**Universidade do Minho**
Escola de Engenharia

Ricardo David Pereira Alves

# Vehicle Routing and Tour Planning Problem: A Cement Industry Case Study

Setembro de 2018

**Universidade do Minho**
Escola de Engenharia

Ricardo David Pereira Alves

# Vehicle Routing and Tour Planning Problem: A Cement Industry Case Study

Dissertação de Mestrado
Mestrado em Engenharia de Sistemas

Trabalho realizado sob orientação do

**Professor Doutor José António Vasconcelos Oliveira**
**Professor Doutor Luís Miguel da Silva Dias**

Setembro de 2018

**DECLARAÇÃO**

Nome: Ricardo David Pereira Alves

Endereço eletrónico: ralves_12_@hotmail.com

Título da dissertação: Vehicle Routing and Tour Planning Problem: A Cement Industry Case Study

Orientadores: Professor Doutor José António Vasconcelos Oliveira e Professor Doutor Luís Miguel Silva Dias

Ano de conclusão: 2018

Mestrado em Engenharia de Sistemas

Universidade do Minho, ___/___/______

Assinatura:

To my Parents.

## ACKNOWLEDGMENTS

Life is like driving a car. This document represents one more road traversed by me. It is therefore necessary to thank all those who helped me in this stage.

First, I want to offer my gratitude to my advisors, J. António Oliveira, Ph.D. and to Luís Dias, Ph.D. for allowing me to grow both academically and personally.

To the University of Minho, for giving me 5 years of great learning.

To my parents, for my education, for the constant support, for teaching me the most important things in the world, and for being an Example to follow everyday.

To Ângela Coutinho, for all the support, patience, friendship, and for encouraging me to be a better person everyday.

To João Fonseca and Ana Regina, for the constant support and for traversing this road together, with me, as colleagues, but, above all, as friends.

Not in order of priority, to my friends José Luís Silva, Afonso Rodrigues, Rui Costa, Miguel Nogueira, Miguel Sanches, Joaquim Santos, José Bruno, Luís Miguel Costa, Ricardo Dias, and João Pedro for the fun moments, and for the learning I had with each one of them.

Finally, I want to show my gratitude to every person who directly, or indirectly, has contributed for the accomplishment of this document.

# ABSTRACT

The transportation, being part of the logistics field, plays a crucial role in the business world. Its impact in the costs and service quality is an increasingly imperative topic. In industry, transportation systems are equally important and can represent a large improvement in the management of the plants and in the service quality of the products, thus bringing advantages for the companies and for the clients.

The cement industry is not an exception. Cement is the second most consumed substance in the world and with the great number of trucks arriving at cement facilities, every day, the supply chain management of this industry must encompass this management as well. With the lack of assistance and guidance clients have inside the cement facilities, both companies incur in additional costs and clients experience reduced levels of service quality. To overcome these issues, three algorithms were developed and implemented. Each algorithm has different specifications and different goals. However, all the developed algorithms improve the service quality, guiding the truck drivers – the clients – inside the plants and giving the routes in shorter periods of time. One algorithm guides the trucks through the minimum distance route and will serve as a comparison term for the other two. The other two algorithms, named equilibrium approaches, are the main contribution of this dissertation. These dynamic algorithms consider not only the traveled distance, but also the workload both in the servers and in the roads. The entrance management in the facilities is also a crucial aspect cement companies must be aware of. Several thought policies are presented and an algorithm for the entrance management is developed and implemented. With a simulation software, the developed algorithms were tested and simulated. The simulation results are reported and discussed.

**Keywords:** Industry 4.0; Supply Chain Management; Vehicle Routing; Tour Planning; Dynamic Routing; Simulation.

## RESUMO

A indústria do transporte desempenha um papel crucial no mundo empresarial. O seu impacto nos custos e na qualidade de serviço são um tópico cada vez mais importante. Na indústria, os sistemas de transporte são igualmente importantes e podem representar uma grande melhoria na gestão das fábricas e na qualidade do serviço dos produtos, trazendo vantagens tanto para as empresas como para os clientes.

A indústria cimenteira não é uma exceção. O cimento é a segunda comodidade mais consumida em todo o mundo, e com o grande número de camiões que chegam às fábricas de cimento todos os dias, a gestão da cadeia de abastecimento desta indústria deve, também, incorporar esta gestão. Com a falta de assistência na orientação que os clientes têm dentro das fábricas, tanto as fábricas incorrem em custos acrescidos como os clientes experienciam uma qualidade de serviço reduzida. Para abordar este problema, três algoritmos foram desenvolvidos e implementados. Cada algoritmo tem objetivos e especificações diferentes. No entanto, todos os algoritmos implementados melhoram a qualidade de serviço guiando os camiões dos clientes dentro das plantas, e calculando as rotas em curtos períodos de tempo. Um dos algoritmos guia os camiões pela rota que permite a menor distância percorrida, e servirá como termo de comparação para os outros dois. Os outros dois algoritmos, chamados abordagens de equilíbrio, são a grande contribuição desta dissertação. Estes algoritmos dinâmicos consideram a ocupação dos servidores e das estradas, além da distância percorrida. A gestão de entrada nas fábricas é também um aspeto importante que as fábricas de cimento devem ter atenção. Diversas políticas de entrada são apresentadas e um algoritmo para a gestão de entrada na fábrica é também desenvolvido e implementado. Com um software de simulação, os algoritmos desenvolvidos foram testados e simulados. Os resultados das simulações são apresentados e discutidos.

**Palavras-Chave:** Indústria 4.0; Gestão da cadeia de abastecimento; Roteamento de veículos; Planeamento de Rotas; Roteamento dinâmico; Simulação.

# CONTENTS

## LIST OF FIGURES

# LIST OF TABLES

# LIST OF ACRONYMS

| | |
|---|---|
| **ECS** | European Committee for Standardization |
| **ICT** | Information and Communication Technologies |
| **UH4SP** | Unified Hub for Smart Plants |
| **GDP** | Gross Domestic Product |
| **INE** | Instituto Nacional de Estatística |
| **VRP** | Vehicle Routing Problem |
| **SPP** | Shortest Path Problem |
| **SSSPP** | Single Source Shortest Path Problem |
| **APSPP** | All pairs Shortest Path Problem |
| **ACO** | Ant Colony Optimization |
| **TSP** | Traveling Salesman Problem |
| **ATSP** | Asymmetric TSP |
| **TSPTW** | TSP with time Windows |
| **TSPPD** | TSP with Pickup and deliveries |
| **TSPPC** | TSP with Precedence Constraints |
| **GA** | Genetic Algorithm |
| **SOP** | Sequential Ordering Problem |
| **GPS** | Global Positioning System |
| **TDTSP** | Time dependent TSP |
| **MILP** | Mixed Integer Linear Programming |
| **TAP** | Traffic Assignment Problem |
| **EU** | European Union |
| **ut** | Units of Time |

# 1. INTRODUCTION

Logistics plays a central role in the micro and macro perspective of a day to day life of a company, organization, or to the economy of a nation. The Comité Européen Normalisation (European Committee for Standardization - CEN) defines logistics as being the concept of plan, execute and control. These tasks are strongly connected, and it is possible to consider logistics as the operational component of the supply chain management (SCM) [1].

Among several definitions of what logistic is, there is one modern definition that applies to most industries [2], and it is presented below.

*"...the efficient transfer of goods from the source of supply through the place of manufacture to the point of consumption in a cost-effective way, whilst providing an acceptable service to the customer..."*

In most industries, one of the crucial stages of logistics is the transportation operation, which is strongly connected to the efficiency of moving products. Usually, the transportation links the several elements in a logistical chain. The use of efficient methods of transportation is one of the key foundations in management techniques for promoting the efficiency and competitiveness of enterprises [3].

The increasingly use and evolution of Information and Communication Technologies (ICT) in industry, and specifically in the support for the logistics operations, have promoted new challenges and introduced a transformation in how organizations are managed [4]. With these aspects, Industry 4.0 is now a familiar term. It is referred to the fourth industrial revolution and is also known as the 'smart manufacturing', or even 'integrated technology'.

The equilibrium between optimizing the supply chain and providing a good service level, is the key aspect when introducing technology into the business world. Thus, the main goal of Industry 4.0 is to connect people, machines, and goods, searching for a more organized environment to, simultaneously, bring advantages for the organizations and for the clients [5]. Industry 4.0 has also a great impact in the transportation sector as well. Using ICT, it is possible to develop a more efficient and profitable transportation system.

The work presented in this dissertation is developed under a scientific project, that aims to develop systems for smart plants, specifically cement plants. The UH4SP – Unified Hub for Smart Plants – aims to develop simulation models and heuristic optimization models to take cement plants to another level [6]. More specifically, one of the main goals of the project is the development of architectures of software and methodologies orientated to services, promoting the corporative and aggregate vision of the operations in each one of the cement plants dispersed by several geographic regions [7]. The UH4SP addresses several segments of the supply chain of a cement plant. The problem addressed in this dissertation is the one dealing with the management of the trucks entering the plant.

A typical cement plant receives hundreds of trucks every day. Each one of them has one or more locations to visit, in order to load or unload materials, depending on each truck. This process is, in this sense, unpredictable, due to the fact that it is not possible to know the locations each truck must visit before arriving at the plant. Besides this, the truck driver usually does not know the plants' map, due to their big dimensions. Even if the driver already knows the facility, the choice of the route will be made only by what he knows of it. Either way, the driver will much probably follow a disadvantageous route, forcing him to stay more time inside the plant, causing delays to him and to other truck drivers that already are inside, or who will still enter the plant. Additionally, the driver may load or unload the materials in wrong locations, causing delays, additional costs to the company, etc. One other big problem caused by the trucks is the congestion in the roads of the plant. Each truck driver chooses its own route, and this 'irreflective' choice will overload some roads in the plant.

## OBJECTIVES

The main goal of this dissertation is to create an algorithm that tackles the routing problem of the trucks. The algorithm must compute a route for each truck, whenever they are entering the plant. This route will guide the drivers inside the plant, to the locations they must visit, reducing its unnecessary times, thus increasing the service quality for the clients.

One other big goal of this dissertation is to test and validate the algorithm and, consequently, the developed program, using a simulation software. This validation will confirm if the algorithm is working as it is required, or to make some adjustments in possible parameters, approximating the solution to what it is expected.

## DISSERTATION OUTLINE

This dissertation is composed by seven chapters. The Chapter 2 and 3 are devoted to the most studied and known routing problems in literature, being them static or dynamic. In these chapters, some examples of algorithms for solving the routing problems, variations of the problems and application examples are also studied and presented. The Chapter 4 presents the cement industry supply chain. It starts by giving a brief overview of the cement industry, presenting the cement life cycle and explaining how this commodity is created, stored and distributed. After that, it suggests how Industry 4.0 and technologies can affect directly the management of the cement supply chain. This chapter ends with the description and the modeling of the trucks routing management problem. With this, the real problem and its impacts in the day to day of a cement facility are outlined. In the Chapter 5, the developed methodologies for solving the routing problem are explained, giving examples of how the trucks will be guided inside the facility. In the Chapter 6, some tests and simulations are presented, testing and comparing the developed methodologies. The Chapter 7 presents an additional problem, the parking management, that can have an impact in the day life of a cement facility. Some entrance policies are presented, and an algorithm and its simulation are developed to tackle this problem. The conclusion of the work and the future research are presented in the Chapter 8.

# 2. Routing Problems

## 2.1. Introduction

In the days we live in, transportation has a big economic impact in almost all companies, organizations, families, and people of most developed countries. Efficient transportation reduces costs in many economic areas. Besides that, the impact that inefficient transportation could bring to the environment is, by itself, a great impact everyone should be aware of. These impacts have motivated companies and academic researchers to vigorously pursue the use of operations research and management science to improve the efficiency of transportation [8].

There are several types of transportation, such as air, rail, road, sea, etc. In this study, the focus will be targeted in the direction of road transportation. This type of transportation has a great impact in the economy of a nation. For example, in Portugal, in 2011, the industry of transportation reached 3.2% of the gross domestic product (GDP) (Instituto Nacional de Estatística – INE).

Road transportation process involves all stages of the production and distribution systems and represents a relevant component (generally from 10% to 20%) of the final cost of the goods [9]. Saving time and/or money is the aim of all organizations. The impact of a successful implementation of a routing software can change a lot in the daily basis of a company.

Several successful implementations of computerized routing software's have been documented in literature. These successes can be attributed in part to algorithmic advances in the field of vehicle routing and also to the development of new software and computer technologies. Vehicle routing is truly one of the great success stories of operations research [10]. There are many examples of routing problems and each one has one purpose, and, because of that, there are inherent constraints and changes that make almost each problem unique. Vehicle Routing Problem (VRP) is described by Laporte [11] as "Unlike what happens for several well-known combinatorial optimization problems, there does not exist a single universally accepted definition of the VRP because of the diversity of constraints encountered in practice." Laporte says as well that researchers may have a difficulty finding their way through the abundant and somewhat disorganized literature in these types of problems. When choosing the best route, it may have to do with distance, with time, with what it is better for the system in that period, etc.

Therefore, in the next sections, some of the most structured routing problems in literature will be addressed.

## 2.2. Tour Planning

The increasing development in technologies lead to a progress in the study and implementation of intelligent transportation systems. Thus, Tour Planning Problems are a vital research area. In [12], it is possible to state the increasingly number of publications in the thematic of Routing Problems since 1954. This increasing interest has focused attention in new and more difficult routing problems.

The tour planning can be generally viewed as a process of assigning resources to requests, for example, vehicles that execute transportation processes, following to some conditions, as capacity, time windows, etc. For each vehicle, the sequence of the requests will be specifically ordered to obtain the minimum cost for that vehicle and for the fleet in general. In the Figure 1 is possible to observe a generic example of a tour planning for a fleet of two vehicles [13]. The objective of the tour plan is connected to a purpose, being that, minimizing the total traveled distance, per example, and the goal is to find the optimal solution, the one that minimizes/maximizes the objective function [11].

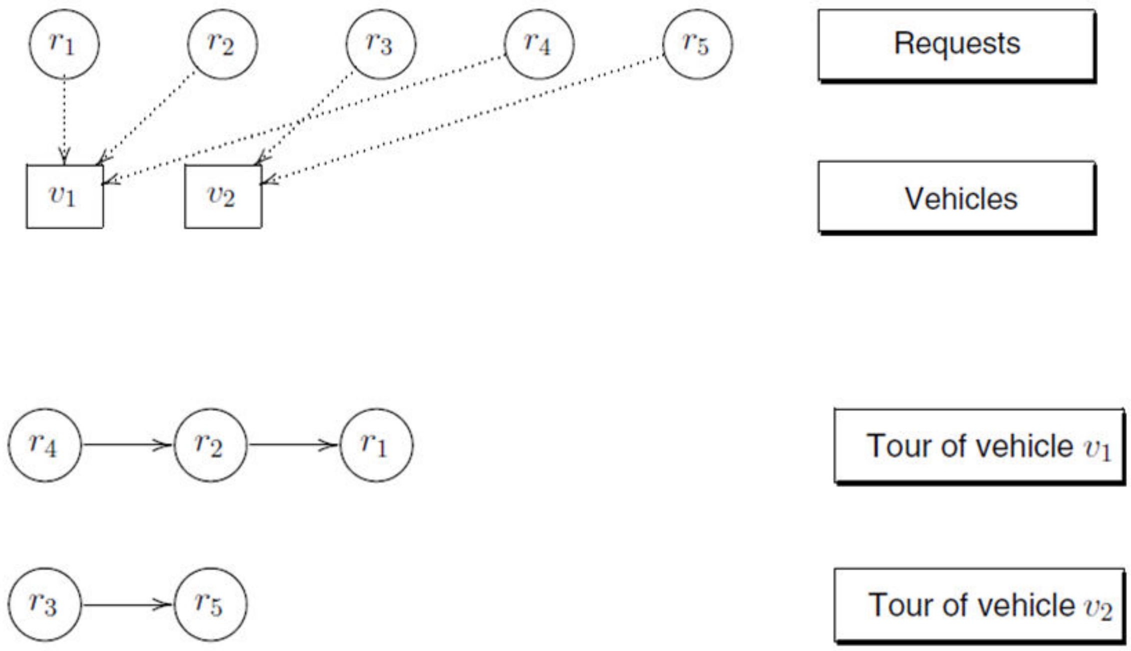


**Figure 1 - Generic Example of a Tour Planning.**

## 2.3. Shortest Path Problem

Shortest Path problems lie at the heart of network flows [14]. The first case of the shortest path is difficult to trace. It is possible to imagine that it was used in very primitive societies, in the search for food, for example. The mathematical research of the problem started later, when compared with other similar problems (like minimum spanning tree, assignment problem, etc.), which could happen due to the relatively easiness of the problem. Yet, when the problem came to the focus of interest, several researchers independently developed methods for solving it [15].

The shortest path problems play a central role in network analysis. Network analysis is one of the most important functions, and because of that, the shortest path problem played an important role in lots of fields, such as electric navigation, traffic tourism, urban planning and electricity, communications, pipe designs, and others. It is important to state that the shortest path is not only the analysis of the shortest distance. This problem extends to other measurements, such as time, cost, or even the capacity of the path. With all this, the 'shortest path analysis' can be turned to the fastest path, the lowest cost, and so on [16]. Besides this, shortest path problems can be applied in other topics. For example, most algorithmic approaches for finding traffic patterns solve a lot of shortest path problems as subproblems [14].

The Shortest Path Problem (SPP) usually involves a network represented by a directed graph G= (N, A), where N is the set of the *n* nodes and A is the set of *m* arcs that connect the nodes. Each one of the arcs $(i,j) \in A$ has an arc cost, which, per example, can be the distance of travelling from *i* to *j*. This cost (weight) can be any measurement [17][18].

Researchers have studied several different types of shortest path problems [14]:

1- Finding shortest paths from one to all other nodes when arc lengths are nonnegative, or Single Source Shortest Path Problems (SSSPP).

2- Finding shortest paths from one node to all other nodes for networks with arbitrary arc lengths.

3- Finding shortest paths from every node to every other node, or All Pairs Shortest Path Problems (APSPP).

4- Various generalizations of the shortest path problem.

In the case of the SSSPP (or simply SPP), the graph contains a distinguished node, named source node. Thus, the problem is to find the shortest path from that node, to all the other nodes [18]. The length of the path is the sum of all the distances (or costs) of each arc that make up the path.

In the case of the APSPPs, it is determined the shortest paths between each pair of nodes presented in the network [17].

A solution to the SPP can be described by a (shortest path) spanning tree rooted in the source node. A spanning tree is a subgraph of G, which includes all the vertices (*n*) of G, but only the necessary number of arcs (*n*-1) for this to happen. In a spanning tree, each node is preceded by another, so that the position of it in the spanning tree is defined by a predecessor label. The predecessor label of a node marks another node that precedes it. The shortest path can be found by following the predecessor labels down to the source node [18]. Thus, not only the shortest path problem gives the minimum cost, but also the route that makes that minimum cost.

As stated earlier, there are different types of shortest path problems (SSSPP, APSPP, etc.). Depending on the context of the problem, different types of algorithms are implemented. Although being a relatively 'easy' problem, advancement in areas of ICT and the increasing of high quality network data, leading to networks involving large amounts of data, containing hundreds of thousands or even millions of nodes [19]. Thus, the algorithms for solving this type of problems are different, depending on the objective and context of the problem.

In the study [14]- Chapters 4 and 5, is stated that there are, in literature, roughly two different major classes of algorithms for solving the SPP. The *label-setting* and *label-correcting* algorithms. These algorithms assign distance labels to each node at each step. The distance labels are upper bounds (estimates) of the shortest path distances. The classes of algorithms vary on the way they approach to the final shortest distance. The *label setting* algorithms designate at each iteration a distance as permanent, while *label correcting* algorithms do not consider any of the label a permanent label till the final iteration, when all the labels become permanent.

The *Dijkstra's* algorithm is one of the most known label setting algorithms. On the other hand, the *Floyd-Warshall* algorithm is one of the most famous label correcting algorithms. A great difference between the two stated approaches is the fact that the label correcting algorithms are more general because they are able to, among other things, solve SPP when negative arc costs are present. On the other side, the label setting algorithms have much better worst-case complexity bounds.

### 2.3.1. DIJKSTRA'S ALGORITHM

In 1959, Edsger Dijkstra came up with an algorithm of finding the shortest path in a network where at least one path between two nodes exists [20].

The Dijkstra's algorithm is one of the most famous algorithms for the SPP. It is part of the label setting algorithms and finds the shortest path from one node to all the other nodes in a nonnegative arc length network, being so part of the SSSPP algorithms stated previously.

Dijkstra's algorithm starts by creating a distance label d(i) for each node i ∈ N. The algorithm divides the nodes into two groups, the permanently labeled and the temporarily labeled. The permanently labeled nodes are the ones who give the shortest distance from the source node to that node. On the other hand, the temporarily labeled nodes, are the ones who give an upper bound on the shortest path from the source node to that node. Thus, the algorithm starts by initializing the source node to be permanently labelled and to have distance of 0. The other nodes are temporarily labeled with their directly distance to the source or labeled with infinity, if there is no connection between the source node and that nodes. In each iteration, the temporarily labeled node with minimum distance is examined. Examining that node means the algorithm scans the arcs A(i), to update the distance labels of the adjacent nodes. This chosen node is also made permanently because none of the arcs from a temporary node can reduce its distance label further due to the nonnegative arc restriction. The algorithm terminates when all the nodes are made permanent [19][22].

In terms of running time, Dijkstra's algorithm has, in his original implementation, a running time of $O(n^2)$, where n is the number of nodes [21]. The most consuming of this time is due to the selection of what node to process next, i.e., the search of the temporarily labeled node with least distance label [18]. The search of all the nodes, in each iteration, makes a great bottleneck. One way to overcome this difficulty, is to implement a priority queue, also named *heap*. A priority queue is a structure that, in a Dijkstra's algorithm implementation, allows to group the arcs by distances and so to overcome the bottleneck of searching all the arcs at each iteration.

There are several applications of heap structures in implementation of Dijkstra's algorithms. There are also several types of heaps and each one of them can have a different computational effort. It is possible to reduce the computing time from $O(n^2)$ to $O(\ m + n\log^2 C)$, where $m$ is the number of arcs, $n$ is the number of nodes and C is the value of the largest arc cost, assuming to be an integer [23][28]. For different applications and running times of heap implementations on Dijkstra's algorithms, see [19] [22] [28]. Besides all the heap-based implementations of Dijkstra's, in a very dense network, the original implementation of Dijkstra's algorithm, without any heaps, achieves the best available running time.

Dijkstra's algorithm has a great spectrum of applications since its creation, in areas such as Traffic information, calculating the shortest path and the shortest distance from a source to a given node, but also, in other problematics such the Open Shortest Path First, used in internet routing [22].

Another well-known algorithm for the single source shortest path is the *Bellman-Ford* algorithm. In this algorithm, it is possible for the network to have also negative arc costs. Besides this, the computational time of the Bellman-Ford is worse than Dijkstra's algorithm [23]. It is possible to see a very vast study on this problematic in Chapter 4 of [14], Chapter 5 of [24], [17] and [18].

### 2.3.2. Floyd-Warshall Algorithm

The Floyd-Warshall algorithm was introduced in 1962, by Robert Floyd [25] and is an example of dynamic programming [26]. The Floyd-Warshall is a simple and widely used algorithm for the SPP. It is part of the label correcting algorithms and allows to compute the shortest path between all pairs of nodes in a weighted graph, being so part of the APSPP stated above [27].

Before explaining how Floyd-Warshall's algorithm works, it is important to have in mind that all pair shortest path problems can be solved by using the repetitive SPP. This means that, by running a single source SPP algorithm *n* times, one for each node of the network, the problem is solved. If the network does not have any arcs with negative cost, the Dijkstra's algorithm could solve this problem. If, on other hand, there are arcs with negative costs, the Bellman-Ford algorithm could be addressed [17].

The Floyd-Warshall algorithm also allows arc costs to be negative. Besides this, the algorithm will give the shortest path for each pair of nodes if there does not exist a negative cycle. If so, the computational effort of the algorithm will pass from polynomial to NP-Hard, unless P=NP [27]. Besides this, it is possible for the algorithm to detect if it does exist a negative cycle.

Let $d_k[i,j]$ represent the shortest path length from node $i$ to the node $j$, using only the nodes 1, 2, …, $k$-1 as internal nodes. It is clear to state that $d_{n+1}[i,j]$ is the shortest path from $i$ to $j$ because any node can be an internal node. The algorithm computes $d_1[i,j]$ for all pairs $i$ and $j$. Then, using $d_1[i,j]$, calculates $d_2[i,j]$ for all node pairs i and j. The algorithm repeats this procedure until the iteration $d_{n+1}[i,j]$, and so obtain the shortest distance between each pair of nodes [19] [22].

The core of the dynamic programing in the Floyd-Warshall algorithm is given by the next equation [27] [32]:

$$\mathrm{d}_k[i,j] = \begin{cases} w_{ij}, & \text{if } \mathrm{k} < 0 \\ \min(\mathrm{d}_k[i,j],\ \mathrm{d}_k[i,k] + \mathrm{d}_k[k,j]) & \text{if } \mathrm{k} > 0 \end{cases} \qquad (1)$$

In each iteration, and just like the Dijkstra's algorithm, the Floyd-Warshall algorithm store a predecessor index of each node, allowing the construction of the shortest path route.

The Floyd-Warshall algorithm has a complexity time of O($n^3$) [24]. Comparing with Dijkstra's algorithm (per example), that, in the original implementation has O($n^2$), the complexity time of the Floyd-Warshall algorithm would be predictable to be greater because while Dijkstra's only computes SSSPPs, Floyd-Warshall algorithm computes ASPPs. Thus, applying $n$ times an algorithm with complexity O($n^2$) will make other algorithm with complexity time of O($n^3$).

The applications spectrum of algorithms as the Floyd-Warshall is very wide, like stated in the beginning of this chapter. In fact, as stated in [28], this type of algorithms is very important in routing the data packets of communications networks to avoid communication delays. In particular, finding the shortest path between each pair of nodes can be a very heavy task, in a network with thousands of nodes.

One other algorithm for solving the ASPP is the *Johnson's* algorithm. The particularity of this algorithm is the fact that it can be faster – with smaller complexity time than Floyd-Warshall algorithm- for sparse graphs. A sparse graph is a graph in which the number of arcs is much lesser than $n^2$, where $n$ represents the number of nodes. As in the Floyd-Warshall algorithm, it gives the shortest paths from all pairs of nodes in a graph with positive or negative arc costs, but with no negative cycles. Similarly, it is possible to report if there exist a negative cycle in the network [24].

It is possible to see more for this problematic in the Chapter 25 of [24], Chapter 5 of [14] and [15].

In some cases, the most traditional algorithms for solving the SPP and APSPP are not viable due to the complexity time for large number of arcs and nodes, so alternative methods are needed and used. The Ant Colony Optimization (ACO) metaheuristic, is a versatile algorithm and proves to be efficient to a lot of NP-Hard problems [29]. Besides not being in focus in this chapter, for these cases, the lecture of [29], [30] and [31] is recommended.

## 2.4. Travelling Salesman Problem

### 2.4.1. Outline of the Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is one of the most widely studied problems in the combinatorial optimization area. It is defined in a graph and states as follows [32]. Given a graph G=(N, A), being N the set of $n$ nodes to be visited, and A the set of arcs, let $C_{ij}$ be the cost of traveling from node $i$ to the node $j$. The objective of the TSP is to determine the minimum cost Hamiltonian circuit, which means that it is necessary to find the minimum cost circuit passing once, and only once, in every node of G. As happens in the SPP, presented in the Chapter 2.3, the cost $C_{ij}$, associated to each arc can be any measurement, like distance, time, capacity, etc. The TSP is often modeled in a complete graph, meaning that exists one arc connecting each pair of nodes. If there is no path between two nodes, adding a fictitious arc, with an infinity cost connecting them, will complete the graph without affecting the optimal tour.

Great effort has been made in literature solving instances with increasingly number of cities (nodes). In [33] and [34], a study of the milestones achieved for the TSP is given.

There are two major types of TSP. The symmetric TSP (STSP) and the asymmetric TSP (ATSP). The STSP is defined in a symmetric graph, which means that travelling from node "A" to "B", per example, has equal cost to travelling from "B" to "A". And this premise happens to every pair of nodes in the graph. On the other hand, in the ATSP, the costs are asymmetric, which means that traveling from "A" to "B" can be different of travelling from "B" to "A" [35][36].

Thus, the total number of possible route solutions for the TSP in a graph, will depend if the graph is asymmetric or symmetric. If the graph is asymmetric the next equation gives the maximum total number of TSP routes [35]:

$$\mathrm{N} = (\mathrm{n} - 1)! \tag{2}$$

Where:

$n$ is the number of nodes

On the other hand, if the graph is symmetric, the next equation gives the maximum total number of TSP routes [35]:

$$\mathrm{N} = \frac{(n-1)!}{2} \tag{3}$$

Where:

$n$ is the number of nodes

Therefore, the TSP is very easy to understand but very hard to solve. It is possible to observe that the number of possible solutions increase exponentially with the number of nodes in the graph, making so extremely difficult to compute optimal solutions, when the number of nodes is large. TSP is so part of the so-called NP-Hard problems due to the great complexity for solving it [36].

TSP is famous due to its complexity but also due to its range of applications. Applications on the TSP are beyond route planning. Areas such electronics, mathematics, computer science, genetics, engineering, machine scheduling, job sequencing, wallpaper cutting, among others, are examples of TSP applications [32][37].

There are different methods to solve the TSP, generally divided in two major classes. Exact algorithms and heuristic methods. Exact algorithms give always the optimal solutions, but these algorithms need very large computational times when compared with other methods. On the other hand, heuristic approaches may give good (or even optimal results) in some cases, or bad results in other cases. The advantage of using heuristic approaches for the TSP relies in the computational time, which is very low when compared with the exact algorithms. Thus, depending on the context of the application (depending on the computational time, the number of nodes, etc.), there are several possible solutions, being them exact or approximation heuristics.

Some examples of exact applications and heuristic approaches and its characteristics are given below.

**Table 1 - TSP Solving Methods.**

| | **Solutions quality** | **Computational time** | **Implementation Difficulty** | **Recommended References** |
|---|---|---|---|---|
| **Integer Linear Programming** | Optimal | Exponential | Relatively Simple | [38] [39] |
| **Brute Force** | Optimal | Exponential O($n!$)) | Simple | [42] [40] |
| **Branch and Bound** | Optimal | Lower when compared with ILP or BB | Relatively Simple | [48] [41] |
| **Concorde** | Optimal | Described as the most performing exact algorithm currently available. | Hard to implement due to the great number of lines of code. The code is open source for academic purposes. | [34] [42] |
| **Greedy Algorithm** | Approximate | O($n^2$ $log^2(n)$)) | Simple | [43] [44] |
| **Nearest Neighbor** | Approximate | Relatively lower time O($n^2$) | Simple | [45] [46] |

| | | | | |
|---|---|---|---|---|
| **K-OPT and its variants** | Very good approximations. 1-2% below optimum | $O(n^k)$ | Hard | [47] [48] [49] |
| **Simulated Annealing (SA)** | Good approximations. 4% below optimum | Higher when compared with K-OPT | Relatively Simple | [43] [50] |

## 2.4.2. Travelling Salesman Problem Variations

There are several variations that make each TSP a particular problem. Usually, these configurations add new features to the "basic" and stated above configuration, making it even more difficult to solve (strongly NP-Hard). These variants have been suggested from various real life or potential applications [37]. In this work, four variations of the problem will be briefly addressed.

### TSP With Time Windows

In most business organizations, there are fixed scheduling's – like opening hour, closing time, etc. - which makes important to define time windows in problems like the TSP.

The TSP with time windows (TSPTW), as the regular TSP, involves a graph G=(N, A), being N the set of nodes to be visited and A the set of arcs. The difference here is that each node has a defined interval $[r_i, d_i]$. The $r_i$ represents the release date, which denotes the earliest possible starting time for visiting the node $i$. On the other hand, the $d_i$ denotes the latest possible time for visiting the node $i$. Thus, this interval is called time window, and its width is given by $d_i - r_i$. For the depot, that is, node 0, the $r_0 = d_0 = 0$ [51]. Allied to this, the constant $p_i$ represents the processing time in the node $i$.

Therefore, the problem is to find the minimum cost route (time, distance, etc.), starting and ending in a specified depot, visiting a set of customers, each one in its predefined time window, having in consideration the arrival time and the processing time of each node. TSPTW can be used for practical applications in bank or postal deliveries, school-bus routing and besides this, it can be also used to model a job sequencing in a single machine, where each job has a release time and a deadline [52] [53].

It is possible to observe that the TSPTW is a special case of the Vehicle Routing Problem with time windows [54], where only one vehicle composes the fleet.

### TSP With Pickup And Delivery

Another extension to the TSP, is the TSP with pickup and deliveries (TSPPD). The TSPPD also involves a graph G = (N, A), being N the set of nodes to be visited, and A the set of arcs. In this case, the set of nodes to visit are divided in two groups. The first group contains the locations requiring amounts of goods to be picked up ($p_i$), and the other contains the amounts of goods to be delivered ($d_i$). The node 0 corresponds to the depot location, being a node of pickup the amount to be delivered in the set of delivery customers. One aspect to have in mind is that, in this case of the TSP, the capacity of the vehicle must be considered. If, for one side, the amount of goods being transported decreases when a delivery node is reached, it increases when a node of picking is reached. For this, the vehicle has a maximum capacity allowed, Q, and a current capacity, c, which represents the capacity being transported at each location. Therefore, c may never exceed Q during the tour. The TSPPD consists of determining the least cost tour (distance, time, etc.), starting and ending in the depot, visiting each node once and having in consideration c and Q, and if the node represents picking up or delivering goods [55] [56].

It is important to state that a node can be simultaneously a pickup and a delivery node. A mathematical formulation for an Integer Programming modeling the TSPPD can be found in [57] [58].

There are various applications for this problem, like school buses scheduling, distribution of goods to supermarkets, cab scheduling, distribution of postal services, etc.

### TSP With Precedence Constraints

Sometimes, in several problems such as scheduling, routing decision, process sequencing, among others, it is necessary to process some tasks before others [59]. In fact, the already explained TSPPD and TSPTW deal with these problematics, that certain nodes must be visited before others, because of the picking and delivering constrains or due to the time windows assigned to each node, respectively. Although TSPPD have this type of constraints, the TSP with Precedence Constraints (TSPPC or PCTSP) tries to solve these problems in a more generalized way. The TSPPC is one of the most difficult combinatorial optimization problems. Thus, given a graph G=(N,A) where N is the set of nodes to be visited and A the arc set, the objective is to find a minimum cost tour, starting and ending in the depot, visiting all the nodes of N-{depot}. Besides that, every node $i$ must be visited before node $j$ (but not necessarily directly), when a precedence constraint exists between these two nodes [60]. When a node must precede other, say $i$ must precede $j$, it is also common to use the notation $i \prec j$. Given a certain node $i \in N \backslash \{Depot\}$ that must precede a set of nodes, $p_i$, and succeed a set of nodes, $s_i$. If $|p_i|+|s_i|= N \backslash \{depot\}$ means that $i$ must have a fixed position in the final tour [61]. Therefore, the precedence's between the nodes can exist for some of them, requiring that only certain nodes need to be visited before others. The precedence's can also exist between all the nodes, when there is a fixed sequence between all the nodes that compose N.

The amount of research and applications on this problem is fewer when compared to other routing problems [61]. A mathematical formulation for the TSPPC can be found in [59], [62] and [63]. In [61], a Branch and Cut algorithm is developed for solving the ATSP with precedence constraints. Different densities structures of precedence's were tested for different network instances, using real life data obtained from industrial applications as well as randomly generated instances. The results for the instances with dense structure precedence's show that it is very difficult to construct optimal solutions, proving so the already mentioned increased difficulty of this variation of the TSP.

In [62], Kubo and Kasugai developed a Branch and Bound algorithm incorporating three different bounding procedures, computed from the Lagrangean Relaxation. The algorithm was tested for different densities of precedence's and different instances size. The algorithm performed well for 49 nodes and with relatively lower computational time.

The TSPPC can be also modeled using a two-commodity network flow problem. To solve a model such this, Moon *et al.* [59] proposed a Genetic Algorithm (GA) in which for small and medium size problems, the algorithm reported optimal solutions. In this example, the path is considered feasible if visits all the vertices, not requiring if it does not return to the initial node. The graph is directed, and the nodes (or vertices) represent activities and the directed arcs (or edges) represent the precedence relation between activities.

Sarin *et al*. [64] developed a new formulation for the ATSP with and without precedence constraints. This algorithm computes tight lower bounds and it is usually required a significantly lesser (by several orders of magnitude) computational effort to reach the optimal solution. Different instances were tested using several densities of precedence's, and the results were presented.

### Sequential Ordering Problem

Although Sequential Ordering Problem may not be a "direct" extension of the TSP, it is a problem with several similarities (like will be demonstrated below) and for that, it will be described in this section.

The Sequential Ordering Problem (SOP) is a well-known combinatorial problem defined on a graph. Given a graph *G*, with *n* vertices and *m* weighted directed edges, the SOP is the problem of finding the minimal cost Hamiltonian path from the start vertex to the terminal vertex, following precedence constraints on the vertex set [65]. In some literature, the SOP is associated to the TSPPC (or the asymmetric TSPPC), presented in the section 0, but it is different, in one characteristic. Both problems have precedence constraints in the vertex set. But, in the case of the SOP, there is only defined a fixed start and end node. While in the PCTSP, as already observed, it is required a closed tour where it is necessary one return to the start node. In the Figure 2 it is possible to observe the difference of two solutions for these problems. It is important to have in mind the precedence constraints presented in both solutions, having so the same sequence of nodes. Thus: (1) in the left for the SOP, with a fixed node and an end node, and (2) in the right for the PCATSP, with a closed tour [66] [67].

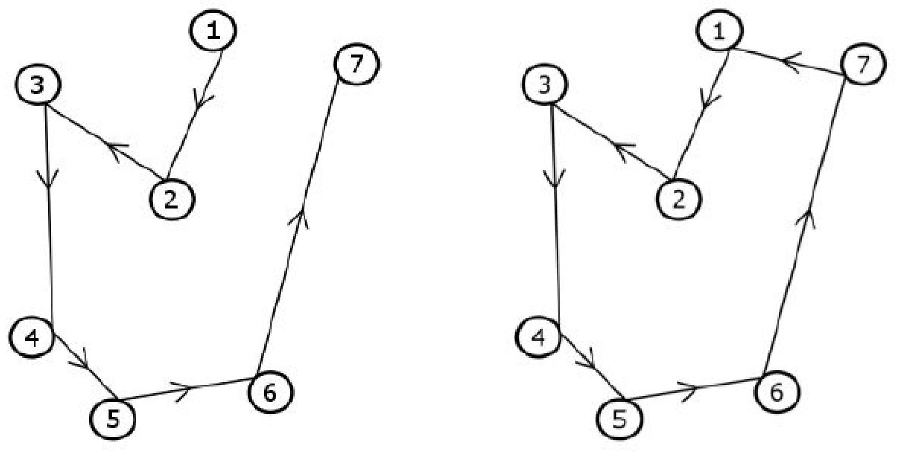


**Figure 2 - Examples of the SOP (right) and the TSPPC (left).**

Thus, SOP is a generalization of the TSP, and so must be NP-Hard [68]. The scope of applications of this problematic is very wide, including, vehicle routing with pickup and deliveries, single-machine scheduling problems with set-up costs and precedence between jobs, among others [69].

A variation of SOP is the capacitated SOP. This problem adds the capacity constraint to the problem. Thus, a vehicle with a capacity $Q$ and a precedence relation $(p,q)$ is associated with a commodity that has a weight of $d_{pq}$, needing so to be collected at $p$ and delivered at $q$. Following the similarity between SOP and TSPPC stated previously, this variation of the problem is related to the Travelling Salesman Problem with Pickup and Deliveries [69].

There are some examples of different applications in literature. In [70], a metaheuristic Ant Colony Optimization algorithm that uses a local search to improve the overall performance of the ACO metaheuristic is developed. It is strongly based on an Ant Colony System and is a building heuristic in the sense that starts from node 0 and adds new nodes until all the nodes have been visited and the last node is reached, always according to the precedence constraints.

To see more applications and state of the art related to the SOP problematic, the reading of [68] is recommended.

# 3. DYNAMIC VEHICLE ROUTING

## 3.1. OUTLINE OF THE DYNAMIC ROUTING PROBLEMS

The development of technologies lead to a model of different routing problems, dealing not only with static routing problems (as the examples already mentioned of the SPP and TSP), but also with dynamic routing problems. The dynamic routing problems arise due to the fact that static network optimization problems do not depend on the time and so, there are some time dependent parameters that are not considered [71].

Thus, a dynamic environment means that the information of the network may be changing during the execution of the algorithm. Moreover, following a dynamic environment, at each time, the choice of a route is based on the information then available [72]. In a dynamic routing problem, what may be the 'best route' for one to follow, may or not be the best, for the same entity, in a different time instant, due to the network information updates.

The Figure  and Figure  illustrate a possible scenario in a dynamic routing problem. Initially (Figure ), at time $t_0$, the best route for a vehicle with characteristics $x_1$, is starting in the node 'S', following the sequence demonstrated in the Figure  and turning back to 'S'. The same vehicle (with characteristics $x_1$), at the time $t_1$ will have a different best route, because the arc connecting the nodes '1' and '2' is unable to be traversed, due to some certain event (Figure ). Therefore, for the same vehicle and network, the assigned the route is different depending on the time.

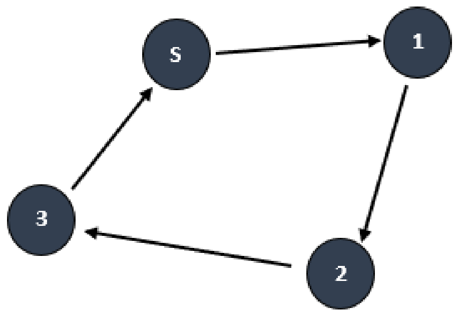


**Figure 3 - Best route at time $t_0$.**

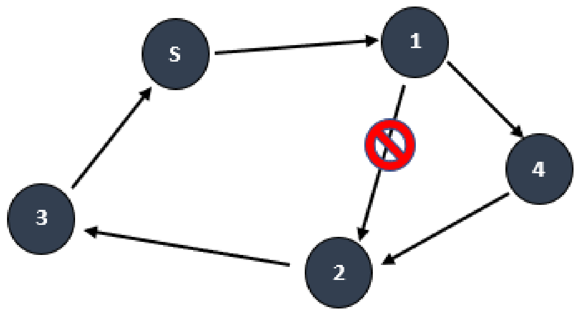


**Figure 4 - Best route at time $t_1$.**

Thus, in the dynamic routing problems there are certain events which may change the network and consequently change the transportation process [13]:

1. New Requests: If it is necessary to visit a new location closed to a planned location, (in a fixed time windows and/or other specified attributes), an adaptation of the tour may be necessary to include this new request.

2. Changes in request attributes: The attributes of each location may be different (requesting a different amount of goods, changing the time window, etc.). Therefore, it will be necessary to rearrange the previous tour, perhaps altering the path of the route, or altering some fleet characteristics.

3. Traffic Congestion and blocked roads: The traffic jam in the roads increase their travel times or can provoke a complete blockage of the affected roads. When this happens, reassigning the vehicles to non-congested roads, or allocating the requests to other vehicles may be necessary.

4. Vehicle disturbances: When a partial or complete deficiency on the vehicle exists, due to an accident or other possible scenario, the routing plan may be rearranged, assigning other vehicles to the requests of the incapacitated one.

A great different between the static and the dynamic routing problems is the objective function. In the static routing problems, usually the objective function tries to minimize the route cost. The dynamic routing problems introduce different scenarios, like service level, throughput time, or revenue maximization [73]. In real-time dynamic routing problems, the objective is sometimes the aggregate of several objectives, combining different measures [74].

Alan Larsen [75], proposes a framework, dividing the dynamic routing problems, depending on their degree of dynamism. The Weakly Dynamic Systems are problems in which the grater part of the information is known in advance, that is, at the time of the tours' construction. The reacting time is considerably longer when compared with others and the tradicional way of solving this problem is to adapting static procedures. Thus, a static routing problem is solved every time an update on the network happens. The Strongly Dynamic Systems are characterized by the fast change of data, and by the urgency of requests received. As examples of this systems, are the emergency services (such as police, fire department and ambulance), and the taxi cabs, in which only a few “customers” are known beforehand. Therefore, in such problems, the reaction time is of great importance.

With this necessity of updated information, dynamic routing problems usually involve more elements than the static routing problems, increasing so the complexity of their decisions and introducing new challenges while judging the merit of a given route plan [73] [76].

If a problem is dynamic, it can also be stochastic or deterministic [77] [74]. In a deterministic and dynamic problem, part or all the information is unknown in advance and depend on time, being revealed during the design or execution of the routes, per example. For this problem, typically is necessary to have technological support, for example cellphones, or global positioning systems (GPS), for real-time communication between the vehicle and the central depot [73]. One example of a deterministic and dynamic problem is the one presented by Daskin [78]. In this problem, a TSP in a time dependent network is addressed. The time dependent TSP (TDTSP) is a generalization of the regular TSP, in which the travel time between two customers or between a customer and the depot depends on the distance between them, but also depends on the time of the day. Here, a Mixed Integer Linear Programing (MILP) formulation is presented, and the results are reported.

In a stochastic and dynamic problem, the uncertain data is represented by a stochastic process. Therefore, the unknown data is a collection of random variables, being so travel times, unknown demands and/or the existence of customers. The data are so gradually revealed during the operational interval, making so that they are not constructed beforehand [75]. One example of this type of problems is the case of the Dynamic Traveling Repairman Problem [79].

In addition to the examples previously presented, one of the most known routing problems in a dynamic network is the dynamic shortest path problem.

## 3.2. Dynamic Shortest Path Problem

The dynamic shortest path problem is the generalization of the static SPP, already explained, where the characteristics of the network may change overtime.

Dynamic shortest path problems are computed in a time-dependent network, instead of a static network as in the static version of the SPP. Thus, a time-dependent Graph is defined as G=(V,E,T), where V is the set of nodes and E is the set of arcs representing the network segments, each one connecting two nodes. For every arc $e=(v_i,v_j) \in E$, and $v_i \neq v_j$, there is a cost function $c_{vi,vj}(t)$, where $t$ is the time variable in time domain T. This cost function represents the travel time from $v_i$ to $v_j$ starting that arc in the time $t$ [80].

Considering that a cost of one (or more) arcs may change during the calculations, the dynamic shortest path problem is to compute the shortest path between one to all the other nodes, or between all the pairs of nodes present in the network. Thus, the dynamic SPP deals with non-fixed arc costs [81].

Dynamic SPP can be further divided into two types, depending on how the time is treated [82]: discrete and continuous. In the discrete type, the time variable is modeled as a set of integers, while in the continuous, the time variable is treated as real numbers. Depending on the type of how the time is treated, the cost function can also be continuous in time, or discrete, whose domain and range are integers [83]. Therefore, dynamic SPP is more about fastest path than shortest path per se. Typically, the objective is to find the fastest path from one node to another, which may not be the shortest one in terms of distance. However, the time of traversing an arc is generally directly proportional to the distance of that arc.

The network can be also FIFO (first in, first out) or non-FIFO. If the condition FIFO holds, no one can depart later at the beginning of one or more arcs and arrive earlier. On the other hand, when the network is non-FIFO, it is possible for an entity to depart later and arrive earlier at the destination. The difference between the former networks lies on the travel functions of the arcs. If the functions are constant or increasing with time, means that the network is FIFO. If there is travel functions that are decreasing with time, the network is non-FIFO [80].

One practical example of such networks can be given by considering a link composed of two physical channels, one being faster than the other. If the policy is to send a message over the first available channel, then a message sent over the slower one may arrive later than another message sent later in the fast channel, meaning that messages arrive in non-FIFO order [83].

These different types of networks bring many implications, such as, if waiting at nodes is possible or not, or if the time of the departure is restricted or unrestricted, etc. For example, if the network is non-FIFO, sometimes it may be preferable to wait a certain amount of time in the node, before entering in one arc. One other example can be the system entering time. This time is the time that the entity starts its route. If this time is restricted means that the entity must enter the system in a fixed time. On the other hand, the entity can have an allowed interval of time before entering the system. Such conditions will have impact on the solution of the shortest path.

An illustration of the dynamic shortest path problem is given in [84] and can be observed in the Figure 3. In this case, it is possible to state that, the edge ‘e’, has a time dependent cost. Therefore, when computing the shortest path from the source, ‘s’, to the destination, ‘d’, the shortest path and cost will depend on the time of the departure. The graph presented in the Figure 3 shows an example of a non-FIFO network.

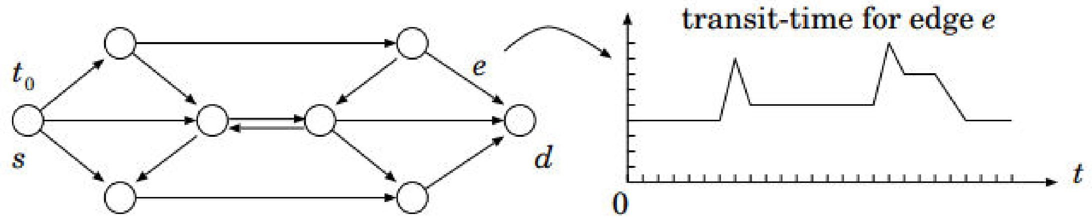


**Figure 3 - Example of a time dependent shortest path.**

In the dynamic version of the SPP, the algorithms can also compute not only the shortest path from one-to-all given a departure time, but also from all-to-all for all departure times. As happens in the static version, this problem can be turned into the fastest path problem, least cost path problem, planning, etc. [82].

Compared to the static SPP, the literature in this problem is surprisingly much more limited. In the study of Cook and Halsey [85], a dynamic Programming algorithm is developed to address the dynamic SPP.  In 1969, Dreyfus [86], is the first to address the time dependent shortest path with a generalization of the well-known Dijkstra’s algorithm.

In this generalization, waiting times at nodes were not allowed. It was proved [87], later, that this generalization is only valid if the network satisfies the FIFO conditions. On the other hand, the time-dependent cost functions of the arcs are usually difficult to be forecasted, thus the link travel times are typically described by random variables.

In [88], a study of the complexity of shortest paths in time-dependent graphs is outlined.

Besides routing, there are an enormous variety of problems were a dynamic modeling may be addressed. Among them are Design of a service network, Repositioning of empty vehicles to anticipate future demands, Production and Inventory Management, Facility planning and design, etc. [77].

Previously, four types of events with most impact in the dynamic routing problems were addressed and explained. The traffic jam was one of them and is mentioned as being one of the principal events in dynamic routing problems and is widely studied in literature.

## 3.3. Traffic Assignment

### 3.3.1. Overview

Since the early 1990's, road traffic has been increasing and causing congestion, delays, accidents, and environmental problems, almost in all large cities [89] [90].

Besides this, congestion also results in a massive delay for the vehicles due to the fact that the time of traversing a road is unpredictably higher whenever congestion is present [91]. Therefore, traffic congestion is a noteworthy problem, and the reduction of the congestion a major challenge [92]. All the costs caused by the traffic can be reduced or even eliminated, by using the transportation systems efficiently. In literature, there are several strategies to avoid traffic congestion, depending on the problem, such as selecting alternative routes, changing the customer-vehicle assignment, among others [93].

To achieve an efficient way to organize the transportation system, the traffic (or transportation) planning problematic can be addressed [18]. The Traffic Planning can be divided into several processes [94], having in consideration goal definition, collection of data, travel forecasting, among others, which are analyzed separately, and often in a predefined sequence.

In what this study concerns, one of the most important processes in Traffic Planning is the so-called Traffic Assignment. The Traffic Assignment is the part of traffic planning that determines traffic loadings on arcs and paths of the road network of interest in a static or dynamic environment [95]. The difference between static and dynamic is, as stated above, that a static approach, by definition, cannot reflect any variation in the traffic flows and any change in the transportation conditions, over time [96].

Therefore, succinctly, the Traffic Assignment Problem (TAP) is stated as follows [97]: Given a directed graph *G*, and a matrix of tours, containing the number of travelers from an origin location to a given destination in *G*, the TAP consists in determining a flow assignment on the links of *G* which satisfies the demand for each pair origin-destination (O-D) and minimizes each traveler's time.

The major aims of TAP are the outlined above [98]:

1. Estimate the volume of traffic on the links.

2. Estimate inter zonal travel cost.

3. Analyze the travel pattern of each origin destination pair.

4. To identify congested links and to collect traffic data useful for the design of the transportation transport system.

The output of the TAP depends on the complexity of the application, but always give an estimate of the traffic volumes and the corresponding travel times or costs on each link of the transportation network. In a more sophisticated technique, the directional turning movements at intersections and route flows may be included to the assignment of traffic [94].

### 3.3.2. All-or-Nothing Assignment

One of the first heuristics to address the TAP was the all-or-nothing technique. This technique consists in the basic procedure of assigning all the traffic to the route with minimum traversing time [99]. In the Figure 4, there are two different routes, $R_1$ and $R_2$, for reaching 'B' from 'A'. Therefore, suppose that an amount of flow (vehicles), $x$, must be assigned to the origin-destination pair A-B, that is, starting in 'A' and traveling to 'B'.

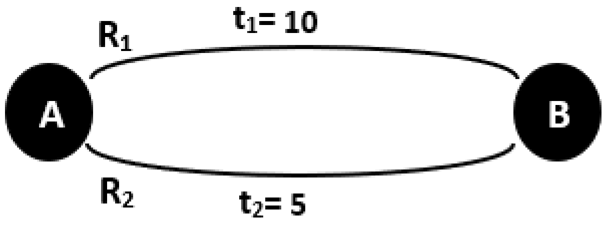


**Figure 4 - Two routes example for the All or Nothing Assignment.**

It is possible to observe in the Figure 4 that the cost (time) of traversing each route does not depend on the flow in the route, having then a constant cost. In the all-or-nothing procedure, all the amount of flow is assigned to $R_2$, being that route the one of minimum cost, regardless of $x$.

It is notable that this technique considers a highly unnatural assumption, that the travel cost is independent of the amount of flow present in the links. If, p. e., there are two alternative routes with a nearly cost, the assignment is always made to the minimum cost route [94].

Moreover, the assigning is made whether or not there is adequate capacity or heavy congestion on the links of the network. Despite this, this procedure may be efficient if the amount of flow to be assigned is low and/or if there are many alternative routes with an accentuated difference in the costs. It may also act as a building block for other models of traffic assignment [98].

This assignment can be made using only a TSP or a SPP instance (depending if there are more than a location to visit or not). Therefore, all the traffic is assigned to that route.

### 3.3.3. LINK COST FUNCTIONS

The results of the all-or-nothing technique are very unrealistic, as stated above. To introduce the concept of congestion, it is necessary to have algorithms considering that travel times on each road of the network are different, depending on the flow in that road. Thus, link cost functions (or link performing functions) need to be developed [100]. As the flows increases in a road, the average stream speed reduces from the free flow speed to the speed corresponding to the maximum flow. The graph presented in the Figure 5 states the typical influence of traffic flow in the average travel time [101] [102].

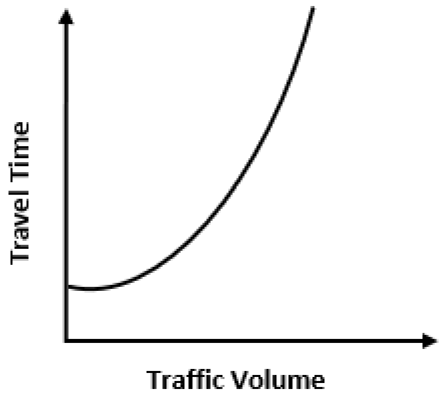


**Figure 5 - Influence of traffic flow in the travel time of a road.**

Different studies developed different link cost functions [97]. The Bureau of Public Roads [103]**,** in 1964, developed the best-known function, taking into consideration that each present vehicle in a road creates an impedance in the road. This equation is presented below.

$$\mathrm{t} = t_0 \cdot \left[1 + \alpha \cdot \left(\frac{x}{k}\right)^{\beta}\right] \qquad (4)$$

where:

$t_0$ is the free flow time

x is the flow on the link

k is the capacity of the link per unit time

t is the average cost time for a vehicle to traverse the road

The α and β are model parameters to be calibrated, but α =0.15 and β=4 are the typically used. The quotient of *x* by *k* is also known as the 'degree of saturation' [104].

### 3.3.4. Capacity-Restraint Heuristic

With the link cost functions, heuristics for TAP including the congestion factor were developed. The Capacity-Restraint heuristic was first developed in the Chicago Area Transportation [105]. In this procedure, a specific origin is randomly selected. Then, the shortest routes are calculated between each O-D pair, being 'O' the selected location. After that, all the traffic containing the latter selected origin is assigned using the all-or-nothing heuristic. At that point, the time of traversing each road is recalculated, having in consideration all the flow assigned so far. This procedure is repeated but now with another specific origin, different from the one chosen in the previous iteration(s), and with different link costs. The algorithm stops after all the origins have been selected, and all the traffic been assigned.

This heuristic differs from the all-or-nothing assignment only in the fact that the travel times are updated after assigning the vehicles from each O-D pair. Thus, the computational times for this method are essentially the same when compared with the all-or-nothing procedure [94].

### 3.3.5. Incremental Assignment

In the incremental assignment procedure, fixed fractions of traffic are assigned to the network in steps, or iterations. Thus, in each step, a fraction of the total flow is assigned to the shortest route, using all-or-nothing assignment. After this, the travel times of each link are recalculated based on the link volumes assigned so far [98]. Usually, the updates in the traffic costs are made by the Equation (4), showed above, or for other link cost functions. The number of iterations is determined in advance, dividing the total amount of tours by the portions of traffic to be assigned in each iteration.

The difference between this method and the capacity-restraint is that, in this method, the portion of the traffic to be assigned is chosen, and so, the travel costs are recalculated at each iteration. In the capacity-restraint method, at each iteration, all the traffic starting at a specific origin is assigned, and only then the travel costs are updated.

The result for this method may resemble an equilibrium of the system, when many iterations are used, that is, when lower portions of traffic are assigned in each iteration. On the other hand, such small increments can increase the computational effort when the number of trips to be assigned is large. The most serious drawback of this approach is that, after an assignment being made, an increment of the flow cannot be reassigned to another path, in the subsequent iterations [106].

Martin and Manheim [107], developed an extension of this method. In their study, the portion of traffic to be assigned is not previously fixed, but determined by a travel-time function, called generation rate characteristic. Consequently, the number of iterations is unknown in advance. With the generation rate characteristic function, when the volume of traffic increases the cost of several paths, in the next iterations, the traffic to be assigned is likely to be reduced, trying to reach an equilibrium in the system.

# 4. CEMENT INDUSTRY CASE STUDY

## 4.1. LIST OF PUBLICATIONS

Before entering in the case study, there are several scientific research studies which compose the basis of the study presented in this dissertation. These research studies are already published, accepted but not published yet, or submitted to publication.

1. **Fonseca, J.**, **Alves, R.**, Macedo, A. R., Oliveira, J. A., Pereira, G. and Carvalho, M. S. (2019), Integer programming model for ship loading management, in J. Machado, F. Soares and G. Veiga, eds, Innovation, Engineering and Entrepreneurship, Springer International Publishing, Cham, pp. 743-749.

2. Macedo, A. R., **Fonseca, J.**, **Alves, R.**, Oliveira, J. A. , Carvalho, M. S., Pereira, G. (2018). The impact of Industry 4.0 to the environment in the cement industry supply chain. Proceedings of ECOS 2018 - The 31st International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS). Presented at the ECOS 2018 Conference.

3. **Alves, R.**, Fonseca, J., Macedo, R., Veloso, H., Dias, L., Pereira, G., Carvalho, M. S., Figueiredo, M., Oliveira, J. A., Martins, C. and Abreu, R. (2018), Cement Industry - A Routing Problem, Cement Update by Daily Cement (5), 10-15.

4. Fonseca, J., Macedo, R., **Alves, R.**, Veloso, H., Dias, L., Carvalho, M. S., Pereira, G., Figueiredo, M., Oliveira, J. A., Abreu, R. and Martins, C. (2018), Rules for Dispatch, BMHR 2018 supplement in World Cement (September).

5. Macedo, A. R., **Alves, R.**, Fonseca, J., Veloso, H., Dias, L., Figueiredo, M., Pereira, G., Carvalho, M. S., Abreu, R. and Martins, C. (n.d.), What can we learn from Industry 4.0: Opportunities in the logistics field on Cement Industry.

6. Veloso, H., Vieira, A., **Alves, R.**, **Fonseca, J.**, Macedo, A., Pereira, G., Dias, L., Carvalho, S., Figueiredo, M. (2018), Simulation in cement industry, CemWeek (July).

## 4.2. Cement Industry Overview

Cement is an inorganic, nonmetallic substance with hydraulic binding properties that is mixed with water to form a paste. After hardening, the cement retains his strength. There are several types of cement products and because of its importance as a construction material, cement is produced in essentially all countries. It is one of the most important materials worldwide and its consumption and production is closely related to construction activity, and, consequently to the general economic activity [108].

All over the world, global cement production grew from 594Mton (Million tons) in 1970 to 2284Mton in 2005, with the vast majority occurring in developing countries, especially China, where the production of cement reached 47% of the overall world production. Besides China, countries like India, Thailand, Brazil, Turkey, Indonesia, Iran, Egypt, Vietnam, and Saudi Arabia accounted for 17% of the 2005 world cement production. Taylor *et al.* [109] shows the production of cement, by country, in 2005. In that same study it is possible to observe the continuous increasing production of cement and the projections till 2050 all over the world.

With this great amount of production, cement is the second most consumed substance in the world, only after water [110]. To produce 1.0 ton of cement, it is necessary to collect about 1.6 tons of main raw materials. This large amount of production makes it so that, usually, plants are located near quarries, which are the source of their main raw materials [111]. Between all raw materials used for cement production, there are limestone, chalk and clay as the most common ones.

The cement industry has also a great impact in the environmental field. This type of industry will come under increasing regulatory pressures to reduce its emissions and to contribute to the reduction of the global warming [112]. The number of articles and the amount of literature review concerning cement industry shows that the impact this industry has in the environment is the immense concern to the scientific world.

In 1995 there were 252 installations producing cement only in the European Union (EU)[113]· The large number of plants all over the world, allied with the almost steady increase in the cement production among the years suggests the importance in the management and the study of the cement industry supply chain.

Moreover, although SCM is a subject with a lot of research and with the technology advances it is also an increasingly investigation topic [110]. Succinctly, in the Figure 6, it is possible to observe the supply chain of cement, from the production till the clients, discriminating the processes involved [114].

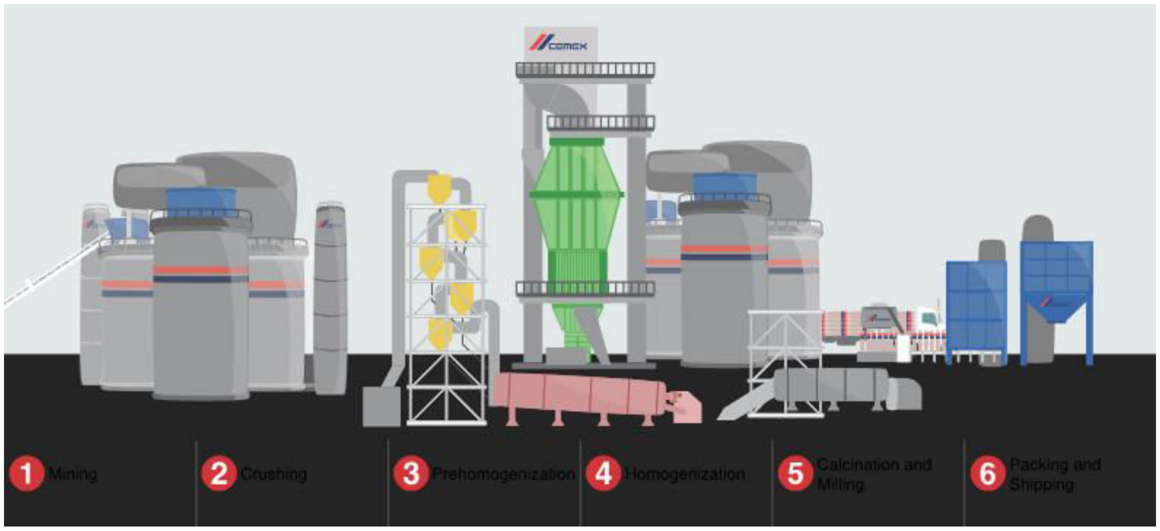


**Figure 6 - Cement Industry supply chain.**

There are two main phases in the production of cement. The first one is relative to the transformation of raw materials in clinker. The second is the production of cement from the clinker [111]. If all the processes composing these steps are geographically apart, there are additional transportation and time costs associated with the supply chain. However, as stated above, usually all the processes are near each other to overcome that disadvantages.

The cement supply chain is complex and somewhat large, as suggested in the Figure 6. However, in this study, the focus is not the processes of extraction and creation of the raw materials and the cement, but the logistic processes inside the plant, when the cement is stored and ready to be shipped - Figure 8 (6).

In the step (6) of the Figure 6, there are two locations where the cement can be stored. The bulk cement, when cement is avulsed, is stored at what is called the cement storage silos. A storage silo is a huge cylindrical structure, like the one presented in the Figure 7, and is used for storing bulk materials, in this case, cement.

**Figure 7 - Cement Storage Silo.**

Pneumatic and mechanical systems can be used to transport cement for the silos. Usually, in a cement plant, there are several storage silos, near each other and/or in different locations. Each silo has one type of cement to be loaded at each instant, and that depends on the current silo configuration [113]. It is also possible for a cement plant to have silos in more than one location and having the same materials, among others.

When the cement is stored in these conditions, a specific type of transportation is also required. At this moment the cement can be transferred by pipelines or tubes to a train, or to a ship, if it is near the plant. If the cement is required to be transported by road, a specific truck is also needed to transport the cement. A cistern or tank truck is used to load the cement stored in the silos. These types of trucks can also be viewed in the Figure 8.

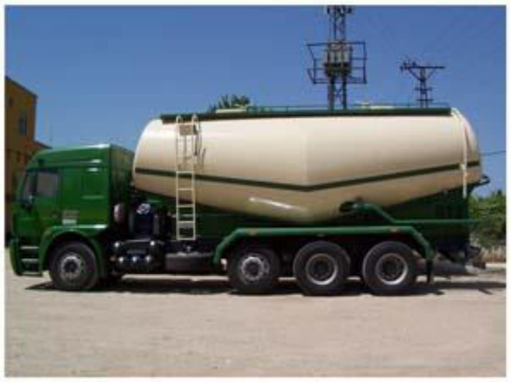


**Figure 8 - Example of a cistern or tank truck.**

The transportation of bulk cement brings some advantages and disadvantages. On one hand, this type of transportation requires a fleet of specific trucks that needs to be owned or outsourced by the clients. Besides this, it is also necessary to have other equipment's to unload the cement. On the other hand, this type of transportation requires less human effort, because the cargo is maneuvered through machinery, thus allowing a relatively easiness in that processes.

The cement can also be bagged and stored in a warehouse. There are several types of bags, with different dimensions and weights, trying to meet the clients' demand. The cement can be stored in several warehouses due to its type and specifications of the product. In a typical cement plant, it is possible that more than one warehouse is present, and it is also possible for different warehouses to have the same products. In that situation, it is not necessary to have a specific type of truck to transport it. Depending on the number of bags to be transported, a "regular" loading truck is necessary. In the Figure 9 it is possible to observe a typical loading truck.

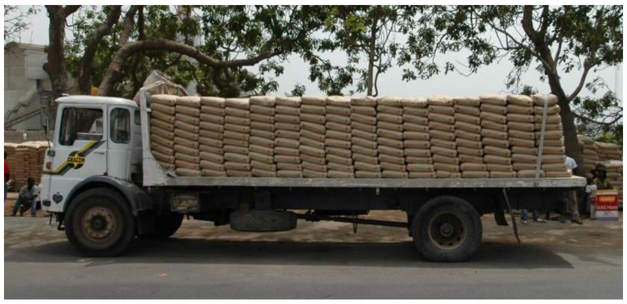


**Figure 9 - Loading truck for bagged cement.**

The transportation of bagged cement can bring some advantages and disadvantages. Bags are more difficult to load and unload. Usually, in a cement plant warehouse, the load is made using palletizers and forklifts. In the client's location to unload, typically the cement in these conditions is unloaded manually. On the other hand, bags are more flexible and can be moved in almost all types of trucks and cars, depending on the quantity of cement to be transported. Bagged cement can also overcome the demand of small and irregular orders.

## 4.3. INDUSTRY 4.0 AND CEMENT INDUSTRY SUPPLY CHAIN MANAGEMENT

With globalization, the market is getting more global and less local. Each time more and more products are available, and their life time is decreasing, by its obsolescence. Also, the quality standards are increasing, and the markets are getting more demanding, imposing short delivery times and at the same time, wanting lower costs. Competitivity is getting increasingly more difficult, and therefore, the companies that use efficiently their SCM are the ones that will survive to this era.

In the cement industry, the lack of SCM is highly present. Cement plants are involved in an unpredictable environment. It is not possible to know, in advance, what materials each client will need, if the required materials are bagged or bulked, what is the day and time each client will arrive at the plant to be served, what are the locations each client must reach, etc. It is important to introduce the concepts of what industry 4.0 represents. Terms like Internet of Things, Cloud Computing, Big Data, integrated systems, Business Intelligence, and so on, are strongly connected to this fourth industrial revolution [115].

Industry 4.0 is an increasing term in the present days. It introduces new technologies, specially information technologies and information techniques. Organization and logistics are implemented in modern business as an aggregated system, which has led to new ways of production, new ways of doing business and better service activities in the sphere of industrial production [116] [117]. The Industry 4.0 allows a 'digital supply chain' and a 'Smart Production', introducing the concept of real-time. This linkage between every stage of the supply chain, between every machine, every sensor, allows to capture and share real-time production data, which could be used for rapid and accurate decision-making [118]. Therefore, and succinctly, Industry 4.0 is about information and how to use technology to efficiently manage that information.

The Figure 10 suggests a general framework of the supply chain for the cement industry, focusing the packing and shipping of the cement. The introduction of the industry 4.0 is represented by the 'information flow' linkage.

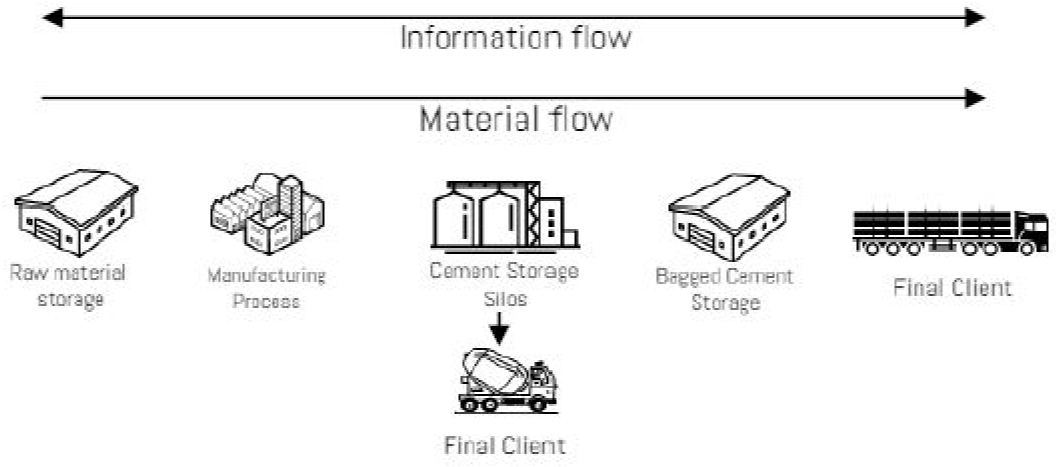


**Figure 10 - Industry 4.0 in cement industry supply chain.**

It is extremely important to have the information flow and material flow connected. With this, it is possible to plan, adapt and execute, making decisions in a much more precise and rapid manner. Industry 4.0 makes SCM easier, increasing also the precision and the efficiency of the available resources, bringing advantages not only for the companies and organizations, but also for the clients, in the service and product quality.

There are several potential applications where technology can have a big impact in the cement industry, among others. In this study, one of them will be focused and is explained below.

## 4.4. PROBLEM DESCRIPTION

The management of the trucks when they are inside the plant is one of the biggest tasks cement plants face nowadays. Whenever a truck enters a cement plant, there are several problems that may come along with it.

Trucks go to cement plants to load or unload materials. If it is necessary to unload various materials, there are locations in the plant designated for each material. A truck can also enter a cement plant to load one or several materials. In that case, the materials can be bulked or bagged, and for that, the type of transportation is different, as mentioned above, and a client can only require bulked or bagged each time.

In addition to this, a typical cement plant receives hundreds of trucks every day. Whenever each truck enters the plant, it must visit one or more locations to be served. The trucks will follow a route, which will lead them to its required services. Due to the great complexity of the cement plants map, usually there are several possible routes for each truck to follow, leading them to the desired locations. Although it is a less significant case, it is also possible for a truck to have some places to visit inside the plant, with a specific and predefined sequence. In this case, the truck must be serviced in all the required locations, following the predefined order.

On the other hand, truck drivers may not be familiar with the plant, and even if they are familiar, they have no assistance or guidance when choosing the routes. This problem leads the drivers to have unnecessary times finding the required locations, and/or making the drivers to loading/unloading wrong materials in incorrect locations, thus resulting in an increased time inside the plant. The congestion inside the plant is also a very important problem cement plants face due to the great number of trucks arriving each day. Overloading some roads of the plant may lead to traffic jams, increasing even more the time each truck spends inside the plant.

It is important to create a routing system, assisting and guiding the truck drivers inside the plants and possibly tackling the traffic jam on the roads. This can only be addressed by introducing technology in the decision-making task, thus allowing each truck to have its own computed route. With this, both service quality and the equilibrium of the facility are improved. The equilibrium of the facility represents the workload level on the servers, and the number of trucks in the roads. When the number of trucks waiting for their services and the number of trucks traversing the same roads at the same time are lower, the facility is in a higher level of equilibrium.

## 4.5. Problem Assumptions

Whenever a problem is modeled, it is always necessary to assume some characteristics. These characteristics encompass all the specifications of the real problem, but also, some inevitable assumptions. The assumptions aim to create robustness in the modeling.

Each cement plant is composed by locations to load, locations to unload, and other possible locations, like for example, areas for the administration. These areas will be, from now on, designated in general as *"service locations"*. Inside of the plant, connecting these areas, there are roads. The roads' surface (i.e. the pavement) is composed by asphalt or only by dirt. Each road can have one or two directions and it is possible to exist more than one road connecting two service locations.

Thus, it is possible to describe a plant through graph theory. Thereby, each one of the service locations will be represented by a node. In the same way, the set of roads will be represented by the arc set of the graph.

The roads, represented by the arc set, will follow a FIFO rule. This means that, when a truck enters a specific road, it will only end its traverse, after all the trucks that already are in that road, in the same direction, have also traversed. Thus, overtaking is not allowed in the roads inside of the plant. Each arc will also have a distance cost of traversing it, and a proportional time cost. It will be assumed that the greater the distance, the longer the time of traversing a road, in terms of simplification.

The average velocity of the trucks will be assumed to be constant and equal when traversing each road inside of the plant, independently on the type of the truck, in terms of simplification.

The cost of traversing the roads will have a lower order of magnitude when compared with the processing time in the service locations. This happens because, typically, the trucks spend more time loading and unloading materials than on the roads inside the plant.

Each one of the service locations has a processing time, depending on the type of service (load, unload, etc.) and depending on the truck (per example, the type of the truck). Therefore, each node will have an associated processing time, depending on that same aspects. The intersections and crossing points will also be treated as nodes, however, in this case, with processing time equals to zero, because there is no service to do in that node.

The type of truck will be independent for the routing system. It is important to know which are the locals that each truck must visit, and the specifications of each service. These specifications are the time each service will require for that truck (depending on the size of the cargo to load or unload) and if the truck has a specific predefined sequence for its services or not.

It is also assumed that after a truck has started being processed in a server, it will be completely served, thus not being considered intermediate stops.

In each service location, only one truck at a time is served. At the same time, if more than one truck reaches the same service location, the trucks will be served one at a time, in the order of arrival at the service location. Thus, the service locations will work following a FIFO order. In a real scenario this may not happen, meaning that, there are service locations that it is possible for more than one truck to be served at the same time. However, in a future possible real implementation, this and all the previous assumptions can be easily altered, thus meeting the real services' characteristics.

In the next sections, three algorithms will be developed to overcome the already mentioned problems, and its implementations addressed. These algorithms will have different considerations as how the routes will be given to the drivers, and when they will be given.

# 5. Approach to the Problem & Applications

Before entering in the algorithms' description, it is necessary to establish and determine how the plants will be organized. Each plant will be organized using graph theory, being the service locations and road intersections represented by the node set, and the arc set representing the roads of the plant. There are several ways to describe the information contained in a graph (see [119]). The data structure that will be used to contain the information presented in the graphs of the plants, will be the "Adjacency Matrix (With Costs)". The Adjacency Matrix is a matrix which indicates, in an organized way, the nodes that are directly connected in the graph. If two nodes are directly connected by some arc (adjacent nodes), the element of the adjacency matrix is 1, and 0 otherwise, thus making this matrix binary.

On the other hand, the Adjacency matrix (with costs) specifies also the cost of the arcs for the adjacent nodes. In this case, the costs of the matrix are distances being the roads of the plants represented by the arc set of the graph. These distances can be represented in meters, kilometers, etc.

It is important to state that the distance between each pair of nodes in that matrix is the distance of the arc that connects that pair of nodes. That distance may or may not be the minimum distance between that pair of nodes. Next, to give a better contextualization, the Figure 11 is an example of a graph. This graph could be an example of a cement facility.

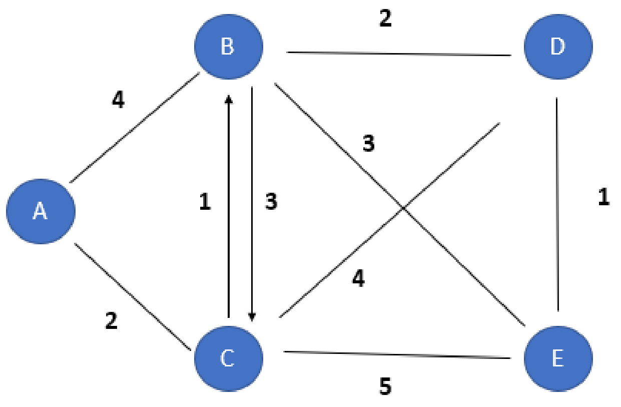


**Figure 11 - Graph example of a cement facility.**

The graph presented in the Figure 11 contains five nodes, representing five service locations, and nine roads. It is important to observe the connections between the nodes "B" and "C". There are two different connections between these two nodes, with different costs and opposite directions, thus making the graph asymmetric.

The respective adjacency matrix is presented in the Table 2. The next algorithms will be relied on this type of matrix.

**Table 2 - Adjacency Matrix for the graph of the Figure 13.**

| | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| **A** | **0** | **4** | ∞ | ∞ | ∞ |
| **B** | **4** | **0** | **3** | **2** | **3** |
| **C** | **2** | **1** | **0** | **4** | **5** |
| **D** | ∞ | **2** | **4** | **0** | **1** |
| **E** | ∞ | **3** | **5** | **1** | **0** |

In the Table 2 it is possible to observe the distance between each pair of nodes. It is important to refer that whenever a distance matrix cell has the symbol "∞", means that there is no connection directly between these two nodes. In a future computational implementation, there is no possible way to include the "infinity". Therefore, a number that is very large when compared with the others is typically chosen.

Next, three algorithms will be presented to approach the already stated problems. When a truck driver arrives at a cement plant, it proceeds to its check-in. When this stage is finished, there will be a system, with an implemented algorithm, that, considering all the information, will compute a specific route for that driver to follow.

There are two general approaches in the next algorithms regarding the possibility of updating the routes. The first one considers that after the driver receives his route, thus entering in the facility, there is no more connection between him and the routing system. Therefore, there is no updates in the given routes, and the only interaction between the driver and the system is in the entrance (check-in) and the exit (check-out).

The second approach considers that after the driver enters the plant, the connection between him and the route system continues. With this, it is possible for the routes to be updated during the trip.

These two general approaches have advantages and disadvantages. For one hand, the update of the routes will encompass a more real time system, considering eventual situations that may change the current state of the facility, thus updating the drivers that are already inside of the plant. On the other hand, for this to happen it is necessary that each driver has an informatic gadget, thus being connected to the route system, in real time, (per example, an app on a smartphone).

One the other hand, if there are no updates after the entrance, the system may not consider some eventual change in the system and update the trucks that already are inside the plant. However, in this scenario, it is not necessary for the driver to be connected to the system, therefore the informatic gadget and the ICT for that to happen are not required.

Each algorithm will consider different scenarios and will have different goals.

## 5.1. Algorithm No.1 – The Distance Approach

The first approach to the problem considers the distance traveled by each truck, and in minimizing it. With this, it is guaranteed that each truck will travel the minimum distance. This approach is, perhaps, the first idea that comes to mind when addressing this problem. By guiding the drivers through this approach, both drivers who do not know the plant and those who already know are guided to the required locations by the route that minimizes their travelled distance. Besides this, this algorithm will serve as a comparison term for the next algorithms.

Therefore, if only one place is required to be visited by some truck, the overall route can be calculated by computing the shortest path between the entrance and the required place, and then calculating the shortest path between that required location and the exit of the facility. Note that the entrance and the exit of the facility can also be in the same location. Hence, this problem can be solved by using a SPP algorithm.

In the Chapter 2.3 of this dissertation, it is possible to observe the most known and used algorithms to address the SPP. The Dijkstra's algorithm is one of the most famous and fast algorithms for solving the SPP, while having a relatively easy implementation. Besides this, it gives always the optimal solution. With all this, a Dijkstra's algorithm is chosen to address this problem.

However, if more than one service location is required to be visited, and for the minimum distance to be achieved, the problem can no longer be solved using only the SPP. In this case, it is necessary to use the traveling salesman problem algorithms. In the Chapter 2.4 of this dissertation, several algorithms addressing the most typical cases of TSP were outlined. In the case of the precedence constrained TSP and the SOP, it is possible to ensure some of the positions of the nodes in the solution sequence. Thus, it is possible to impose some precedence's in the possible route solutions, while in a regular TSP that is not possible (see Chapter 0).

It is necessary to ensure the node representing the entrance of the facility to be the first node, and to ensure the node representing the exit of the facility to be the last node of the possible route solutions. In the case the entrance and the exit of the plant are in the same location, the ensured node to start and end is the same.

To solve this problem, a Brute Force algorithm was designed. This Brute Force algorithm will test all the possible route solutions and give the one with the minimum distance. Besides this, each possible route solution must respect the fact that the node of the entrance and the exit of the facility must be the first and the last in each route, respectively.

As stated earlier, brute force algorithms are not very advised when the problem is complex due to its higher computation times. Still, even if the number of nodes of a plant is high, it is not assumable that a truck requires more than four services in the same visit. Therefore, a brute force algorithm will not compute a higher number of different routes, thus giving always the optimal solution in shorter periods of time.

Before entering in the Brute force algorithm, it is necessary to compute all the minimum distances between all pairs of nodes in the graph. As stated in the Chapter 2.3.2, one of the most famous algorithms for computing the APSPP is the Floyd-Warshall algorithm. However, and as stated earlier, in the case there is only one service location to be visited, a Dijkstra's algorithm will compute the minimum distances. In this case, and to use the same algorithm, that Dijkstra's will be applied to all the nodes, thus computing the all pairs shortest path problem. The running time of computing the Dijkstra's $n$ times, is similar to what would it be if the Floyd-Warshall was used (see Chapter 2.3.1).

With this, a matrix with the same dimensions of the adjacency matrix is created. Hence, in this matrix, each element has the minimum distance between each pair of nodes. It is possible that the minimum distance between two nodes not corresponds to the distance of travelling directly from one to the other, thus passing in intermediate nodes. The overall route must also convey that.

Thus, whenever a truck must visit more than one service location in the facility, it is necessary to compute all the different possible routes it can go and choose the one that offers the minimum distance. When the driver informs the system about the required places, the algorithm will calculate all the combinations of the possible routes for that truck, leading him to its required places. The algorithm will calculate the minimum distance route, passing in all the required locations, starting at the entrance node of the facility, and terminating in the exit node of the facility.

This algorithm must also consider the fact that, as already stated, the service locations may have a fixed sequence to process the trucks. In this case, this approach will compute a route, following the order of the service locations defined by the truck driver. However, the algorithm of this approach will also compute the minimum routes connecting all the required servers, not changing the required sequence by the driver. This will probably not result in the overall minimum distance route but will decrease the travelled distance for the trucks in that conditions.

Therefore, generally, whenever a truck arrives at the facility, it proceeds to its check-in. During the check-in, the service locations are transmitted to the system, and the minimum distance route is calculated, and communicated to the driver again. As the distances of each road are fixed and do not update during the visit, the route it is only transmitted to the driver at the entrance and the driver follows that route. It is also possible for a plant to have the entrance and exit in different locations and the algorithm will consider that as well. This approach can be observed in the next flowchart.

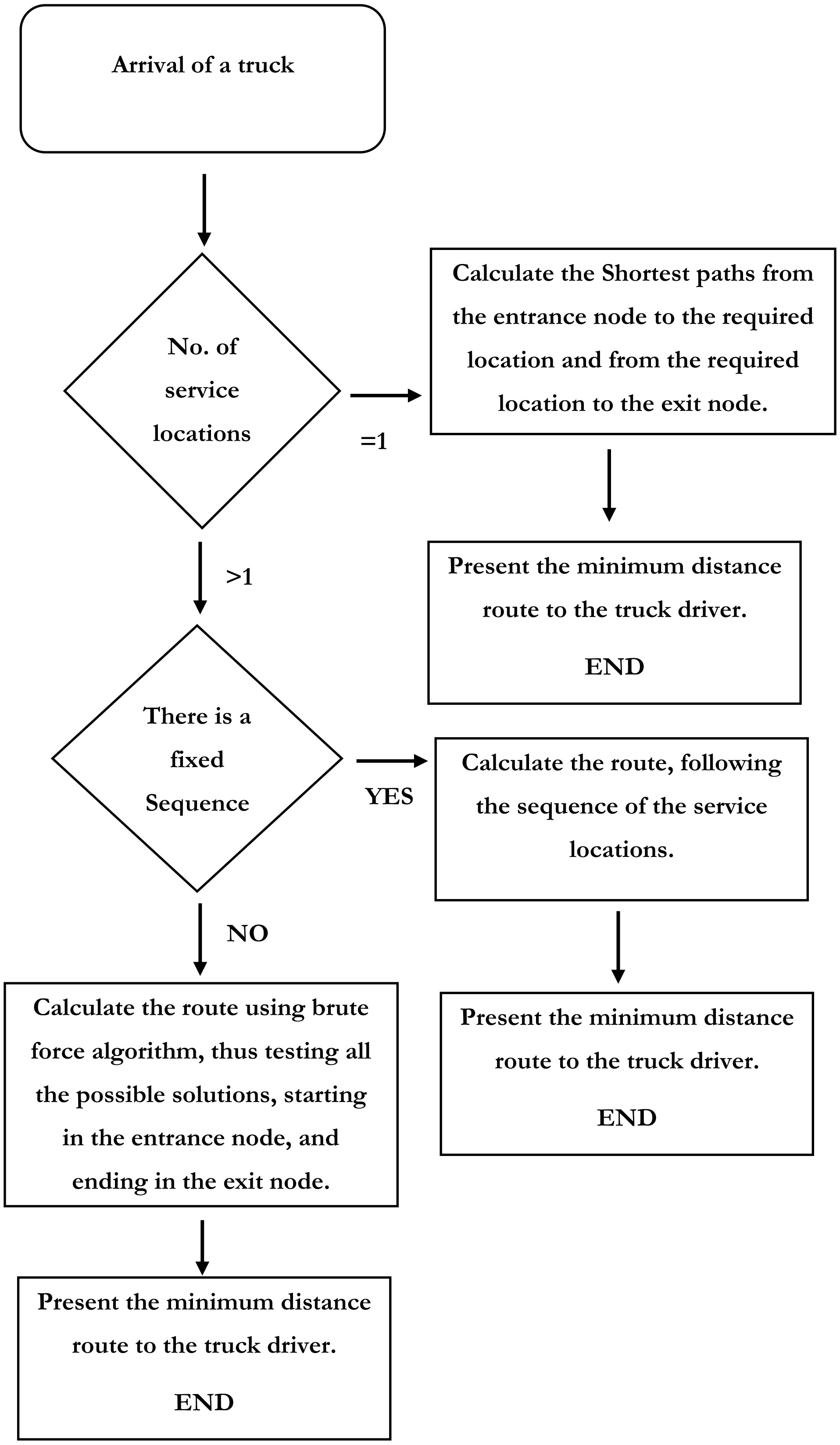

Arrival of a truck
No. of service locations
=1
Calculate the Shortest paths from the entrance node to the required location and from the required location to the exit node.
Present the minimum distance route to the truck driver.
END
>1
There is a fixed Sequence
YES
Calculate the route, following the sequence of the service locations.
Present the minimum distance route to the truck driver.
END
NO
Calculate the route using brute force algorithm, thus testing all the possible solutions, starting in the entrance node, and ending in the exit node.
Present the minimum distance route to the truck driver.
END

**Figure 12 – Flowchart of the Algorithm No.1.**

## 5.2. Algorithm No. 2 – The Equilibrium Approach (without updates)

The algorithm presented in this approach is the greatest contribute of this dissertation, as well as the Algorithm No.3 -being the algorithm No.3 a variation of this one- and, to the best of my knowledge, there is no procedure gathering the characteristics of this approach.

In the algorithm No.1, the distance travelled by each one of the trucks is minimized. Besides this, if each driver follows its associated route, every truck will reach its required places. With this, it is possible to overcome problems like the one where the drivers do not know where the service locations are and how to reach them.

However, the problem of the congestion inside the plant is not addressed by that approach. If, on one hand, the congestion may decrease because the trucks are not getting lost inside the plant, on the other hand, due to this static choice of the routes (always choosing the minimum distance one) the congestion will increase in some roads of the facility.

The algorithm of this approach is concerned with two principal and connected objectives. The servers' (service locations) workload and the congestion and traffic jam in the roads. This approach will not consider any updates in the routes after being given at the entrance. Thus, the control of the trucks is made in the entrance and in the exit of the plant, being the route given to the driver during its check-in.

There is a question that arises from the approach of the algorithm No.1. If, for example, two trucks arrive at the plant, requiring visiting the same service locations, the first approach will give the two trucks the exact same route, minimizing each truck traveling distance. Thus, the trucks will enter the facility, in FIFO sequence, following the same route. This, following the assumption that, in each service location, only one truck is served at a time, the first one will be served and the second one must wait for the service. Therefore, one good principle would be to divide them between the required service locations. Thus, the first truck will travel to a service location, and the second to other service location with minimum workload, therefore avoiding congestion and unnecessary waiting times. This illustrative example shows the problem for only two trucks. But, with hundreds of trucks arriving at the facilities every day, this problem becomes even more relevant.

A dynamic approach is necessary to overcome the unnecessary waiting times. When a truck enters a facility, besides the minimization of the traveling distance, it is also necessary to consider the workload of the required service locations.

As stated earlier, this approach will not consider the updates of the routes after being given to the drivers. Therefore, for the workload of the servers being properly considered in the calculation of the routes, it is necessary to store information relative to the given routes, since the beginning of the calculations (for example, using the algorithm since the beginning of the day, in a possible application).

Thus, it is necessary to consider the times of traversing each road of the facility. Hence, besides the distance of each road, the time of traversing a road will also be considered. It is important to remind that the service time of each truck in the servers is grossly higher than the time of crossing the roads.

Therefore, when associating a truck to a route, each time of reaching and leaving a destination will be stored. Consider the situation when a truck enters a cement plant, at time 0 (with no more trucks inside the plant), with required service locations “A” and “B”. Assume that the route given to the driver is E-A-B-O (going from the entrance, E, to A, from A to B, and from B to the exit, O, consequently). Thus, if the road connecting E and A takes 10ut (units of time), and if the server A is empty at the time of its arrival, the truck will be served at the time of 10 in the server A. If its service time in A is 20ut, the truck will leave the server A at the time 30 (10+20). Thus, following the route, assume that the road connecting A and B takes 5ut. The truck will now reach the server B at time 35. If the server B is empty at that time, and if the service time of the truck in server B is 20ut the truck will leave the server B at the time 55 (35+20). Following the route, assume that the road connecting B and O is 5ut, so, the truck will leave the facility at time 60.

Therefore, it is necessary to store the times this truck reaches and leaves its required locations, for the servers' workload level to be properly considered. Thus, this truck will be in the server A within [10,30] and will be in the server B within [35,55].

Thus, in this approach, every time a route is given to a truck driver, the interval of times that each truck reaches and leaves each server will be stored. With this, as the routes are given at the entrance, for the next trucks to arrive, the algorithm will be able to consider the future workload of the servers.

In what concerns the servers' workload, this approach will, whenever a truck enters a plant, choosing the servers' sequence by following the next premise. The first (or next) server to be served will be the one with least workload within a range. That is, the server with the minimum number of units of time occupied in the interval:

**[arrival time in the server, arrival time + service time]**

In order to clarify this situation, consider the following example. Assume that a truck X1, entering the facility at the time 25, has a service time of 20ut in the server A and a service time of 20ut in the server B. The servers A and B are the required service locations for the truck X1. The algorithm must choose what will be the first server and the second server for this truck, thus creating the servers' sequence.

Following any route, X1 would reach the server A at the time of 30. However, it is necessary to remember that every time a route is given, all the intervals of reaching and leaving the servers are stored. Thus, each server may or not have already scheduled trucks. The server A is occupied by some other truck in [12,32], and after that time is empty for the whole time. If the truck X1 goes to the server A first, it will be served at the time of 32 and will end its service on that server at 52.

On the other hand, if the truck X1 follows some other route, it would reach the server B at the time of 31, and that server is unoccupied at that time, thus ready to serve the truck X1. However, the server B has also a scheduled truck. The server B will be occupied in [33,53] by some other truck, that is already following its own route, and serving the truck X1 at 31 will delay the service of the other truck, that is expecting to be served on that server at 33.

It will be assumed that, if the server with minimum workload level is not occupied at the arrival time, the truck will occupy it during its service time, what may delay the service of other trucks. This happens so that the servers are working constantly, in the maximum occupancy level possible, without breaks. Yet, despite this, if this assumption can not be made in a future real implementation, it is relatively easy to modify it by changing the way the algorithm will works.

In the previous example case, the workload of the server A is one truck during 2ut in the above-mentioned interval. While the server B is occupied 18ut within the same range. Thus, the algorithm will choose the server A to be the first server for the truck X1 due to its lower workload in the interval. Therefore, X1 will reach the server A and wait for its service, that will occur in [32,52].

Therefore, the unnecessary waiting times are decreased for the truck X1, and the congestion in the servers is also decreased, thus increasing the equilibrium state of the facility.

The choice of this interval is to overcome problems like the one described above. By choosing the least occupied server in that interval and not only at the arrival time, it is possible for the algorithm to consider a more concise workload of the servers. As stated in that example, a server can be unoccupied at the arrival time but heavily occupied in the next units of time.

The goal is to consider the trucks that have entered the facility already. Choosing the sequence of servers by its workload, will allow the facility to be in a high level of equilibrium, tackling situations where some servers are overloaded, and others empty.

Therefore, the sequence of the servers will be computed following that premise. The algorithm will always choose the least occupied server in that interval. After defining the first server to go, the algorithm will compute the same proceeding but for the remaining required services. In that case, having always in consideration the update of the times, that is, the time of ending the previous service.

In the case there are two servers with the same minimum workload, the choice will fall in the nearest server, considering the travel time. The routes are given at the entrance, and as the trucks are following its given routes, other trucks may enter the plant and change the servers' workload. When two servers have the same minimum workload, by choosing the nearest one, the probability of their workload to be altered is lower because the truck will reach that server in a shorter period of time. Consequently, the sequence of the servers is computed by choosing always the nearest minimum occupied server in the above-mentioned interval. The algorithm will choose the servers, one by one, until all are chosen. This can be viewed as creating a precedence constraint between them (as in the TSPPC – see section 0), but, in this case, all the nodes will have a fixed position, defined by its workload.

It is now necessary to define how the algorithm will deal with the choice of the roads to reach the required servers. The choice of the roads will also affect the sequence of the nodes. When calculating the workload level of the servers in the range stated above, the servers' workload depends on the arrival time, and the route chosen to reach the servers will affect the arrival time.

Each road has an associated time cost. The time of traversing each road is proportional to the distance of that road. Using the Traffic Assignment Problem, and to address the problem of the congestion inside the plant, the time of traversing each road will also be dependent on the number of trucks that are already in that road. That is, when a truck enters a road in a specific time, the cost of traversing that road is increased if there are some trucks crossing the road. As the flows increase in a road, the average velocity tends to decrease, causing increased traversing times (see Chapter 3).

There are several methods to address the Traffic Assignment Problem. The incremental assignment procedure (see Chapter 3.3.5) is to assign fixed portions of traffic to the network, in steps. After assigning each portion of flow to the shortest route, the cost of that route is recalculated, following a link cost function. This link cost function will consider the volume assigned to that route so far.

Due to great unpredictability regarding the number of trucks that will arrive and the locations they require to visit, the incremental assignment allows to overcome these characteristics by assigning only one truck in each step. That is, considering that only one truck enters at a time in the plant and its route is calculated at that time, the portion to be assigned will only consider that truck.

When assigning a route to a truck, it is necessary to, as made in the servers, store all the times the truck will traverse each road that composes its route. Per example, assume a truck that must travel from a specific node A to B, following the route A-E-B. The road A-E has a time cost of 5ut and the E-A has a time cost of 3ut. Thus, assuming that the truck starts the route at the time of 10, the road A-E will have one more truck in [10,15] and the road E-B will have one more truck within [15,18].

Following the incremental assignment procedure, each portion of traffic (in this case, one truck) is assigned to the minimum time route. In this case, the time costs of each arc will be updated, for each time, considering all the trucks already inside the plant following its own routes. Thus, what may be the minimum time cost route connecting two nodes at a specified time, may or not be the minimum time cost at a different time, due to the trucks that are already inside the plant. The equation (4), developed by the Bureau of Public Roads, in 1964, will be used to calculate the updates in the travelling times, for each road. Therefore, every time a truck is associated to a route, the times each road is traversed are stored, for the costs to be properly updated when computing new minimum cost routes.

When computing the minimum cost routes, a modified dynamic Dijkstra's will be addressed. In this case, the difference is that, when comparing the time costs of the roads, the comparison can not be made directly. Whenever comparing the time costs of two roads, it is necessary to use the number of trucks that are in that roads, at the time of entering the roads. With this, the time costs are properly updated, using the link cost function stated above.

Consider, as stated earlier, the times of traversing each route are stored in intervals. When counting the number of trucks traversing each road, the count must contemplate the time the truck will enter in that road. With this, it is necessary to compare the time of entering the road, and all the previous stored intervals, thus updating the costs of the roads. Therefore, assume that the algorithm is calculating the number of trucks traversing a road in a specific time. If that time is not within any of the previous stored intervals for that road, means that there is no truck traversing that road, at that time. In that case, the time of traversing that road will be the free flow time. Thus, it is not necessary to decrement the number of trucks in the roads, because the number of trucks traversing a road will depend on all the previous intervals stored for each road.

It is important to clarify that waiting times at the servers will not be allowed. Sometimes, when comparing two different routes for one location, the algorithm will calculate the number of trucks that are in the roads at the times of traversing the roads, and, depending on that, the algorithm will choose the route according with the time cost in that instant. If it was allowed to wait some units of time before entering the roads, the resultant route could be other with a possible shorter time cost. However, if it was allowed for the vehicles to wait in the nodes, the congestion in the nodes would increase. This problem is even more highlighted because in some servers the space is limited for the trucks to be there waiting. Therefore, even if waiting in the nodes would represent a shorter time cost route, that is not allowed.

The algorithm that will compute the routes, considering the number of trucks in each road, is described in the next steps.

1. Create two sets of nodes. The temporary ones, and the permanent.
   C is the current node.
   Assign temporarily T(x)=infinity, for all x.
   T(x) is the current time of going to the node x.
   T(C) is the time cost of reaching C.

2. The current time is $t_0$. Calculate the number of trucks in each road, at the current time, connecting the current node and the neighbor nodes. Update the time costs of that roads, for the current time, using the link travel function (4).
   Find the node x, connecting the current node, with the smallest temporarily value of T(x).
   If there are no temporary nodes, then stops.
   Node x is labeled as permanent. Node x is now the current node. And the current time is updated for $t_0=t_0+T(x)$.

3. For each neighbor y of x, make the comparison:
   if $T(x)+t(x,y)<T(y)$, then T(y) is changed to $T(x)+t(x,y)$.
   To calculate t(x,y), the time of going from x to every neighbor y, it is necessary to update the time costs, but for the current fictious time. It is so necessary to calculate the number of trucks in each one of that roads, at the time of entering the roads, and to calculate the updated costs, using the function (4).

4. Return to Step 2.

Using this algorithm, the congestion in the roads is highly decreased due to the fact that, when the number of trucks traversing one road, for a specific time, becomes higher, the algorithm will choose other roads, with less trucks, thus diminishing both the congestion and the travel time for the entities (each client of the cement plant). Therefore, from this point on, when calculating a “minimum (or shortest) time route”, the route will follow the previous steps.

It is now possible to explain the overall procedure of the algorithm No.2, considering the sequence of the nodes and the choice of the roads. As stated earlier, whenever a truck arrives at a plant, it will proceed to its check-in. During this part, the truck driver will give the information regarding the required service locations and the service time in each service.

After this, the algorithm will calculate, considering all the trucks already in the facility following their own routes, the shortest time routes to reach every required location. At this point, the algorithm will calculate the workload of each one of the required service locations. It is important to remember that the workload level of each server depends on the arrival time at each server, and that depends on the route to reach the server. The minimum occupied server will be chosen to be the first server to visit. In the case there are two servers with the same minimum workload, the nearest server will be the first one to be visited.

The algorithm will update the time value, for the time the truck will exit the first chosen server. This is possible because the algorithm considers the time of traversing each road composing the computed route, and the service time of the truck in that server.

After that, it will calculate the minimum time routes, considering the updated times, of going from the current server to the remaining required servers. At the time of reaching that servers, the least occupied server will be chosen to be the next server to go. The algorithm will, always updating the times, making this calculations and choices until there are no required locations left. At this point, the algorithm will now calculate the minimum time route connecting the current server and the exit of the plant. When this happens, the overall route is given to the driver and the times of traversing each road and reaching/leaving each service location are stored. With this, it is possible for the system to have control of all the trucks already inside the plant, because all the already given routes are stored.

It is possible to understand that, as the overall route is given at the entrance, all the time updates made during the algorithm calculations, are made assuming fictitious future times. With all the times traversing each road and reaching/leaving in each server stored, it is possible to calculate the number of trucks that will be occupying each road, and each server, for a specified instant of time. Thus, the routes given now, are given considering that future events.

The overall procedure can also be observed in the next steps.

1. Create two sets of nodes, containing the visited and unvisited nodes, representing the required locations.
   Place all the nodes in the unvisited set.

2. Calculate the minimum time routes, for the current time, between the current node and all the unvisited locations.

3. Calculate the workload of each server in the interval [arrival time in the server, arrival time + service time in the server] for each one of the required places.

4. Choose to least occupied server to be the next server to visit. If there are more than one server with minimum workload, choose the nearest one, choosing the route with shortest time cost.
   Add that server to the visited set. Update the time value for the time of leaving that server (considering the time of the route to reach the server, the waiting time in the server, and service time in that server).

5. If the unvisited set is empty, then stops. Present the overall route to the truck and store all the times the truck will traverse each road of the route, and the interval of time it will be in each one of the required locations, considering the waiting time and the processing time in the server.
   Else, go to step 2.

To give a practical example of the overall algorithm, assume a truck X2, entering a facility at the time 5, and with three required locations, being them represented by the nodes C, D and E. The truck X2 requires a service time of 20ut in C, 15ut in D and 20ut in E. The algorithm will start computing its calculations. First, considering the current time (5), the algorithm will compute three shortest time routes (one for each required location), considering all the trucks inside the plant so far. Assume that a route r1 will lead X2 to C and takes 3ut, r2 leads X2 to D having a time cost of 4ut and r3 leads X2 to E and takes 3ut. The Figure 13 presents the schema for this example.

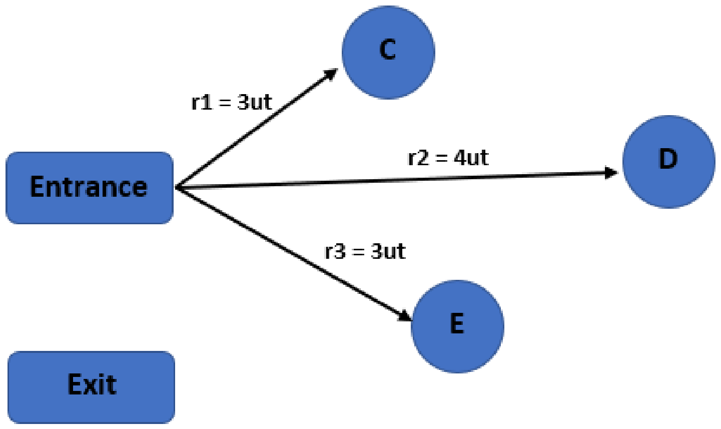


**Figure 13 - Minimum shortest routes for the truck X2.**

The algorithm will now calculate the workload of C, D and E, in 8ut (5+3), 9ut (5+4) and 8ut (5+3), respectively. Assume now that the least occupied server at that times is D, and the truck can be served when arrives at that server, which may not hold in other cases. Thus, the first server for X2 to follow is D, following the route r2. Now, the algorithm must update the time, fictitiously, for 24 (9+15), because X2 reaches D at 9, and has a service time of 15ut. In the Figure 14, it is possible to observe, marked in red, the choice of the first service location.

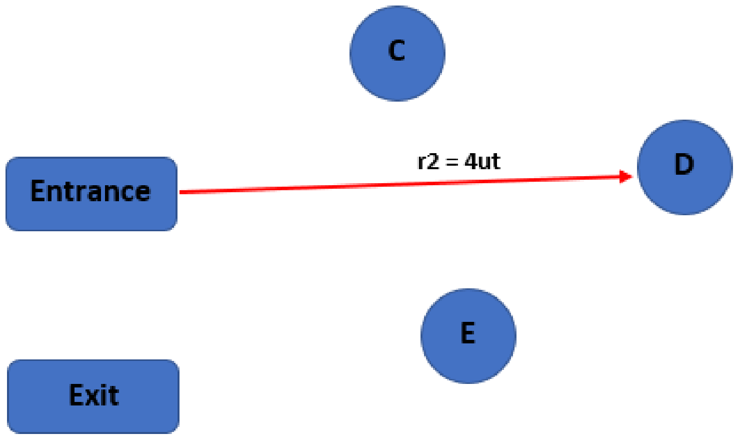


**Figure 14 - The first chosen route for the truck X2.**

This previous process will be repeated, but in this case the current fictitiously time is 24, and the routes to be computed are now between D and C and between D and E. The algorithm will repeat these processes till all the required places are "visited", with shortest time routes connecting them. At this point, the algorithm gives X2 the route for it to follow. In the Figure 15-Figure 17 it is possible to observe the sequence of the remaining steps by the algorithm to find the overall route containing all the service locations.

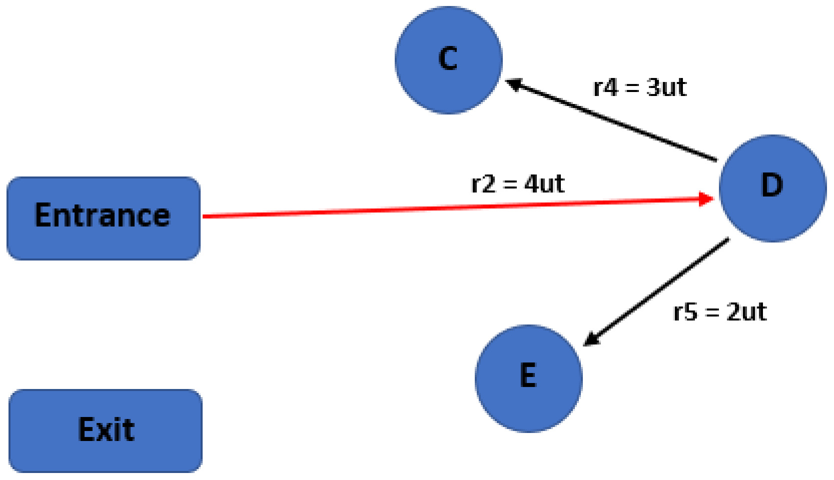


**Figure 15 - The routes for the truck X2 - 2nd iteration.**

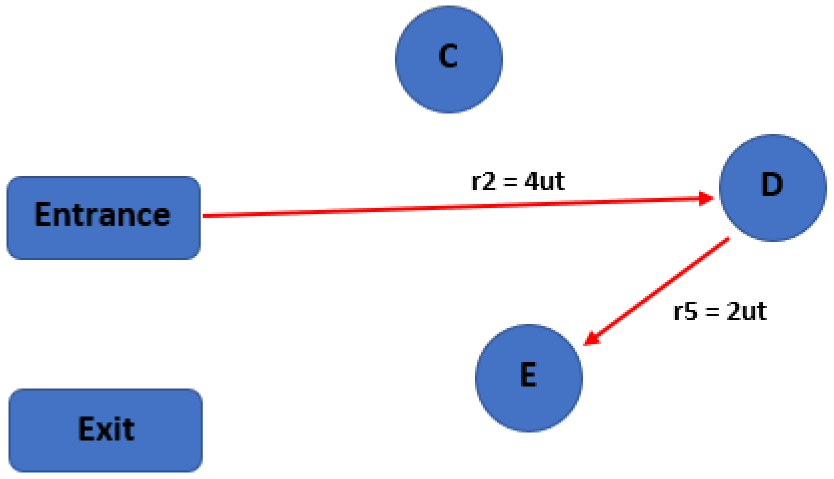


**Figure 16 - Second chosen route for the truck X2.**

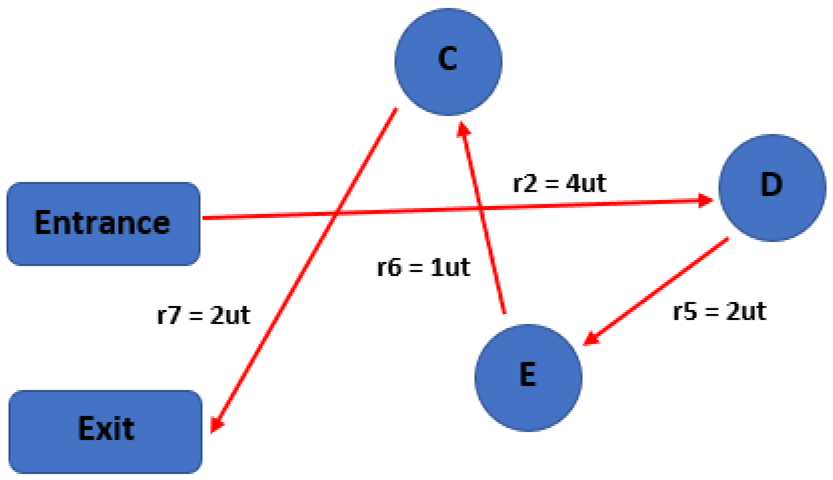


**Figure 17 - Overall route for the truck X2.**

At the time of giving the route to the driver, the algorithm will also store all the times the truck will pass in each road, and the intervals of time it will be in each one of the required locations. Thus, the system will have all the information considering all the trucks inside the plant for the next trucks to arrive.

In this case, if the routes connecting the servers were only composed by one road, which probably will not hold in other examples, the road connecting the entrance and D will have one more truck in [5,9]. The server D will have a truck within [9,24]. The road connecting the servers D and E, will have one more truck in [24,26]. The server E will have one more truck in [26,46]. The road connecting E and C will have one more truck in [46,47]. The server C have one more truck [47,67]. The road connecting C and the exit will have one more truck within [67,69]. Assuming there was only one road composing each route, the algorithm will, at the time of transmitting the route for the driver, store all the previously stated information.

This approach will also tackle the case when a truck has a fixed sequence for the service locations. In this case, the algorithm will work in a similar way, but it will not calculate the servers' sequence, because that choice is already predefined. Every time a truck arrives at a facility with a fixed sequence for the service locations, the algorithm will calculate the minimum time routes, but the next server to visit is the server in the sequence predefined by the driver. Besides this, the algorithm will also store all the times traversing the roads and servicing in each one of the servers.

It is important to remind that, as there are no updates in the routes, being all the calculations made at the time of each trucks' check-in, the servers' workload could change and the servers with less occupation now may or may not be in the same servers in the next units of time, due to the changing of the workload configuration. In a little example, assume that a truck X3 enters a plant in a specific time, and must be served on the servers A, B and C. Assume now that, considering all the trucks already inside the plant, the sequence of servers for X3 is B-C-A, being the routes connecting the servers the minimum time ones.

Consider now that, after X3 enters, several trucks arrive at the plant. These trucks have only one required place, and it is the server C. The algorithm will give the minimum time routes for that trucks to reach C, because there are no other required places for them to be guided. Therefore, when X3 reaches C, after being served in B, the server C will be more occupied than what have been previously calculated. This may result in a delayed service for the truck X3. In this particular situation, it would be preferable, both for the system and for the truck X3, for X3 to go first to the server A, if the workload in A is less than C at the time of reaching C.

The algorithm No.2 does not consider the updates on the routes, and so, after a truck entering the plant, there is no possible way to change the course of the truck, not addressing these particular problems. However, since the algorithm stores all the times of all the trucks since the beginning of the work, the workload equilibrium of the facility is expected to be largely increased. Even when the servers' workload changes drastically, the overall state of the factory is to be in a higher state of equilibrium, with less congestion, both in the service locations and in the roads, because each decision is made considering all the trucks that have entered the facility so far.

Therefore, this approach leads each truck to its required locations. Above that, it chooses the servers by its workload, thus tackling the problem of the congestion of the servers.  Besides this, it also decreases the congestion in the roads of the plant, avoiding the ones with great number of trucks.

## 5.3. Algorithm No.3 – The Equilibrium Approach (with updates)

The third approach is, in a matter of fact, a variation of the algorithm No.2. The approach of the algorithm No.2 considers that all the routes are given to the drivers at the time of the trucks' entrance. Therefore, it is not necessary for the trucks to be connected to the routing system when they are inside the plant. That approach, besides addressing the servers' workload and the roads' congestion, it does not consider the fact that it is possible for the workload of the servers to change after the trucks enter the plant, as in the example of the truck X3. Thus, what may be the best sequence of services and route (considering the workload configuration) now, may or not be the best over time. With this, it would be profitable for the trucks, and to the overall system, if it is possible to change the routes of the trucks already inside the plant.

The third approach will consider that each truck driver has an interface (possibly a mobile application or other related gadget) connecting it to the system. Consequently, each route can be updated in every moment and the driver is always connected to the route system. Besides all of this, and to minimize the number of updates, not requiring for the driver to be always "looking" if there are updates, this approach will consider that the updates will be given at specific times, as will be explained below.

When a truck arrives at a cement plant, and proceeds to its check-in, the algorithm will, as in the algorithm No.2, calculate the minimum time routes for each required location, considering all the trucks already inside the plant. That minimum time routes are calculated in the same way as in the algorithm No.2. After that, the algorithm will find the least occupied server at the time of reaching the servers, following the premise of the algorithm No.2, which means, the least occupied server in the above-mentioned interval (see algorithm No.2).

The route leading the truck to the least occupied server is chosen, and that server is the first service location for the truck to be served. After that, and contrary to what happens in the algorithm No.2, the algorithm No.3 will not continue calculating the sequence of the occupied servers, and the routes connecting all the servers. Instead, the algorithm No.3 will present the route for the first server to the truck driver, and it will store the sime of traversing each road and reaching/leaving the first service location. That same truck will follow the route for that server, and it will be serviced there. After that, the truck driver will require the new route (through its connection to the system). Thus, the algorithm will repeat the previous step but, in this case, considering all the remaining required locations, and the server on which the truck is currently located. The algorithm is going to calculate the new least occupied server and respective route for there, for the truck to follow. This procedure will continue until all the required services are reached, and so the algorithm will find the route for the truck to the exit of the plant.

Thus, while some trucks are traversing each road to reach its destinations, the trucks at the entrance or in the servers are requiring for the system to calculate what will be the next server for they to go. Therefore, the control of the facility is much more precise. With this, the facility will be in a higher level of equilibrium.

Thus, whenever a truck is on the entrance or finishes being served in some server, it requires to the system the next route to reach the next service location. The system must know which vehicle is requiring to know -by and identifier, per example- thus knowing the remaining required locations for that vehicle and its current location, to properly compute the minimum time route for the next server. As the requirements to the system will only be made at the entrance and in each server, instead of a mobile application or a gadget, other possibility is a fixed communication system placed in each service location. Thus, the truck driver will go, after loading or unloading its goods, to that machine, requiring the route to the next server (or to the exit).

When a truck driver requires to the system the route for him to follow next, the algorithm will follow the next steps.

1. Calculate the minimum time routes following the premise of the algorithm No.2, considering all the trucks inside the plant, between the current location of the truck, and all the required unvisited locations.
   If there are no more unvisited locations, calculate the minimum time route between the current location and the exit of the plant. Present that route for the truck driver and stops.

2. Calculate the workload of each server in the interval [arrival time in the server, arrival time + service time in the server] for each one of the required places.

3. Choose to least occupied server to be the next server to visit. If there are more than one server with the minimum workload, choose the nearest one, choosing the route with shortest time cost.
   Add that server to the visited set, removing it from the unvisited set.

4. Store the times of traversing each road and reching/leaving the service location.
   Present the route for the next service to the truck driver and stops.

This algorithm tackles the problems of the congestion of the roads and the servers, but not only with the trucks that have entered the factory until the entrance of a truck. This approach will consider all the trucks inside the plant each time there is a requirement by a truck driver, which happens at the entrance of a vehicle or in the end of each service.

Using the example given in the algorithm No.2, assume again the truck X2 arriving in the facility at the time 5. Its required locations are C, D and E. The truck X2 requires a service time of 20ut in C, 15ut in D and 20ut in E. The algorithm will start its calculations by compute all the minimum time routes, considering all the trucks already inside the plant, and the results of that routes are given in the Figure 13.

At this point, the algorithm will calculate the workload of C, D and E, at the times of 8ut (5+3), 9ut (5+4) and 8ut (5+3), respectively. The calculation of the servers' workload is made in the interval stated in the Step 2. Assume, as in the example of the algorithm No.2, that the least occupied server is D. The algorithm will, contrary to what happens in the previous approach, give the route for that server to the truck X2.

Besides this, the algorithm will store the times for the truck X2, since the entrance until its service in D. In this example, and again assuming the route connecting the entrance to the server D it is only composed by one road, that road will have one more truck in [5,9]. The server D will have one more truck within the range [9,24].

Therefore, the truck X2 will follow that route and arrives in D at the time of 9. As its processing time in D is 20ut, the truck X2 will end its service in D at the time of 24.

While this happens, it is possible for other trucks to arrive at the plant, and the algorithm will give them routes for the least occupied servers, following minimum time routes.

After being served in D, at the real time of 24, the truck driver will require its new route. The algorithm will now compute all the minimum time routes for the servers C and E. After that, the algorithm will calculate which one of them is the least occupied server. In this case, assume that new trucks have entered the plant while X2 traversed the route $r_2$, and now, at the time of 24, the least occupied server is C and not E, contrary to the same example of the algorithm No.2. Now the algorithm will present other route, $r_4$, for the truck X2 to follow and reach C. The algorithm will also store all the times of traversing the road $r_4$, and the interval of reaching/leaving C. In this case, the road connecting D and E will have one more truck in [24,27] and C will have one more truck within the range [27,47].

The truck driver will now follow its route for C, and after being serviced there, at the time of 47, it will require its new route. The algorithm will do the same as before, calculating the least occupied server and presenting the minimum time route for that server to the truck driver. Always storing the times of traversing each road, and reaching/leaving each service location, for other trucks entering now the plant or finishing its services to consider all the trucks and all the roads that are being traversed at each moment. When all the locations are visited, the algorithm will give the minimum time route for the truck to exit the plant, as observed in the Figure 18.

It is important to state, as already mentioned, that in this little example, the truck X2 was serviced at the time of reaching the server, which could not happen if the server was occupied already. If a server is the least occupied server, but it is occupied at the arrival time, the truck must wait for its turn for being served.

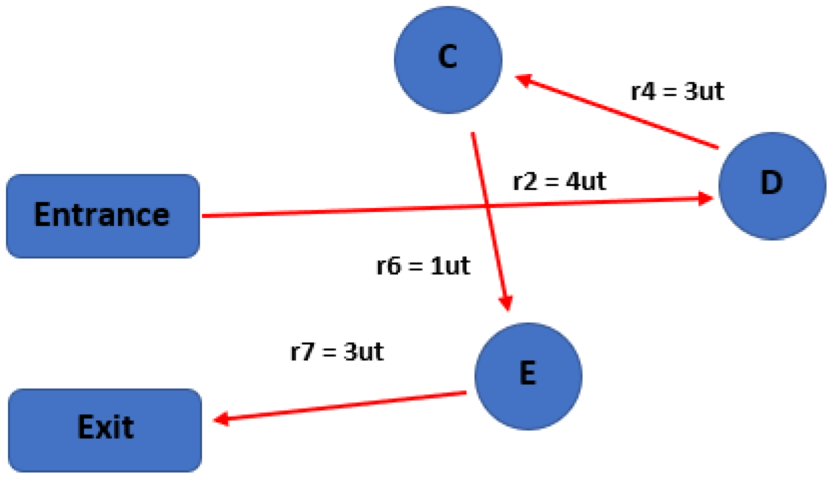


**Figure 18 - Overall Route for the truck X2 in the Algorithm No.3.**

This approach will, as in the other approaches, overcome the case when a truck has a predefined sequence for the service locations. In that case, this approach will calculate the minimum time route for the next server (following the predefined sequence), and present that route to the truck driver. After the truck being served in that location, the algorithm will compute the new route for the next server, always following the sequence until all the required locations are visited.

Therefore, the greatest difference between this algorithm and the algorithm No.2 is the fact that the algorithm No.2 makes all the calculations assuming future fictitious times, considering all the trucks that have entered the facility until the time of each trucks' arrival. This approach only considers future fictitious times to calculate what will be the next server to visit, and the minimum time route to reach that server.

# 6. Implementation: Tests and Results

## 6.1. Algorithm No. 1

The Dijkstra's algorithm was implemented using Java programming language. This algorithm can be applied to compute the minimum distance between a pair of nodes, or to all the pairs on the graph, depending if the truck has one or more services to visit, respectively.

If the truck has more than one required location to visit, a spread sheet was developed, containing the brute force algorithm. The algorithm in the spread sheet will compute, for a given number of locations, the minimum distance route, testing all the possible sequences of the required service locations. Thus, the shortest distance route is always found and given to the driver.

Consider again the facility given by the graph presented in the Figure 11. Assume that the server represented by the node A is the entrance and exit of the facility, and the remaining nodes represent the service locations. The algorithm must assign the entrance and exit nodes to be the beginning and end of each one of the computed routes. In this case, the node A is assigned to be the beginning and the end of each one of the possible routes.

Assume a given truck arriving at this facility, being B, C and E its required service locations. The answer given by the brute force algorithm is presented in the Table 3.

**Table 3 - Possible routes for a truck with required locations B, C and E - Algorithm No.1.**

| N=3 | B,C,E | | |
|---|---|---|---|
| B | C | E | 17 |
| B | E | C | 13 |
| C | B | E | 13 |
| C | E | B | 13 |
| E | C | B | 16 |
| E | B | C | 14 |

It is possible to observe all the feasible route solutions computed by the algorithm, as the minimum distance one, marked in a different color. In this case, there are more than one route with the same minimum cost.

Considering all the minimum routes between each pair of the nodes, the overall route, calculated by the algorithm No.1 is: A-C-**B-E-C-**A with the cost of 13.

Consider now a truck that needs to be served in the servers C and D, but in a predefined sequence. The truck must be served in C and only then, in D. For that case, the algorithm will compute the minimum distance route from A to C, from C to D, and then from D to A. Thus, the overall computed route is: A-**C**-B-**D**-B-A.

With the computed tests, it is possible to state that this approach is calculating everything as expected and as defined in the algorithm explanation, for each truck arriving at each cement facility. Besides this, the computation time of this algorithm is low, computing the routes in few seconds, as expected.

## 6.2. ALGORITHM NO. 2

The Algorithm No.2 was implemented using Java programming language. Due to the great extension of the resultant code, it will not be presented in the Appendix section.

As stated in the previous sections, each road has a distance, and an associated time. In this approach, the time of traversing a road it is not only proportional to the distance. The time of traversing a road is dependent on the number of trucks traversing it, at the moment of entering the road, thus being time dependent. Using the equation (4) there are some values that must be measured and determined.

The free flow time, $t_0$, represents the time of traversing each road, directly proportional to the distance of the road, and with no trucks traversing it.

The $x$ represents the flow for that road in that specific moment of time, which means, the number of trucks that are traversing the road at that moment. This constant may be different for the same road in different times, depending on the number of trucks that are, at that moment, traversing the road.

It is also important to define what will be the capacity of the link per unit time, $k$. This constant is fixed for a road. As the roads may not be so large inside the facility, and the trucks have big dimensions, a capacity per unit time of 9 is chosen. This choice can be easily modified and can be different for each road. However, in this implementation it is assumed to be equal for every road composing the plant.

The Bureau of Public Roads has also defined two model parameters, α and β. Typically, these values are 0.15 and 4, respectively. As it is possible to observe in the equation (4), these parameters are proportional to the sensibility of the equation. This means that, for the same *x* (the number of trucks in the road), the higher these values, the higher the time cost that road will have in that moment. Thus, and to consider again that the roads inside the factory are dealing with vehicles with big dimensions, the value of α is set to be 0.15, and the value of β is 5. With this, since β is in the exponent, the function will become more "sensible" to the number of trucks, originating routes with different roads, for a smaller number of trucks traversing it. It is important to mention again that these parameters can be easily modified if it is required to.

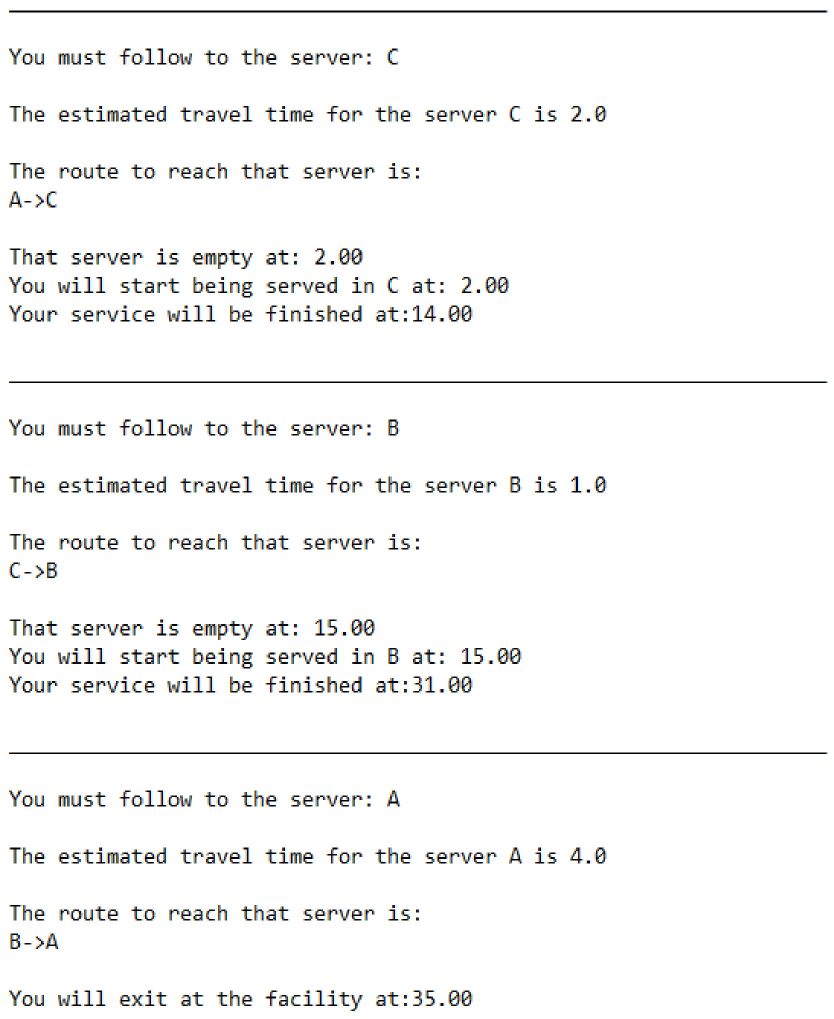
Thus, the average time cost, *t*, for a truck to traverse a road at a specific time is presented in the equation (5).

$$t = t_0 \cdot \left[1 + 0.15 \cdot \left(\frac{x}{9}\right)^5\right] \qquad (5)$$

The inputs of the developed program are, as explained in the algorithm explanation, the entrance time, the required service locations and the time the truck must spend in each service.

With the implementation addressed, it is possible to compute some tests. Assume the example where a cement facility is represented by the graph presented in the Figure 11.

Consider a truck T1, entering in that facility at the time 0, and having the servers B and C as its required locations. Assume that the required service time for this truck in B is 16ut, and in C is 12ut. In the Figure 19 it is possible to observe the response by the developed program.



```
You must follow to the server: C

The estimated travel time for the server C is 2.0

The route to reach that server is:
A->C

That server is empty at: 2.00
You will start being served in C at: 2.00
Your service will be finished at:14.00
```

```
You must follow to the server: B

The estimated travel time for the server B is 1.0

The route to reach that server is:
C->B

That server is empty at: 15.00
You will start being served in B at: 15.00
Your service will be finished at:31.00
```

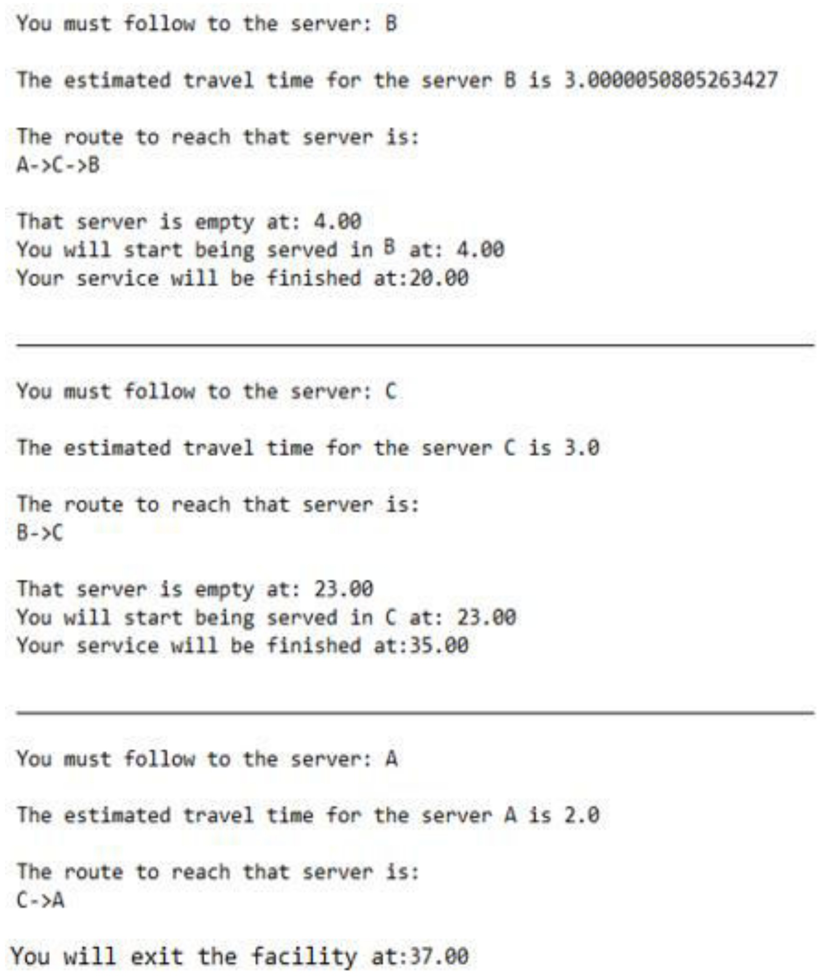
```
You must follow to the server: A

The estimated travel time for the server A is 4.0

The route to reach that server is:
B->A

You will exit at the facility at:35.00
```

**Figure 19 - Computed route for the truck T1.**

The truck T1 will first follow the route A-C. At the time of 2 and until the time of 12, the truck will be served in C. The server C is chosen to be the first server. This happens because the workload of B and C is equal and so the nearest server is the chosen to be the next. After this, that truck will follow the route C-B, being served in B in 15 and exiting that server at 31. The truck will exit the facility at 35.

In this example, the truck T1 will be served at the arrival time in each one of the required servers. This happens because T1 is the first truck entering the facility (at time 0), and each server is empty at the arrival times.

In the same example, after 1ut, consider another truck, T2, arriving at the same facility with the same required places, B and C. The truck T2 has the same service times of 16ut in B, and 12ut in C. Thus, the program will compute the route presented in the Figure 20.



```
You must follow to the server: B

The estimated travel time for the server B is 3.0000050805263427

The route to reach that server is:
A->C->B

That server is empty at: 4.00
You will start being served in B at: 4.00
Your service will be finished at:20.00

______________________________________________________________

You must follow to the server: C

The estimated travel time for the server C is 3.0

The route to reach that server is:
B->C

That server is empty at: 23.00
You will start being served in C at: 23.00
Your service will be finished at:35.00

______________________________________________________________

You must follow to the server: A

The estimated travel time for the server A is 2.0

The route to reach that server is:
C->A

You will exit the facility at:37.00
```

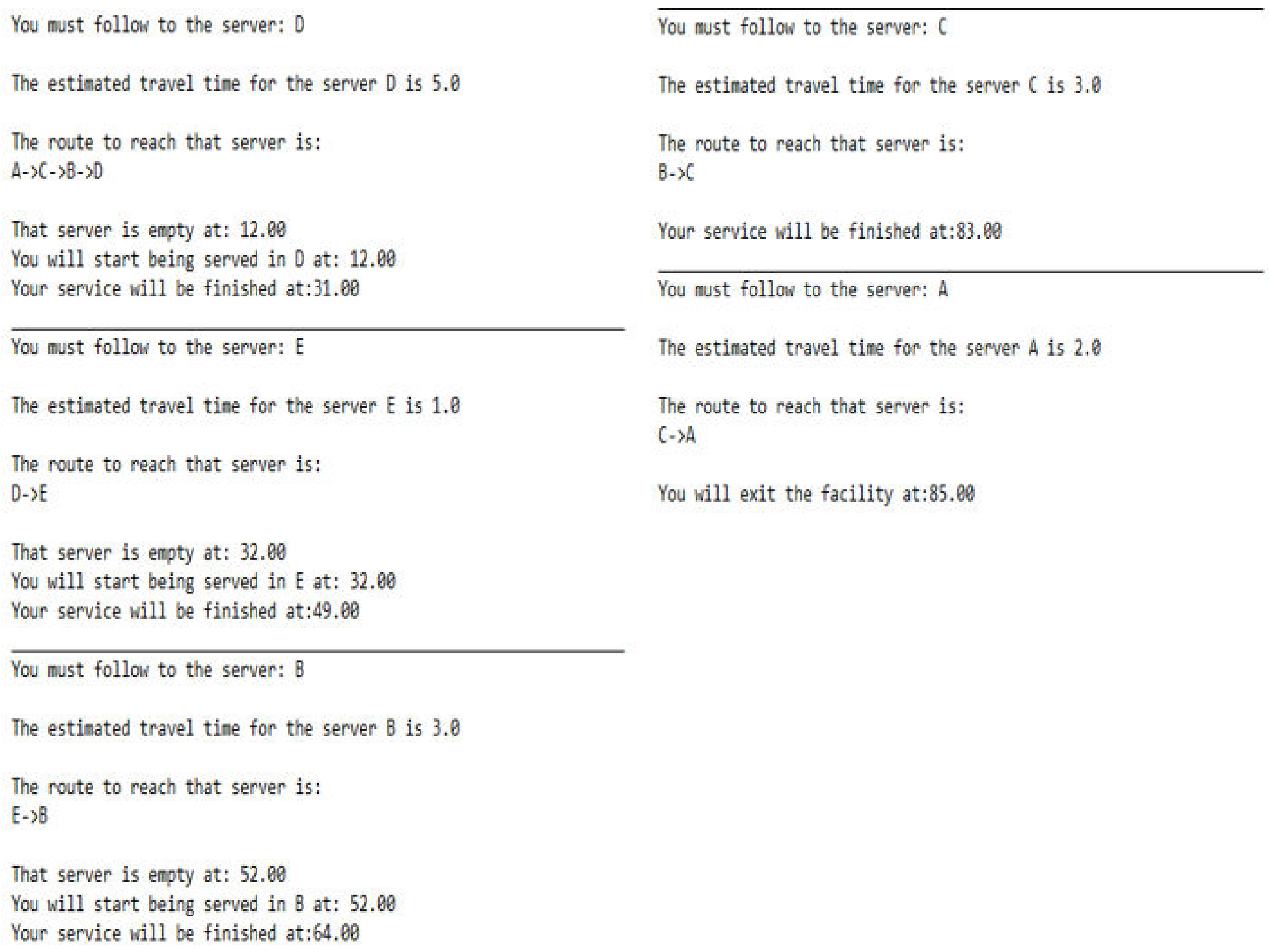
**Figure 20 - Computed route for the truck T2.**

In the computed route, it is possible to observe that, as this truck is entering at the time 1, at the times of reaching the required locations, the server C is occupied for one truck and B is empty for the whole time. Thus, this truck will be served first in B, and then in C.

One important aspect about this route is the traveling time for the server B. Instead of being exactly 3ut of time, it is slightly higher than that. This happens because T2 is entering the facility at the time 1, traversing the road A-C at the time [1,3], and T1 have entered the facility at 0 and is traversing the same road in [0,2]. Thus, due to the updates in the roads' costs, the cost of traversing that road is increased. It is also possible to observe that it is only slightly increased because there is only one truck traversing the road at that time. However, due to the exponential type of the function, with a higher number of trucks, the cost will reach greater values.

Assume now that, in the same facility, the server B is occupied in [3,15], and the server C is occupied in [11,20] by some trucks that already are inside the plant.

Consider that other truck T3, at the time 7, is entering the facility with the required servers B, C, D and E and with a required service time of 15ut in each one of the servers. The computed route for that truck is presented in the Figure 21.

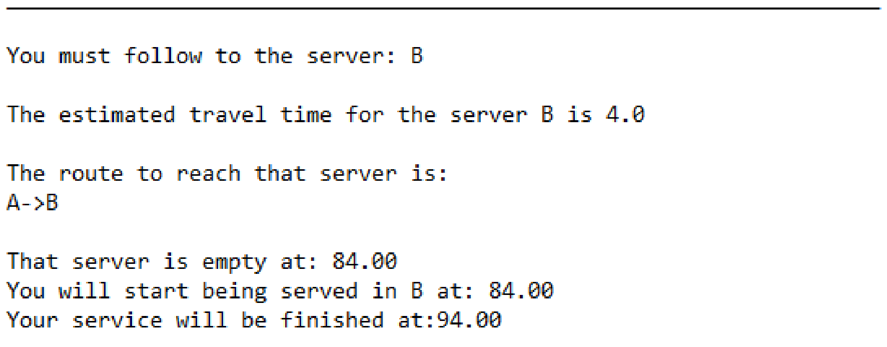


```
You must follow to the server: D

The estimated travel time for the server D is 5.0

The route to reach that server is:
A->C->B->D

That server is empty at: 12.00
You will start being served in D at: 12.00
Your service will be finished at:31.00
_______________________________________________
You must follow to the server: E

The estimated travel time for the server E is 1.0

The route to reach that server is:
D->E

That server is empty at: 32.00
You will start being served in E at: 32.00
Your service will be finished at:49.00
_______________________________________________
You must follow to the server: B

The estimated travel time for the server B is 3.0

The route to reach that server is:
E->B

That server is empty at: 52.00
You will start being served in B at: 52.00
Your service will be finished at:64.00
_______________________________________________
You must follow to the server: C

The estimated travel time for the server C is 3.0

The route to reach that server is:
B->C

Your service will be finished at:83.00
_______________________________________________
You must follow to the server: A

The estimated travel time for the server A is 2.0

The route to reach that server is:
C->A

You will exit the facility at:85.00
```

**Figure 21 - Computed route for the truck T3.**

The first chosen server was the server D. This happens because the time for reaching B is 3ut and for reaching C is 2ut. At the time 10 (7+3), the server B is occupied. At the time of 9 (7+2) the server C is unoccupied but will be occupied at the time 11. Choosing this server will provoke a delay in the truck that is already scheduled for that time. As the servers D and E have no trucks scheduled and are empty, the algorithm has chosen the server D to be the first and E to be the second. Thus, when the truck arrives in each one of the servers (including B and C), it will not delay any already scheduled truck and will not have to wait any time to be served.

In the previous examples the trucks were always served at the time of reaching the servers, not having to wait for its services. However, as illustrated in the Figure 22, there are cases when this may not happen. The Figure 22 shows part of the route for a truck T4, entering the facility at the time of 10, and having the server B as its required location. In this case, the truck must wait for its service, due to the already scheduled trucks.



```
You must follow to the server: B

The estimated travel time for the server B is 4.0

The route to reach that server is:
A->B

That server is empty at: 84.00
You will start being served in B at: 84.00
Your service will be finished at:94.00
```

**Figure 22 - Example of a truck having to wait for its service.**

The developed program has also the functionality which allows the driver to have already a predefined sequence for the service locations, working as described in the algorithm explanation.

Besides all the printed information in each route, it is possible to know some additional informative data, such as the time of reaching the servers and the time of traversing each one of the roads for each truck, if it is required to.

Therefore, the algorithm No.2 is working as explained in the Chapter 5.2. Besides that, the algorithm computes the routes in few seconds (1-2 seconds), thus having a short computation time for a real application.

The implementation of the algorithm No.3 is not presented due to its similarity with the algorithm No.2. With this, the algorithm No.3 has a short computation time, as happens with the algorithm No.2.

## 6.3. Simulation: Algorithms Comparison

So far, the algorithms were developed and implemented. Thus, it is now necessary to test these algorithms with several trucks as in a real-life scenario would happen.

Through simulation, it is possible to test and analyze different settings and potential impacts in the productive systems. This technique is done virtually, thus allowing to test different scenarios and overcome difficulties that a first real implementation could bring [120]. A simulation software (the *Simio Simulation Software*) will simulate a real work day for different specified sets of trucks, entering the same facility.

In one case, each truck entering the facility will follow a route computed by the algorithm No.1 and some results will be measured. In the other case, the algorithm No.2 will calculate the routes for the same set of trucks and the same results will be measured. With this, some of the results will be presented and discussed, and the remaining ones will be presented in the Appendix section.

Using the facility represented by the graph in the Figure 11, consider the entrance and the exit of the facility to be again represented by the node A, while the remaining nodes represent the service locations.

In the Figure 23, the facility is represented in the *Simio* environment. The node A is represented by two servers, being one for the entrance and the other for the exit of the plant.

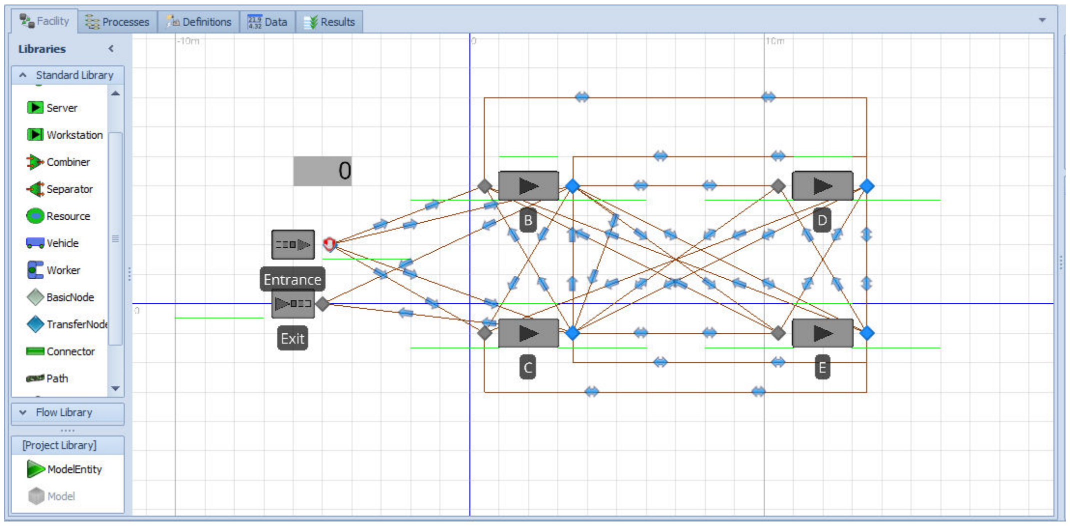


**Figure 23 - Representation of the cement plant in *Simio* software.**

Other important aspect about the modeling is the one considering the paths (roads) between each pair of servers. In *Simio* software, it is necessary to create paths connecting the entrance/exit of the servers and the entrance/exit of the other servers. Thus, it is possible for a truck to pass through the servers without being serviced there. In a little example, assume that a truck needs to reach the server E, following the route: A-C-B-E. In this case, this truck will only be served in E, but it will pass through the servers B and C. Consequently, the truck needs to pass through those servers without being serviced there, and it is so necessary to create those paths for the simulation to be properly executed. Moreover, each path in the simulation is a time path. This means that each truck will spend a time cost to traversing the paths. In this case, and to simplify, the time of traversing each road will be equal, in number, to the distance of the path.

The first set to be simulated, composed by 5 trucks, is presented in the Table 4– Appendix section. Each truck has four required locations, which, in this case, means that each truck requires to visit all the servers in the plant. In the same table, it is also possible to observe the entrance time of each one of the trucks. The service time for each one of the required places is presented in the Table 5 – Appendix section. In this example, each one of the trucks will have the same required service time in each location.

In the algorithm No.1 scenario, the simulation occurred in a total time of 141ut. The simulation time represents the time the last truck exited the plant after 141 units of time, meaning that the facility was able to serve that trucks in that amount of time. Usually, in literature concerning scheduling and job sequencing, this time is also named "Makespan". The maximum time that a truck stayed inside the facility was 137ut. In this simulation, the entities (trucks) had an average time in the system of 101ut.

In the simulation of the algorithm No.2 for the same set, the results were different. The simulation occurred in the total time of 118ut. The average time in the system was 94ut and the maximum time that a truck stayed inside the facility was 114ut. It is possible to observe all the collected results of these two simulations in the Appendix section –Table 6 and Table 7.

The next sets of trucks will be fully randomized, both in the number of required services as in which services each truck must be served. The service times in each server will also be random and will be fixed for all trucks in the same set, as in the previous simulation. Trucks with the predefined sequences will not be allowed, being all the routes entirely calculated by the two algorithms. The first one is composed by 15 trucks. In the Table 8 and Table 9 – Appendix section, it is presented the arriving time, the required service(s) locations and the service time in each server, for each truck.

In the case of the algorithm No.1, the simulation occurred in 242ut. The average time that a truck stayed inside the facility was 145ut, approximately, and the maximum time that a truck spent in the system was 225ut. In the algorithm No.2, the simulation took 221ut, the maximum time that a truck stayed inside the plant is 213ut and the average time that a truck stayed inside the facility was 144ut, approximately. The remaining collected data of these simulations is also presented in the Appendix section (Table 10 and Table 11).

The next set is composed by 20 trucks and is presented in the Table 12 – Appendix section. In the Table 13- Appendix section, it is also presented the service times of each truck in the service locations. As in the previous examples, the service time in each server will be the same, independently of the truck.

The simulation using the algorithm No.1 occurred in 298ut. The maximum time that a truck stayed inside the facility was 291ut and the average time inside the facility was 169ut. The simulation of the same set, but now using the algorithm No.2, took 292ut. The maximum time that a truck stayed inside the plant was 282ut and the average time was 162ut. As in the previous simulations, the remaining collected data of these simulations is presented in the Appendix section (Table 14 and Table 15).

Consider now other facility represented by the graph in the Figure 24. Assume that the entrance of the facility is represented by the node 'A' and the exit is represented by the node marked as 'G', being the service locations represented by the remaining nodes.

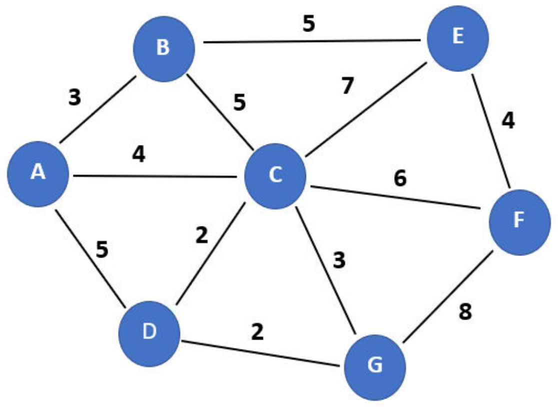


**Figure 24 – Cement facility graph.**

The representation of this facility in the *Simio* environment can be observed in the Figure 25. It is possible to observe the node A and G representing the entrance and the exit of the facility.

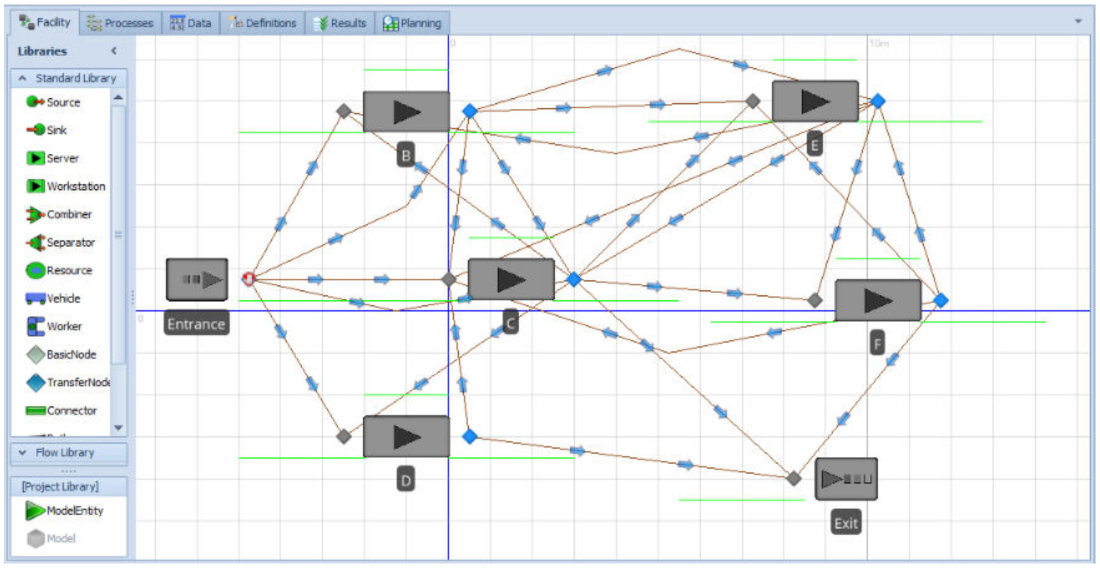


**Figure 25 - *Simio* representation of the facility.**

The first set to be simulated in this facility is presented in the Table 16– Appendix section and it is composed by 10 trucks. As in the previous examples, the service time of each truck in the servers will be independent on the truck, as it is possible to observe in the Table 17- Appendix section. Using the algorithm No.1, the simulation took 612ut and the maximum time that a truck stayed inside the plant was 603ut. On the other hand, the simulation of the algorithm No.2 occurred in 491ut and the maximum time a truck stayed inside the plant was 483ut. The results of these simulations are present in the Appendix section- Table 18 and Table 19.

Consider now the set of trucks presented in the Table 20- Appendix section. This set is composed by 16 trucks and the service time each truck will take in each server is presented in the Table 21– Appendix section. The simulation of the algorithm No.1 occurred in 839ut, and the maximum time a truck stayed inside the plant was 821ut. The simulation of the algorithm No.2 took 768ut and the maximum time a truck stayed inside the plant was 763ut. As in the previous simulations, the remaining collected data is present in the Table 22 and Table 23– Appendix section.

The next set to be simulated, composed by 30 trucks, is presented in the Table 24- Appendix section. The service times of each one of the service locations are presented in the Table 25- Appendix section. The simulation of the algorithm No.1 occurred in 1731ut, corresponding to the time the facility was able to serve the 30 trucks. The maximum time a truck stayed inside the facility was 1699ut. In the simulation of the algorithm No.2, the facility was able to serve the same set of trucks in 1687ut. In this simulation, the maximum time a truck stayed inside of the facility, was 1682ut. The remaining results of these two simulations are presented in the Table 26 and Table 27- Appendix section.

## 6.4. DISCUSSION

The results of the presented simulations show that, in general, the algorithm No.2 had better results than the algorithm No.1. The simulated sets took always more time in the case of the algorithm No.1 when compared with the algorithm No.2. This means that, for the same simulated facilities in the same conditions, the sets of trucks were served in shorter periods of time, when the routes have been computed by the algorithm No.2.

In the computed simulations, it was possible to notice that, although some trucks may have shorter times inside the facility using the algorithm No.1, some other trucks have very higher times using this algorithm. This happens due to the fact that the algorithm No.1 does not choose the routes considering the servers' workload, while the algorithm No.2 does. Besides that, the algorithm No.2 chooses the roads to reach the servers by considering the number of trucks traversing each road, at each time. Therefore, using the algorithm No.2, not only the facilities are able to serve the trucks in shorter periods of time, as it is possible for the congestion in the roads and in the servers to be decreased.

These results are somehow expected due to the fact that the algorithm No.2 does not give always the same route for trucks requiring the same locations. By choosing always the least occupied server at the time of reaching the servers and by computing different routes, avoiding the congested ones, the facility reaches a higher level of equilibrium. Usually, when computing a system that is static (always giving the same route, in the case of the algorithm No.1, the minimum distance one), there will be one (or several) road(s) and server(s) that will represent the so-called bottleneck(s). The algorithm No.2 tries to equilibrate the occupation both in the servers and in the roads, not overloading any of them, thus eliminating these bottlenecks or, at least, mitigating its effects. Besides the results of the simulations, it was possible to observe that, in the simulations using the algorithm No.2, the trucks were much more dispersed inside the facilities, decreasing the queues in the servers, per example. Also, it was possible to observe that, in some cases, the algorithm No.2 has chosen some roads that in the simulations of the Algorithm No.1 had not been used at all. In the roads case, as the sets have a limited and relatively lower number of trucks, the division may not be so present. However, in a day to day of a factory, with hundreds of trucks, the roads would become much more congested. In that case, the algorithm No.2 will provoke a much more highlighted division of the trucks in the roads.

With all stated so far, it is possible to conclude that these two approaches have better results when compared with the lack of assistance drivers have in cement plants. In both approaches the trucks are guided to the required locations. However, the algorithm No.2 seems to be a most robust solution, because, although in some cases the trucks are not following the minimum distance routes, they are following a route leading them to servers with least workload levels, minimizing its unnecessary waiting times. Besides this, the roads for that servers are chosen, always considering the number of trucks traversing them, at each time, thus decreasing the congestion levels inside the plant, both in the roads as in the service locations. With this, a better journey for the truck is provided, also improving the organization levels and equilibrium of the facility.

Although not being simulated in this work, the results of the Algorithm No.3 are expected to be even better, or in the worst case, equal, when compared with the algorithm No.2. This prediction relies on the fact that the algorithm No.3 computes the routes the same way the algorithm No.2 does. However, as this algorithm gives always the route for the next server, it encompasses possible modifications in the occupation of the servers and in the roads after entering the facility. If the servers' workload does not change during the tours, the computed routes of the two algorithms will be the same.

# 7. Parking Management

The algorithms No.2 and No.3 tackle the congestion in the facilities, both in the roads and in the service locations. However, as hundreds of trucks arrive at cement facilities every day, the trucks may not enter the facilities at the arrival time, waiting for their turn in the parking lot. In the simulations presented in the Chapter 6.3, the trucks have entered the facility in FIFO order, that is, the first trucks reaching the parking lot of the facility and proceeding to the check-in, are the first ones to enter. This may not be the best solution, both for the truck itself and for the facility. With this, a parking management is also necessary to study.

Next, there are several thought policies for the entrance management in the facility:

1. FIFO: The first trucks arriving at the facility, are the first ones to enter, and thus consequently.
2. Shortest Processing Time (SPT): The trucks with the least service times in the required servers are the first entering the facility, and thus consequently.
3. Minimum number of required locations: The first trucks entering the facility are the ones with the minimum number of required locations, and thus consequently.
4. Minimum exiting time: Considering all the times inside the facility (waiting times + service times + roads' traversing times), the truck with the least required time inside the plant, will be the first truck to enter. That rule is applied to the remaining trucks, in the same way. The exiting time for a truck it also depends on the chosen route for it to follow. Thus, to apply this policy, it is necessary to implement another algorithm to compute the route for the truck inside the plant.
5. Least workload levels: The first truck entering the facility is the truck that requires the servers with least workload levels, at that moment, and thus consequently.
6. Other related policies.

When implementing one of these policies, an aspect that it is necessary to have in mind is that, as the trucks are always arriving at the facility, the choice of the next trucks to enter can not consider all the trucks in the park every time. If all the trucks in the park are always considered when the choice is made, some trucks may greatly delay its entrance, or worse, may never be chosen to be the next to enter. To overcome this issue, regardless the chosen entrance rule, the entrance system must include also a FIFO order. This means that the entrance policy must be applied to batches of trucks in the park. The batches will be serviced in FIFO rule (the first group of trucks arriving at the facility, is the first group to be processed), and the order of the batches will not be changed. The order of the trucks composing the batches will be changed respecting the chosen entrance rule. With this, a truck can be delayed at the entrance if it is not advantageous for it and for the system, but it will not be delayed more than a fixed number of trucks.

With all stated so far, an algorithm for the entrance management was designed. This algorithm will consider batches of 5 trucks and will sort the trucks composing the batches, following the policy number 4. If the next truck entering the facility is the truck requiring the least time inside the plant, the congestion both in the servers and roads is expected to be decreased. Besides this, it is expected that the trucks will wait shorter times for their services. This because the time inside the plant includes all the waiting times, service times and routing times. Thus, if the next truck to enter is the one with least required time inside the plant, it is assumable that or its service time is quicker and/or the required service locations for that truck are with lower workload levels. Thus, by using this entrance rule it is expected that the trucks with the required service locations having higher workload levels are delayed at the parking lot instead of entering the facility, not contributing for the congestion inside the plant.

Therefore, the algorithm starts by creating a batch of 5 trucks, composed by the trucks that have arrived earlier at the facility. After that, a variation of the Algorithm No.2 will calculate the routes for each truck composing the batch. This variation will proceed exactly as the algorithm No.2. However, these calculations are made in an auxiliary way, aiming to know the time each one of the trucks would leave the plant, if they followed the computed routes. By calculating the routes with the Algorithm No.2, the problem of the congestion inside the plant is also tackled, as observed in the previous sections. Thus, the truck with the minimum required time inside the facility is chosen to be the first truck to enter. For the one with the minimum exiting time, the Algorithm No.2 will compute its route, but this time for that truck to follow. Besides this, the algorithm will store the times traversing each road and reaching/leaving the required service locations. After that, the algorithm will proceed in the same way, but now, for the remaining trucks composing the batch. The algorithm only ends when all the trucks composing the batch are inside the facility. When this happens, the algorithm will compute the new routes but now for a new batch, containing the next 5 trucks that have arrived in FIFO order.

The algorithm can also be observed in the next steps.

1. Create a batch composed by 5 trucks. These trucks are the 5 that arrived earlier at the facility.
2. There are more trucks in the batch? If there are not, Stops.
   Calculate, in an auxiliary way and for each one of the trucks in the batch, the routes for each truck using the Algorithm No.2 and measure the time each truck will exit the facility if they follow that route.
3. Choose the truck with the minimum required time inside the facility (the minimum exiting time) and present the route for that truck driver, storing all the times the truck will reach/leave the servers, and will traverse each road.
4. Remove that truck for the batch.
   Go to Step 2.

This algorithm was developed, aiming to be an extension of the Algorithm No.2, considering also an entrance management. With this, it is expected for the facility to reach a higher equilibrium level when compared with the algorithm No.2, and, besides that, for the truck driver to have a better journey, with shortest waiting times in the servers, while inside the facility.

Using the implementation of the Algorithm No.2, and by computing some modifications, the designed algorithm englobing the parking management was also implemented using Java programing language. This implemented algorithm has the information of the trucks that are currently in the parking lot as input, thus creating the batch and computing the routes for each truck composing the batch, as already explained.

Consider a batch of trucks, numbered from one to five, following the arrival order. Thus, the truck number one was the first truck arriving at the facility, and the fifth the last truck of that batch. The algorithm will present the entrance order in the facility for that batch. In this example, the computed entrance order for that batch is: 3-2-4-1-5. It is possible to observe that the trucks are sorted and will not enter the facility in FIFO order. Besides the entrance order, the routes for each truck are also presented, in the same format as observed in the tests of the Chapter 6.2.

This algorithm was also simulated using the *Simio* simulation software, as presented in the Chapter 6.3. The results of the entrance management algorithm will be compared with the already presented results of the algorithm No.2, for the same sets of trucks. Therefore, the entrance time will be the same, but the entrance order and the routes will be different, depending on the algorithm.

All the simulations were made using the facility presented in the Figure 24. The first set to be simulated is the one presented in the Table 16- Appendix section. The simulation time using the entrance management algorithm occurred in 461ut while the simulation of the same set using the algorithm No.2 took 491ut. As already stated, the entrance management algorithm aims to decrease the waiting times in the servers. Thus, the average waiting time in the servers using the algorithm No.2 was 42ut, while using the entrance management algorithm was 41ut. The simulation results for this set, using the entrance management algorithm can be observed in the Table 28-Appendix Section.

Using the set presented in the Table 20- Appendix section, the simulation using the entrance management algorithm took 742ut, while the simulation of the algorithm No.2 occurred in 768ut, as already observed. The average waiting time in the servers was also decreased by using the entrance management algorithm, being 73ut, while in the algorithm No.2 simulation was 82ut. The simulation data of the entrance management algorithm for this set is presented in the Table 29- Appendix section.

The simulation of the set presented in the Table 24- Appendix section, occurred in 1697ut using the entrance management algorithm, while using the algorithm No.2 took 1687ut. The average waiting time in the servers was 226ut using the entrance management algorithm, while using the algorithm No.2 was 247ut. The collected simulation data of the entrance management algorithm for this set is presented in the Table 30- Appendix section.

The simulation results of the developed entrance management algorithm are promising for a real implementation. Although in one of the simulated sets the time the facility has served all the trucks was higher using this algorithm, the simulation time has decreased for the remaining simulated sets. The average waiting times in the servers has decreased in all the simulations, as previously expected, thus proving the lower servers' workload levels and the better journey provided for the clients.

Besides this, by using the developed algorithm for the entrance management, the congestion inside the plant is decreased even further when compared with situations where "only" a route system is used. This situation is even more emphasized because the developed algorithm for the entrance management uses the algorithm No.2 as one of its steps, which tackles the congestion by itself, as already illustrated.

# 8. Conclusion and Future Research

One of the goals of this dissertation was to understand the supply chain of the cement industry and to perceive how the plants are dealing with the trucks management inside the plants. Hereupon, with the lack of assistance drivers have when they enter the facilities, it was imperative to create a routing system, to improve the SCM of the cement industry. A literature review about the most famous static and dynamic routing problems was made aiming to tackle the management of the trucks inside the plants. It was possible to notice the differences between the static and dynamic routing problems, regarding their purposes, implementations, features, and others.

With this literature research, it was possible to develop and implement three different algorithms to tackle the problem of the trucks' management. The computation time was low in all the implementations. This is an important achievement, since in the current digitalization era decisions must be made quickly. The algorithm No.1 guides the trucks to the required locations through the minimum distance route and serves as a comparison term for the other two. The algorithms No.2 and No.3, named equilibrium approaches, are the main contribution of this dissertation. These two dynamic algorithms, being one a variation of the other, consider not only the travelled distance, but also the servers' workload and the roads' congestion to compute the routes.

Using a simulation software, it was possible to test the algorithms, with sets of trucks, as in a real-life scenario would happen. The simulation compared, for the same sets of trucks and for two different facilities, the algorithm No.1 and No.2. The algorithm No.3 is a variation of the second one, and it was not simulated. The results of the algorithm No.2 were better than the algorithm No.1, in all the simulated tests and facilities, reducing the unnecessary waiting times, thus making the facility serve the trucks in shorter periods of time. The results of the algorithms No.2 shown the equilibrium of the facilities and the service quality for the clients was highly increased, as it is suggested and evidenced by the computed tests and simulations. Besides this, it was possible to notice that the trucks were much more dispersed inside the facilities in the simulation of the algorithm No.2, when compared with the algorithm No.1.

With this, the algorithm No.2 is very promising for a real implementation. The algorithm No.3, although not simulated, due to its characteristics, is expected to be even better or, at least equal, than the results of the algorithm No.2.

Concerning now the parking management, several policies for the entrance were also studied and an algorithm addressing this problem was developed and implemented. This algorithm allows an entrance management, different from using a FIFO rule, as it is currently used. This management aims to decrease even further the wasted times inside the facility, already reduced by the developed routing algorithms. Using the same simulation software, the developed entrance management algorithm was compared with the algorithm No.2. The simulation results of the entrance management algorithm are also promising, reducing the average waiting times in the servers in all the simulated sets.

As future work, there are several important topics whose investigation must continue. The results of the algorithm No.2 are promising, but it is necessary to test and simulate the algorithm using larger instances, different facilities, and different service times for each truck, for the results to be even more reliable. Yet, as the sets were randomized, even with instances containing hundreds of trucks and different facilities, it is expected that the results would be even better when compared with the algorithm No.1. This because with a greater number of trucks, the effects of the bottlenecks will be even more accentuated. The algorithm No.3 must also be simulated, in the same sets and facilities as the others, aiming to prove that its results would be better, or in the worst case, the same, as the algorithm No.2. If the results continue to be reliable and promising, a real application of the algorithm No.2 or No.3 should be considered and addressed.

The study of further applications for the algorithm No.2 and No.3 must also be considered. Applications in supermarkets (or related) that have hundreds of entities arriving each day, each one with required locations, having queues in that locations, and routes connecting each one of the locations, are examples of the similarity of these processes. One other example of other possible application is also in industry, but in the job sequencing machines problem. In these facilities, each entity must be served in one or several machines before being ready to be dispatched. Each entity, depending on its characteristics, must follow a route inside the facility, traveling through the required machines. Besides this, each entity can require a different service time in each machine.

The developed algorithm for the entrance management of the trucks must also be tested and simulated, with larger and different instances, thus proving its reliability for a further real application. Besides that, this algorithm uses one of the several suggested entrance rules. Other algorithms using different entrance rules should also be implemented and simulated as a comparison term. With this, it would be possible to confirm if the chosen policy is the one that gives the best results.

# BIBLIOGRAPHY

[1] U. D. Project, "The Logistics Handbook A Practical Guide for the Supply Chain Management of Health Commodities," 2011.

[2] A. Rushton, P. Croucher, and P. Baker, "The handbook of logistics and distribution management," *Proj. Manag. J.*, 2006.

[3] Y. Tseng, W. L. Yue, and M. A. P. Taylor, "The role of transportation in logistics chain," *East. Asia Soc. Transp. Stud.*, 2005.

[4] L. Barreto, A. Amaral, and T. Pereira, "Industry 4.0 implications in logistics: an overview," *Procedia Manuf.*, 2017.

[5] PwC, "Industry 4.0 – Opportunities and Challenges of the Industrial Internet," no. Industry 4.0, p. 52, 2014.

[6] "UH4SP - Unified Hub for Smart Plants." [Online]. Available: http://uh4sp.com/. [Accessed: 08-Jul-2018].

[7] Cachapuz, "Parte B ( Anexo Técnico ) Sistema De Incentivos À Investigação E Desenvolvimento Tecnológico ( Si I & Dt )," pp. 1–75, 2015.

[8] M. Fisher, "Vehicle routing," *Handbooks Oper. Res. Manag. Sci.*, vol. 8, no. C, pp. 1–33, 1995.

[9] P. Toth and D. Vigo, *The vehicle routing problem*, vol. 9. 2002.

[10] G. Laporte, "Routing problems : A bibliography Routing problems : A bibliography," vol. 61, no. JANUARY 1995, pp. 227–262, 2016.

[11] G. Laporte, "What you should know about the vehicle routing problem," *Nav. Res. Logist.*, vol. 54, no. 8, pp. 811–819, 2007.

[12] B. Eksioglu, A. Volkan, and A. Reisman, "The vehicle routing problem : A taxonomic review," *Comput. Ind. Eng.*, 2009.

[13] F. Ferrucci, *Pro-active Dynamic Vehicle Routing*. 2013.

[14] R. Ahuja, T. Magnanti, and J. Orlin, "Network Flows: Theory, Algorithms, and Applications," p. 846, 1993.

[15] A. Schrijver, "On the History of the Shortest Path Problem," *Doc. Math. · Extra Vol. ISMP*, vol. I, pp. 155–167, 2012.

[16] Z. Fuhao and L. Jiping, "An algorithm of shortest path based on Dijkstra for huge data," *6th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2009*, vol. 4, pp. 244–247, 2009.

[17] A. K. Nemani and R. K. Ahuja, "Shortest Path Problem Algorithms," *Wiley Encycl. Oper. Res. Manag. Sci.*, pp. 1–13, 2010.

[18] J. Holmgren, "Efficient Updating Shortest Path Calculations for Traffic Assignment," 2004.

[19] F. B. Zhan and C. E. Noon, "A Comparison Between Label-Setting and Label-Correcting Algorithms for Computing One-to-One Shortest Paths," *Inf. Syst.*, vol. 4, no. 2, pp. 1–11, 2000.

[20] E. W. Dijkstra, "A Note on T w o Problems in Connexion with Graphs," vol. 271, no. 1, pp. 269–271, 1959.

[21] R. K. Ahuja, K. Mehlhorn, J. Orlin, and R. E. Tarjan, "Faster algorithms for the shortest path problem," *J. ACM*, vol. 37, no. 2, pp. 213–223, 1990.

[22] H. Reddy, "PATH FINDING - Dijkstra ' s and A * Algorithm ' s," pp. 1–15, 2013.

[23] A. V. Goldberg and T. Radzik, "A heuristic improvement of the Bellman-Ford algorithm," *Appl. Math. Lett.*, vol. 6, no. 3, pp. 3–6, 1993.

[24] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms , Second Edition*. 2001.

[25] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, 1962.

[26] G. Algorithms, "All Pairs Shortest Paths •," no. m, pp. 1–4, 2015.

[27] S. Hougardy, "The Floyd-Warshall algorithm on graphs with negative cycles," *Inf. Process. Lett.*, vol. 110, no. 8–9, pp. 279–281, 2010.

[28] A. Pradhan and G. Mahinthakumar, "Finding all-pairs shortest path for a large-scale transportation network using parallel floyd-warshall and parallel Dijkstra algorithms," *J. Comput. Civ. Eng.*, vol. 27, no. June, pp. 263–273, 2012.

[29] M. Głąbowski and B. Musznicki, "Shortest Path Problem Solving Based on Ant Colony

Optimization Metaheuristic," *Image Process. Commun.*, vol. 17, no. 1, pp. 7–18, 2012.

[30] S. R. Kolavali and S. Bhatnagar, "Ant Colony Optimization Algorithms for," pp. 37–44, 2009.

[31] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, 1997.

[32] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 2, pp. 231–247, 1992.

[33] A. R. Saiyed, "The Traveling Salesman problem History of The TSP," pp. 1–15, 2012.

[34] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "The Traveling Salesman Problem: A Computational Study," *Princet. Univ. Press*, p. 593, 2006.

[35] T. S. Problem, *Travelling salesman problem: a foot-in-the-door?*, vol. 1, no. 4. 1997.

[36] M. M. Abid and I. Muhammad, "Heuristic Approaches to Solve Traveling Salesman Problem," *TELKOMNIKA Indones. J. Electr. Eng.*, vol. 15, no. 2, 2015.

[37] G. Gutin and A. Punnen, *The Traveling Salesman Problem and Its Variations*. 2004.

[38] M. a. S. Casquilho, "Travelling Salesman Problem," *Tech. Univ. Lisbon, Ave. Rovisco Pais, 1049-001 Lisboa, Port.*, vol. 43, no. 2, pp. 431–449, 2012.

[39] "A Comparative Study of Tabu Search and Simulated Annealing for Traveling Salesman Problem Project Report Applied Optimization MSCI 703 Sachin Jayaswal Student ID : 20186226 Department of Management Sciences University of Waterloo," *Constraints*.

[40] R. Stanec and O. Trenz, "Solving of Travelling Salesman Problem for large number of cities in environment with constraints," 2011.

[41] E. Balas, "Branch and Bound Method for Traveling Salesman Problem," *Carnegie Mellon Univ.*, 1983.

[42] J. Cirasella, D. S. Johnson, L. a Mcgeoch, and W. Zhang, "The Asymmetric Traveling Salesman Problem : Algorithms , Instance Generators , and Tests," *Lect. Notes Comput. Sci.*, vol. 2153, pp. 32–59, 2001.

[43] D. S. Johnson and L. A. McGeoch, "The traveling Salesman Problem: A Case Study in

Local Optimization," *Local Search Comb. Optim.*, pp. 215–310, 1997.

[44] C. Nilsson, "Heuristics for the traveling salesman problem," *Linkoping Univ.*, pp. 3–8, 2003.

[45] B. Golden, L. Bodin, T. Doyle, and W. S. Jr., "Approximate Traveling Salesman Algorithms," *Oper. Res.*, 1980.

[46] K. Arora, S. Agarwal, and R. Tanwar, "Solving TSP using Genetic Algorithm and Nearest Neighbour Algorithm and their Comparison," *Int. J. Sci. Eng. Res.*, vol. 7, no. 1, pp. 1014–1018, 2016.

[47] Á. N. Prestes, "Uma Análise Experimental de Abordagens Heurísticas Aplicadas ao Problema do Caixeiro Viajante," p. 84, 2006.

[48] K. Helsgaun, "Effective implementation of the Lin-Kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, 2000.

[49] S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem," *Oper. Res.*, 1973.

[50] M. Malek, M. Guruswamy, M. Pandya, and H. Owens, "Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem," *Ann. Oper. Res.*, 1989.

[51] N. Ascheuer, M. Fischetti, and M. Grötschel, "Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut," *Math. Program.*, 2001.

[52] M. López-Ibáñez, C. Blum, J. W. Ohlmann, and B. W. Thomas, "The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization," *Appl. Soft Comput. J.*, 2013.

[53] Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon, "An Optimal Algorithm for the Traveling Salesman Problem with Time Windows," *Oper. Res.*, 1995.

[54] B.-I. Kim, S. Kim, and S. Sahoo, "Waste collection vehicle routing problem with time windows," *Comput. Oper. Res.*, 2006.

[55] M. Gendreau, G. Laporte, and D. Vigo, "Heuristics for the traveling salesman problem with pickup and delivery," *Comput. Oper. Res.*, 1999.

[56] B. Kalantari, A. V. Hill, and S. R. Arora, "An algorithm for the traveling salesman problem with pickup and delivery customers," *Eur. J. Oper. Res.*, 1985.

[57] G. Mosheiov, "Theory and Methodology Tlhe Travelling Salesman Problem with pick-up and delivery," vol. 79, pp. 299–310, 1994.

[58] I. Dumitrescu, S. Ropke, J. F. Cordeau, and G. Laporte, "The traveling salesman problem with pickup and delivery: Polyhedral results and a branch-and-cut algorithm," *Math. Program.*, 2010.

[59] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *Eur. J. Oper. Res.*, vol. 140, no. 3, pp. 606–617, 2002.

[60] M. Fischetti and P. Toth, "An Additive Bounding Procedure for Combinatorial Optimization Problems," *Oper. Res.*, 1989.

[61] N. Ascheuer, M. Jünger, and G. Reinelt, "Branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints," *Comput. Optim. Appl.*, 2000.

[62] M. Kubo and H. Kasugai, "The precedence constrained traveling salesman problem," no. 2, pp. 152–172, 1991.

[63] A. Chentsov, M. Khachay, and D. Khachay, "Linear time algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 651–655, 2016.

[64] S. C. Sarin, H. D. Sherali, and A. Bhootra, "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints," *Oper. Res. Lett.*, vol. 33, no. 1, pp. 62–70, 2005.

[65] I. T. Hernádvölgyi, "Solving the sequential ordering problem with automatically generated lower bounds," *Oper. Res. Proc.*, no. July 2003, pp. 355–362, 2004.

[66] R. Salman, "Algorithms for the Precedence Constrained Generalized Travelling Salesperson Problem."

[67] L. F. Escudero, "An inexact algorithm for the sequential ordering problem," *Eur. J. Oper. Res.*, vol. 37, no. 2, pp. 236–249, 1988.

[68] V. Papapanagiotou, J. Jamal, R. Montemanni, G. Shobaki, and L. M. Gambardella, "A comparison of two exact algorithms for the sequential ordering problem A comparison of two exact algorithms for the sequential ordering problem," 2015.

[69] A. N. Letchford and J.-J. Salazar-González, "Stronger multi-commodity flow formulations of the Capacitated Vehicle Routing Problem," *Eur. J. Oper. Res.*, vol. 244, no. 3, pp. 730–738, 2015.

[70] L. M. Gambardella and M. Dorigo, "An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem," *INFORMS J. Comput.*, 2000.

[71] T. Hasuike, H. Katagiri, H. Tsubaki, and H. Tsuda, "Tour Planning for Sightseeing with Time-Dependent Satisfactions of Activities and Traveling Times," vol. 2013, no. May, pp. 369–379, 2013.

[72] A. Ephremides, P. Varaiya, and J. Walrand, "A Simple Dynamic Routing Problem," *IEEE Trans. Automat. Contr.*, 1980.

[73] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*. 2013.

[74] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno, "Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies," *European Journal of Operational Research*. 2003.

[75] L. Allan, "The dynamic Vehicle Routing Problem," 2000.

[76] H. N. Psaraftis, "Dynamic vehicle routing: Status and prospects," *Ann. Oper. Res.*, 1995.

[77] W. B. Powell, P. Jaillet, and A. Odoni, "Stochastic and dynamic networks and routing," *Handbooks in Operations Research and Management Science*. 1995.

[78] C. Malandraki and M. S. Daskin, "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms," *Transp. Sci.*, 1992.

[79] D. Bertsimas and G. Van Ryzin, "The Dynamic Traveling Repairman Problem," *MIT Sloan Sch. Work. Pap.*, 1989.

[80] U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, "A case for time-dependent shortest path computation in spatial networks," in *Proceedings of the 18th SIGSPATIAL*

*International Conference on Advances in Geographic Information Systems - GIS '10*, 2010.

[81] L. S. Buriol, M. G. C. Resende, and M. Thorup, "Speeding up dynamic shortest-path algorithms," *INFORMS J. Comput.*, 2008.

[82] I. Chabini, "Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time," *Transp. Res. Rec.*, 1997.

[83] A. Orda and R. Rom, "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length," *J. ACM*, 1990.

[84] H. N. L. Zhao, T. Ohshima, "A* algorithm for the time-dependent shortest path problem," *11th Japan-Korea Jt. Work. Algorithms Comput.*, 2008.

[85] K. L. Cooke and E. Halsey, "The shortest route through a network with time-dependent internodal transit times," *J. Math. Anal. Appl.*, 1966.

[86] S. E. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms," *Oper. Res.*, 1969.

[87] D. E. Kaufman and R. L. Smith, "FASTEST PATHS IN TIME-DEPENDENT NETWORKS FOR INTELLIGENT VEHICLE-HIGHWAY SYSTEMS APPLICATION∗," *I V H S J.*, 1993.

[88] L. Foschini, J. Hershberger, and S. Suri, "On the complexity of time-dependent shortest paths," *Algorithmica*, 2014.

[89] A. Mohan Rao and K. Ramachandra Rao, "MEASURING URBAN TRAFFIC CONGESTION – A REVIEW," *Int. J. Traffic Transp. Eng.*, 2012.

[90] CEPAL, *Traffic Congestion: The Problem and how to deal with it*. 2015.

[91] K. Olagunju, "Evaluating Traffic Congestion in Developing Countries – a Case Study of Nigeria," *2015 Chart. Inst. Logist. Transp. Africa forum held Mt. Meru Hotel. Arusha, Tanzania*, 2015.

[92] Thomas A. Rubin and Fatma Mansour, "Transit Utilization and Traffic Congestion : Is There a Connection ?," no. December, 2014.

[93] A. L. Kok, E. W. Hans, and J. M. J. Schutten, "Vehicle routing under time-dependent travel times: The impact of congestion avoidance," *Comput. Oper. Res.*, vol. 39, no. 5, pp. 910–918, 2012.

[94] M. Patriksson, "The Traffic Assignment Problem: Models and Methods," *Ann. Phys. (N. Y).*, vol. 54, no. 2, p. xii, 223 p., 1994.

[95] T. Friesz and D. Bernstein, "Analytical dynamic traffic assignment models," *Handb. Transp. Model.*, pp. 1–15, 2000.

[96] a Primer, "Dynamic Traffic Assignment," *Transp. Netw. Model. Comm.*, no. June, pp. 1–39, 2011.

[97] O. Di, "School of Industrial and Information Engineering The Asymmetric Traffic Assignment Problem on Large-Scale Networks," 2015.

[98] T. V Mathew and K. V. K. Rao, "Traffic Assignment," *Introd. to Transp. Eng.*, pp. 1–8, 2007.

[99] D. S. Leftwich and C. L. Heimbach, "Traffic assignment by trip type using volume restraint and link restraint for application in small urban areas," *J. Adv. Transp.*, vol. 18, no. 1, pp. 55–75, 1984.

[100] S. Gonzalez, "Estudio Integral de Transporte (III): Multimodal Transportation Study," 1999.

[101] M. L. Hazelton and J. Pueschel, "Estimation of link performance functions from incomplete flow data," *J. Adv. Transp.*, vol. 33, no. 3, pp. 323–334, 1999.

[102] N. A. Irwin, N. Dodd, and G. H. Von Cube, "Capacity Restraint in Assignment Programs," in *Highway Research Board Bulletin, 40th Annual Meeting of the Highway Research Board*, 1961.

[103] United States Bureau of Public Roads, "Traffic assignment manual for application with a large, high speed computer.," *U.S. Dept. Commer. Bur. Public Roads, Off. Planning, Urban Plan. Div.*, 1964.

[104] E. T. Mtoi and R. Moses, "Calibration and Evaluation of Link Congestion Functions: Applying Intrinsic Sensitivity of Link Speed as a Practical Consideration to Heterogeneous Facility Types within Urban Network," *J. Transp. Technol.*, 2014.

[105] A. V Plummer, "The Chicago area Transportation Study: Creating the First Plan ( 1955-1962 )," *Andrew V. Plummer*, 1962.

[106] J. A. Ferland, M. Florian, and C. Achim, "On incremental methods for traffic assignment," *Transp. Res.*, 1975.

[107] B. V Martin and M. L. Manheim, "A research program for comparison of traffic assignment techniques," 1965.

[108] E. Worrell, L. Price, N. Martin, C. Hendriks, and L. O. Meida, "CARBON DIOXIDE EMISSIONS FROM THE GLOBAL CEMENT INDUSTRY," *Annu. Rev. Energy Environ.*, 2001.

[109] M. Taylor, C. Tam, and D. Gielen, "Energy Efficiency and CO 2 Emissions from the Global Cement Industry," *IEA-WBCSD Cem. Energy Effic. Ind. Work.*, no. September, 2006.

[110] B. Noche and T. Elhasia, "Approach to Innovative Supply Chain Strategies in Cement Industry; Analysis and Model Simulation," *Procedia - Soc. Behav. Sci.*, vol. 75, pp. 359–369, 2013.

[111] I. Agudelo, "Supply Chain Management in the Cement Industry," 2008.

[112] R. Rehan and M. Nehdi, "Carbon dioxide emissions and climate change: Policy implications for the cement industry," *Environ. Sci. Policy*, 2005.

[113] European Commission, "Integrated Pollution Prevention and Control (IPPC) Best Available Techniques Reference Document on the Production of Iron and Steel December 2001," *Production*, 2001.

[114] "Cement - High Performance Cementitious Solutions | CEMEX UK." [Online]. Available: https://www.cemex.co.uk/cement.aspx. [Accessed: 16-Jul-2018].

[115] H.-C. Pfohl, B. Yahsi, and T. Kuznaz, "The impact of Industry 4.0 on the Supply Chain," *Proc. Hambg. Int. Conf. Logist. (HICL)-20*, 2015.

[116] E. Hozdić, "Smart factory for industry 4.0: A review," *Int. J. Mod. Manuf. Technol.*, 2015.

[117] B. Tjahjono, C. Esplugues, E. Ares, and G. Pelaez, "What does Industry 4.0 mean to Supply Chain?," *Procedia Manuf.*, 2017.

[118] P. Zheng *et al.*, "Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives," *Frontiers of Mechanical Engineering*. 2018.

[119] E. J. Henley and R. A. Williams, “Chapter 10: Matrix Representation of Graphs,” in *Graph Theory in Modern Engineering: Computer Aided Design, Control, Optimization, Reliability Analysis*, 1973.

[120] D. C. De Sena, E. F. Soares, I. V. L. De Paiva, and B. B. T. Do Carmo, “Queue balancing of load and expedition service in a cement industry in Brazil,” *Indep. J. Manag. Prod.*, vol. 4, no. 2, pp. 452–462, 2013.

# APPENDIX

Table 4 - Set composed by 5 trucks.

| Truck Id | Entrance Time | No. Of Required Locations | Required Service Locations |
|---|---|---|---|
| 1 | 0 | 4 | B,C,D,E |
| 2 | 1 | 4 | B,C,D,E |
| 3 | 2 | 4 | B,C,D,E |
| 4 | 3 | 4 | B,C,D,E |
| 5 | 4 | 4 | B,C,D,E |

Table 5 - Service times for the Set composed by 5 trucks.

| Server | Service Time |
|---|---|
| B | 16 |
| C | 12 |
| D | 19 |
| E | 17 |

Table 6 - Algorithm No.1 results of the simulation for the set composed by 5 trucks.

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 65 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 83 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 101 |
| Trcuk4 | [Population] | FlowTime | TimeInSystem | Average | 119 |
| Trcuk5 | [Population] | FlowTime | TimeInSystem | Average | 137 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 1.06383 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 30 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.141844 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 4 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 0.070922 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 2 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 101 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 137 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 65 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 5 |

**Table 7 - Algorithm No.2 results of the simulation for the set composed by 5 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 103.05 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 86 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 88 |
| Trcuk4 | [Population] | FlowTime | TimeInSystem | Average | 80.05 |
| Trcuk5 | [Population] | FlowTime | TimeInSystem | Average | 114.0333 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 0.448743 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 10.59333 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.254448 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 6.006667 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 5 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 0.016662 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 0.393333 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 94.22667 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 114.0333 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 80.05 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 5 |

**Table 8 - Set composed by 15 trucks.**

| Truck Id | Entrance Time | No. Of Required Locations | Required Service Locations |
|---|---|---|---|
| 1 | 0 | 3 | B,D,E |
| 2 | 1 | 2 | B,C |
| 3 | 2 | 3 | C,D,E |
| 4 | 3 | 4 | B,C,D,E |

|  |  |  |  |
|---|---|---|---|
| 5 | 5 | 4 | B,C,D,E |
| 6 | 6 | 2 | D,E |
| 7 | 7 | 1 | B |
| 8 | 8 | 4 | B,C,D,E |
| 9 | 10 | 4 | B,C,D,E |
| 10 | 11 | 2 | B,E |
| 11 | 12 | 3 | B,D,E |
| 12 | 13 | 3 | B,C,D |
| 13 | 15 | 2 | B,C |
| 14 | 16 | 3 | C,D,E |
| 15 | 17 | 4 | B,C,D,E |

**Table 9 - Service times for the Set composed by 15 trucks.**

| Server | Service Time |
|---|---|
| B | 16 |
| C | 12 |
| D | 19 |
| E | 17 |

**Table 10 - Algorithm No.1 results of the simulation for the set composed by 15 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 89.01667 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 134 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 121.0167 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 119.0333 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 136.0333 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 49.01667 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 64 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 171.0333 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 207.0333 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 172 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 179.0167 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 185.0333 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 184 |

| | | | | | |
|---|---|---|---|---|---|
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 141.0167 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 225 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 12 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 3.706612 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 74.75 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 10 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 0.173554 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 4.2 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 11 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.393113 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 8.648485 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 11 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 1.181129 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 25.98485 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 145.0833 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 225 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 49.01667 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 15 |

**Table 11 - Algorithm No.2 results of the simulation for the set composed by 15 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 114.0167 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 70 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 124.0333 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 193 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 197.0333 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 82.03333 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 32 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 213.0333 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 170 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 140 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 187 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 170.0333 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 152 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 148.0333 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 182.0167 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 12 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 2.755241 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 50.75 |

| | | | | | |
|---|---|---|---|---|---|
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 10 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 0.787212 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 17.4 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 11 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.932589 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 18.73939 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 11 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 1.401749 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 28.16667 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 144.9489 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 213.0333 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 32 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 15 |

**Table 12 - Set composed by 20 trucks.**

| Truck Id | Entrance Time | No. Of Required Locations | Required Service Locations |
|---|---|---|---|
| 1 | 0 | 1 | B |
| 2 | 0 | 2 | D,E |
| 3 | 1 | 2 | B,E |
| 4 | 1 | 1 | C |
| 5 | 2 | 3 | B,D,E |
| 6 | 3 | 4 | B,C,D,E |
| 7 | 4 | 4 | B,C,D,E |
| 8 | 4 | 4 | B,C,D,E |
| 9 | 4 | 2 | D,E |
| 10 | 5 | 3 | B,D,E |
| 11 | 5 | 1 | E |
| 12 | 6 | 2 | B,C |
| 13 | 7 | 4 | B,C,D,E |
| 14 | 8 | 4 | B,C,D,E |
| 15 | 8 | 3 | B,D,E |
| 16 | 8 | 3 | C,D,E |
| 17 | 9 | 4 | B,C,D,E |
| 18 | 10 | 1 | B |
| 19 | 10 | 4 | B,C,D,E |
| 20 | 10 | 3 | B,D,E |

**Table 13 - Service times for the Set composed by 20 trucks.**

| Server | Service Time |
|---|---|
| B | 19 |
| C | 10 |
| D | 12 |
| E | 17 |

**Table 14 - Algorithm No.1 results of the simulation for the set composed by 20 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 26 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 62.01667 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 291 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 14 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 111.0167 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 154.0333 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 168 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 185 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 75.01667 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 176.0167 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 40.01667 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 267 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 216 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 232 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 224.0167 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 88.01667 |
| Truck17 | [Population] | FlowTime | TimeInSystem | Average | 265 |
| Truck18 | [Population] | FlowTime | TimeInSystem | Average | 206 |
| Truck19 | [Population] | FlowTime | TimeInSystem | Average | 288 |
| Truck20 | [Population] | FlowTime | TimeInSystem | Average | 282.0167 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 15 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 6.275168 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 124.6667 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 10 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 0.060403 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 1.8 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 14 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.100671 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 2.142857 |

| | | | | | |
|---|---|---|---|---|---|
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 16 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 1.448993 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 26.9875 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 168.5083 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 291 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 14 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 20 |

**Table 15 - Algorithm No.2 results of the simulation for the set composed by 20 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 26 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 79.01667 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 82 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 14 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 157 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 195.0167 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 269 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 250 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 125.0333 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 176.0167 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 57.01667 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 96 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 259.0167 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 275.0167 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 189 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 207.0167 |
| Truck17 | [Population] | FlowTime | TimeInSystem | Average | 240.0167 |
| Truck18 | [Population] | FlowTime | TimeInSystem | Average | 54 |
| Truck19 | [Population] | FlowTime | TimeInSystem | Average | 282 |
| Truck20 | [Population] | FlowTime | TimeInSystem | Average | 206 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 15 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 2.277169 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 44.32889 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 10 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 1.085388 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 31.69333 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 14 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 2.140525 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 44.64524 |

| | | | | | |
|---|---|---|---|---|---|
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 16 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 2.057192 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 37.54375 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 161.9083 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 282 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 14 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 20 |

**Table 16 - Set composed by 10 trucks.**

| Truck Id | Entrance Time | No. Of Required Locations | Required Service Locations |
|---|---|---|---|
| 1 | 0 | 2 | B,E |
| 2 | 1 | 3 | B,C,E |
| 3 | 2 | 3 | C,E,F |
| 4 | 3 | 3 | B,E,F |
| 5 | 4 | 4 | B,C,E,F |
| 6 | 5 | 1 | D |
| 7 | 6 | 4 | B,C,E |
| 8 | 7 | 2 | C,F |
| 9 | 8 | 3 | B,C,E,F |
| 10 | 9 | 4 | B,C,E,F |

**Table 17 - Service times for the Set composed by 10 trucks.**

| Server | Service Time |
|---|---|
| B | 48 |
| C | 53 |
| D | 45 |
| E | 55 |
| F | 50 |

**Table 18 - Algorithm No.1 results of the simulation for the set composed by 10 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 114.0061 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 283 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 226 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 335 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 443 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 52 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 494 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 169 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 549 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 603 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 7 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 1.596405 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 139.5714 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 7 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 0.161765 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 14.14286 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 1 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 8 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 0.852941 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 65.25 |
| ServerF | [Resource] | Capacity | UnitsAllocated | Total | 6 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 326.8006 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 603 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 52 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 10 |

**Table 19 - Algorithm No.2 results of the simulation for the set composed by 10 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 131 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 295 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 239 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 296 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 432 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 52 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 320 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 266 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 483 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 372 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 7 |

| | | | | | |
|---|---|---|---|---|---|
| ServerB | InputBuffer | Content | NumberInStation | Average | 0.782077 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 54.85714 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 7 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 0.661914 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 46.42857 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 1 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 8 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 0.270876 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 16.625 |
| ServerF | [Resource] | Capacity | UnitsAllocated | Total | 6 |
| ServerF | InputBuffer | Content | NumberInStation | Average | 0.613035 |
| ServerF | InputBuffer | HoldingTime | TimeInStation | Average | 50.16667 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 288.6 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 483 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 52 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 10 |

**Table 20 - Set composed by 16 trucks.**

| Truck Id | Entrance Time | No. Of Required Locations | Required Service Locations |
|---|---|---|---|
| 1 | 0 | 1 | D |
| 2 | 1 | 3 | C,E,F |
| 3 | 2 | 2 | E,C |
| 4 | 3 | 3 | B,E,F |
| 5 | 4 | 4 | B,C,E,F |
| 6 | 5 | 4 | B,E,F,C |
| 7 | 6 | 2 | C,F |
| 8 | 7 | 1 | D |
| 9 | 8 | 3 | B,E,F |
| 10 | 9 | 2 | E,C |
| 11 | 11 | 1 | D |
| 12 | 12 | 3 | B,E,C |
| 13 | 13 | 4 | B,C,E,F |
| 14 | 15 | 3 | C,E,F |
| 15 | 17 | 4 | B,C,E,F |
| 16 | 18 | 4 | B,C,E,F |

**Table 21 - Service times for the Set composed by 16 trucks.**

| Server | Service Time |
|---|---|
| B | 42 |
| C | 56 |
| D | 48 |
| E | 60 |
| F | 40 |

**Table 22 - Algorithm No.1 results of the simulation for the set composed by 16 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 55 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 301 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 230 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 239 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 415 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 474 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 165 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 96 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 474 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 279 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 140 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 604 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 706 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 527 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 762 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 821 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 8 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 1.334923 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 140 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 11 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 0.401669 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 30.63636 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 3 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.150179 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 42 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 12 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 2.628129 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 183.75 |

|  |  |  |  |  |  |
|---|---|---|---|---|---:|
| ServerF | [Resource] | Capacity | UnitsAllocated | Total | 10 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 393 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 821 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 55 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 16 |

**Table 23 - Algorithm No.2 results of the simulation for the set composed by 16 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---:|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 55 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 287 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 230 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 379 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 704 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 763 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 296 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 96 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 340 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 279 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 140 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 576 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 555 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 633 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 607 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 552 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 8 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 0.891927 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 85.625 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 11 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 1.579427 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 110.2727 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 3 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.164063 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 42 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 12 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 1.377604 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 88.16667 |
| ServerF | [Resource] | Capacity | UnitsAllocated | Total | 10 |
| ServerF | InputBuffer | Content | NumberInStation | Average | 1.071615 |
| ServerF | InputBuffer | HoldingTime | TimeInStation | Average | 82.3 |

| | | | | | |
|---|---|---|---|---|---|
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 405.75 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 763 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 55 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 16 |

**Table 24 - Set composed by 30 trucks.**

| Truck Id | Entrance Time | No. Of Required Locations | Required Service Locations |
|---|---|---|---|
| 1 | 0 | 2 | C,E |
| 2 | 1 | 1 | D |
| 3 | 2 | 3 | B,E,F |
| 4 | 3 | 3 | C,E,F |
| 5 | 4 | 4 | B,C,E,F |
| 6 | 5 | 4 | B,C,E,F |
| 7 | 6 | 4 | B,C,E,F |
| 8 | 8 | 3 | B,C,E |
| 9 | 10 | 2 | C,F |
| 10 | 11 | 4 | B,C,E,F |
| 11 | 12 | 2 | C,E |
| 12 | 13 | 4 | B,C,E,F |
| 13 | 14 | 2 | C,F |
| 14 | 15 | 3 | C,E,F |
| 15 | 17 | 1 | D |
| 16 | 18 | 4 | B,C,E,F |
| 17 | 19 | 3 | C,E,F |
| 18 | 20 | 2 | C,E |
| 19 | 21 | 4 | B,C,E,F |
| 20 | 22 | 4 | B,C,E,F |
| 21 | 23 | 2 | C,E |
| 22 | 24 | 3 | B,E,F |
| 23 | 25 | 4 | B,C,E,F |
| 24 | 26 | 4 | B,C,E,F |
| 25 | 27 | 1 | D |
| 26 | 29 | 2 | C,E |
| 27 | 30 | 4 | B,C,E,F |
| 28 | 31 | 4 | B,C,E,F |
| 29 | 32 | 3 | B,C,E |
| 30 | 33 | 1 | D |

**Table 25 - Service times for the Set composed by 30 trucks.**

| Server | Service Time |
|---|---|
| B | 58 |
| C | 70 |
| D | 50 |
| E | 65 |
| F | 60 |

**Table 26 - Algorithm No.1 results of the simulation for the set composed by 30 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 430 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 57 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 468 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 532 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 776 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 845 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 914 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 982 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 211 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 1049 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 488 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 1117 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 277 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 845 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 91 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 1182 |
| Truck17 | [Population] | FlowTime | TimeInSystem | Average | 971 |
| Truck18 | [Population] | FlowTime | TimeInSystem | Average | 550 |
| Truck19 | [Population] | FlowTime | TimeInSystem | Average | 1249 |
| Truck20 | [Population] | FlowTime | TimeInSystem | Average | 1318 |
| Truck21 | [Population] | FlowTime | TimeInSystem | Average | 617 |
| Truck22 | [Population] | FlowTime | TimeInSystem | Average | 1291 |
| Truck23 | [Population] | FlowTime | TimeInSystem | Average | 1426 |
| Truck24 | [Population] | FlowTime | TimeInSystem | Average | 1495 |
| Truck25 | [Population] | FlowTime | TimeInSystem | Average | 131 |
| Truck26 | [Population] | FlowTime | TimeInSystem | Average | 681 |
| Truck27 | [Population] | FlowTime | TimeInSystem | Average | 1561 |
| Truck28 | [Population] | FlowTime | TimeInSystem | Average | 1630 |

| | | | | | |
|---|---|---|---|---|---:|
| Truck29 | [Population] | FlowTime | TimeInSystem | Average | 1699 |
| Truck30 | [Population] | FlowTime | TimeInSystem | Average | 175 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 16 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 3.878683 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 419.625 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 24 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 1.641248 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 118.375 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 4 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.13056 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 56.5 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 24 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 5.290583 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 381.5833 |
| ServerF | [Resource] | Capacity | UnitsAllocated | Total | 19 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 835.2667 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 1699 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 57 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 30 |

**Table 27 - Algorithm No.2 results of the simulation for the set composed by 30 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---:|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 541 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 57 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 1189 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 739 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 1512 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 1682 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 1575 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 979 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 612 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 1396 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 555 |
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 1305 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 623 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 1436 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 91 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 1319 |
| Truck17 | [Population] | FlowTime | TimeInSystem | Average | 1248 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Truck18 | [Population] | FlowTime | TimeInSystem | Average | 687 |
| Truck19 | [Population] | FlowTime | TimeInSystem | Average | 1222 |
| Truck20 | [Population] | FlowTime | TimeInSystem | Average | 1426 |
| Truck21 | [Population] | FlowTime | TimeInSystem | Average | 754 |
| Truck22 | [Population] | FlowTime | TimeInSystem | Average | 1033 |
| Truck23 | [Population] | FlowTime | TimeInSystem | Average | 1158 |
| Truck24 | [Population] | FlowTime | TimeInSystem | Average | 1522 |
| Truck25 | [Population] | FlowTime | TimeInSystem | Average | 131 |
| Truck26 | [Population] | FlowTime | TimeInSystem | Average | 902 |
| Truck27 | [Population] | FlowTime | TimeInSystem | Average | 1587 |
| Truck28 | [Population] | FlowTime | TimeInSystem | Average | 1516 |
| Truck29 | [Population] | FlowTime | TimeInSystem | Average | 1165 |
| Truck30 | [Population] | FlowTime | TimeInSystem | Average | 175 |
| ServerB | [Resource] | Capacity | UnitsAllocated | Total | 16 |
| ServerB | InputBuffer | Content | NumberInStation | Average | 1.820984 |
| ServerB | InputBuffer | HoldingTime | TimeInStation | Average | 192 |
| ServerC | [Resource] | Capacity | UnitsAllocated | Total | 24 |
| ServerC | InputBuffer | Content | NumberInStation | Average | 4.299941 |
| ServerC | InputBuffer | HoldingTime | TimeInStation | Average | 302.25 |
| ServerD | [Resource] | Capacity | UnitsAllocated | Total | 4 |
| ServerD | InputBuffer | Content | NumberInStation | Average | 0.133966 |
| ServerD | InputBuffer | HoldingTime | TimeInStation | Average | 56.5 |
| ServerE | [Resource] | Capacity | UnitsAllocated | Total | 24 |
| ServerE | InputBuffer | Content | NumberInStation | Average | 6.093657 |
| ServerE | InputBuffer | HoldingTime | TimeInStation | Average | 428.3333 |
| ServerF | [Resource] | Capacity | UnitsAllocated | Total | 19 |
| ServerF | InputBuffer | Content | NumberInStation | Average | 1.847659 |
| ServerF | InputBuffer | HoldingTime | TimeInStation | Average | 164.0526 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 1004.567 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 1682 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 57 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 30 |

**Table 28 - Entrance management algorithm results of the simulation for the set composed by 10 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 186 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 350 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 294 |

| | | | | | |
|---|---|---|---|---|---|
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 248 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 402 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 52 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 214 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 292 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 429 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 452 |
| B | [Resource] | Capacity | UnitsAllocated | Total | 7 |
| B | InputBuffer | Content | NumberInStation | Average | 0.533623 |
| B | InputBuffer | HoldingTime | TimeInStation | Average | 35.14286 |
| C | [Resource] | Capacity | UnitsAllocated | Total | 7 |
| C | InputBuffer | Content | NumberInStation | Average | 0.668113 |
| C | InputBuffer | HoldingTime | TimeInStation | Average | 44 |
| D | [Resource] | Capacity | UnitsAllocated | Total | 1 |
| E | [Resource] | Capacity | UnitsAllocated | Total | 8 |
| E | InputBuffer | Content | NumberInStation | Average | 0.859002 |
| E | InputBuffer | HoldingTime | TimeInStation | Average | 49.5 |
| F | [Resource] | Capacity | UnitsAllocated | Total | 6 |
| F | InputBuffer | Content | NumberInStation | Average | 0.496746 |
| F | InputBuffer | HoldingTime | TimeInStation | Average | 38.16667 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 291.9 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 452 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 52 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 10 |

**Table 29 - Entrance management algorithm results of the simulation for the set composed by 16 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 55 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 201 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 440 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 319 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 417 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 98 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 226 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 255 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 494 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 553 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 140 |

| | | | | | |
|---|---|---|---|---|---|
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 612 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 369 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 607 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 665 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 724 |
| B | [Resource] | Capacity | UnitsAllocated | Total | 8 |
| B | InputBuffer | Content | NumberInStation | Average | 0.402965 |
| B | InputBuffer | HoldingTime | TimeInStation | Average | 37.375 |
| C | [Resource] | Capacity | UnitsAllocated | Total | 11 |
| C | InputBuffer | Content | NumberInStation | Average | 1.530997 |
| C | InputBuffer | HoldingTime | TimeInStation | Average | 103.2727 |
| D | [Resource] | Capacity | UnitsAllocated | Total | 3 |
| D | InputBuffer | Content | NumberInStation | Average | 0.172507 |
| D | InputBuffer | HoldingTime | TimeInStation | Average | 42.66667 |
| E | [Resource] | Capacity | UnitsAllocated | Total | 12 |
| E | InputBuffer | Content | NumberInStation | Average | 1.107817 |
| E | InputBuffer | HoldingTime | TimeInStation | Average | 68.5 |
| F | [Resource] | Capacity | UnitsAllocated | Total | 10 |
| F | InputBuffer | Content | NumberInStation | Average | 1.570081 |
| F | InputBuffer | HoldingTime | TimeInStation | Average | 116.5 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 385.9375 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 724 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 55 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 16 |

**Table 30 - Entrance management algorithm results of the simulation for the set composed by 30 trucks.**

| Object Name | Data Source | Category | Data Item | Statistic Type | Value |
|---|---|---|---|---|---|
| Truck1 | [Population] | FlowTime | TimeInSystem | Average | 57 |
| Truck2 | [Population] | FlowTime | TimeInSystem | Average | 411 |
| Truck3 | [Population] | FlowTime | TimeInSystem | Average | 735 |
| Truck4 | [Population] | FlowTime | TimeInSystem | Average | 669 |
| Truck5 | [Population] | FlowTime | TimeInSystem | Average | 1157 |
| Truck6 | [Population] | FlowTime | TimeInSystem | Average | 423 |
| Truck7 | [Population] | FlowTime | TimeInSystem | Average | 772 |
| Truck8 | [Population] | FlowTime | TimeInSystem | Average | 1119 |
| Truck9 | [Population] | FlowTime | TimeInSystem | Average | 1188 |
| Truck10 | [Population] | FlowTime | TimeInSystem | Average | 1181 |
| Truck11 | [Population] | FlowTime | TimeInSystem | Average | 95 |

| | | | | | |
|---|---|---|---|---|---|
| Truck12 | [Population] | FlowTime | TimeInSystem | Average | 625 |
| Truck13 | [Population] | FlowTime | TimeInSystem | Average | 484 |
| Truck14 | [Population] | FlowTime | TimeInSystem | Average | 903 |
| Truck15 | [Population] | FlowTime | TimeInSystem | Average | 1461 |
| Truck16 | [Population] | FlowTime | TimeInSystem | Average | 1329 |
| Truck17 | [Population] | FlowTime | TimeInSystem | Average | 848 |
| Truck18 | [Population] | FlowTime | TimeInSystem | Average | 548 |
| Truck19 | [Population] | FlowTime | TimeInSystem | Average | 1236 |
| Truck20 | [Population] | FlowTime | TimeInSystem | Average | 1465 |
| Truck21 | [Population] | FlowTime | TimeInSystem | Average | 134 |
| Truck22 | [Population] | FlowTime | TimeInSystem | Average | 518 |
| Truck23 | [Population] | FlowTime | TimeInSystem | Average | 972 |
| Truck24 | [Population] | FlowTime | TimeInSystem | Average | 1662 |
| Truck25 | [Population] | FlowTime | TimeInSystem | Average | 1390 |
| Truck26 | [Population] | FlowTime | TimeInSystem | Average | 178 |
| Truck27 | [Population] | FlowTime | TimeInSystem | Average | 678 |
| Truck28 | [Population] | FlowTime | TimeInSystem | Average | 1027 |
| Truck29 | [Population] | FlowTime | TimeInSystem | Average | 1665 |
| Truck30 | [Population] | FlowTime | TimeInSystem | Average | 1594 |
| B | [Resource] | Capacity | UnitsAllocated | Total | 16 |
| B | InputBuffer | Content | NumberInStation | Average | 2.486152 |
| B | InputBuffer | HoldingTime | TimeInStation | Average | 263.6875 |
| C | [Resource] | Capacity | UnitsAllocated | Total | 24 |
| C | InputBuffer | Content | NumberInStation | Average | 4.183854 |
| C | InputBuffer | HoldingTime | TimeInStation | Average | 295.8333 |
| D | [Resource] | Capacity | UnitsAllocated | Total | 4 |
| D | InputBuffer | Content | NumberInStation | Average | 0.139069 |
| D | InputBuffer | HoldingTime | TimeInStation | Average | 59 |
| E | [Resource] | Capacity | UnitsAllocated | Total | 24 |
| E | InputBuffer | Content | NumberInStation | Average | 1.282852 |
| E | InputBuffer | HoldingTime | TimeInStation | Average | 90.70833 |
| F | [Resource] | Capacity | UnitsAllocated | Total | 19 |
| F | InputBuffer | Content | NumberInStation | Average | 3.848556 |
| F | InputBuffer | HoldingTime | TimeInStation | Average | 343.7368 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Average | 884.1333 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Maximum | 1665 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Minimum | 57 |
| Exit | [DestroyedEntities] | FlowTime | TimeInSystem | Observations | 30 |