achieving a high return on equity, aspiring to be recognized as the "Best Business Place in Türkiye" [1]. Services such as mobile and internet banking, telephone banking, and branches prioritize simplicity and speed, reinforced by collaborations across industries to expand service points. Subsidiaries like Fiba Portföy Yönetimi A.Ş. and Finberg Araştırma Geliştirme ve Danışmanlık Yatırım Hizmetleri A.Ş. complement Fibabanka's offerings, managing mutual funds and corporate venture capital. The bank's ownership structure comprises International Finance Corporation (IFC), European Bank for Reconstruction and Development (EBRD), and TurkFinance B.V., acquiring shares in 2015 and 2016, while Fiba Holding A.Ş. maintains the majority stake of 69.22% [1]. Specific shareholding percentages include IFC at 6.21% [2], EBRD at 8.96% [2], and TurkFinance B.V. at 9.95% [2]. As an industry leader, Fibabanka heavily invests in technology, driving innovation in products and services, while fostering a positive workplace culture to deliver superior banking experiences.



## Chapter 3 Current Situation and the Problem

Fibabanka has two data facilities (data storage centers) that were put into service 12 months ago in Istanbul and Ankara. There is simultaneous one-way data transfer from Istanbul to Ankara. Since the transfer process between the two centers is new, it does not have a fully established system. They transfer the data with two software, Zerto and Veeam, and replication in the database section is done manually by admins. Since this transmission system is not fully synchronized, there is a delay (approximately 15 minutes). Also, the bank uses the Recovery Point Objective (RPO) transfer duration limit approach; in case of a disruption (fire, earthquake, power outage, etc.) that may occur in Istanbul, which will lead to data loss as the data in Ankara is not backed up in a timely manner as there still may be data waiting to be sent. Since all systems used are directly or indirectly dependent on data, it is very important that the system works optimally, ensuring that data is sent on time. Fibabanka's request is to find a solution to reduce data loss by minimizing the delay in case of any problem or emergency, that is, by maximizing the transmission speed.

Regarding customer experience, following the problems that may occur in the facilities shortly after a large amount of deposit or withdrawal, the data of these operations may not be stored, resulting in disruptions in business continuity. This data loss can create a negative experience and a major problem for the customer. Since the number of customers who may be affected by this situation is quite high, numerous negative customer experiences can lead to major negative consequences for the bank, such as financial losses. These financial losses not only can occur due to customer dissatisfaction but also treasury-wise. This is because if the total amount of money entering or leaving the bank does not match the amount in the treasury, it may result in large fines at the end of the audits from legal institutions such as the Banking Regulation and Supervision Agency (BRSA). Considering all of these factors, it is likely that the bank's reputation will be negatively affected, which may reduce its competitive advantage.

## Chapter 4 Problem Description

Fibabanka has placed its data facilities in both Istanbul and Ankara with the ability to operate from one center's data if access to the data in the other center is unavailable. These data centers cannot perform real-time synchronized data transfer, and there is a delay in data transfer between the two centers. The reason for this delay is the limited capacity of the Network Bandwidth, which restricts the bank's system capabilities of the current Disaster Protocol as it was designed to enable data backup between the facilities. Even though Recovery Point Objective (RPO) transfer duration is determined according to the priority level of the data, there is a risk of data loss due to this network bandwidth constraint. Hence, Fibabanka's major desire is to minimize data loss between its data facilities in Istanbul and Ankara while maintaining reliable and seamless business operations during unforeseen circumstances. To satisfy this request, the design process involves developing a strategy for resolving an existing complex problem.

### Keyword Definitions:

- **Recovery Point Objective (RPO):** The maximum allowed times for sending data according to their priority levels (U1 --> 15 minutes, U2 --> 30 minutes, U3 --> 60 minutes).
- **Data Volume:** The data volume of each server.
- **Server Capacity:** The capacity allocated to each server for sending data. (changing each minute)
- **Network Bandwidth:** The maximum amount of data a network can send in every minute.
- **Total Sending Time:** The total data transfer time it took to send the whole dataset.

### Problem Formulation Inputs and Assumptions:

This project operates under the assumption that the number of servers aligns with the quantity of data units to be transferred. Additionally, it is presumed that this information, along with the priority levels (U1, U2, U3), is available at the outset of each minute. Specific data transfer times are associated with each priority level: 15 minutes for U1, 30 minutes for U2, and 60 minutes for U3. Data transfer between the Istanbul and Ankara facilities is restricted by the overall network bandwidth capacity, which is a crucial input. The optimal data transmission time depends on

allocating each server its network bandwidth capacity. The order of data transfer is also influenced by the system's operation under the RPO transfer duration principle, which considers each server's predetermined priority level and the maximum amount of time each priority level is allowed to transfer the data.

### **Outputs:**

The system produces the output of each server's data transfer duration at the end of the transfer. Moreover, it also gives the output of the total amount of data that is transferred to Ankara, satisfying the constraint of the total network bandwidth capacity that can be used.

### **Components:**

Important elements of the system include data centers in Ankara and Istanbul, servers in charge of data transfer, network bandwidth, application levels (U1, U2, U3), and data volume of each data that is going to be transferred.

### **Goals/Objectives:**

The main objective is to minimize the time it takes for each server to transfer its allocated data to the recovery center in order to minimize data loss during emergencies. This includes having certain objectives, including assigning servers to network bandwidth capacity according to data volume and priority levels. It also involves making sure that, for every priority level, the overall data transfer duration stays within the allowed RPO transfer duration. Additionally, in order to achieve a continuous solution, obtaining a continuous code was aimed to be achieved by using the Genetic Algorithm (GA) method along with simple heuristics.

### **Constraints / Conditions:**

The approach taken to find a solution to the problem at hand is restricted by a number of constraints and limitations. Particularly, there is an approximately 15-minute delay due to non-real-time synchronized data transfer and a network capacity constraint between the data facilities in Istanbul and Ankara. As it is mentioned above, the capacity allocation for each server is restricted by the system's total network bandwidth limit. Furthermore, each server can only transfer data of a specific priority level and each server can only transfer one criticality level data.

### **Performance Measures:**

Key metrics that will be considered during the analysis are the amount of capacity allocated to each server, the total data transfer time of the system, (for continuous solution) the number of iterations

that it took to send the whole dataset, average capacity utilization, and whether the data sent within in the RPO constraint of the application (Success / Failure).

### **Alternative Courses of Action:**

The model provides control over decision variables, which are the allocation of the capacities for each server, including application numbers and server allocations. This allows the model to facilitate the generation of diverse outcomes and the identification of feasible solutions. The objective is an in-depth evaluation of possibilities, exploring various paths of action with regard to factors under control. This objective makes it possible to cultivate a comprehensive understanding of implications associated with decisions on application levels and server provisions.

According to the definition of the problem, it is observed that there are a number of constraints, such as technical, social, environmental, and financial ones. First of all, the Turkish Banking Regulation and Supervision Agency (BRSA) restricts the use of cloud storage due to parties that provide cloud services in foreign countries, which creates a risk of unauthorized access to personal data by third parties. Hence, data centers can only communicate with one another via physical cables. Reliance on physical data transfer exposes companies to environmental risks like infrastructure problems or natural disasters. The system uses a social framework that reflects a predefined organizational structure and decision-making process by assuming a predefined set of servers and priority levels. Financially, the selected strategy must minimize expenses related to possible data loss and the ensuing recovery operations while optimizing data transfer within the specified constraints and parameters. Therefore, the suggested solution attempts to address the distinct difficulties identified by Fibabanka, taking into account the physical, technical, social, environmental, and financial limitations particular to the enterprise's data transfer necessities.

The group utilized the expertise of IE342 to establish a mathematical model for the problem. Additionally, they leveraged the concepts obtained from IE343 to formulate algorithms and pseudo codes for the heuristics in Approaches 1 and 2 (Simple Heuristics and Genetic Algorithm). These approaches were seamlessly converted into operational heuristic programs with guidance from IE246. Furthermore, IE246 facilitated the development of a simulation coding approach in Python, offering an alternative method to solve the problem. This model not only identified constraints but also represented project limitations, furnishing the group with a robust foundation for the optimization process. Drawing upon the knowledge acquired from IE383, the group employed

Excel Solver to ascertain an optimal solution. Lastly, IE325 played a pivotal role in guiding the creation of the Arena simulation model, which, in turn, inspired the development of simulation code in Python.

**Table 4.1: List of IE Courses and their Utilization in the Project**

<b>Industrial Engineering Courses</b>	<b>Where is it used?</b>
IE 343 (Mathematical Modelling and Heuristic Methods)	Algorithm
IE 383 (Decision Support Systems)	Excel Solver
IE 246 (Programming for Operation Research)	Heuristic (Java Codes) and Simulation Code (Python)
IE 342 (Mathematical Modelling and Exact Methods)	Math Modeling
IE 325 (Simulation Modelling and Analysis)	Simulation Model

## Chapter 5 Design (or Solution) Approach

Initially, a comprehensive mathematical model was formulated, outlining its associated constraints and limitations, serving as the foundation for all subsequent approaches. The project encompasses three distinct approaches: one exact and two heuristic approaches (simple heuristic and genetic algorithm). Furthermore, a simulation-based approach is being implemented via Arena, facilitating the creation of a simulation model. Moreover, based on the Arena model, a Python code is created. The exact approach involves the use of Excel Solver as a tool to address the problem at hand. This choice is motivated by several factors. Firstly, Excel Solver is efficient and simple to use with modest sample sizes. Importantly, it was selected due to the non-linear nature of our mathematical model, and Excel Solver provides the capability to address non-linear models. Java's versatility, efficiency, and platform independence make it the preferred choice for heuristic coding, enabling the implementation of complex algorithms across diverse domains with speed and reliability.

Moreover, Arena Simulation has been chosen for the simulation approach. This decision is attributed to Arena's comprehensive features for modeling and analyzing complex systems. Arena provides a user-friendly interface for creating simulation models, allowing for a detailed representation of the systems under study whilst changing controllable factors. The decision to utilize Python for developing simulation-based code reflects a strategic choice based on its user-friendly syntax, and robust scientific libraries. Despite potential performance trade-offs, Python's overarching strengths in versatility and ease of use render it a rational selection for agile and collaborative simulation attempts.

### **Adaptation of All Approaches in Terms of Achieving Continuity in the System:**

Since approaches give results for a discrete point of time, to make the approach applicable to continuous flow, if there are any pieces of remaining data volume after the run of the algorithm, then the remaining data volume updated to get processed again. According to the results of the solution, this update happens when the current server data volumes are reduced by the capacity

allocated by the algorithm at that minute. Also, the remaining time for the server to send the data is reduced by one minute. The updated dataset gets processed again, and approaches generate the necessary outputs. This cycle is followed until no remaining data is left, and each cycle is called one iteration. Since the model is run at one-minute intervals, one iteration corresponds to one minute of sending time. At the end of the cycle, when no remaining data is left, the number of iterations done is counted to determine how many minutes it took to send all the data to the other data storage center. This is because the capacities are changing every minute, meaning every iteration, according to the data volume of each server and the output of each iteration's minimum total data transfer time, actually gives the estimated total time for each server to transfer its allocated data to the recovery center if the model has static capacity. Hence, the number of iterations must be counted during approaches to understand the total data transfer time.

## 1. Mathematical Modeling

### Sets:

$i$  be the set of servers, denoted as  $i = \{1,2,3,\dots,1000\}$

$j$  be the set of priority level applications, denoted as  $j = \{U1, U2, U3\}$

### Parameters:

$Z_{ij}$ : 1 if the application  $j$  is transferred from server  $i$ , and 0 otherwise.

$V_i$ : Amount of data volume transferred from server  $i$ .

$T_j$ : Recovery Point Objective (RPO), also known as the total time it takes to transfer application  $j$ .

$N$ : Total Network Bandwidth

### Decision Variable:

$X_{ij}$ : Amount of capacity used by server  $i$  for priority level application  $j$ , where  $i = \{1,2,\dots,n\}$  and  $n$  is the number of servers (20,100 or 1000), and  $j = \{1, 2, 3\}$  represents criticality levels (U1, U2, U3).

### Auxiliary Variable:

$S_i$ : Capacity of server  $i$  for each minute (planning horizon)

### Objective Function:

$$\min \sum_{i=1}^n \frac{V_i}{S_i} \quad (5.1)$$

s.t:

$$\sum_{i=1}^n \frac{V_i}{S_i} \leq T_j \quad \forall_j \in \{1,2,3\} \quad (5.2)$$

$$\sum_{j=1}^3 X_{ij} * Z_{ij} \leq T_j \quad \forall_i \in \{1,2,\dots,n\} \quad (5.3)$$

$$\sum_{i=1}^n S_i \leq N \quad \forall_i \in \{1,2,\dots,n\} \quad (5.4)$$

$$V_i, T_j, X_{ij}, S_i \geq 0 \quad (5.5)$$

#### **Explanation of the mathematical model:**

In (5.1), the objective function's goal is to minimize the summation of the ratio of the amount of data volume transferred from the server  $i$  to the amount of capacity used by the server  $i$ , which yields the total time each server takes to transfer the data. Subsequently, in the constraints part of the mathematical model, first (5.2) checks whether the time it takes server  $i$  to transfer the data is less than or equal to the Recovery Point Objective (RPO) allocated to application  $j$ . Then, in (5.3), if the server  $i$  has the application  $j$ , then  $Z_{ij}$  will be equal to 1, and this equation will give the amount of capacity of the server  $i$  assigned to  $S_i$ . If the server  $i$  does not contain a certain application,  $Z_{ij}$  will be equal to 0, and it will not assign a capacity to the server  $i$ . In (5.4), the equation checks if the total capacity allocated to all servers is less than or equal to the total bandwidth constraint. Lastly, in (5.5) the non-negativity constraints are defined.



## **2. Exact Approach**

- Inputs: List of servers, application levels of each server, data volume, RPO constraint, and bandwidth constraint
- Outputs: The minimum total data transfer time, The total capacity allocated to all servers, Remaining Data Volume (if there is any)

### **Explanation of the Approach:**

In this approach utilizing Excel's Solver tool, the focus is on optimizing how server capacities are assigned to minimize data transfer time while meeting specific criteria. The process involves creating a mathematical model within Excel, aiming to minimize the sum of ratios calculated from data volume divided by server capacity. This ensures efficient data transfer among servers by adjusting their allocated capacities while maintaining non-negativity within these ratios. Servers are designated to handle only one application, enhancing operational efficiency. Additionally, the model considers the constraints of Recovery Point Objective (RPO) time, ensuring that the calculated ratios do not exceed this predefined threshold. Data volume for each server is upheld at zero or higher, and the total capacity allocated across servers is constrained not to exceed the available network bandwidth. Leveraging the "GRG Nonlinear" solving method in Excel, this approach yields the optimal solution, providing the most efficient distribution of server capacities to minimize data transfer time while satisfying all specified conditions.

## **3. Heuristics**

The problem of scheduling data transfers between multiple servers with different capacities and Recovery Point Objective (RPO) times can be formulated as a mathematical optimization model. The model has variables representing factors such as server capacities, data transfer times, and RPO times, and an unpredictable number of servers that increases or decreases depending on the amount of data. Because of these constraints, it is difficult to solve this problem completely using traditional solvers. Therefore, two heuristic approaches are proposed in this report. These heuristic approaches are based on the idea of iteratively improving servers' sending times until a solution that satisfies all constraints is found.

### 3.1 Simple Heuristic

- **Inputs:** List of servers, Each server with its own server ID, Application Level, Data volume, RPO, Sending time, Best sending time, Bandwidth
- **Outputs:** The best sending time for each server, The capacity of each server, The total volume of data transferred in that minute, The total sending time
- **The following flowchart outlines the steps of the heuristic solution algorithm:**

Start → Read Excel file → Initialize the iteration → Set the repetition number → Initialize sending times → Calculate total volume and sending time → New best solution found? → Yes: Set best sending times to current sending times; No: Keep the best sending times → Update total volume and sending time → Increase the iteration → Is an iteration less than the repetition number → Yes: Repeat the process at calculating the total volume and sending time; No: Print results → End

- **Explanation of the heuristic solution algorithm:**

The heuristic solution algorithm begins by initializing the sending times of all servers to their respective RPOs. Subsequently, the algorithm iteratively refines these sending times to find an optimized solution. In each iteration, the algorithm checks if the current configuration satisfies the given bandwidth and time constraints. If the best solution is identified, it updates the best sending times of the servers accordingly. The algorithm then proceeds to decrement the sending time of each server by one unit, ensuring that it does not fall below zero. In case a server's sending time becomes zero, it is reset to its RPO. The process continues for a predefined number of repetitions, allowing the algorithm to explore different configurations. The goal is to converge towards a solution that minimizes the total sending time while sticking to the specified constraints. Throughout the iterations, as the algorithm finds a better result, it keeps the sending times as the best sending times. Upon reaching the specified number of repetitions, the algorithm calculates the total sent data and total time based on the best sending times of all servers. The resulting configuration represents the best solution that satisfies the constraints and minimizes the overall sending time. The algorithm ends with printing the calculated results, providing information on the solution obtained within the defined computational constraints.

### 3.2 Genetic Algorithm

- **Inputs:** Sending time values of the servers generated from the Simple Heuristic approach.

- **Outputs:** The optimal sending time values array closest to the results of the crossover sending time values, with the smallest sending time and capacity that does not exceed the bandwidth.

- **The following flowchart outlines the steps of the heuristic solution algorithm:**

Start → Initialize the population → Crossover Method → Fitness Test → Updating the best solution → Result

- **Explanation of the Genetic Algorithm:**

- Population Initialization: A population of server configurations is created, each representing a potential solution to the optimization problem. These configurations consist of sending time values and server capacities (taken from the Simple Heuristic approach).
- Crossover Method: In each generation, two-parent configurations are randomly selected from the population. A crossover operation is performed between the selected parents to generate two child configurations. This crossover operation combines parts of the parent configurations to create new ones, mimicking genetic recombination.
- Fitness Test: After crossover, each child configuration's total sending time values are summed. If the sum exceeds the maximum bandwidth capacity, the child configuration is deemed invalid (-1 fitness value). Otherwise, the sum of sending time values serves as the fitness value for the child configuration.
- Updating the Best Solution: In each generation, the best-performing child configuration (lowest fitness value, valid) is retained. This process repeats over multiple generations to continually improve the server configuration.
- Results: After a specified number of generations, the algorithm prints the best-performing server configuration along with its fitness value and capacities and passes it to the Simple Heuristics + Genetic Algorithm.

This code demonstrates the application of a genetic algorithm to address the complex problem of server allocation optimization. By iteratively evolving potential solutions through crossover and fitness evaluation, the algorithm aims to converge towards an optimal server configuration that balances performance and resource constraints within the network infrastructure. Through this

process, it seeks to minimize sending time while ensuring that server capacities do not exceed available bandwidth, thereby improving overall network efficiency and reliability.

**Table 5.1: Comparisons of Heuristic Approaches**

Heuristic \ difference	Strategy	Solution Representation	Evaluation	Information Sharing
Simple Heuristic	Iterative improvement	Sending times & Server capacities	Meets constraints & minimizes sending time	No explicit sharing
Simple Heuristic + Genetic Algorithm	Feasible results gathered from Simple Heuristic are used during population-based exploration	Sending times & Server capacities	Sum of sending times values (fitness) - penalized for exceeding bandwidth	Crossover exchanges information between solutions

#### 4. Simulation Approach

- **Inputs:** number of servers, number of entities, data volume of entities, RPO constraint, and server capacities
- **Outputs:** The total data transfer time, minimum, maximum, and average transfer times, and the count of entities transferred within and outside their respective RPOs.

##### **Explanation of the Approach:**

Fibabanka's strategy for addressing data transfer challenges employs an Arena Simulation framework. This simulation divides applications into three levels: U1, U2, and U3, each with distinct priorities and transfer time objectives. To manage this, the simulation employs process modules tailored to each entity type. These modules ensure that applications are allocated to servers according to their level and processed within designated timeframes. Additionally, the simulation randomly assigns data volumes and server capacities within predefined ranges to mimic real-world variability. After processing, an assign module deducts transferred data volumes and prioritizes entities for further processing based on their remaining data volume. This ensures that applications are efficiently managed and transferred within their Recovery Point Objectives (RPOs). Decide modules are then employed to verify if transfer criteria are met. These modules check if an entity's

data volume is fully transferred and if the transfer time falls within the specified RPO. Entities failing to meet these criteria are directed to a dispose module and documented accordingly. Throughout the simulation, record modules track the progress of application transfers for each entity level, recording the number of applications transferred and the time spent within the system. This data provides insights into the efficiency of the transfer process. Finally, the simulation runs for a 24-hour period to assess the effectiveness of the approach in handling data transfer challenges.

## 4.1 Simulation Code

- **Inputs:** Excel files consisting of server capacity, data volume, and entities
- **Outputs:** Minute-by-minute entity state, Total Entities Processed, Entities, Not Processed Within RPO Time, Entities Processed Within RPO Time, Total Entities Processed Within RPO Time, Total Entities Not Processed Within RPO Time, Total Data Volume Transferred for Each Entity Type, Total Server Capacity Transferred for Each Entity Type, Total Data Volume Transferred, Total Server Capacity Transferred, Average Utilized Capacity, Transferring Time Statistics for Each Entity Type, Total Data Volume Not Transferred, Total Transfer Time for All Entities
- **The following flowchart outlines the steps of the simulation code algorithm:**  
 Start → Import Libraries → Initialize Simulation Environment → Create Entities → Process Entities → Run Simulation → Generate Report → Optimization Insights

### Explanation of the Simulation Code:

- **Import Libraries:**
  - Import "SimPy" and "pandas" libraries
- **Initialize Simulation Class:**
  - Create a class named "DataTransferSimulation".
  - Initialize dictionaries to track metrics like data transferred, processing status, and transferring times.
  - Set parameters such as RPO, priority levels, data volume, and server capacity.
- **Create Entities:**
  - Call the create\_entities method to read entity data from an external Excel file.
  - Validate the data for integrity and consistency.

- Generate unique entity IDs for each entity.
- Schedule entities within the simulation environment.
- **Process Entities:**
  - Implement the process\_entity method to simulate entity processing within the data transfer system.
  - Request access to a server.
  - Calculate transfer time based on data volume and server capacity.
  - Progress simulation by the calculated duration.
- **Run Simulation:**
  - Execute the simulation until a specified time duration ("sim\_time") is reached.
  - Progress simulation environment by advancing time.
  - Create, process, and record entity actions.
  - Generate a comprehensive report containing various recorded data and statistics.
- **Generate Report:**
  - Include metrics such as total entities processed, entities processed within RPO, data volume transferred, server capacity transferred, average utilized capacity, transferring time statistics, data volume not transferred, and total transfer time.
- **Optimization Insights:**
  - Utilize the generated report to gain insights into optimizing and improving the efficiency of data transfer systems.

In summary, the script demonstrates the construction of a discrete-event simulation model using SimPy to analyze data transfer processes. By incorporating real-world data and simulating data transfer scenarios, the script provides actionable insights for optimizing and improving the efficiency of data transfer systems.

## 5. Literature Review

The difficulties we encountered in trying to solve Fibabanka's problems with regard to maximizing data transfer speed while minimizing data loss in emergency situations are very similar to the difficulties discussed in the research paper by Ahani et al. [3] In the context of the mathematical

model, a similar theoretical basis to Fibabanka's problem formulation is provided by the routing and scheduling problem mentioned in the paper [3]. Both scenarios emphasize the value of simulating network flows while taking capacity, deadlines, and flow rate constraints into account, showing a common restriction between them. The article's method for creating columns is similar to Fibabanka's focus on maximizing data transfer speed. In order to reach Recovery Objective Points (RPO) within specified timeframes, it is necessary to efficiently distribute network bandwidth capacity. Column Generation Approach (CGA) accomplishes this by splitting the issue down into smaller, controllable parts.

Fibabanka's goal of minimizing data loss during emergencies is similar to that of the Maximum-Flow-Based Algorithm (MFA) described in the article. MFA prioritizes flows according to deadlines, which aligns with Fibabanka's requirement to follow Recovery Objective Points (RPO) for every priority level, maximizing the overall flow rate in emergency scenarios.

Furthermore, the main focus of both the group's solution to Fibabanka's problem and Vdovin et al.'s research paper [4] is the optimization of critical information center operations. In order to allocate resources effectively in data centers, the paper [4] emphasizes the significance of efficient schedulers. This idea is consistent with Fibabanka's objective of minimizing data loss during emergencies by optimizing data transfer times between facilities. Also, the paper's [4] analysis of several algorithms for data center resource allocation aligns with Fibabanka's strategy of considering multiple priority levels (U1, U2, U3) in their data transfer optimization project. Both emphasize the significance of flexible and comprehensive approaches that strike a balance between numerous domains of resource distribution.

The mathematical model presented by Vdovin et al. [4] lays the groundwork for developing algorithms by defining physical and virtual resource requirements. Similarly, the mathematical model generated by the IE 402 group assumes a predefined set of servers, priority levels, and network bandwidth capacities, forming a mathematical foundation for their data transfer optimization. Vdovin et al. [4] and this project takes into account how their suggested solutions might be used in real-world situations. While Fibabanka prioritizes data transfer based on predetermined criticality levels in order to satisfy urgent business demands first, the paper [4] investigates efficiency under various load conditions and request patterns.

The problem of effectively allocating spectrum in cognitive radio networks is addressed in the research paper "Hybrid Optimization Algorithms for Resource Allocation in Heterogeneous Cognitive Radio Networks" by Yuvaraja Teekaraman et al.[5]. The suggested system seeks to optimize network capacity, data rates, and interference reduction by including a hybrid optimization algorithm and decision-making process. The group's project approaches draw inspiration from the research article [5], employing alternative methods to achieve improved outcomes while recognizing the complexities of resource allocation in cognitive radio networks. In addition to addressing the challenges of spectrum sharing and interference reduction in heterogeneous cognitive networks, both the group's project with Fibabanka and the project mentioned in the paper [5] seek to increase data transfer rates. It is common to validate hypotheses through simulations, and both projects use simulations to demonstrate how effectively their suggested models work.[5]

Quality of Service (QoS) improvement is a shared focus, emphasizing the optimization of network parameters for an enhanced user experience. In summary, the group's project aligns with existing research, contributing to the ongoing optimization of data transfer and improving the cognitive networks.[5]

A method for effectively allocating applications to physical resources in data centers is described in the article "Algorithm for Resource Allocation in Data Centers with Independent Schedulers for Different Types of Resources" by P. M. Vdovin et al. (2014)[6]. The algorithm tackles the problem of striking a balance between computational complexity and assignment quality by integrating greedy and limited search strategies. This evaluation of the literature examines relevant work that supports the objectives of the project, which is to optimize data transport in a manner similar to the optimization of various resources in data centers.

An organized way of representing data center resources and requests is provided by the math model presented in the article. This is in accordance with the approach used in this report, in which, the server capacities are maximized for enhanced data transfer efficiency using a mathematical model [6]. The significance of complex algorithms for efficient utilization of resources is emphasized in both the project and the reviewed article [6]. Although the criteria are different, both aim to maximize resource utilization and furnish an analysis of the results in terms of complexities.



The exploration of solutions for optimizing data transfer in complex network systems can be significantly advanced through the use of genetic algorithms. These algorithms, which replicate natural selection processes including selection, crossover, and mutation, are adept at navigating and enhancing solutions within extensive and dynamic search landscapes [7]. This methodology is crucial in addressing optimization challenges similar to those encountered by Fibabanka, where the objective is to minimize data loss while maximizing data transfer speed across network bandwidth constraints. The inherent flexibility and adaptability of genetic algorithms make them ideal for such complex scenarios, allowing for continuous improvement and adaptation of solutions over time.

In the research conducted by Keklik and Özcan (2023) [8], the Two-Point Crossover method is highlighted as a critical advancement in genetic algorithm techniques. This method entails swapping genes between two specific points on a chromosome, thereby significantly increasing the diversity of the genetic pool and the quality of potential solutions [8]. This increased diversity is crucial in environments with stringent constraints, such as the network systems at Fibabanka, where optimizing data transfer to minimize sending time and meet specified Recovery Point Objectives (RPO) is essential.

Applying this method within Fibabanka's context, the Two-Point Crossover technique is used to develop diverse server configurations that optimally utilize the limited network bandwidth. By recombining server data in this manner, the genetic algorithm not only introduces a broad range of potential solutions but also increases the likelihood of finding an optimal configuration that can effectively manage network resources. This is particularly important for ensuring that each data transfer not only adheres to network capacity limitations but also aligns with the bank's priority levels, thus optimizing the overall data transfer duration within the allowed RPO timeframes.

The adoption of the Two-Point Crossover in Fibabanka's optimization strategy effectively harnesses the genetic algorithm's capability to explore and refine solutions, thus ensuring that the server configurations meet the critical balance between minimizing sending times and adhering to recovery objectives. According to Keklik and Özcan [8], such a technique directly

contributes to avoiding local optima and achieving superior solutions under dynamic conditions, which is pivotal for maintaining robust and efficient operations during network-critical scenarios.

All in all, the review of the literature emphasizes how crucial effective resource allocation is, particularly in data centers. The findings are consistent with the project's initiative, demonstrating a common focus on achieving efficiency through careful heuristics and an understanding of the mathematical difficulties involved in resource optimization.[6]

**Table 5.2: List of Articles Referenced in the Project**

<b>Research Article</b>	<b>Chapter</b>
[3]“Routing and scheduling of network flows with deadlines and discrete capacity allocation,”	The research article is cited in Chapter 4 to highlight the similarities between it and this project.
[4] “Comparing various approaches to resource allocation in data centers.”	The research article is referred to in Chapter 5 to emphasize the alignment of its mathematical model with Fibabanka's project, highlighting the shared objectives of both projects.
[5]“Hybrid Optimization algorithms for resource allocation in heterogeneous cognitive radio networks.”	Chapters 4 and 5 mention the article to highlight similarities across projects and provide proposals for heuristic methods employed in this project
[6] “Algorithm for resource allocation in data centers with independent schedulers for different types of resources”	The study article is cited in Chapter 5 to highlight how the project and it share influenced methods and heuristics.
[7] “IE 504- Heuristic Methods VIII. Genetic Algorithms”	This presentation discusses the steps of genetic algorithms extensively from page 12 onwards, which have been employed in the heuristic approach of this project.
[8] “Genetik Algoritmaların İşleyişi ve Genetik Algoritma Uygulamalarında Kullanılan Operatörler”	This article is cited to highlight the Two-Point Crossover method we have implemented in this project, which involves designating two fixed points on a chromosome and exchanging the genes between them to enhance genetic diversity and optimize solutions.

## Chapter 6 Computational Results

For this project, solution methods were tested with three different approaches. These solution methods were examined in various scenarios to determine their efficacy under different conditions and to observe the extent of their impact on results. Scenarios were designed to assess the performance of solution methods based on varying numbers of servers. The success rates of solution methods were initially measured in terms of the Key Performance Indicator (KPI) of total sending time. The capacity utilization rate was also selected as an additional parameter that shows how much the system has improved. In order to analyze the results, 6 different scenarios were constructed. The characteristics of these scenarios are as follows.

**Table 6.1 Scenarios**

<b>Scenarios / Characteristics</b>	<b># of Servers</b>	<b>Distribution of Applications (U1, U2, U3)</b>	<b>Initial Data Volume (mb)</b>	<b>Bandwidth</b>
<b>Scenario 1</b>	20	Randomly	6.759 mb	2250
<b>Scenario 2</b>	30	Randomly	5.443 mb	2250
<b>Scenario 3</b>	20	U1 Abundant	6.528 mb	2250
<b>Scenario 4</b>	20	U2 Abundant	5.648 mb	2250
<b>Scenario 5</b>	20	U3 Abundant	7.383 mb	2250
<b>Scenario 6</b>	20	Randomly	11.351 mb	2250

Scenarios 1 and 2 are constructed for performance evaluation based on varying server counts. Scenario 3, 4, and 5, meanwhile, are designed to understand the impact of the dominant application on the system. Scenario 6 is designed for internal comparison of three approaches within a high-volume data context.

**Table 6.2 Distribution of Applications Levels**

Scenario / Abundance Metrics	# of Applications (U1, U2, U3)	Percentages in Terms of mb
Scenario 3	(14,1,5)	<p>A pie chart for Scenario 3 showing the distribution of applications. The chart is divided into three segments: a large blue segment for U1 (68%), a green segment for U3 (30%), and a small orange segment for U2 (2%).</p>
Scenario 4	(1,14,5)	<p>A pie chart for Scenario 4 showing the distribution of applications. The chart is divided into three segments: a large orange segment for U2 (60%), a green segment for U3 (37%), and a small blue segment for U1 (3%).</p>
Scenario 5	(3,3,14)	<p>A pie chart for Scenario 5 showing the distribution of applications. The chart is divided into three segments: a large green segment for U3 (80%), and two smaller segments for U1 (10%, blue) and U2 (10%, orange).</p>

**Table 6.3 Total Sending Time S1 vs. S2 (based on minutes)**

Instance	S1 (20 server)	S2 (30 server)
Exact	3	Inconclusive
Simple Heuristic	4	7
Simple Heuristic + GA	4	5

In the comparison between Scenario 1 and Scenario 2, as outlined in Table 6.3 and Figure 6.1, the larger server size in Scenario 2 presents a challenge for the exact solution, rendering it unable to

produce results. However, heuristic methods, particularly the simple heuristic, demonstrate an increase in sending time from 4 to 7 minutes when transitioning from Scenario 1 to Scenario 2. The integration of the genetic algorithm into the heuristic approach shows promising outcomes, resulting in a reduction of sending time from 7 to 5 minutes in Scenario 2 compared to relying solely on the simple heuristic. This improvement highlights the genetic algorithm's effectiveness in optimizing server allocation, especially in scenarios with larger server size where exact solutions may falter.

**Table 6.4 Total Sending Time S3 vs. S4 vs. S5 (based on minutes)**

	<b>S3</b> <b>(U1 Abundant)</b>	<b>S4</b> <b>(U2 Abundant)</b>	<b>S5</b> <b>(U3 Abundant)</b>
<b>Exact</b>	3	3	4
<b>Simple Heuristic</b>	4	4	5
<b>Simple Heuristic + GA</b>	4	3	4

Table 6.4 and Figure 6.2, presents the outcomes of solution methods across all scenarios. The dominant application indirectly influences the system through diverse pathways. Owing to the application-based varying RPO constraint, disparities in the results of solution methods emerge as the system must allocate more capacity to these applications.

When comparing different approaches across scenarios, the exact solution consistently demonstrates the lowest total sending time, highlighting its optimality. In Scenario 3, both the exact solution and the GA approach achieve a total sending time of 3 minutes, surpassing the Simple Heuristic. Similarly, in Scenario 4, both the exact solution and the GA approach achieve a total sending time of 4 minutes, outperforming the Simple Heuristic. These findings underscore the effectiveness of integrating the genetic algorithm into the heuristic approach, particularly evident

in scenarios where the Simple Heuristic falls short. However, it's important to note that the genetic algorithm doesn't consistently outperform the Simple Heuristic across all scenarios, emphasizing the significance of considering specific context and dataset characteristics when selecting optimization strategies. Consequently, applying the genetic algorithm to the Simple Heuristic produces equivalent results to the optimal solution in Scenarios 4 and 5.

Upon reviewing the tables above, it becomes apparent that all solution methods produce comparable results. However, a noteworthy observation is that the exact approach consistently delivers the optimal outcomes across all scenarios. Furthermore, the genetic algorithm generates identical results to the exact approach in all scenarios except for Scenario 3. These findings lead to the conclusion that the heuristic approach, augmented by the genetic algorithm, closely approximates the exact solution in nearly all scenarios.

**Table 6.5 Average Utilization of Capacity (percentage)**

	<b>S3</b> <b>(U1 Abundant)</b>	<b>S4</b> <b>(U2 Abundant)</b>	<b>S5</b> <b>(U3 Abundant)</b>
<b>Exact</b>	97%	84%	82%
<b>Simple Heuristic</b>	73%	63%	66%
<b>Simple Heuristic + GA</b>	73%	84%	82%

Table 6.5 and Figure 6.3 illustrates the average capacity utilization rates, where the capacity isn't fully utilized due to the remaining data in the last iteration being less than the total capacity. However, as evident from the table, the simple heuristic values range from 60% to 70% on average, which falls below an acceptable level of capacity utilization. Conversely, the genetic algorithm has managed to elevate this rate to approximately above 80% and has nearly reached the exact solution method.

**Table 6.6 Scenario 6 Outputs of the Approaches**

<b>Simulation Code Output</b>	<b>Exact</b>	<b>Simple Heuristics</b>	<b>Genetic Algorithm</b>
<b>Total Transfer Time(min)</b>	<b>5.045</b>	<b>7.084</b>	<b>6.112</b>
<b>Average Utilized Capacity</b>	<b>90.48%</b>	<b>75.65%</b>	<b>83.25%</b>
<b>Total Entities processed within RPO</b>	<b>20</b>	<b>20</b>	<b>20</b>
<b>Total Entities processed Outside RPO</b>	<b>0</b>	<b>0</b>	<b>0</b>

According to Table 6.6, Figure 6.4 and Figure 6.5, the approach exhibiting the smallest total transfer time is deemed optimal as it yields the most efficient solution. Furthermore, the genetic algorithm demonstrates superior performance compared to the simple heuristic in Scenario 6. Notably, the genetic algorithm's output closely aligns with the optimal solution in terms of average utilized capacity, surpassing the simple heuristic's performance. In essence, the genetic algorithm consistently produces outputs that are closer to the optimal solution than those generated by the simple heuristic approach.

In summary, for systems with 20 servers or fewer, the exact solution should be prioritized. However, for systems with a larger number of servers, the genetic algorithm emerges as the most suitable choice. The company operating with an unspecified number of servers or those aiming to maintain algorithm consistency may find the genetic algorithm, which is compatible with all systems, to be the most suitable approach.

## Chapter 7 Implementation

The regular updates of server arrivals, data volumes, and RPO values within the company are performed through a dedicated list. In the project, the process was streamlined by transferring the server list to an Excel file. This facilitated a more convenient approach during coding, as the Excel file could be read to progress. To get the best solution for scenarios involving more than 20 servers, Genetic Algorithm (GA) approach presents a viable solution. While Solver's limitations restrict its applicability to modest server sizes, employing GA enables efficient processing across larger server counts. This approach ensures the company can effectively address optimization challenges while maintaining computational feasibility.

The resources for this project include Microsoft Excel software, programming languages such as Java and Python, and a set of data (server ID, application name, data volume, and RPO values).

Unlike the software methods employed by the company, a licensing fee may be required for the approaches. Therefore, financial support may be deemed necessary from this perspective. The technical and operational traceability of the application has been organized by adopting an Excel file as the shared document among various approaches, including simulation python code, genetic algorithm approach, and simple heuristic. This decision enhances collaboration and ensures consistency across different methods, facilitating efficient tracking and analysis of results.

Due to the unavailability of Fibabanka's original data, the opportunity to thoroughly examine their systems was not available. During discussions with the company representative, the team proposed the genetic algorithm approach as the best approach to address the problem. This selection followed a thorough comparison with other methods, affirming the genetic algorithm approach's suitability for the company's needs and requirements. This collaborative process ensures alignment between the chosen solution and the company's objectives.



## Chapter 8 Conclusion

Following the determination of the problem statement, various solution approaches were developed based on the mathematical modeling. In the context of these solution approaches, different scenarios were tested, and variations of approaches based on parameters were explored. The exact solution was calculated.

The privacy of personal data in banks holds undeniable significance. Therefore, the delay in data transfer and data loss are crucial both from the perspective of the bank and the customer. In this context, minimizing the delay in data transfer is of great importance for maintain trust in operations among the shareholders of the bank during a disaster recovery, as it also minimizes data los of the banks. It is anticipated, in the solution alternatives, that this problem will be resolved in the most effective way.

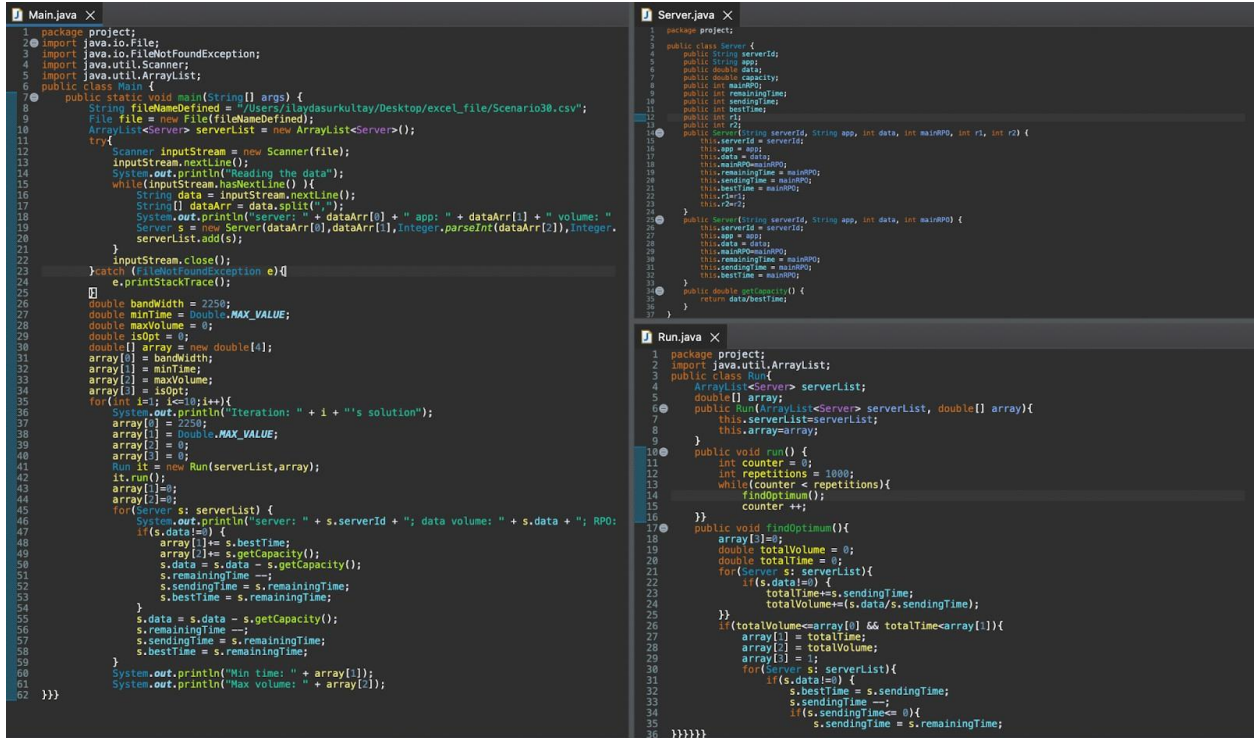
According to the comparisons made, it can vary which approach provides results closer to the optimal. For instance, if the number of servers is 20 or less than 20, exact approach is the optimal, while for more servers, simple heuristic + GA tends to yield closer results to optimal. Since the number of servers is generally greater than 20, the most logical approach is Simple Heuristic + Genetic Algorithm approach.

# References

- [1] Fibabanka A.Ş., “AboutFibabanka,” Fibabanka, <https://www.fibabanka.com.tr/en/about-us/meet-fibabanka/about-fibabanka> (accessed Dec. 2, 2023).
- Fibabanka A.Ş., “2022 Faaliyet Raporu - Fibabanka,” Fibabanka, [https://www.fibabanka.com.tr/docs/default-source/faaliyet-raporlar%C4%B1-4/faaliyetraporu2022.pdf?sfvrsn=5805755c\\_9](https://www.fibabanka.com.tr/docs/default-source/faaliyet-raporlar%C4%B1-4/faaliyetraporu2022.pdf?sfvrsn=5805755c_9) (accessed Dec. 2, 2023).
- [2] G. Ahani, P. Wiatr, and D. Yuan, “Routing and scheduling of network flows with deadlines and discrete capacity allocation,” *Networks*, vol. 76, no. 1, pp. 54–74, Jul. 2020. Accessed: Dec.1, 2023. [Online]. Available: <https://onlinelibrary-wiley-com.offcampus.ozyegin.edu.tr/doi/10.1002/net.21939#>
- [4] P. M. Vdovin, I. A. Zotov, V. A. Kostenko, A. V. Plakunov, and R. L. Smelyanskiy, “Comparing various approaches to resource allocation in data centers. ,” *Journal of Computer and Systems Sciences International*, vol. 53, no. 5, pp. 689–701, 2014. doi:<https://doi.org/10.1134/s1064230714040145>
- [5] Teekaraman, Y., Manoharan, H., Basha, A. R., & Manoharan, A. (2020). Hybrid Optimization algorithms for resource allocation in heterogeneous cognitive radio networks. *Neural Processing Letters*, 55(4), 3813–3826. <https://doi.org/10.1007/s11063-020-10255-2>
- [6] Vdovin, P. M., & Костенко, B. A. (2014). Algorithm for resource allocation in data centers with independent schedulers for different types of resources. *Journal of Computer and Systems Sciences International*, 53(6), 854–866. <https://doi.org/10.1134/s1064230714050141>
- [7] B. Balçık, “Genetic Algorithms Presentation IE 504,” in *Ozyegin University IE 504*, Mar. 20, 2024
- [8] G. Keklik and B. D. Özcan, “Genetik Algoritmaların İşleyişi ve Genetik Algoritma Uygulamalarında Kullanılan Operatörler” *ResearchGate*, [https://www.researchgate.net/publication/369142132\\_Genetik\\_Algoritmalarin\\_Isleyisi\\_ve\\_Genetik\\_Algoritma\\_Uygulamalarinda\\_Kullanilan\\_Operatorler](https://www.researchgate.net/publication/369142132_Genetik_Algoritmalarin_Isleyisi_ve_Genetik_Algoritma_Uygulamalarinda_Kullanilan_Operatorler) (accessed Mar. 20, 2024).

# Appendices

## Chapter 5: Simple Heuristic



```
1 package project;
2 import java.io.File;
3 import java.io.FileNotFoundException;
4 import java.util.Scanner;
5 import java.util.ArrayList;
6 public class Main {
7     public static void main(String[] args) {
8         String fileNameDefined = "Users\\laydasurkultay\\Desktop\\excel_file\\Scenario30.csv";
9         File file = new File(fileNameDefined);
10        ArrayList<Server> serverList = new ArrayList<Server>();
11        try {
12            Scanner inputStream = new Scanner(file);
13            inputStream.nextLine();
14            System.out.println("Reading the data");
15            while(inputStream.hasNextLine()) {
16                String data = inputStream.nextLine();
17                String[] dataArr = data.split(",");
18                System.out.println("server: " + dataArr[0] + " apps: " + dataArr[1] + " volume: "
19                Server s = new Server(dataArr[0],dataArr[1],Integer.parseInt(dataArr[2]),Integer.
20                serverList.add(s);
21            }
22            inputStream.close();
23        } catch (FileNotFoundException e) {
24            e.printStackTrace();
25        }
26        double bandwidth = 2250;
27        double minTime = Double.MAX_VALUE;
28        double maxVolume = 0;
29        double isOpt = 0;
30        double[] array = new double[4];
31        array[0] = bandwidth;
32        array[1] = minTime;
33        array[2] = maxVolume;
34        array[3] = isOpt;
35        for(int i=1; i<=10; i++){
36            System.out.println("Iteration: " + i + "'s solution");
37            array[1] = Double.MAX_VALUE;
38            array[2] = 0;
39            array[3] = 0;
40            Run it = new Run(serverList,array);
41            it.run();
42            array[1]=;
43            array[2]=;
44            array[3]=;
45            for(Server s: serverList) {
46                System.out.println("server: " + s.serverId + " data volume: " + s.data + " RPO:
47                if(s.data!=0) {
48                    array[1]+= s.bestTime;
49                    array[2]= s.getCapacity();
50                    s.data = s.data - s.getCapacity();
51                    s.remainingTime --;
52                    s.sendTime = s.remainingTime;
53                    s.bestTime = s.remainingTime;
54                }
55                s.data = s.data - s.getCapacity();
56                s.remainingTime --;
57                s.sendTime = s.remainingTime;
58                s.bestTime = s.remainingTime;
59            }
60            System.out.println("Min time: " + array[1]);
61            System.out.println("Max volume: " + array[2]);
62        }
63    }
64 }
```

```
1 package project;
2 public class Server {
3     public String serverId;
4     public String app;
5     public double data;
6     public double capacity;
7     public int mainRPO;
8     public int remainingTime;
9     public int sendTime;
10    public int bestTime;
11    public int r1;
12    public int r2;
13    public Server(String serverId, String app, int data, int mainRPO, int r1, int r2) {
14        this.serverId = serverId;
15        this.app = app;
16        this.data = data;
17        this.capacity = capacity;
18        this.remainingTime = mainRPO;
19        this.sendTime = mainRPO;
20        this.bestTime = mainRPO;
21        this.r1=r1;
22        this.r2=r2;
23    }
24    public Server(String serverId, String app, int data, int mainRPO) {
25        this.serverId = serverId;
26        this.app = app;
27        this.data = data;
28        this.capacity = capacity;
29        this.remainingTime = mainRPO;
30        this.sendTime = mainRPO;
31        this.bestTime = mainRPO;
32    }
33    public double getCapacity() {
34        return data/bestTime;
35    }
36 }
```

```
1 package project;
2 import java.util.ArrayList;
3 public class Run {
4     ArrayList<Server> serverList;
5     double[] array;
6     public Run(ArrayList<Server> serverList, double[] array){
7         this.serverList=serverList;
8         this.array=array;
9     }
10    public void run() {
11        int counter = 0;
12        int repetitions = 1000;
13        while(counter< repetitions){
14            findOptimum();
15            counter ++;
16        }
17    }
18    public void findOptimum(){
19        array[1]=;
20        double totalVolume = 0;
21        double totalTime = 0;
22        for(Server s: serverList){
23            if(s.data!=0) {
24                totalTime+=s.sendTime;
25                totalVolume+=(s.data/s.sendTime);
26            }
27            if(totalVolume<array[0] && totalTime<array[1]){
28                array[1] = totalTime;
29                array[2] = totalVolume;
30                array[3] = 1;
31                for(Server s: serverList){
32                    if(s.data!=0) {
33                        s.bestTime = s.sendTime;
34                        s.remainingTime --;
35                        if(s.sendTime== 0){
36                            s.sendTime = s.remainingTime;
37                        }
38                    }
39                }
40            }
41        }
42    }
43 }
```

Figure 5.1: Code of Simple Heuristic

## Chapter 5: Genetic Algorithm

[illegible]

### Figure 5.2: Code of Genetic Algorithm

## Chapter 5: Exact Approach

### Detailed Execution Steps of Exact Approach Using Excel Solver

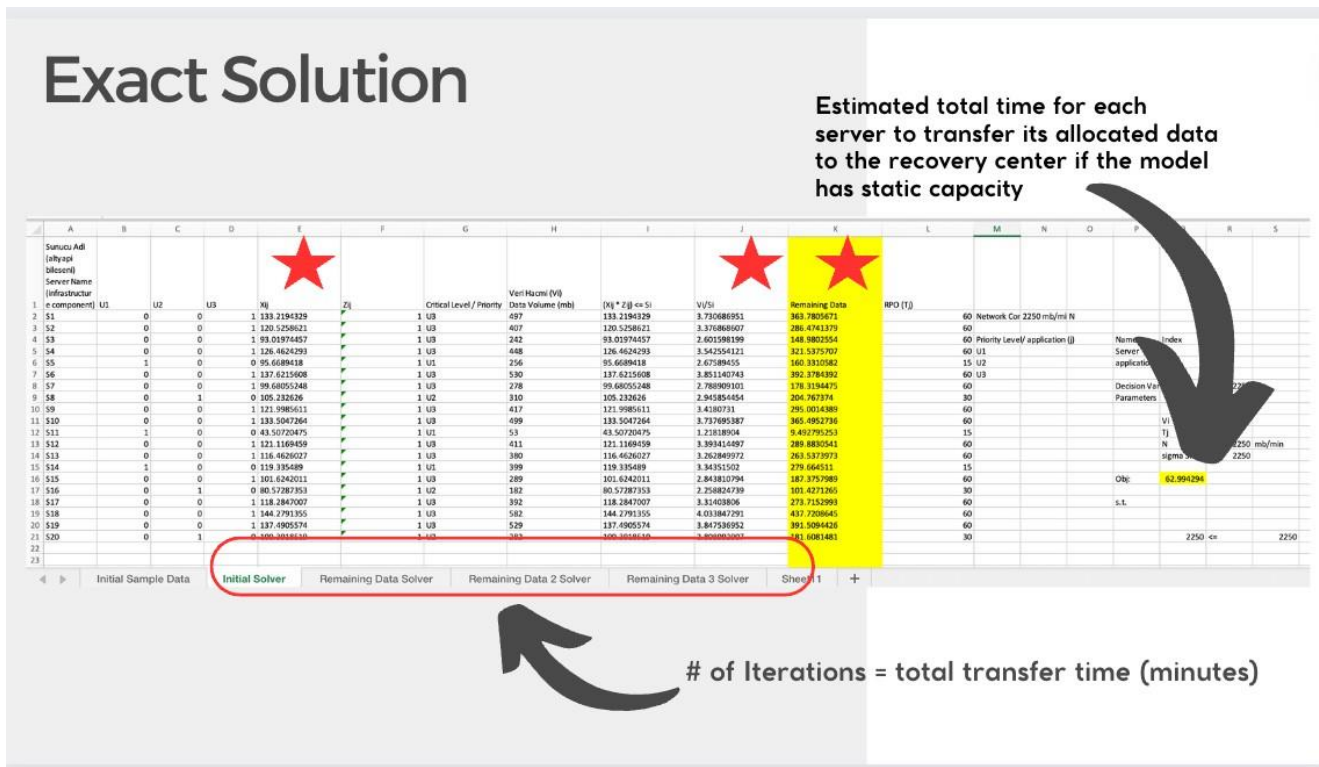
- **Inputs:** List of servers, application levels of each server, application, data volume, RPO constraint and bandwidth constraint
- **Outputs:** The minimum total data transfer time, The total capacity allocated to all servers
- The following flowchart outlines the steps of the Optimal solution algorithm:
  - Step 1: Define Variables
    - Capacity Allocation Variables: Create cells to represent the amount of capacity allocated to each server. For example, these are in cells B2, B3, B4, and so on.
  - Step 2: Define Objective Function

- Calculate Ratios: In adjacent cells, calculate the ratio of data volume to allocated capacity for each server. This could be in cells C2, C3, C4, etc. Use a formula like Data Volume / Capacity Allocated.
- Calculate Sum of Ratios: In a cell (e.g. D2), sum up all these ratios. The formula might look like =SUM(C2:Cn), where n is the last row with ratios.
- Step 3: Set Constraints
  - Capacity Constraints: Ensure that the capacity allocated to each server (cells B2, B3, B4, etc.) is greater than or equal to zero.
  - One Server, One Application: Assure that each server handles only one application. This might involve categorical constraints if there are data indicating server-application relationships.
  - Non-Negative Time: Ensure the ratios calculated in step 2 remain greater than or equal to zero (cells C2:Cn).
  - RPO Time Constraint: Define a constraint for each server's ratio being less than or equal to the total allowed RPO time of the application. For instance, C2 <= RPO\_Time\_App1.
  - Non-Negative Data Volume: Ensure data volume for each server is greater than or equal to zero.
  - Total Network Bandwidth Constraint: Set a constraint for the sum of capacity allocations (B2:Bn) being less than or equal to the total network bandwidth capacity.
- Step 4: Set Solver Parameters
  - Open the Solver tool in Excel (usually found in the Data or Solver tab).
  - Set the Objective: Select the cell where the sum of ratios is calculated (cell D2) as the objective to minimize.
  - Set Variables: Choose the cells representing the capacity allocation (B2:Bn) as the changing cells.
  - Add Constraints: Add all the constraints defined in step 3.
  - Choose the Solver Method: Select "GRG Nonlinear" as the solving method.
  - Set Options if required (iterations, precision, etc.).

- Click Solve.
- Step 5: Solve
  - Click on the "Solve" button within the Solver dialog box. The Solver finds a solution, it'll show the optimized values for the capacity allocations (in cells B2:Bn) that minimize the objective function (cell D2).

## Chapter 5: Exact Approach

### Exact Approach using Excel Solver



**Figure 5.3.1: Excel File with Corresponding Data**

# Exact Solution

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																				
2	51	0	0	1	214.9500196		1	U3	125.3985661	214.9500196	0.583368479	0								
3	52	0	0	1	153.3400512		1	U3	63.69074632	153.3400512	0.415617096	0								
4	53	0	0	1	0		1	U3	0	0	0	0								
5	54	0	0	1	183.611829		1	U3	91.4604181	183.611829	0.498118333	0								
6	55	1	0	0	0		1	U1	0	0	0	0								
7	56	0	0	1	234.0639487		1	U3	148.6540027	234.0639487	0.635126395	0								
8	57	0	0	1	0		1	U3	0	0	0	0								
9	58	0	1	0	18.38153386		1	U2	0.917032492	18.38153386	0.049888519	0								
10	59	0	0	1	161.0240956		1	U3	70.38847715	161.0240956	0.437319132	0								
11	510	0	0	1	216.2432948		1	U3	126.8063843	216.2432948	0.586407011	0								
12	511	1	0	0	0		1	U1	0	0	0	0								
13	512	0	0	1	116.3170788		1	U3	66.36310772	116.3170788	0.434167121	0								
14	513	0	0	1	129.9067453		1	U3	45.74825525	129.9067453	0.352162777	0								
15	514	1	0	0	146.9067041		1	U1	58.33536222	146.9067041	0.397947213	0								
16	515	0	0	1	0		1	U3	0	0	0	0								
17	516	0	1	0	0		1	U2	0	0	0	0								
18	517	0	0	1	140.6447811		1	U3	53.67989223	140.6447811	0.38166999	0								
19	518	0	0	1	261.5739242		1	U3	185.884462	261.5739242	0.71063835	0								
20	519	0	0	1	253.5338967		1	U1	147.9548724	253.5338967	0.639601992	0								
21	520	0	1	0	0		1	U2	0	0	0	0								
22																				
23																				

4 Iterations = 4 Minutes

Figure 5.3.2: Excel File with Corresponding Data

# Exact Solution

Solver Framework

Solver Parameters

Set Objective:

To: ☐ Max ☒ Min ☐ Value Of:

By Changing Variable Cells:

Subject to the Constraints:

\$E\$2:\$E\$21 >= 0

\$H\$2:\$H\$21 >= 0

\$I\$2:\$I\$21 >= 0

\$J\$2:\$J\$21 <= \$L\$2:\$L\$21

\$Q\$21 <= 2250

Add  
Change  
Delete  
Reset All  
Load/Save

☒ Make Unconstrained Variables Non-Negative

Select a Solving Method: ☒ GRG Nonlinear

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Close Solve

Equations in the Model

(5.1)

(5.5)

(5.5)

(5.3) & (5.5)

(5.2)

(5.4)

Figure 5.3.3: Excel Solver Framework

## Chapter 5: Simulation Approach

### Detailed Execution Steps of Simulation Approach Using Arena Simulation Software

- **Inputs:** number of servers, number of entities, data volume of entities, RPO constraint, and server capacities
- **Outputs:** The total data transfer time, minimum, maximum, and average transfer times, the count of entities transferred within and outside their respective RPOs.

#### Explanation of the Approach:

In addressing Fibabanka's data transfer challenges, Arena Simulation Model has been implemented. This simulation model begins by modeling three application levels, which are named U1, U2, and U3. All entities are accessible from the outset and are facilitated through the implementation of create modules. The model comprises diverse applications, each classified by type and systematically administered within the system. Allocation of these applications is distributed across a specified number of servers, with each server exclusively designated to manage a specific single application. This allocation strategy aligns with the mathematical model we have implemented. To describe specific characteristics for these entities, the assign module is utilized. Priorities are set, with U1 assigned the highest priority, followed by U2 and U3 in succession. Key attributes encompass time, denoted by TNOW, signifying the current simulation time, representing the immediate moment within the simulated environment, and associated Recovery Point Objective (RPO) times. Specifically, U1 is allocated an RPO time of 15 minutes, U2 has 30 minutes, and U3 is allotted 60 minutes, ensuring prompt transfer for each entity. Moreover, attributes include data volume, representing predefined volumes for entities as follows: U1 ranges from 100 to 400, U2 from 100 to 550, and U3 from 200 to 650. Additionally, server capacity is assigned, defining the allocated server volume for each entity. Specifically, servers allocated to U1 transfer data within the range of 50 to 500, U2 servers transfer within 100 to 400, and U3 servers process data within 200 to 600 per transfer. As these attributes are assigned, the system randomly selects data volumes for each entity and allocates server capacities to the servers at random between the defined uniform ranges.

Additionally, three distinct process modules have been defined, each designed to handle a specific type of entity. Across all three processes, the sequence of actions involves seize delay release.



Notably, for the U1 process, entities are assigned a high priority (1), signifying that U1 entities take precedence over others. In contrast, the U2 process designates a medium priority (2), positioning U2 entities for processing between high and low priority entities. Lastly, the U3 process is characterized by a low priority (3), indicating that U3 entities are processed last. Within the process modules, dedicated servers are assigned as resources to each entity. In process module 1, server 1 exclusively handles U1 entities. Process module 2 designates server 2 for U2 entities, and in process module 3, server 3 manages U3-related entities. The delay expression for process module 1 is  $\text{DataVolume} / \text{ServerCapacity1}$ , where the delay is determined by the ratio of DataVolume to ServerCapacity1. Similarly, for process module 2, the delay expression is  $\text{DataVolume} / \text{ServerCapacity2}$ , and for process module 3, it is  $\text{DataVolume} / \text{ServerCapacity3}$ . This expression guarantees that entities will be processed within their designated RPOs, avoiding unnecessary queue delays.

Subsequent to each process module, an additional assign module is deployed. This module is tasked with deducting the data volume transferred as entities traverse their designated process modules. The attribute assigned in each module is “datavolume”, featuring distinct expressions for U1, U2, and U3 entities. For U1, it is  $\text{DataVolume} - \text{ServerCapacity1}$ ; for U2, it is  $\text{DataVolume} - \text{ServerCapacity2}$ ; and for U3, it is  $\text{DataVolume} - \text{ServerCapacity3}$ . Moreover, the assign module incorporates the priority attribute to prioritize entities that undergo processing until their data volume is fully transferred (i.e.,  $\text{dataVolume}=0$ ) over new arrivals. This ensures that entities are not withheld unnecessarily and transferred within their designated RPO's.

Following the assign modules, three distinct "decide" modules were integrated, each configured with a 2-way condition. These modules are pivotal in using the "datavolume" attribute for each critical level, determining whether an entity's data volume has been entirely transferred. The decide module, with the expression " $\text{datavolume} \leq 0$ ," ensures that for all three entities, if their data volume is zero, the transfer is considered complete. Otherwise, the entities are rerouted back to the process module for further processing. Moreover, three more distinct "decide" modules were incorporated, each configured with a 2-way condition. These modules play a crucial role in utilizing the "RPO" attribute for each critical level, determining whether entities have been transferred within their designated Recovery Point Objective (RPO) times. Specifically, for U1 entities, the transfer should be completed in less than or equal to 15 minutes, for U2 in less than or equal to 30

minutes, and for U3 within 60 minutes. In cases where this criterion is not fulfilled, the "decide" module intervenes, redirecting the specific entity to the dispose module. Prior to disposal, they are documented using record modules as "Entities not processed within RPO." A record module is utilized subsequent to each decide module to distinguish between entities transferred within their respective RPO (Recovery Point Objective) and those which are not, for each entity type.

Additionally, prior to the completion of all application transfers, we employ record modules for each critical level. These record modules serve to capture and document the number of applications transferred for each entity. Furthermore, they record the time duration that each entity spends within the system. Finally, to streamline the process, we integrate a dispose module to manage the disposal of the transferred applications. Finally for the setup, we run the simulation for 24 hours.

## **Chapter 5: Simulation Approach**

### **Detailed Execution Steps of Simulation Approach Using Python Programming Language**

- **Inputs:** excel files consisting of server capacity, data volume and entities
- **Outputs:** Minute-by-minute entity state, Total Entities Processed, Entities ,Not Processed Within RPO Time, Entities Processed Within RPO Time, Total Entities Processed Within RPO Time, Total Entities Not Processed Within RPO Time, Total Data Volume Transferred for Each Entity Type, Total Server Capacity Transferred for Each Entity Type, Total Data Volume Transferred, Total Server Capacity Transferred ,Average Utilized Capacity, Transferring Time Statistics for Each Entity Type, Total Data Volume Not Transferred, Total Transfer Time for All Entities

### **Explanation of the Simulation Code:**

The script begins by importing two essential libraries: "SimPy" and "pandas". These libraries provide functionalities for simulation and data handling, respectively. "SimPy" enables the creation of discrete-event simulations, while pandas facilitates reading and manipulating data, especially in tabular formats like Excel spreadsheets. The core of the simulation procedure is provided by a class in the script called "DataTransferSimulation". In Python, classes serve as the building blocks for objects that contain both functionality and data. This class has a number of properties and functions to easily manage the simulation.

The "data\_transferred" dictionary is created to track the amount of data transferred for each entity type during the simulation. Similarly, the "total\_processed", "not\_processed\_within\_rpo", and "processed\_within\_rpo" dictionaries are initialized to monitor the processing status of entities. These metrics provide insights into whether entities are processed within their Recovery Point Objective (RPO), a critical factor in data recovery scenarios where meeting specific time constraints is essential. The "transferring\_times" dictionary is set up to record the transferring times for each entity type. Additionally, parameters such as RPO, priority levels, data volume transferred, and server capacity transferred are initialized to predefined values. These parameters play crucial roles in shaping the behavior of the simulation and influencing the decision-making processes within the data transfer system. Finally, the create\_entities method is called within the \_\_init method to initiate the process of generating entities and preparing them for simulation.

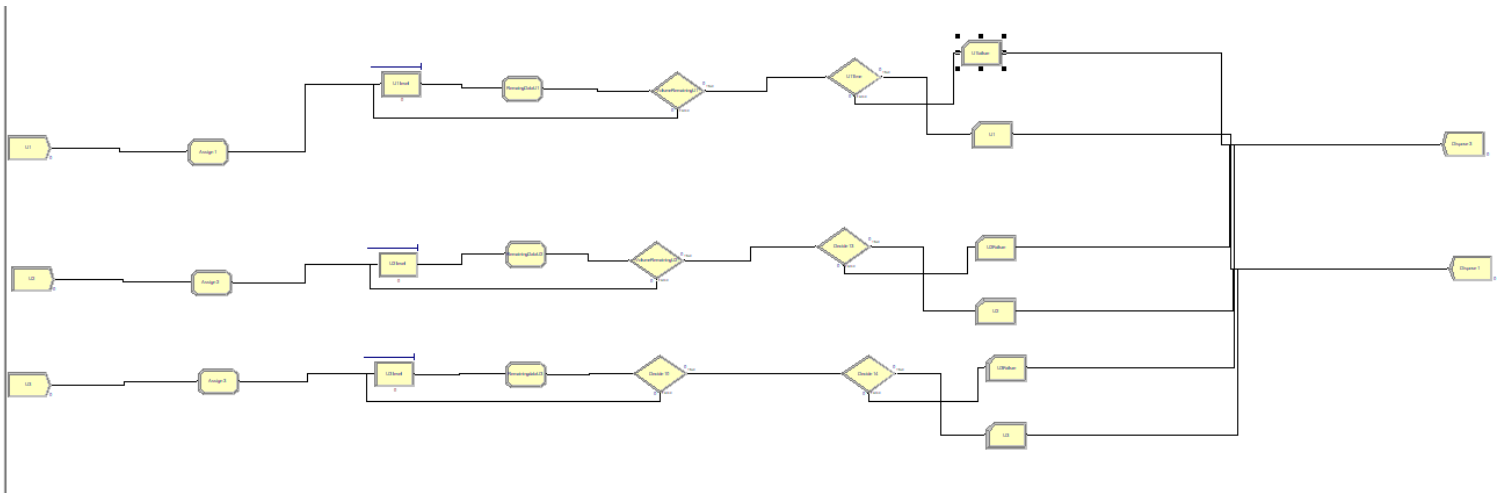
The create\_entities method serves as the initial step in the simulation process. Its primary responsibility is to read entity data from an external Excel file, which typically contains information such as entity type, data volume, and server capacity. Once the data is retrieved, the method validates it to ensure its integrity and consistency. This validation step is crucial for preventing errors and maintaining the accuracy of the simulation. Subsequently, the method generates unique entity IDs for each entity based on their type and count. These IDs play a vital role in tracking and managing entities throughout the simulation. After the IDs are assigned, the entities are scheduled within the simulation environment, mirroring the real-world process of entity creation and scheduling.

Upon entity creation, the process\_entity method simulates the processing of each entity within the data transfer system. This method encapsulates the core data transfer process within the simulation environment. Firstly, the entity requests access to a server, reflecting the initiation of a transfer request in the system. Subsequently, the method calculates the transfer time based on the data volume of the entity and the capacity of the server. This calculation simulates the time required to transfer the entity's data, considering the available resources. Once the transfer time is determined, the simulation environment progresses by the calculated duration, mimicking the passage of time during the data transfer process. Throughout this process, various metrics such as total processed entities and transferring times are updated to track the progress and performance of the data transfer system.

The "run\_simulation" method serves as the moderator of the entire simulation process. Its primary function is to execute the simulation until a specified time duration ("sim\_time") is reached. This method encapsulates the core logic of the simulation, coordinating the creation, processing, and analysis of entities within the simulation environment. During the execution of the simulation, the method progresses the simulation environment by advancing time until the specified sim\_time is reached. As time progresses, entities are created, processed, and their actions are recorded. Once the simulation completes, the method generates a comprehensive report containing various recorded data and statistics. Key metrics included in the report encompass details such as the total number of entities processed, entities processed within Recovery Point Objective (RPO), data volume transferred, server capacity transferred, average utilized capacity, transferring time statistics, data volume not transferred, and total transfer time.

In summary, the script demonstrates the construction of a discrete-event simulation model using SimPy to analyze data transfer processes. By incorporating real-world data and simulating data transfer scenarios, the script provides actionable insights for optimizing and improving the efficiency of data transfer systems.

## Chapter 5: Simulation Approach



**Figure 5.4.1: Arena Simulation Model**

untitled0.py\*

```

1 import simpy
2 import pandas as pd
3 class DataTransferSimulation:
4     def __init__(self):
5         self.servers = {}
6         self.data_transferred = {'id': 0, 'v': 0, 't': 0}
7         self.total_processed = {'id': 0, 'v': 0, 't': 0}
8         self.not_processed_within_rpo = {'id': 0, 'v': 0, 't': 0}
9         self.processed_within_rpo = {'id': 0, 'v': 0, 't': 0}
10        self.transferring_times = {'id': [], 'v': [], 't': []}
11        self.rpo = {'id': 0, 'v': 0, 't': 0}
12        self.priority = {'id': 1, 'v': 2, 't': 3}
13        self.data_volume_transferred = {'id': 0, 'v': 0, 't': 0}
14        self.server_capacity_transferred = {'id': [], 'v': [], 't': []}
15        self.create_entities(self)
16    def create_entities(self):
17        file_path = r'D:\ayyegin\university 4th Year\18th Semester\18402\Arena Code\data\fixed\Scenario1.4.xlsx'
18        try:
19            entities_data = pd.read_excel(file_path)
20        except FileNotFoundError:
21            print("Error: File not found.")
22            return
23        if "Data Volume" not in entities_data.columns or "Server Capacity" not in entities_data.columns:
24            print("Error: 'Data Volume' or 'Server Capacity' column not found in the excel file.")
25            return
26        for index, row in entities_data.iterrows():
27            entity_type = row["Entity Type"]
28            data_volume = row["Data Volume"]
29            server_capacity = row["Server Capacity"]
30            if pd.isnull(entity_type) or pd.isnull(data_volume) or pd.isnull(server_capacity):
31                print("Warning: Skipping row with empty values.")
32            else:
33                entity_id = f"{entity_type}_{self.data_transferred[entity_type]}"
34                self.data_transferred[entity_type] += 1
35                data_volume = int(data_volume)
36                server_capacity = int(server_capacity)
37                if entity_type not in self.server_capacity_transferred:
38                    self.server_capacity_transferred[entity_type] = {}
39                if entity_id not in self.server_capacity_transferred[entity_type]:
40                    self.server_capacity_transferred[entity_type][entity_id] = 0
41                self.sim.process(self.process_entity(entity_id, entity_type, data_volume, server_capacity))
42    def process_entity(self, entity_id, entity_type, data_volume, server_capacity):
43        self.sim.request(self, entity_id, capacity=1)
44        start_time = self.sim.now
45        transfer_time = round(data_volume / server_capacity, 4)
46        yield self.sim.timeout(transfer_time)
47        self.total_processed[entity_type] += 1
48        delay_time = self.sim.now - start_time
49        if delay_time > self.rpo[entity_type]:
50            self.not_processed_within_rpo[entity_type] += 1
51        else:
52            self.processed_within_rpo[entity_type] += 1
53            self.data_volume_transferred[entity_type] += data_volume
54            self.server_capacity_transferred[entity_type][entity_id] += server_capacity

```

untitled0.py\*

```

55        self.not_processed_within_rpo[entity_type] += 1
56        self.processed_within_rpo[entity_type] += 1
57        self.data_volume_transferred[entity_type] += data_volume
58        self.server_capacity_transferred[entity_type][entity_id] += server_capacity
59        self.transferring_times[entity_type].append(transfer_time)
60        server.release(self, entity_id)
61        del self.servers[entity_id]
62    def run_simulation(self, sim_time):
63        self.sim.run(until=sim_time)
64        print("Recorded Data:")
65        print("Total Entities Processed:")
66        for entity_type in self.total_processed:
67            print(f"Entity Type: {entity_type}, Total Processed: {self.total_processed[entity_type]}")
68        print("Entities Not Processed Within RPO Time:")
69        for entity_type in self.not_processed_within_rpo:
70            print(f"Entity Type: {entity_type}, Total Not Processed Within RPO Time: {self.not_processed_within_rpo[entity_type]}")
71        print("Entities Processed Within RPO Time:")
72        for entity_type in self.processed_within_rpo:
73            print(f"Entity Type: {entity_type}, Total Processed Within RPO Time: {self.processed_within_rpo[entity_type]}")
74        total_processed_within_rpo = sum(self.processed_within_rpo.values())
75        total_not_processed_within_rpo = sum(self.not_processed_within_rpo.values())
76        print(f"Total Entities Not Processed Within RPO Time: {total_not_processed_within_rpo}")
77        print(f"Total Data Volume Transferred for Each Entity Type:")
78        for entity_type in self.data_volume_transferred:
79            print(f"Entity Type: {entity_type}, Total Data Volume Transferred: {self.data_volume_transferred[entity_type]}")
80        print(f"Total Server Capacity Transferred for Each Entity Type:")
81        for entity_type in self.server_capacity_transferred:
82            print(f"Entity Type: {entity_type}, Total Server Capacity Transferred: {self.server_capacity_transferred[entity_type]}")
83        total_data_volume_transferred = sum(self.data_volume_transferred.values())
84        total_server_capacity_transferred = sum(self.server_capacity_transferred.values())
85        print(f"Total Data Volume Transferred: {total_data_volume_transferred}")
86        print(f"Total Server Capacity Transferred: {total_server_capacity_transferred}")
87        total_server_capacity_transferred = sum(self.server_capacity_transferred.values())
88        average_utilized_capacity_percentage = (total_data_volume_transferred / total_server_capacity_transferred) * 100
89        print(f"Average Utilized Capacity: {average_utilized_capacity_percentage:.2f}%")
90        print("Transferring Time Statistics for Each Entity Type:")
91        for entity_type in self.transferring_times:
92            if self.transferring_times[entity_type]:
93                min_transfer_time = min(self.transferring_times[entity_type])
94                max_transfer_time = max(self.transferring_times[entity_type])
95                avg_transfer_time = round(sum(self.transferring_times[entity_type]) / len(self.transferring_times[entity_type]), 4)
96                total_transfer_time = round(sum(self.transferring_times[entity_type]), 4)
97                print(f"Entity Type: {entity_type}, Min={min_transfer_time}, Max={max_transfer_time}, Avg={avg_transfer_time}, Total={total_transfer_time}")
98        print("Total Data Volume Not Transferred for Each Entity Type:")
99        for entity_type in self.data_transferred:
100            data_volume_not_transferred = max(0, self.data_transferred[entity_type] - self.data_volume_transferred[entity_type])
101            print(f"Entity Type: {entity_type}, Data Volume Not Transferred: {data_volume_not_transferred}")
102            total_data_volume_not_transferred = max(0, sum(self.data_transferred.values()) - total_data_volume_transferred)
103            print(f"Total Data Volume Not Transferred: {total_data_volume_not_transferred}")
104            total_transfer_time_all_entities = round(sum(self.transferring_times.values()), 4)
105            print(f"Total Transfer Time for All Entities: {total_transfer_time_all_entities}")
106            simulation = DataTransferSimulation()
107            simulation.run_simulation(sim_time)

```

Figure 5.4.2: Code of Genetic Algorithm

## Chapter 6: Comparisons

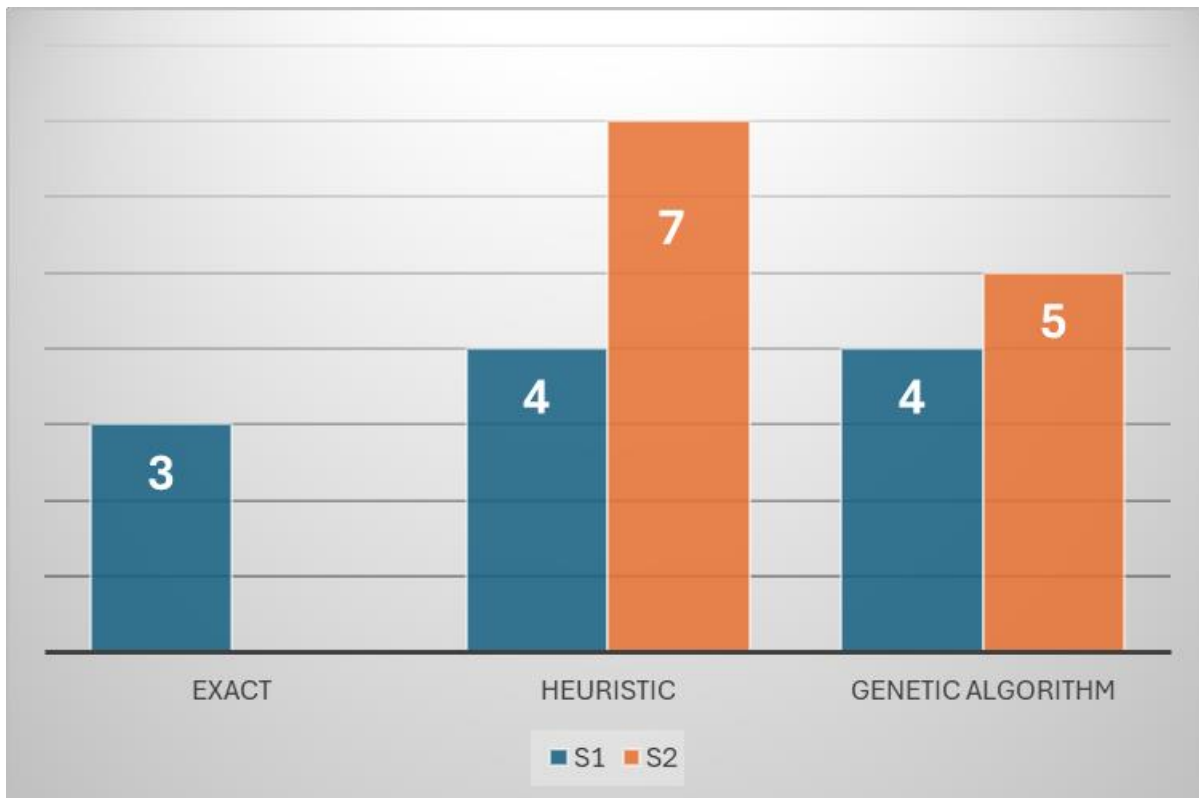
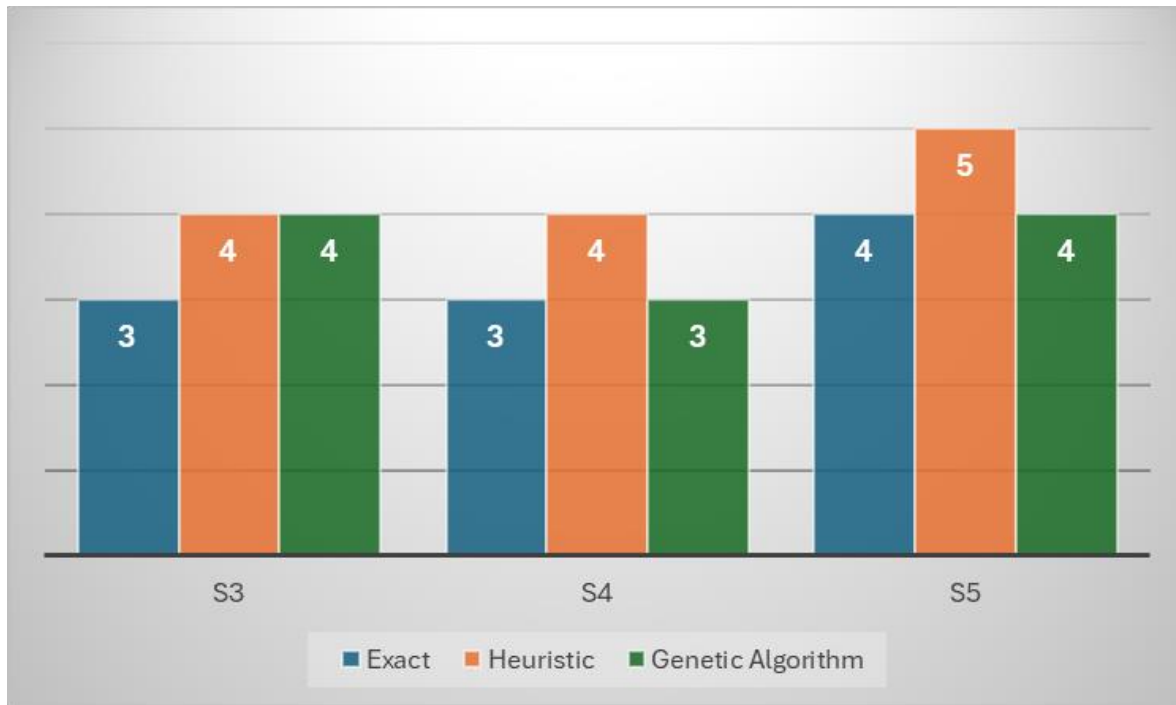
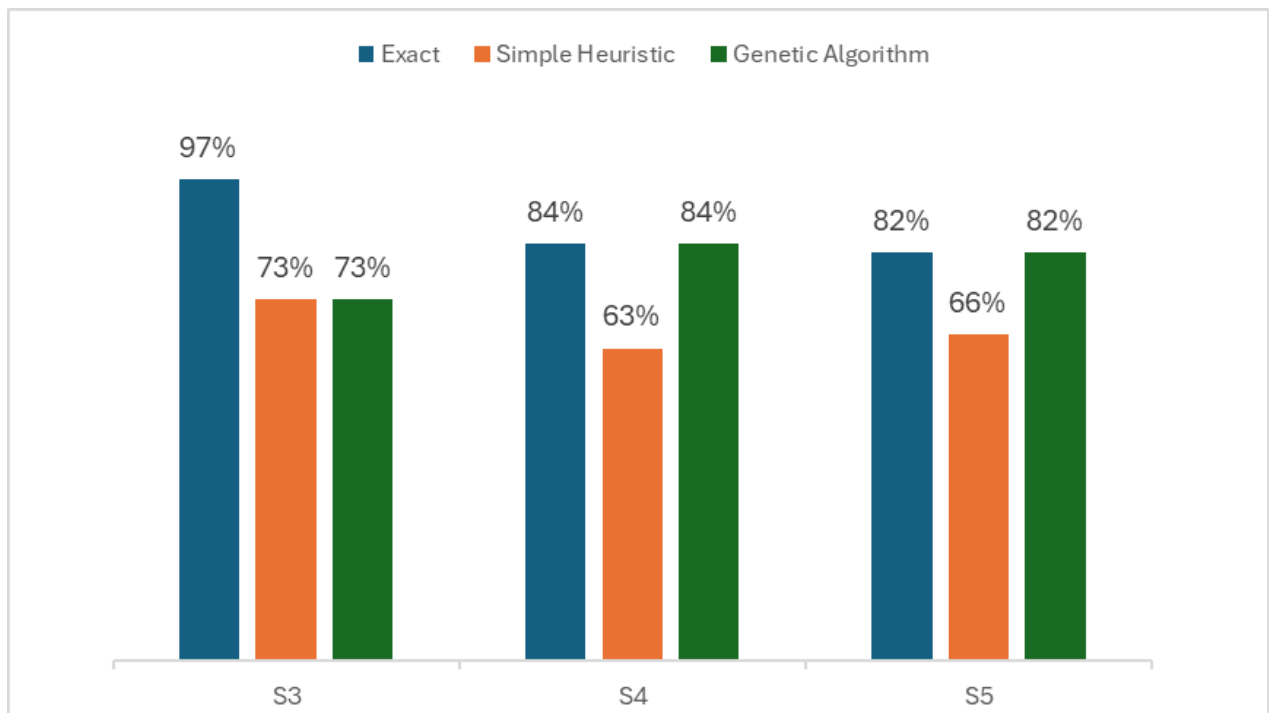


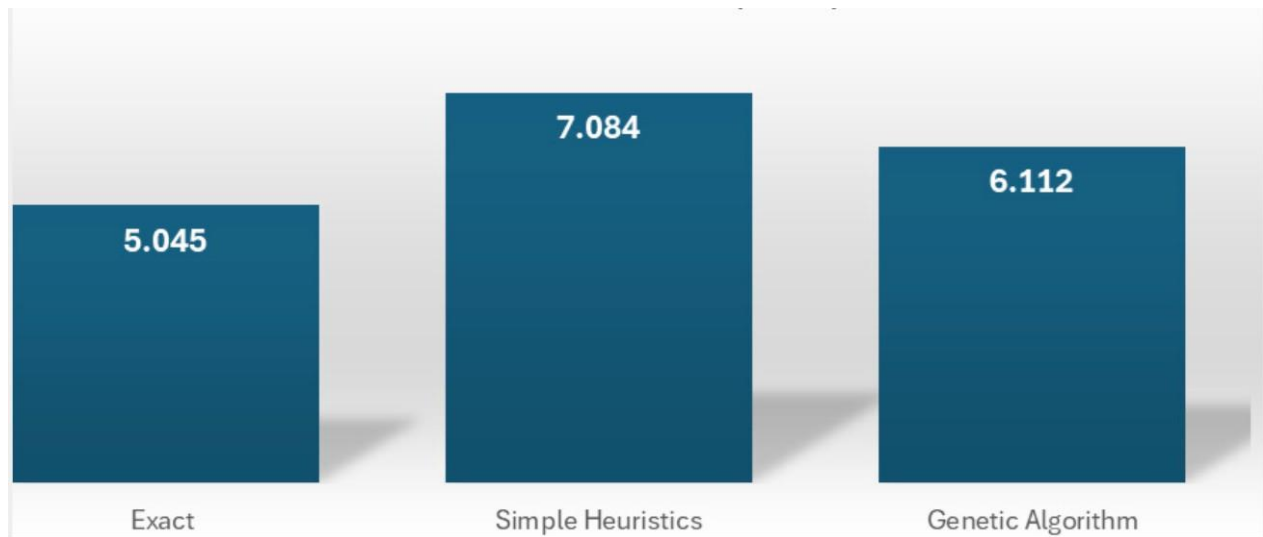
Figure 6.1 Total Sending Time (based on minutes)



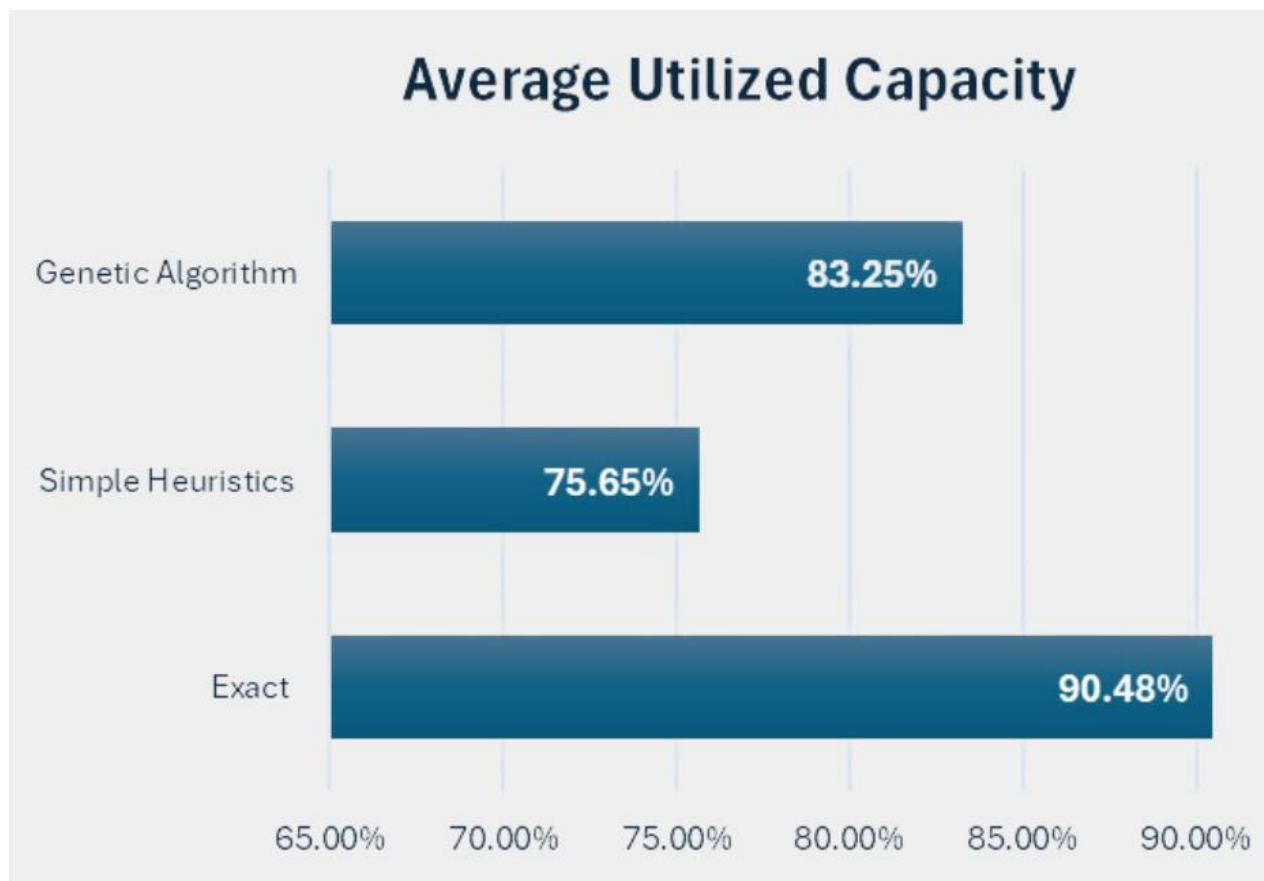
**Figure 6.2 Total Sending Time (based on minutes)**



**Figure 6.3 Average Utilization of Capacity**



**Figure 6.4 Total Transfer Time (based on minute)**



**Figure 6.5 Average Utilized Capacity (in percentages)**