GLOBAL SUPPLY CHAIN OPTIMIZATION: A MACHINE LEARNING PERSPECTIVE TO IMPROVE CATERPILLAR'S LOGISTICS OPERATIONS

Marco Veluscek

Electronic and Computer Engineering

College of Engineering, Design and Physical Sciences

Brunel University London

A thesis submitted for the degree of Doctor of Philosophy

July 2016

First supervisor: Dr. Tatiana Kalganova

Second supervisor: Peter Broomhead

DECLARATION

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This thesis has not previously been presented in identical or similar form to any other examination board. The thesis work was conducted from February 2013 to February 2016 under the supervision of Dr. Tatiana Kalganova, at Brunel London University.

Marco Veluscek, BSc, MSc

Uxbridge, London, United Kingdom

ABSTRACT

Supply chain optimization is one of the key components for the effective management of a company with a complex manufacturing process and distribution network. Companies with a global presence in particular are motivated to optimize their distribution plans in order to keep their operating costs low and competitive. Changing condition in the global market and volatile energy prices increase the need for an automatic decision and optimization tool.

In recent years, many techniques and applications have been proposed to address the problem of supply chain optimization. However, such techniques are often too problem-specific or too knowledge-intensive to be implemented as in-expensive, and easy-to-use computer system. The effort required to implement an optimization system for a new instance of the problem appears to be quite significant. The development process necessitates the involvement of expert personnel and the level of automation is low. The aim of this project is to develop a set of strategies capable of increasing the level of automation when developing a new optimization system. An increased level of automation is achieved by focusing on three areas: multi-objective optimization, optimization algorithm usability, and optimization model design.

A literature review highlighted the great level of interest for the problem of multi-objective optimization in the research community. However, the review emphasized a lack of standardization in the area and insufficient understanding of the relationship between multi-objective strategies and problems. Experts in the area of optimization and artificial intelligence are interested in improving the usability of the most recent optimization algorithms. They stated the concern that the large number of variants and parameters, which characterizes such algorithms, affect their potential applicability in real-world environments. Such characteristics are seen as the root cause for the low success of the most recent optimization algorithms in industrial applications. Crucial task for the development of an optimization system is the design of the optimization model. Such task is one of the most complex in the development process, however, it is still performed mostly manually. The importance and the complexity of the task strongly suggest the development of tools to aid the design of optimization models.

In order to address such challenges, first the problem of multi-objective optimization is considered and the most widely adopted techniques to solve it are identified. Such

techniques are analyzed and described in details to increase the level of standardization in the area. Empirical evidences are highlighted to suggest what type of relationship exists between strategies and problem instances. Regarding the optimization algorithm, a classification method is proposed to improve its usability and computational requirement by automatically tuning one of its key parameters, the termination condition. The algorithm understands the problem complexity and automatically assigns the best termination condition to minimize runtime. The runtime of the optimization system has been reduced by more than 60%. Arguably, the usability of the algorithm has been improved as well, as one of the key configuration tasks can now be completed automatically. Finally, a system is presented to aid the definition of the optimization model through regression analysis. The purpose of the method is to gather as much knowledge about the problem as possible so that the task of the optimization model definition requires a lower user involvement. The application of the proposed algorithm is estimated that could have saved almost 1000 man-weeks to complete the project.

The developed strategies have been applied to the problem of Caterpillar's global supply chain optimization. This thesis describes also the process of developing an optimization system for Caterpillar and highlights the challenges and research opportunities identified while undertaking this work. This thesis describes the optimization model designed for Caterpillar's supply chain and the implementation details of the Ant Colony System, the algorithm selected to optimize the supply chain. The system is now used to design the distribution plans of more than 7,000 products. The system improved Caterpillar's marginal profit on such products by a factor of 4.6% on average.

.

To my family

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincerest gratitude to my family for their endless and unconditional support. Thank you to my dad Antonio and my mum Giovanna for always believing in me and supporting me, especially during difficult times. Thank you to my sisters Paola and Giulia for being the best sisters I could ever ask for. My heartfelt thanks and appreciation to my grandparents Mario and Giuliana.

I would like to thank my first supervisor Dr. Tatiana Kalganova for giving me the opportunity of undertaking my PhD degree and for her supervision throughout these three years. I would also like to thank my second supervisor Peter Broomhead for his support in writing my papers and this thesis.

I am grateful to Caterpillar Inc. for allowing me to work on this project and for the financial support. I am very thankful to the Caterpillar team: Anthony Grichnik , Adam Byerly, Karunya Jasti, Santhosh Vamaraju, Lusine Baghdasaryan, and all the people who worked with me on this project. It has been a pleasure working alongside so many talented people. Moreover, thank you for being so friendly and helpful during my visit at Caterpillar. You made my visit one of the best experiences of my PhD.

I would like to thank Prof. Sune Lehmann, Prof. Alessandro Sperduti and Dr. Andrea Burattin for first introducing me to the research world. If I was able to succeed is in no small part because of the things I have learnt working with you.

Finally, I would like to acknowledge the PhD committee who have reviewed my work.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
SCO	Supply Chain Optimization
LP	Linear Programming
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
ILP	Integer Linear Programming
MOO	Multi-Objective Optimization
BB	Branch and Bound
HE	Heuristic
ACO	Ant Colony Optimization
ACS	Ant Colony System
ES	Evolutionary Strategy
PSO	Particle Swarm Optimization
WIP	Work In Progress (often referred to inventory and shipped product)
MIP	Mixed-Integer Programming
NSGA-II	Non-dominated Sorting Genetic Algorithm-II
MOEA	Multi-Objective Evolutionary Algorithm
DSS	Decision Support System
MOES	Multi-Objective Evolutionary Strategy
TSP	Travelling Salesman Problem
DC	Distribution Centre
3PL	Third-party Logistics
4PL	Fourth-party Logistics

RoRo	Roll-on-Roll-off
WTI	West Texas Intermediate
VAM	Vogel's Approximation Method
MO	Multi Objective
GS	Goal Synthesis
EX	Exploration
IS	Incremental Solving
SP	Superposition
DU	Delta Unification
WFW	Weighted Frontier Walk
IT	Incremental Solving via Tuning
IR	Incremental Solving via Retention
QLF	Quality Loss Function
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
GP	Gaussian Process
MCS	Monte Carlo Sampling
LR	Linear Regression
SVR	Support Vector Machine for Regression
RBF	Radial Basis Function

LIST OF APPENDICES

1 INTRODUCTION

Companies with complex supply chains are being forced to reconsider their operations and transportation network designs in light of elevated and unpredictable operating costs. Transportation costs in particular are affected by the recent volatility in energy prices. As such costs rises are increasingly becoming a significant factor in the strategic planning and decision making processes for product sourcing. In addition to improving their overall profitability, companies must also strive to ensure that their supply chain networks are resilient to unexpected disruptions, whilst at the same time maintaining high service levels.

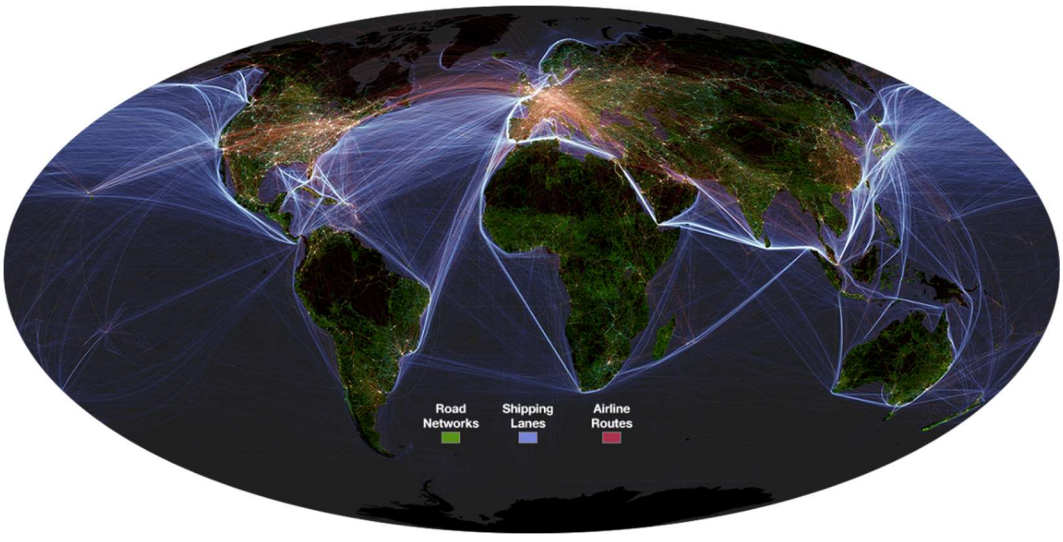

Figure 1.1 - Global transportation system.

The domain of the work presented in this thesis is global supply chain optimization. The project consists of the development of a set of strategies to increase the level of automation when developing a new optimization system. Specifically, three areas of this problem have been addressed:

1. Identification of general strategies for the problem of multi-objective optimization.
2. Improvement of the usability of the optimization algorithm.
3. Development of an algorithm to automatically design the optimization model.

The developed strategies have been applied when implementing an optimization system to solve Caterpillar's global supply chain optimization problem. All developed techniques have been tested on data provided by the logistics department of Caterpillar.

This thesis is structure as follows. Section 1.1 outlines the main motivations for this work. Section 1.2 states the objectives that the project aimed to accomplish. Sections 1.3 and 1.4 highlight the contribution to the knowledge and the list of achievements of this project. Chapter 2 presents a literature review of the problem of supply chain optimization. A general description of the problem is first provided. Then the most adopted and successful methods to address it are reported. Throughout the review, research works and surveys are referenced as they offer more in-depth analysis to specific aspects of the problem. The purpose of the chapter is to highlight the limitations and research opportunities that emerge from such methods. Chapter 3 addresses the task of the optimization model definition and describes in details the model designed for representing Caterpillar's global supply chain. The chapter shows how the limitations and opportunities discussed in chapter 2 have been addressed while designing Caterpillar's model. The chapter continues with the description of the implemented optimization algorithm and establishes its effectiveness. Due to the potential high complexity of the Caterpillar's scenario, several constraints have been encountered on the algorithms and technologies that could be implemented and deployed in the long term. The optimization algorithm adopted is the Ant Colony System (ACS), a meta-heuristic algorithm from the area of artificial intelligence.

Chapters 4, 5, and 6 describe the strategies to increase the level of automation. Specifically, chapter 4 addresses the problem of multi-objective optimization. Chapter 5

proposes an algorithm capable of automatically tuning one of the most influential parameters of the optimization: the termination condition. Chapter 6 presents an idea for the development of an algorithm that is able of automatically design the optimization model, given readily available information and data.

Chapter 7 draws the conclusions of this work and suggests some possible future work.

The systems presented in the following chapters have been deployed into the production environment of Caterpillar. At the time of writing, the process is employed to design distribution plans of over 7,000 products. The optimization system increased Caterpillar's marginal profit on such products by 4.6% on average.

1.1 Motivations

In the last few decades, many different aspects of the problem of supply chain optimization have been addressed both from a research and from an industrial stand point. Nevertheless, there is still high friction when implementing a new optimization system. The main tasks when developing an optimization system are:

1. Data collection and preparation.
2. Optimization model definition.
3. Optimization algorithm selection and implementation.
4. Optimization parameters configuration.
5. Results analysis and implementation.

Figure 1.2 shows the process of implementing an optimization system. The input of the process is logistics data and knowledge of the logistics operations and business characteristics, and the output a distribution plan. The figure also depicts the organization of the thesis.

Almost all of the previously listed tasks are still performed manually. It is necessary to involve personnel expert in very different areas. For instance, when defining the optimization model, it is needed to involve personnel with a good understanding of the logistics operation of the company. They must also have experience in designing mathematical models and know how to describe logistics operations with a mathematical formulation.

Even following the example of the work done on an application similar to the problem at hand is typically not enough. The most common techniques to address optimization problem are characterized by many variants and parameters. Very often a new application requires a specialized combination of algorithm variants and parameters.

As anticipated, three areas have been identified as challenging when developing an optimization system. The three areas are as follows:

1. Multi-objective optimization.
2. Optimization algorithm implementation and parameters tuning.
3. Optimization model definition.

During the literature review, the topic of multi-objective optimization has been identified as being of particular interest to the research community, but also lacking a general definition of the problem domain and the necessary techniques to address it. The use of multi-objective supply chain optimization has led to a more accurate and realistic solution in comparison to scenarios where only a single objective is considered. From a review of the state of the art in multi-objective optimization, four generic strategies have appeared to be the most common ones. They are referred to as *goal synthesis*, *superposition*, *incremental solving* and *exploration*. From the literature, the preferred approach lies in the combination of goals into a single optimization model (a.k.a. goal synthesis). Despite its popularity as a multi-objective optimization method and in the context of the current problem domain, the experimental results achieved by this method resulted in poor quality solutions when compared to the other strategies. This was particularly noticeable in the case of the superposition method, which significantly outperformed goal synthesis. Such experimental results suggest that the relationship between multi-objective optimization strategies and the type of problem where they are more likely to succeed is still unknown. Given a problem instance, it is difficult to predict what would be the most promising method to solve it.

Regarding the second point about the optimization algorithm, a version of the Ant Colony System (ACS) has been implemented. The ACS is a well-known bio-inspired optimization algorithm, which has been successfully applied to several NP-hard optimization problems, including supply chain optimization. As for many of the optimization techniques from the area of artificial intelligence, the ACS is also known

for its large number of variants and parameters. Such large number of parameters that need to be configured significantly reduce the algorithm usability. The termination condition is one of the key parameters of the ACS. It is also a parameter that is often neglected in research work. Meta-heuristics experts claim that a general termination criterion does not exist for most of such methods. In practice, some rules of thumb are adopted to set the termination conditions. Rules of thumb may be effective in practice. However, they reduce the usability of the algorithm. Personnel with experience in such rules of thumb need to be involved when implementing and maintaining the algorithm. Moreover, if the instances on which the algorithm is applied change, the termination condition cannot be automatically adapted. In order to address such issue, first, a definite procedure to evaluate improvement should be design. Then, a set of termination conditions can be implemented based on this metric for improvement. Ideally, a formalization of improvement measurements and termination conditions should allow automatic setting of them.

Finally, the task of optimization model definition is a crucial one, which encapsulate many areas of expertise. Despite the large number of applications that exist in the literature, designing the model for a new supply chain optimization problem is still very time consuming and resources intensive. The typical process for defining a model consists of either hiring simulation engineers and building internal simulation teams, or contract consultants. Time and resources are invested so that the team can become familiar with the logistics operations. After a while, the team produces a model than needs to be evaluated and tested. The model is likely to be inaccurate or overly simplified after the first step of development. More iterations of development and evaluation are likely to be required to create a model of satisfactory quality and accuracy. What the literature is lacking are general-purpose models and tools to aid the definition and specialize such models. All supply chain contexts should have some degree of similarity. There should be information that is common throughout all of them and data that should be easily available. Formalizing such information allows the definition of general-purpose models and tools capable of instantiate them with a reduce personnel interaction.

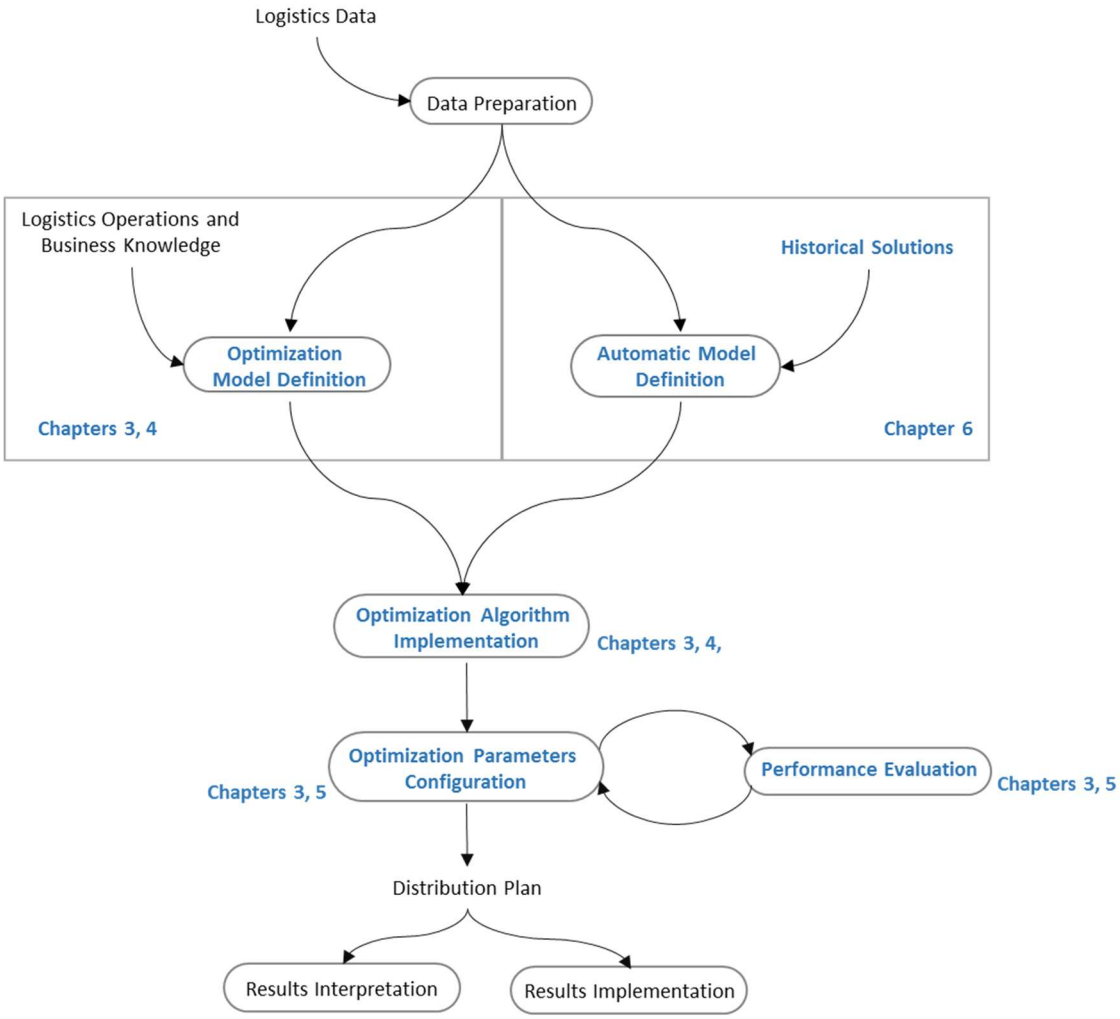

Figure 1.2 – Visual representation of the process of implementing an optimization system. The inputs of the process are logistics data and knowledge of the logistics operations and business characteristics. The major tasks are the definition of the optimization model and the implementation of the optimization algorithm. The graph also shows the thesis organization. Each task addressed in this work is highlighted in blue and the chapters where they are covered are indicated.

1.2 Aim and Objectives

The work presented in this thesis aims to develop a set of strategies capable of increasing the level of automation when implementing a new optimization system. The application of such strategies should reduce the expert interaction required. In order to accomplish this objective, three areas have been investigated and they are as follows:

1. Multi-objective optimization (MOO).

2. Automatic termination condition tuning to improve usability of the optimization algorithm.
3. Optimization model design.

The objectives in the three areas are respectively:

1. To provide standardization in the area of MOO. Standardization may be achieved by the identification of the most promising strategy to solve the problem of supply chain optimization when a multi-objective perspective is taken. The performance of such strategies is analyzed on the test case problems of this work. Such analysis is aimed to provide more understanding on the relationship between MOO strategies and problem instances.
2. To propose algorithms capable of improving the usability of the optimization algorithm. Usability may be improved by developing systems to automatically handle the many parameters of the typical optimization algorithms. The work in this area is particularly focused on the automatic tuning of the termination condition. More concretely, the current objective is to define evaluation metrics to measure improvement during the search for the optimal solution. Such evaluation metrics allow to formalize termination conditions and develop algorithms to automatically tune them given an instance of the problem.
3. To formalize information easily-available in an applied context of supply chain optimization. The objective then consists of taking advantage of such information to develop automatic tools to aid the design of an optimization model. The algorithm result of this activity should produce a good approximation of a manually-designed optimization model. The requirement for expert personnel interaction should be reduced overall.

The strategies developed to reach such objectives should work when applied in a real-world context. As previously discussed, Caterpillar's global supply chain has been selected as test case. An additional aim of this project is that the developed methods should provide meaningful benefits to the application of Caterpillar's optimization problem.

The following sections summarize how these objectives have been accomplished and what has been achieved throughout this project.

1.3 Contribution to the Knowledge

Contributions to the knowledge in this thesis are focused on the three areas listed above. Analysis, strategies, and algorithms have been designed to achieve the objectives in this three areas. To summarize, the main contributions are as follows:

1. In order to provide standardization in the area of MOO, four generic strategies have been identified and explained with thorough details. This work is described in chapter 4. For referencing purposes, they have been named as *goal synthesis*, *superposition*, *incremental solving* and *exploration*. Seven instances of these four strategies have been implemented and tested on three test cases of the problem of supply chain optimization. The superposition strategy has been found to be the best performing one on the given test cases. An introductory analysis has been provided to increase the understanding of the properties of the four strategies. Empirical evidences have been highlighted that suggest what type of relationship exists between strategies and problem instances. This work will serve as a reference on multi-objective methods for real-world 'industrial' supply chain optimization problems
2. In order to improve usability of the optimization algorithm, a method to automatically tune the termination condition has been proposed. This work is described in chapter 5. The purpose of the method is to predict the best termination iteration for an unseen instance by analyzing the performance of the optimization process across solved instances. A fitness landscape analysis is used to understand the behavior of the optimizer on all given instances. A comprehensive set of features are presented to characterize instances of the transportation network optimization problem. This set of features are associated with the results of the fitness landscape analysis through a machine learning-based approach, so that the behavior of the optimization algorithm may be predicted before the optimization starts and the termination iteration may be set accordingly. The proposed system has been tested on three instances of the problem of supply chain optimization. The proposed method has drastically

reduced the computational times required by the optimization algorithm in finding high quality solutions. Arguably, the usability of the algorithm has been improved as well, as one of the key configuration tasks can now be completed automatically.

3. In order to reduce the effort when designing an optimization model, the development of tools to aid during the completion of this task has been discussed. This work is described in chapter 6. A scenario is illustrated where a supply chain is being managed manually, but there is an intent to applying an optimization system. In such a scenario, information on the supply chain already exists in the form of distribution plans implemented in the past. An algorithm is proposed to "mine" the knowledge available in such solutions and automatically design an optimization model. A regression analysis has been employed to approximate the function representing the supply chain model. The approximated function is then implemented within the optimization process to evaluate potential solutions. The regression analysis yielded quite promising results: the approximated functions in all the tested scenarios appear to be very close to the original model. However, the overall optimization process produced solutions of unexpected lower quality. This discrepancy is currently being investigated. The proposed system is described in chapter 6 because the approach has the potential to significantly improve the development of optimization process. The intent is to suggest the development of techniques to aid the design task.

1.4 List of Publications and Project Achievements

Most of the contributions presented in this thesis are based on published or patented material. The model and the optimization algorithm described in chapter 3 are covered by the following patent applications:

- T. Grichnik, T. Aguilar, K. Jasti, S. Vamaraju, C. Nikolopoulos, A. Byerly, T. Kalganova, M. Veluscek, and P. Broomhead (Filing Date: 27/03/2015) Method and System for Managing Supply Chain Network with Multiple Supply Layers, CAT Ref. No.: 08350.1735-00000 (14-1941). *Provisional Patent Application*.

- T. Kalganova, T. Grichnik, A. Ogunbanwo, A. Williamson, M. Veluscek, R. Izsak, P. Broomhead (Filing Date: 15/10/2013) Hybrid Supply Chain Modelling, Optimisation and Presentation, CAT Ref.: 13-0770. *Provisional Patent Application*.
- T. Kalganova, T. Grichnik, A. Ogunbanwo, A. Williamson, M. Veluscek, R. Izsak, P. Broomhead (Filing Date: 15/03/2014) Supply Chain Network Modeling Using an Evolutionary Strategy, CAT Ref.: 14-0155. *Provisional Patent Application*.

The analysis hinted in chapter 2 regarding the network complexity of the work developed in the last decade for supply chain optimization is published in [1]:

- A. Ogunbanwo, A. Williamson, M. Veluscek, R. Izsak, T. Kalganova, P. Broomhead. "Transportation Network Optimization, Encyclopedia of Business Analytics and Optimization." *Encyclopedia of Business Analytics and Optimization (1st Edition 2014)*, John Wang, IGI Global. DOI: 10.4018/978-1-4666-5202-6. ISBN: 9781466652026.

The multi-objective methods summarized in chapter 4 have been published in [2]:

- Marco Veluscek, Tatiana Kalganova, Peter Broomhead, Anthony Grichnik. "Composite Goal Methods for Transportation Network Optimization." *Expert System with Applications*, Volume 42, Issue 8, 15 May 2015, Pages 3852–3867. DOI:10.1016/j.eswa.2014.12.017.

The methods have been used to deploy a multi-goal optimization system for Caterpillar's problem. The system is described in the following patent application:

- T. Grichnik, C. Nikolopoulos, A. Byerly, E. Hill, B. Kelly, T. Kalganova, M. Veluscek, P. Broomhead (Filing Date: 15/05/2014) Supply Network Optimization Method and System for Multiple Objectives, CAT Ref.: 08350 1265 (13-1756). *Provisional Patent Application*.

Chapter 5 is based on the work in [3]:

- M. Veluscek, T. Kalganova, P. Broomhead. "Improving Ant Colony Optimization Performance through Prediction of Best Termination Condition."

Industrial Technology (ICIT), 2015 IEEE International Conference, Pages 2394-2402, 17-19 March 2015, DOI: 10.1109/ICIT.2015.7125451. ieeexplore.ieee.org

The main datasets utilized throughout this work have been published on the free online repository Figshare (figshare.com). The main datasets are:

1. The original dataset from Caterpillar published in [4] for a medium-size excavator. The distribution network consists of 200 dealerships locations, 40 production facilities and assembly points, and 68 shipping ports. The connections between nodes in the network are determined by real-world transportation routes. The demand and capacity is from a twelve months period from January 2015 to December 2015.
2. Dataset generated according to an uniform distribution published in [5]. The demand figures, the production capacities, the transportation times and costs and the sale prices have been randomly generated according to a normal distribution with the same mean and standard deviation as in the original dataset.
3. Dataset generated according to a random distribution published in [6]. The demand figures, the production capacities, the transportation times and costs and the sale prices are randomly generated in an interval between 0 and an upper limit which is a random increase over the maximum value in the original data, according to a negative exponential distribution.
4. Extended dataset generated according to an uniform distribution published in [7]. The demand figures, the production capacities, the transportation times and costs and the sale prices have been randomly generated according to a normal distribution with the same mean and standard deviation as in the original dataset. This dataset has been extended with 9 more years of demand distributions. The additional 9 years have been created with the same random problem generator. The purpose of the dataset is to provide more instances of the problem (e.g. this dataset has been adopted in a machine learning context and it was necessary to have a larger and more comprehensive training set.). The dataset may be interpreted as containing 10 years of demand for one product or the demand figures of 10 similar products.

In March 2016, the project team won the Chairman's Innovation Award 2016. The Chairman's Innovation Award is an annual competition at Caterpillar. The goal of the competition is to award the team that delivers the most innovative technologies that impact the business results.

In June 2014, the author of this thesis has been awarded 3rd prize for best presentation at ResCon 2014, the annual research conference held in Brunel University. The Brunel University student conference aims to bring together research students from all subject areas to showcase the high caliber of research across the University. The presentations are evaluated by a committee of experts in the respective subject areas.

In June 2013, the author of this thesis presented a poster at ResCon 2013, the annual research conference held in Brunel University. The poster reached the top ten of best posters.

2 State of the Art: Supply Chain Optimization

The current chapter presents a general introduction to the problem of Supply Chain Optimization, or, as sometime referred to, the problem of Transportation Network Optimization. The chapter starts with a general definition of the problem of supply chain optimization, and provides several examples and references to the most relevant work in this area (section 2.1). The chapter continues with a description of the most common methods to address the optimization problem (section 2.2). Section 2.2 starts with a mathematical formulation of the problem. Sections from 2.2.1 to 2.2.4 provide a general overview of the most common approaches from the fields of mathematical programming, operational research, and artificial intelligence. In particular, the sections are limited to a brief description of the concepts. For more detailed discussions, surveys and literature reviews are referenced for the respective arguments.

2.1 Supply Chain Optimization

A supply chain or distribution network is a dynamic, stochastic and complex system that can be modelled as an entity containing a set of manufacturers, distribution centres and target customers [8]. Transportation models play an important role in logistics and supply chain management. Manufacturing enterprises are typically organized into

networks of manufacturing and distribution centres that acquire raw material, process it into finished goods and distribute those goods to end customers [9]. Cross-docking facilities are often utilized during the transportation of finished goods from source to destination whereby products and materials move through distribution centres without being stored. Goods are consolidated at cross-docking facilities to minimize operational costs [10].

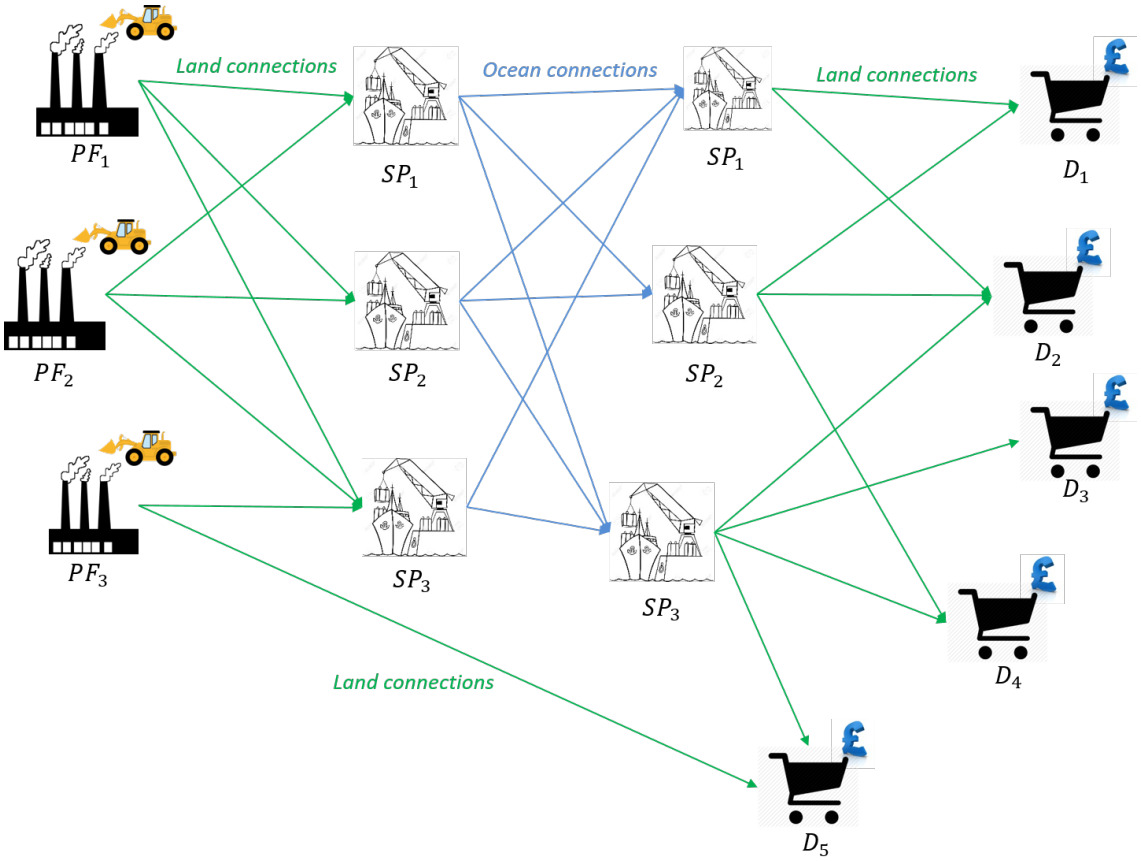

Figure 2.1 - Graphical representation of a supply chain as a network, where nodes PF_i are the production facilities, SP_j are the shipping ports, and D_k are the dealerships. The connections between nodes indicate the presence of a transportation link.

At the highest level, a supply chain is comprised of two basic, integrated processes: the Production Planning and Inventory Control Process, and the Distribution and Logistics Process [11]. Figure 2.1 shows a simple example of supply chain represented as a network. In the case of Figure 2.1, the nodes labelled PF_i represent production facilities, SP_j are shipping ports, and D_k are the customers or dealerships. Production capacities

and costs associated with each production facility allow integrating the process of production planning into the supply chain representation. Inventory typically is placed at the location of the production facilities and shipping ports. The distribution and logistics process consists of choosing paths on the network between production facilities and dealerships for sending products to the customers.

Optimizing a transportation network involves several competing factors, such as facility location, demand allocation, route and production planning [12]. Minimizing cost of logistics operations is not the sole objective of such optimizations, but may also include factors important to route planning such as minimizing total route distance, time, and environmental impact as well as maximizing profit, and network resilience to unplanned disruptions. Given such wide range of decision variables, the problem of transportation network optimization is known to be a difficult and complex problem to solve [13] [14].

The most frequently addressed objectives are the minimization of transportation/distribution *cost* and *time*. Minimizing cost aims to reduce total transportation costs, and therefore, increase profit. Minimizing time aims to reduce the total time required to transport finished goods from manufacturers to customers. A faster distribution plan usually entails a reduced need for inventory and often a higher service level. Cost and time are seen as the two most important measures in the optimization of transportation networks [15]; as such it's not surprising that they are the most frequently reported objectives in the literature.

Anghinofi et al. [16] analyzed a supply chain for the transportation of fruit and vegetables to satisfy the demand of a general market. The study deals with operational planning of freight transportation operations in an intermodal network. Freight transportation plays a key role in the economy of any country. In the context of supply chain and especially when dealing with long-haul transportation, intermodal transportation refers to the movement of goods in a network by exploiting different transportation modes. In the problem addressed in [16], the goods are moved in special refrigerated containers on either trains or trucks. Similar work has been presented by Boudahri et al. [17] for the transportation of meat products. The work is particularly focused on product quality and safety, and health regulations. High distribution standards do not only affect the agri-foods sector. In Zhang et al. [12], a distribution

network optimization including structure type, facility location and demand allocation was designed for a compound fertilizer supply chain. The market of compound fertilizer is client driven and the scientific and rational optimization design of the distribution network is an extremely important approach to make clients more satisfied and enterprises more competitive [12].

As natural resources continue to be depleted, environmental impact is also becoming a more important optimization objective. Manufacturing industries are being obliged to continuously improve their networks in order to reduce environmental waste [18]. Yeh et al. [19] introduced a green criteria to select the supplier partner, which involves four objectives such as cost, time, product quality and green appraisal score. In many countries, to address environmental damage and resource wastage issues, manufacturers are required to take the responsibility for the whole process of product life cycle, especially for recycling and reuse treatment of waste products. Du et al. [20] presented a closed-loop supply chain which considers the remanufacturing process and becomes a more effective way for coordinating development of economy and environment.

Many of the existing approaches which address environmental issues take a multi-objective optimization approach, combining several (possibly competing) objectives while optimizing the supply chain network. In general, multiple performance measures have to be taken into account simultaneously to properly design a supply chain and its operations [21]. Supply chain decisions are much more complex than treating them as single-objective optimization problems [21]. Many single-goal metrics are by definition in conflict with each other. A simple example is provided in Aslam et al. [21] as follows:

> *A short average lead time means the total time a product stored in the system is short, which means customer orders can be fulfilled within a shorter time and thus leverages the overall performance of the supply chain. A low Work-In-Progress (WIP) means the cost spent on transportation and inventory is lowered and thus is also highly desired. Therefore, to a decision maker, an ideal configuration is the one that maximizes delivery service level while simultaneously minimizing lead time and WIP.*

> *[…] Modeling a system using traditional optimization techniques in which one optimizes a single objective or a single weight-based objective to combine multiple objectives would very likely lead to misleading results in a dynamic system such as a supply chain.*

In a general Multi-Objective Optimization (MOO) problem, there exists no single best solution with respect to all objectives; a solution might be optimal in one objective but worst in the other objectives [21]. Usually, a set of non-dominated solutions is presented to the decision maker. The set of such solutions is called the set of Pareto-optimal solutions. A solution is defined as non-dominated (or Pareto optimal), if any improvement of one objective function results in a degrading of the value of some other objective function. If no additional ranking policy is provided, all non-dominated solutions are considered to be of equivalent quality. Figure 2.2 depicts the search space of a bi-objective optimization problem. The solutions belonging to the Pareto-optimal set are highlighted in red, the remaining dominated solutions in green.

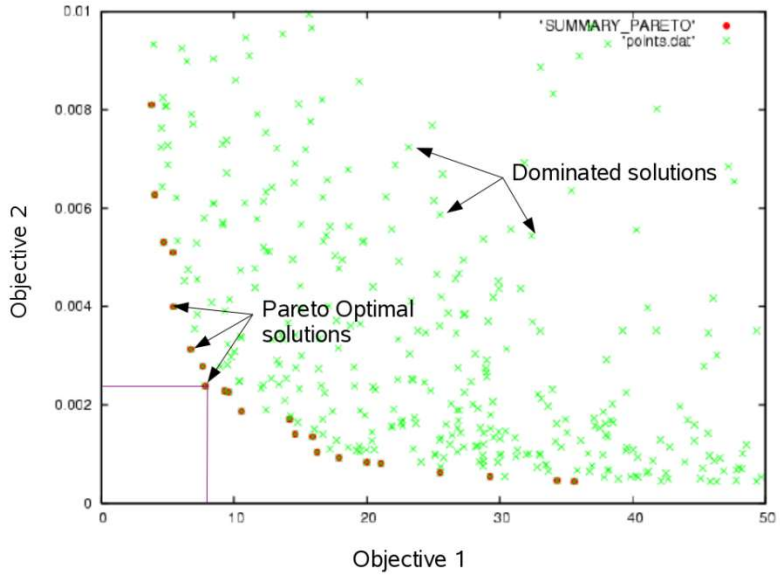

Figure 2.2 – Visualization of the search space of a bi-objective optimization problem. A solution is defined as Pareto optimal (or non-dominated), if any improvement regarding the first objective results in a degrading of the value of the second objective. The Pareto optimal solutions are highlighted in red, the dominated solutions in green.

A detailed discussion about multi-objective optimization is presented in chapter 4. Many methods have been proposed for multi-goal optimization in general, and for supply chain optimization specifically. Most of such methods, however, are very specific to either the problem or the optimization algorithm used. An example is the well-known Non-dominated Sorting Genetic Algorithm-II (NSGA-II) multi-objective optimization algorithm from Deb et al. [22]. NSGA-II is an improvement of a multi-objective evolutionary algorithm (MOEA), which has been proven to find a much better spread of solutions and has better convergence near the true Pareto-optimal front [22] than previously defined MOEAs. Despite its popularity and its success, the principles of NSGA-II may be difficult to apply in a context where, for instance, constraints on the optimization algorithm do not allow the employment of an evolutionary strategy.

The work presented in chapter 4 presents a classification of general multi-objective methods, independent from the problem or the optimization algorithm. The analysis provides a starting point for the selection and development of multi-goal solutions in real-world “industrial” applications.

In addition to the study of multi-objective optimization for supply chains, the study of creating resilient networks has grown in popularity in recent years [23]. The resilience of a transportation network can be expressed in terms of its ability to maintain operations and connectedness when faced with losses in structure or function, such as the unavailability of a manufacturer or route closures [24]. Transportation network disruptions can result in longer lead-times, a reduction in the ability to meet customer demand and increased operating costs. Building a resilient transportation network may result in increased flexibility and decrease the likelihood of node outages, but such functionality is often at the expense of increased operational costs.

Zhao et al. [24] proposed new network resilience metrics to measure the effects of disruptions on a military logistic network. The resilience is measured against both random and targeted disruptions, with particular focus on the case when disruptions are unknown. Previous research has revealed that the topology of the network has a great impact on its resilience [24]. However, the authors found little research to support this, and therefore, their study take a topological perspective for resilience metric definition. The new metric is based on the assumption that roles and function of nodes in a supply

network are heterogeneous, i.e. different types of nodes play different roles – an assumption often neglected by most research work. The authors developed a taxonomy to represent the different types of nodes, and two critical resilience metrics: availability and connectivity. Availability shows whether nodes in the supply network can get the supplies that they need to maintain normal operations. Typically, connectivity measures the size of the largest connected component. The authors extend such metric and consider the size of the largest functional subnetwork instead. The subnetwork is defined according to the topology of different nodes function.

While addressing the problem of resilience modelling for manufacturing enterprises, Holloway et al. [23] also found that a highly heterogeneous network is close to the optimal design. Disruptions on a manufacturing supply network are typically related to loss of capacity, loss of inventory, or significant change in demand. The significance of disruption is measured in degree of severity and duration. The authors in [23] identified two ways of achieving resilience: operational redundancy and inventory redundancy. The first consists of allocating sufficient additional capacity or having a different set of operations able to produce the same final product using independent resources. The second resides in allocating enough inventory to address the loss of capacity. However, retaining large inventory is often scorned at by many manufacturing companies.

Jiang et al. [25] and Ip [26] analyzed the resilience of a meat-food supply chain and an aircraft servicing and maintenance logistic network respectively. Again, the authors found that a network with redundant resources has the ability to recover more quickly to potential disruptions. Both in [25] and [26], a multi-supplier/multi-sourcing network model is discussed where several suppliers are available for selection.

2.2 Optimization Methods

Advanced analytical methods from the area of Operations Research may be applied to find the best distribution plan among all feasible plans of a supply chain. Typically, such analytical methods require the problem to be expressed in mathematical terms. The most common approach consists of defining a Mixed Integer Programming (MIP) model to describe the optimization problem. The main components of a MIP model are:

- A set of decision variables.

- The objective function to be either minimized or maximized.
- A set of constraints over the decisions variables.

The optimal or near-optimal solution may be then found via the application of exact optimization algorithms such as the Simplex algorithm designed for linear programming problems, or iterative methods such as the gradient descent for non-linear programming, or heuristics to provide approximated solutions to large-scale problem in a reasonable amount of time.

A basic example of MIP model for the supply chain optimization problem is as follows:

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \qquad \text{Objective function} \tag{2.1}$$

$$s.t.: \ \sum_{j=1}^{m} x_{ij} \leq S_i \qquad \forall i \in [0, n]. \text{ Capacity constraints} \tag{2.2}$$

$$\sum_{i=1}^{n} x_{ij} = D_j \qquad \forall j \in [0, m]. \text{ Demand constraints} \tag{2.3}$$

$$x \in \mathbb{N}_{+}^{n \times m},\ c \in \mathbb{R}_{+}^{n \times m},\ S \in \mathbb{R}_{+}^{n},\ D \in \mathbb{R}_{+}^{m} \qquad \text{Domains} \tag{2.4}$$

where, $m \in \mathbb{N}^{+}$ is the number of manufacturers, $n \in \mathbb{N}^{+}$ is the number of dealers with demand, S_i is the production capacity at manufacturer $i \in [0, i]$, D_j is the demand from dealer $j \in [0, m]$, x_{ij} is the number of units transported from manufacturer i to dealer j, c_{ij} is the cost of sending x_{ij} units of product from manufacturer i to dealer j. The above MIP model is from Dantzig 1963, chapter 3.3 [27] and $x \in \mathbb{N}_{+}^{n \times m}$ is the set of decision variables, eq. (2.1) is the objective function of the problem, eq. (2.2) and (2.3) are the constraints and eq. (2.4) the domains. All possible routes of the supply chain are represented by the variables $\boldsymbol{x}$ and a transportation cost c is associated with each route. The goal is minimizing the total transportation cost by deciding on which route and how much product to send from a production facility to a customer. The solution is acceptable only if the total amount of product shipped from a manufacturer is not more than its capacity and all customers receive the exact amount of product requested.

$x \in \mathbb{N}_{+}^{n \times m}$ is the set of all possible routes in the network between each production source to each dealer. Given a network configuration as shown in Figure 2.1, all possible routes must be listed in order to represent the problem with the model in

equations (2.1)-(2.4). The process of listing all the available routes may be computationally expensive. In the case of a four-layer network, the worst case complexity is the number of nodes in one layer multiplied by the number of nodes in each remaining layer – $O(n \times p \times p \times m)$ where $p \in \mathbb{N}^+$ is the number of shipping ports.

The problem of supply chain optimization may be seen as a specialization of the minimum-cost flow problem, a well-known optimization problem model, where the goal is to find a feasible flow of minimum cost in a network with capacity constraints and edge costs [28]. Let $D = (V, E)$ be an oriented graph, where V is the set of vertexes of the graph, and E the set of edges. Let be $n = |V|$, and $m = |E|$. For such graph, it is given the costs c_e for each edge $e \in E$, the capacities u_e for each edge $e \in E$, and the demands b_v for each node $v \in V$. An admissible flow (or pseudo-flow) in D is a vector $(x_e)_{e \in A}$ such that $x_e \geq 0, x_e$ is not greater than the capacity u_e, and, for each $v \in V$, the flow on the incoming edges minus the outgoing flow is exactly the demand b_v. The flow x_e on the edge e has a cost of $c_e x_e$. The goal of the minimum-cost flow problem is to find an admissible flow with the minimum overall cost. Equations from (2.5) to (2.7) define the linear programming model for the minimum-cost flow optimization problem. Such a model may represent a supply chain optimization problem where the supply network is defined as in Figure 2.1. Bevilacqua et al. [29] discussed in details a multi-goal optimization model for a distribution network where the number of layers is not set a priori. The model in [29] extends the one defined in (2.5)-(2.7).

$\min \sum_{(v,w) \in E} c_{vw} x_{vw}$	Objective function	(2.5)
$\sum_{w:(w,v) \in E} x_{wv} - \sum_{w:(v,w) \in E} x_{vw} = b_v$	$v \in V$. Flow constraints	(2.6)
$x_{vw} \in \mathbb{N}^+, 0 \leq x_{vw} \leq u_{vw}, c_{vw} \in \mathbb{R}^+, b_v \in \mathbb{N}^+$	$\forall (v, w) \in E$. Domains and capacity constraints	(2.7)

Both models defined by equations (2.1)-(2.4) and (2.5)-(2.7) are classified as pure-integer programming models, as the decision variables are non-negative integers. It is

well known that a binary integer programming model is NP-hard [30]. Thus, because every integer programming model can be written as a binary integer programming model by transforming one to the other in polynomial time [30], any integer programming model including the problem addressed in this research is also NP-hard. What characterized any NP-hard problem is that it is impossible to predict whether a solution can be computed in less than exponential time. Finding an optimal or near optimal solution can be computationally expensive [24]. Approximate algorithms are often employed to solve NP-hard optimization problems as they can often find high quality solution in a reasonable amount of time.

Since 1946-1947, when George B. Dantzig developed a general linear programming formulation to be used for planning problems by the US Air Force, many models for supply chain optimization have been proposed. Vidal et al. [31] identified a classification of this body of research into three categories: buyer-vendor coordination, production-distribution coordination, and inventory-distribution coordination. Moreover, the authors distinguished three level of planning depending on the time horizon: strategic if the time horizon considered is more than one year, operational for short-term decisions, and tactical, which falls in between. A comprehensive strategic problem consists of determining:

- the number, location, capacity, and type of manufacturing plants and warehouses to use;
- the set of suppliers to select;
- the transportation channels to use;
- the amount of raw materials and products to produce and ship among suppliers, plants, warehouses, and customers;
- the amount of raw materials, intermediate products, and finished goods to hold at various locations in inventory.

With some consideration and the proper assumptions, the model defined by equations (2.1)-(2.4) may arguably represent all the strategic problems summarized above. Deciding which route x_{ij} to ship product through inherently determines also the manufacturing plants and warehouses to use, the capacity to allocate, and the transportation mode. Similarly, if x_{ij} connects a supplier to the production facility or

warehouse, the model represents the problem of supplier selection. The amount of raw materials and products to produce and ship is given by the solution vectors – the values assigned to the variables x_{ij}. A comprehensive example of a binary-integer programming model for a three-tier supply chain is given in Musa et al. [30]. The three tiers in the network represent the set of external supplier, the set of cross-docking facilities, and the customers. The objective is to minimize the total transportation cost.

If the set of routes x includes also routes to new potential sites, the model in (2.1)-(2.4) describes a problem of new locations selection and supply network expansion. The cost coefficients c would include the setup costs for installing a new production facility or warehouse. Ko et al. [32] presented a binary-integer model which considers the problem of new potential sites allocation. The model actually describes a dynamic integrated forward/reverse supply network. A logistics network is defined as reverse, if returns policy and repairs operations are also considered in the optimization. The simultaneous consideration of forward and reverse flow is quite important when improving the customers' experience.

The problem of inventory placement is one of the most challenging in the area of supply chain optimization, and it has many interpretations. Typically, the amount of inventory placed at specific location is based on predictions of customer demand, and its evolution over time. The amount of inventory allocated to guarantee a minimum service level is known as *safety stock* [33]. Safety stock is a buffer, which assures the availability of excess product in the case the consumer demand exceeds the expectation. Two different types of inventory, which significantly affect the transportation times and costs, are *cycle stock* and *in-transit inventory*. Cycle stock is often defined as the inventory used to satisfy the customer demand in a given time window and making the shipment process less time-consuming. In-transit inventory is the en-route goods, the amount of product "trapped" in transition from one node of the supply chain to the next.

The amount of time that product must be held as cycle stock inventory depends on the promised time of delivery. If the travel time from the beginning of the supply chain to the customer is greater that the promise time, inventory must be allocated on a node (e.g. warehouse, or distribution center) closer to the customer. Altiparmak et al. [34] defined a programming model which considers the delivery time and the maximum

allowable delivery time. The difference between travel time and promise time defines the amount of time inventory has to be held to a given location. The optimization occurs on the product quantity. The cost of cycle stock inventory is a function of the quantity and duration for which product must be stored.

When optimizing safety stock inventory, the decision variable is usually the rate of replenishment of the inventory in a given location. The objective is to minimize the expected lost sales due to stock-outs, and simultaneously inventory holding cost. The demand is modelled through a statistical distribution and the expected value is considered. Nozick et al. [33] presented a model of a multi-product two-echelon inventory optimization problem. Both the demand and stock-out probability are modelled with a Poisson distribution. Nozick et al. [33] used the results of the inventory optimization to address a problem of location selection of distribution centers. This type of decision rule on inventory allocation allows the safety stock inventory costs for each potential distribution center to be estimated and factored into a fixed-cost facility location model [33].

Sadeghi et al. [35] provide another example of model for safety stock optimization. The aim in [35] is to find the order size, replenishment frequency of the retailers, and the routing tour to maximize the production system reliability of a supply chain consisting of a single vendor and multiple retailers.

Vidal et al. [31] focused their review on strategic production-distribution MIP models, distinguishing between local and global logistics systems. Global supply chains are inherently more challenging to manage in an international scenario, the flow of cash and the flow of information are more important and difficult to coordinate than they are in a single-country environment. The inclusion of different taxes and duties, differential exchange rates, trade barriers, transfer prices, and duty drawbacks is fundamental for a model to more accurately represent the real system. In addition, sources of uncertainty and qualitative factors, such as government stability and general infrastructure of a particular country, are critical issues for the strategic design of a global supply chain [31].

During the last decades of the twentieth centuries, many industries witnessed a market globalization and the need for expanding their supply network arose. Meixell et al. [36]

selected 18 major research articles from a set of more than a hundred publications on the subject of global supply chain management. From the review in [36], the main decision variables for global supply networks appear to be:

- Facility selection;
- Production/shipment quantities;
- Supplier selection.
- Placement of inventory.

The performance of management operations is mostly measured as:

- Maximize after-tax/operating profit;
- Minimize production/material/labour/transportation/utility costs;
- Robustness and value of flexibility across pre-defined scenarios;
- Maximize utility for manufacturers and retailers.

Supply networks are also characterized by the degrees of integration and accuracy in representing the business environment. Factors to measure degrees of integrations are:

- Supply chain level;
- Use of Bill of Materials (BOM);
- Coordination of decision.

Typically, transportation costs are the largest contributors for the total logistics costs in global supply networks and the integration of freight transportation function is one of the most studied issues in the scientific logistics literature. Bravo et al. [37] investigated the management of freight transport costs in supply chain optimization models. The authors analyzed two groups of papers from high-ranking journals. The first group of papers spans from 2009 to 2012 to analyze current trends. The second one provides a perspective of the historical behavior and comprises papers from 1974 to 2008. Both groups of papers have been analyzed against a novel taxonomy, whose main aspects are: model characteristics, decision variables, objective functions, product details and transportation modes. Model characteristics consist of network and chain configuration, modelling approach, degree of supply chain integration and level of uncertainty. The supply chain configuration is quite heterogeneous in all considered papers; nevertheless,

it is always a graph of at least two layers, where the nodes are a subset of Supplier, Plant, Distribution Centre, Retailer and Customer selected in such order. The most widely modelling approach is the design of a mixed integer-programming model. The preference is slightly in favor of linear models instead of non-linear and the main objective strategy is cost minimization. In practice, it is desirable to perform trade-off analysis between different types of objectives, e.g. finding the solution that both minimize transportation cost and maximize service level. It is also common to have uncertainty in real-world data that may be handled with stochastic variables. Despite their importance in practical applications, multi-goal and stochastic models due to their intrinsic higher computational complexity have not been addressed properly in the past decades. According to the taxonomy in [37], location selection, the amount of product to be shipped on a given route, and inventory placement are the most considered decision variables. Cost minimization is the preferred strategy in optimization processes, and it is typically preferred over customer service level maximization. Service level is often measured as percentage of demand satisfaction and inventory replenishment policies or safety stocks. Both metrics impact the transportation function as the former forces the distribution function to find the most efficient ways to prevent any type of non-fulfillment, and the availability of inventory depends on the effectiveness of the replenishment strategy from the sources. In recent papers, more attention has been paid to service time (i.e. time windows when delivery has to be made) with occasionally including parameters such as types of vehicle, loading and unloading times, and limited transportation capacity. On the contrary, very few studies have been pursued involving physical characteristics of the product (e.g. volume and weight), and microeconomics considerations. The type of market may influence the price point of products, and therefore plays an important role when the objective is profit maximization. Moreover, very few studies explicitly deal with fuel consumption, topography of the roads and CO_2 emissions.

A summary of the trends and shortcomings emerging from the body of research reviewed in [37] is as follows:

1. The objective of cost minimization is often preferred over profit maximization.
2. Transportation costs are effectively represented through cost per unit to be shipped and cost per unit distance.

3. From the literature, transportation costs appear to be in direct conflict with in-transit inventory, shipment size, and the number of locations to be opened. Such conflicts indirectly affect multi-echelon inventory, service level, transportation times, and economies of scale in production. Despite the level of consensus for the above mentioned relationships, the in-transit inventory is often neglected when modelling supply chains.
4. Employing outsourced vehicle fleets increased the importance of factors such as product characteristics and route attributes when defining freight transportation rate, factors which are often omitted in mathematical models.
5. In a scenario where demand is stochastic, the overall profit of the organization will also be stochastic. Even if uncertainty of demand is modelled, the minimization of costs scenario does not provide insights for the case of extremely low demand. A substantial demand decrease would certainly affect the profit and the organization would be significantly affected by a cost minimization strategy.

Chapter 3 depicts the model of Caterpillar's global supply chain. Points from 1 to 4 have been considered in the definition of the model and addressed. The uncertainty of the demand has not been directly addressed. Several possible demand scenarios are generated by Caterpillar before the optimization process starts. Each scenario is then solved independently. Having available several distinct computing platforms, this approach allows avoiding the overhead due to a stochastic computation. Moreover, the results of the distinct scenarios may be presented separately to the managers or decision makers, who can perform a what-if analysis.

Regarding point 1, both an objective function for cost minimization and profit maximization are considered. The transportation costs are a function of cost per unit shipped and cost per distance (point 2), and the in-transit inventory is included in the transportation cost calculation (point 3). Finally, all the main physical and financial characteristics of the products and components have been factored in the model. Energy consumption and average travel time, combined with transportation mode, provide indirect information on routes characteristics. As suggested by the authors in [37], time arguably provides more information than distance. If the vehicle speed is low, this may

suggest some type of road congestion and might result in an increase in fuel consumption and an increase in the opportunity cost of the in-transit inventory [37].

The following sections (from 2.2.1 to 2.2.4) describe the most adopted optimization techniques found in the literature to solve the problem of supply chain optimization. The methods are:

1. Exact optimization algorithms based on linear programming.
2. Approximated methods or heuristics.
3. Meta-heuristics.
4. Hyper-heuristics.

Ogunbanwo et al. [1] analyzed the most relevant work in the area of supply chain optimization applied to real business environment, developed in the last decade. The authors listed the objective functions considered in each analyzed paper, and the method employed for optimization. As the analysis in [1] is part of the work presented, the list is reported in Table 2.1. In the table, LP refers to any exact optimization algorithm which is based on a linear programming representation of the problem. BB stands for Branch and Bound, HE for Heuristic, ES for Evolutionary Strategy, ACO for Ant Colony Optimization, and PSO for Particle Swarm Optimization. The objective functions identified in the considered works are: minimization of travel distance, transportation time and cost, and environmental impacts, and maximization of resilience, service level, and product quality.

Wherever the information is available, Ogunbanwo et al. [1] measured the complexity of the supply network developed. In graph theory, there are several ways to measure the complexity of a graph or network. They are usually based on either the length of whole network, the length of the shortest path, the number of cycles in the graph, or its level of connectivity. The *beta index* and the *number of cycles measures* have been selected as measure of complexity. The *beta index* takes into account the level of connectivity of a graph, and requires knowing the number of both nodes and edges. Let e be the number of edge in a graph, and let v be the number of vertex in the same graph, the beta index is defined as $\beta = \frac{e}{v}$. The *number of cycles* is based on the maximum number of independent cycles in a graph, and it requires to know the number of nodes, links, and sub-graphs. Let p be the number of sub-graph, the *number of cycles* measure is defined

as $u = e - v + p$. Table 2.2 reports the value for these measures for the works where it was possible to extract the information needed. The same metrics have been used to measure the complexity of Caterpillar's network. The results are reported in chapter 3.

Table 2.1 – List of papers analyzed in Ogunbanwo et al. [1]. The methods employed to optimize the problem and the objective functions considered are reported for each work investigated. LP stands for exact optimization algorithm based on linear programming. BB stands for Branch and Bound, HE for Heuristic, ES for Evolutionary Strategy, ACO for Ant Colony Optimization, and PSO for Particle Swarm Optimization. The objective functions identified in the considered works are: minimization of travel distance, transportation time and cost, and environmental impacts, and maximization of resilience, service level, and product quality.

Author, year	Algorithms	Objective(s)						
		Distance	Cost	Resilience	Time	Service level	Product quality	Environmental issues
Xiang *et al*, 2012 [38]	ACO				✓			
Bevilacqua *et al*, 2012 [29]	ES		✓		✓			
Che, 2012 [39]	PSO		✓		✓			
Sadjady *et al*, 2012 [40]	LP		✓		✓			
Zhao *et al*, 2011 [24]	PSO			✓				
Boudahri *et al*, 2011 [17]	LP	✓	✓					
Huang *et al*, 2011 [41]	PSO		✓	✓				
Utama *et al*, 2011 [42]	ACO	✓	✓			✓	✓	✓
Anghinolfi *et al*, 2011 [16]	IP and ACO		✓					
Yeh *et al*, 2011 [19]	ES		✓		✓		✓	✓
Georgiadis *et al*, 2011 [43]	BB		✓					

Zhao *et al*, 2011 [44]	PSO		✓					
Musa *et al*, 2010 [30]	ACO		✓					
Han *et al*, 2010 [45]	ACO	✓						
Chen *et al*, 2010 [46]	ES		✓					
Chang, 2010 [47]	ES		✓					
Ghoseiri *et al*, 2010 [48]	ACO	✓	✓					
Che *et al*, 2010 [49]	ES		✓		✓		✓	
Wang, 2009 [50]	ACO			✓				
Jiang *et al*, 2009 [25]	ES			✓				
Ding *et al*, 2009 [8]	ES		✓			✓		
Lin *et al*, 2009 [51]	ES		✓					
Chan *et al*, 2009 [52]	ACO				✓			
Bidhandi *et al*, 2009 [53]	LP		✓					
Du *et al*, 2009 [20]	LP		✓					
Lau *et al*, 2009 [54]	ES		✓					
Lin *et al*, 2008 [55]	ACO		✓		✓	✓	✓	✓
Wen *et al*, 2008 [14]	PSO		✓					
Qin *et al*, 2008 [56]	PSO		✓					
Huang *et al*, 2008 [57]	PSO		✓					
Farahani *et al*, 2008 [58]	ES		✓					
Caldeira *et al*, 2007 [59]	ACO		✓					
Fu *et al*, 2007 [60]	ACO		✓					
Wang , 2009 [50]	ACO		✓	✓				
Chen *et al*, 2007 [61]	LP		✓		✓			
Altiparmak *et al*, 2006 [34]	ES		✓		✓			
Ding *et al*, 2004 [62]	ES		✓		✓			

Ji *et al*, 2005 [63]	ES				✓			
Pimentel *et al*, 2013 [64]	HE		✓					

Table 2.2 – Supply network complexity analysis. LP stands for exact optimization algorithm based on liner programming. BB stands for Branch and Bound, HE for Heuristic, ES for Evolutionary Strategy, ACO for Ant Colony Optimization, and PSO for Particle Swarm Optimization. Let e be the number of edge in a graph, and let v be the number of vertex in the same graph, the *beta index* is defined as $\beta = \frac{e}{v}$. Let p be the number of sub-graph, the *number of cycles* measure is defined as $u = e - v + p$. Source of the analysis is A. Ogunbanwo et al. [1].

Author, Year	Algorithms	No. of Nodes	No. of Edges	No. of Sub-graph	Beta-index	No. of Cycles
Yu, 2005 [65]	ACO	2300	3200	61	1.391304348	961
Ko *et al*, 2007 [32]	ES	28	46	1	1.642857143	19
Lin *et al*, 2009 [51]	ES	35	210	1	6	176
Wang *et al*, 2011 [66]	ES	14	91	1	6.5	78
Hosseinzadeh *et al*, 2012 [67]	ES	15	44	1	2.933333333	30
Zhao *et al*, 2011 [44]	PSO	1000	1815	1	1.815	816
Huang *et al*, 2011 [41]	BB	14	40	1	2.857142857	27
Jiang *et al*, 2009 [25]	ES	20	64	1	3.2	45
Wang, 2009 [50]	ES	6	9	1	1.5	4
Sadjady et al, 2012 [40]	LP	150	3600	1	24	3451
Bevilacqua et al, 2012 [29]	ES	9	18	1	2	10
Chang, 2010 [47]	ES	80	120000	1	1500	119921
Ghoseiri et al, 2010 [48]	ACO	4000	61783	1	15.44575	57784
Wang , 2007 [50]	ACO	14	39	1	2.785714286	26

2.2.1 Linear Programming

As defined by Dantzig in [27], liner programming is concerned with describing the interrelations of the components of a system. The linear programming approach is to consider a system as decomposable into a number of elementary functions, the *activities*. Each activity is associated with a quantity called *activity level*. Acting on the activities level affects the value of the objective function. For instance, the activities of a transportation optimization problem are the actions of shipping from a production facility to a customer. Representing the set of activities often results in a system of linear inequalities and equations; when this is so, it is called a linear programming model.

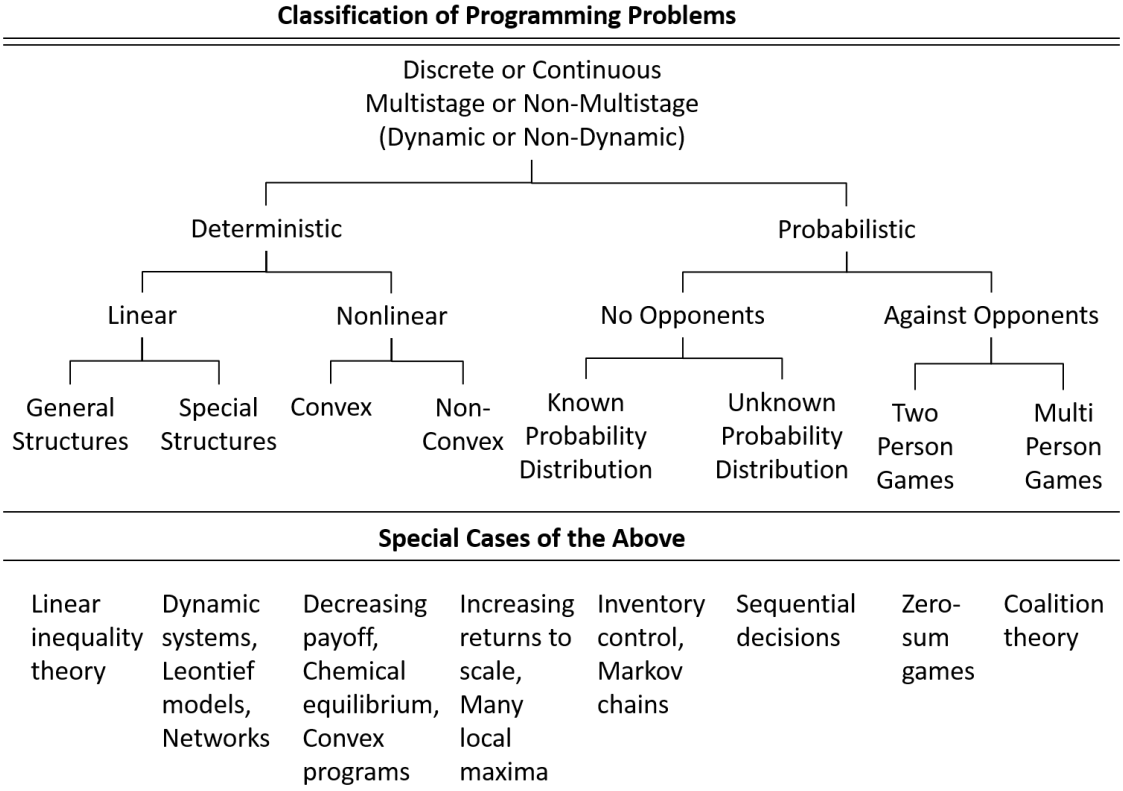

Figure 2.3 – Classification of programming problems from Dantzig [27].

Figure 2.3 shows a basic classification of possible programming problems. This session focuses on methods for solving linear programming problems. Occasionally, nonlinear problems may be solved too by the same techniques, if the system of constraints defines

a convex set: a nonlinear convex programming problem may be reduced to a linear one. With proper transformations and assumptions, certain probabilistic problems may also be reduced to exact linear programming problems.

To be classified as a linear programming model, the system must possess certain properties of proportionality, non-negativity, and additivity. The system satisfies the property of proportionality if a change in a decision variable results in a proportional change in the variable's contribution to the system. Negative quantities of variables are not possible and the value of the system is equal to the sum of the contributions of the value of each variable.

The basic form for the problem of linear programming consists of finding values for a set of non-negative variables that satisfy a system of linear equations and minimize an objective function. Developed in 1947 by George Dantzig, the simplex algorithm is one of the most popular methods to solve linear programming problems and possibly the most implemented by commercial solutions. An additional requirement for the application of the simplex method is that the linear program is expressed in standard form. The standard form of a linear program is as follows:

$$\max c^T \cdot x \tag{2.8}$$

$$s.t.: \ Ax \leq b \qquad \text{Feasible region} \tag{2.9}$$

$$x_i \geq 0 \qquad \forall i \in [0, n]. \text{ Non-negativity constraints} \tag{2.10}$$

with $x \in \mathbb{R}^n$ being the vector of decision variables, and $c \in \mathbb{R}^n$ the cost coefficients. The matrix $A \in \mathbb{R}^{m \times n}$ and the vector $b \in \mathbb{R}^m$ together define the set of feasible solutions. Any linear program may be converted into one in standard form. From a geometric perspective, eq. (2.9) and (2.10) define a convex polytope. A convex polytope is a polytope which satisfies the property of a convex set. A set is said to be convex if, for every pair of points in the set, all the points in the segment joining the pair are also part of the set. It can be proved that if the convex polytope is bounded, then the optimal solution is at least one of its vertices. The simplex algorithm consists of

iteratively visiting all the vertices of the polytope until the optimal solution is found. Figure 2.4 depicts the feasible region of a simple bi-dimensional linear program and the steps of the simplex algorithm to find the optimal solution. For details and the mathematical proofs of the properties of the simplex algorithm refer to [27].

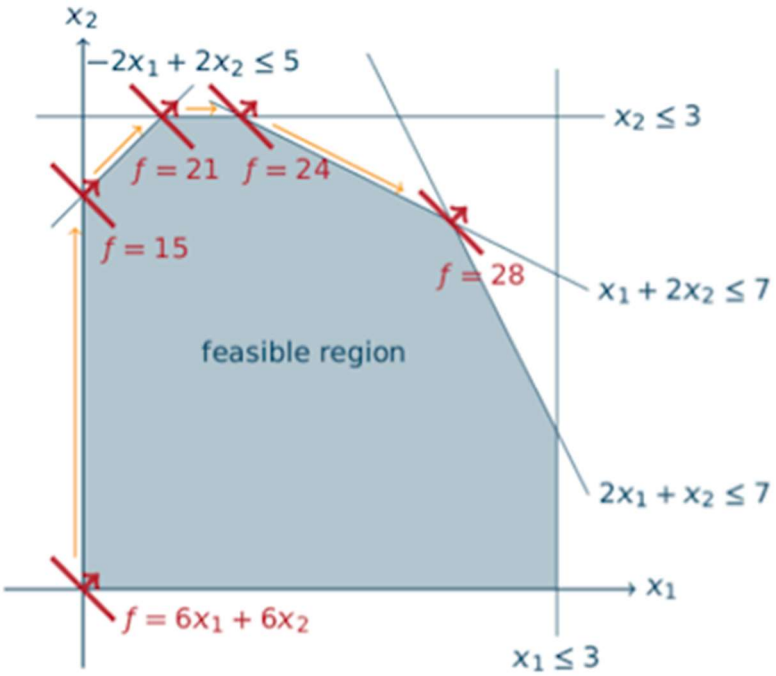

Figure 2.4 – Bi-dimensional linear program with *f* being the objective function. The yellow arrows show the steps of the simplex algorithm while searching for the optimal solution.

The simplex method has been shown to be remarkably efficient in many real-world applications. Efficiency tests on random problems showed that the algorithm solve most problems in a cubic number of steps. It has been proved however that the worst case complexity of the algorithm is exponential in the number of variables. Nevertheless, the algorithm is still applicable to real-life problems by accepting solutions for which the quality is within a pre-determined distance from the optimum. According to Cordeau et al. [68], data estimates are prone to errors and therefore solving real-world problems to optimality is usually not meaningful.

The most used commercial implementations of the simplex algorithm are IBM CPLEX [69], and LINGO [70]. A free and open source alternative from GNU is GLPK [71].

Alternatives to the simplex method are special-purpose exact techniques such as branch-and-bound, branch-and-cut, column generation, and decomposition methods [72]. Realistically sized problems often result in models with a large number of variables and constraints. When it is not practical to solve a problem to optimality, heuristic methods may be engaged and the most popular ones are: Lagrangian relaxation, linear programming based heuristics and metaheuristics [72].

Sadjady et al. [40] designed a mixed integer linear programming model to solve a two-echelon multi-commodity supply chain network problem, which has been solved using LINGO [70]. The same tool has been applied by Boudahri et al. [17] to optimize a supply chain for agri-foods products. It is worth noticing that the authors decomposed the entire problem into two sub problems, and each sub problem was solved in a sequential manner, to get the final solution. Exact methods can be flexible, and solve complex problem. Georgiadis et al. [43] proposed a detailed mathematical formulation for the problem of designing supply chain networks comprising multiproduct production facilities with shared production resources, warehouses, distribution centers and customer zones and operating under time varying demand uncertainty. They captured uncertainty in terms of a number of likely scenarios possible to materialize during the lifetime of the network. The problem was formulated as a mixed-integer linear programming problem and was solved to global optimality using standard branch-and-bound techniques.

2.2.2 Heuristics

Although, it is desirable to have the theoretical certainty that the solution found is also the optimal one, exact methods are not always applicable. As discussed in section 2.2.1, simplex-based methods require the problem model to satisfy certain properties. Moreover, for specific types of problems, the runtime complexity could be exponential in the number of variables. Even for methods which are not based on the simplex algorithm, the dimensionality of real-world problems might make their application impractical.

It is worth mentioning that, when developing a new optimization application, it is a good practice to always attempt to design the problem model as a linear program. The

process of designing the linear program is itself a powerful analysis tool which is likely to highlight many aspects of the problem. Moreover, the increasing power of computing hardware and efficiency of solvers might make it a practical solution.

Nevertheless, when it is not possible or is impractical to apply exact methods, finding approximate solutions may be a good tradeoff, if the solution is close enough to the optimal and the time/resource required to find it, is reasonable. Heuristic techniques are approaches which implement practical methods to solve optimization problems. Typically, there is no mathematical proof that the optimal solution will be found, but the methods are expected to work well in practice. Heuristics take advantage of specific properties of the problem, and knowledge acquired from past experience.

In real-world applications specifically, finding approximated high-quality solutions in a reasonable time is sufficient, due to some intrinsic characteristics of real-world situations, such as:

- In real domains, most of the *key parameters* are unknown, and they are usually approximated. Therefore, the application of an exact method is likely to find a non-optimal solution anyway, but requiring more time and resources.
- Sometimes, the goal of analysts might be of *finding a possible solution* in a very short time, with the purpose of evaluating the situation. This is the case of a Decision Support System (DSS). A DSS is a computer-based information system used to support business or organizational decision-making activities. These systems often rely on optimization algorithms and help to make decisions, which may be rapidly changing and not easily specified in advance.
- Optimization algorithms are often applied in *real-time contexts*, in which a good solution is needed in matter of few seconds.

Pimentel et al. [64] developed a mathematical model for the stochastic capacity planning and dynamic network design problem. In [64], the network configuration is dynamic and the facility location decisions, the network design, and the capacity allocation are addressed simultaneously in an integrated framework. Such integration scales up the complexity of the problem. The authors proposed a Lagrangian heuristic to determine reasonable bounds in a fair amount of time. The basic idea is to relax a subset of constraints in order to derive a Lagrangian problem which should be easier to solve

and whose optimal value provides a bound on the optimal solution of the original problem [64].

For the majority of combinatorial optimization problems, it is possible to design a specific heuristic to leverage specific properties of the problem and the knowledge acquired from past experience. As a matter of fact, an optimization algorithm very often is simply the set of steps which are manually performed to find a solution. The quality of the solution produced by heuristics strongly depends on the level of experience used to design the procedure.

In the last couple of decades, both academics and practitioners have grown an interest in approaches based on general heuristics. These approaches are applicable to many different kind of problems and they usually have similar if not better performance than specific heuristics, in terms of both quality of the solution and time required to find it. Approaches based on general heuristic may be categorized as:

- *Constructive heuristics*: build the solution in an iterative fashion. They start from the empty set, and add an element at each iteration. Usually, the element to add is chosen based on a local optimality criterion. Greedy algorithms are constructive heuristics.
- *Meta-heuristics*: general methods which are completely independent from the application domain. These methods define components and their interactions. The components must be specialized for the specific problems. The most well-known meta-heuristics are: *Local Search*, *Simulated Annealing*, *Tabu Search*, *Variable Neighborhood Search*, *Greedy Randomized Adaptive Search Techniques*, *Genetic Algorithms*, *Scatter Search*, *Ant Colony Optimization*, *Swarm Optimization*, *Neural Network*, etc.
- *Approximated algorithms*: heuristic methods which assure a minimum level of performance, i.e. it is possible to formally prove that the computed solution would never be worse than the optimal (unknown) solution for more than a fixed percentage (which most of the times is quite high).
- *Hyper-heuristics*: methods based on Artificial Intelligence and Machine Learning techniques, which are able to discover optimization methods, and

automatically adapt them to several problems. The search is performed in the heuristics space, not in the solution one.

2.2.3 Meta-Heuristics

Meta-heuristics are search algorithms which combine basic heuristic methods in higher level frameworks aiming at efficiently and effectively exploring the search space [73]. Blum et al. [73] collected some definitions of meta-heuristics from the most prominent research in the area of combinatorial optimization. To summarize the analysis in [73], a meta-heuristic is a high-level iterative process which guides and modifies the operations of subordinate heuristics. The resulting strategy efficiently produces high-quality solutions and may be applied to a wide set of different problems. Meta-heuristics typically are non-deterministic as they rely on probabilistic decisions during the search. Moreover, they incorporate mechanisms to avoid local minima and getting trapped in confined areas of the search space. Nowadays, more advanced meta-heuristics employ memory mechanisms and use the knowledge acquired to guide the search.

The specific details of meta-heuristics may vary significantly from one to another; however, two steps are always implemented and dynamically balanced: *diversification* and *intensification*. As defined in [73], diversification generally refers to the exploration of the search space, whereas intensification consists of exploiting the accumulated knowledge of the search space. The balance between diversification and intensification as mentioned above is important, on one side to quickly identify regions in the search space with high quality solutions and on the other side not to waste too much time in regions of the search space which are either already explored or which are unlikely to provide high quality solutions [73].

Blum et al. [73] summarized the main classifications of meta-heuristics. The classifications depend on the viewpoint or characteristic selected and they are as follows:

- *Nature-inspired vs. non-nature* – Classification based on the origins of the algorithm. Examples of nature-inspired algorithms are the Evolutionary Strategy and the Ant Colony Optimization. Non-nature inspired algorithms are Tabu Search and Iterated Local Search.

- *Population-based vs. single point search* – Classification based on the number of solutions used at the same time. Methods which use only one solution describe a trajectory in the search space. Population-based meta-heuristics describe the evolution of a set of points in the search space.
- *Dynamic vs. static objective function* – One method to escape from local minima is to modifying the search landscape by altering the objective function with the new information collected during the search. A meta-heuristic that implements this mechanism is Guided Local Search.
- *One vs. various neighbourhood structures* – Some meta-heuristics swap between fitness landscapes in order to increase the chance of diversify the search.
- *Memory-usage vs. memory-less methods* – A key aspect of meta-heuristics is the use they make of the search history. Some algorithms employ memory mechanisms in order to make better either short or long term decisions. Methods which do not remember the search history decide on the next action only based on the current state and, in a sense, they perform a Markov process.

Blum et al. [73] identified the population-based vs. single point search classification to be one allowing a more significant description and comparison of the algorithms. Moreover, many new meta-heuristics have been developed by integrating single point search algorithms with population-based ones (see Blum et at. 2011 [74]).

The following sections provide a general overview of the best known meta-heuristics with a particular focus on the Ant Colony Optimization algorithm, which is the main optimization method used in the present work.

A. Local Search

As defined in [73], the Local Search meta-heuristic is a single-trajectory search method where each move is only performed if the resulting solution is better than the current one. The algorithm is also known as iterative improvement and it stops as soon as a local minimum is found. Since the algorithm has no mechanism to recognize and avoid local minima, the performance is usually quite unsatisfactory. As a result, in literature, several techniques have been developed to allow meta-heuristics avoid local minima. They typically consist in improving the termination conditions such that the search is

not terminated when simply a local minimum is found, but rather, for instance, restarted from a random location of the search space. As shown in chapter 5, termination conditions play a critical role in the effectiveness of a meta-heuristic-based optimization system.

B. Simulated Annealing

Blum et al. [73] identify Simulated Annealing being the oldest among the meta-heuristics and one of the first to explicitly tackling the problem of local minima. The fundamental idea is to allow moves resulting in solutions of worse quality than the current one. In order to assure the search terminates in a finite amount of moves, the probability of worsening moves is reduced over time. The algorithm is inspired by the annealing process of metal and glass where the temperature is gradually reduced until reaching a low energy state. The temperature represents the probability of worsening moves and it is computed following the Boltzmann distribution. In practice, the search process consists of two strategies: random walk and iterative improvement. At the beginning, when the temperature is still high, random moves are more likely and the bias towards improvement is low. As the temperature is reduced, less random moves are accepted and the algorithm behaves more like a local search.

The cooling schedule is crucial to the effectiveness and efficiency of the algorithm. It has been proven that a logarithmic schedule allows the search to converge to the global optimum. When implemented however in real-world scenarios, such a schedule is too slow and a lower runtime is preferred over convergence guarantee. In practice, cooling schedule and initial temperature are adapted to the problem instance, since a strong relationship exists between such parameters and the search landscape.

C. Tabu Search

Tabu Search is a meta-heuristic based on Local Search which is capable of avoiding local minima by systematically employing memory methods. The history of the search is kept in a *tabu list* which is a list of the most recently visited solutions. The neighbourhood of the current solution is thus restricted to the solutions that do not belong to the tabu list [73]. The use of a tabu list prevents the search from returning to recently visited solutions; therefore it prevents from endless cycling and forces the search to accept even up-hill moves [73]. The length of the tabu list is the key parameter

of the algorithm. A long tabu list will force the search process to explore larger regions of the search space as the number of solutions which are forbidden to revisit is high. On the contrary, a small tabu list will follow the search and be focused in a concentrated area of the search space. Blum et al. [73] review methods to dynamically manage the length of the tabu list and the information stored in it.

D. Evolutionary Strategy

Evolutionary Strategy (ES) is a meta-heuristic inspired by evolutionary theories of natural selections and genetics. The theory of evolution by natural selection formulated by Charles Darwin in "Origin of Species" in 1859, states that only the elements of a population that are the fittest to their environment will live and reproduce, passing on their genetic information. Any mutation that makes an individual better suited for its environment will be inherited by the new generation. After a few generations, the new population will be well adopted to live in the environment and the individuals will have the most chance of survival.

Evolutionary Strategy is a computational model which implements the key ideas of the theory of evolution. In the context of optimization, a feasible solution to the problem would be an element of the population. Mutation operators would change aspects of the solution and only the most promising ones will be combined to produce a new "generation" of feasible solutions which should be closer to the optimal one.

The first step when applying an ES to an optimization problem is to define the elements of the population and their genetic makeup (i.e. the information store by the genes of the individual). As in biology, the value of genes within a subject will affect the characteristics of the individual and therefore impact on the solution to the problem. In the case of a supply chain, as investigated in this study, the genes may represent the nodes of the transportation network, i.e. the manufacturing facilities, the shipping ports, and the dealers that constitute the network. The genes then are grouped together based on the individual they belong to. An element of the population represents a full solution to the problem under investigation. In the case of supply chains, the units of the population represent the entire path travelled by all the shipped goods of the proposed solution; in the context of the problem under investigation this refers to the production

facilities, the dealerships through which they are sold and all the sea ports they pass through when travelling across the network. As said, the algorithm starts with a population, a set of feasible starting solutions. Typically, such solutions are generated randomly or by some simple heuristic. Some of the individuals from the initial population will be more fit to the environment than others. According to the theory of evolution and natural selection, the fittest elements will survive, reproduce and therefore spread their genetic heritage. Usually ES algorithms use operators called *recombination* or *crossover* to recombine two or more individuals to produce the next generation of population [73]. The "fitness" of each individual is determined by the calculation of a fitness value. The fitness value is defined as the quality a solution has with respect to the objective function of the optimization problem and the amount of infeasibility. The generation of infeasible solutions can be handled in one of two ways: either constraints are incorporated in the algorithm such that infeasible solutions are not possible, or a penalty can be applied to the fitness value to account for the level of infeasibility. Once the fitness of each individual is determined, the new population is generated from the fittest solutions, with newer, more fit elements replacing the less fit ones in the population.

The process is repeated and at each iteration the "fittest" individuals are saved and reproduced from until such time as the termination criteria are satisfied. The termination criterion is determined by the designer when the ES algorithm is implemented, and traditionally uses one the following conditions:

- An upper limit on number of generations has been met.
- An upper limit on number of evaluations of fitness has been reached.
- The probability of achieving significant improvement in the next generation is negligible.

The first 2 criteria require prior knowledge about the problem domain and how the algorithm will perform, whereas the third criterion is dynamic and is a function of the algorithm used. This option is the more commonly used of the three described. The first two may appear to be the simpler, but the algorithm designer needs to understand the algorithms convergence characteristics in order to determine how many generations

must be completed before the cost of computation is no longer offset by the generation of a more optimal solution.

The third criterion is based on the actual convergence of the solutions produced by the algorithm and once they have converged to an 'optimal' solution the algorithm terminates. For example, iteration can be terminated when the same solution has been found for 'n' previous iterations.

The most simple diversification strategy for ES is to use a mutation operator [73]. Typically, the mutation operator consists of applying a small random permutation to an individual, introducing some noise. In order to avoid premature convergence towards sub-optimal solutions, maintaining the population diversity high is the key strategy [73]. Several studies have concentrated on preserving this population diversity throughout the evolutionary process by modifying the mutation step size in order to transverse from a local optimal solution to the global optimum [75].

The basic definition of ES does not include an intensification step. Nevertheless, it has been proved that applying a step of intensification (e.g. local search – ESs which implement local search are also called Memetic Algorithms) is quite beneficial in many applications [73]. Exploring the search space helps to quickly identify promising areas.

The mathematical models that may take into account several perspectives simultaneously are quite complicated; it is often difficult or even impossible to build linear models equivalent to the non-linear ones. In such scenario, MILP methods are not always applicable, without introducing simplifications or assumptions. On the other hand, evolutionary strategies can deal with non-linear problems, and it is possible to apply them to multi-objective problems. Yeh et al. [19] developed a multi-objective evolutionary strategy to solve their problem of green supply chain optimization. The authors showed how a large number of Pareto-optimal (see Figure 2.2 for an example and definition of Pareto-optimal solution) solutions have been generated in a reasonable amount of time.

Ding et al. [8] addressed the design of production-distribution networks including both supply chain configuration and related operational decisions such as order splitting, transportation allocation and inventory control. The goal is to find the best compromise

between cost and customer service level, and, for that, the authors developed a simulation framework based on multi-objective evolutionary strategy (MOES).

Bevilacqua et al. [29] developed a multi-objective evolutionary strategy based on a real-valued chromosome that may be applied to the case of constrained nonlinear function, a case where it is usually not easy to apply exact methods. The authors analyzed the results of their algorithm with regard to the results produced by an exact ILP method applied on a linear approximation of the problem. They found that the ILP method produce a downwards rounded estimate of the optimal performance, while the MOES approach found an upwards rounded estimate.

Evolutionary algorithms have been shown to be also flexible. Chang et al. [47] proposed a combination of co-evolutionary mode with constraint-satisfaction mode to narrow down the possible solutions, hence reduce the space to explore. The co-evolutionary mode can adjust evaluation constraints dynamically to match a complex reality. Another example is in Che et al. [49], where the addressed problem is an optimization of build-to-order supply chain plans. The chosen evaluation criteria are namely costs, delivery time, and quality. The defined models are non-linear and define complex spaces. In [49], the authors were able to increase the quality of the Pareto-optimal solutions, by revision of crossover and mutation operators.

E. Ant Colony Optimization

The Ant Colony Optimization (ACO) algorithm is a meta-heuristic inspired by the social behavior of ants finding the shortest paths from their nest (colony) to a food source [30]. Proposed by Dorigo et al. [76] in 1996, it is an approach to stochastic combinatorial optimization [76]. Ants have the ability to find food sources using the shortest paths emanating from their nest. Whilst making their way to a food source ants deposit a chemical marker (pheromone), which affects the probability that other ants searching for food will follow the same path. The more ants that pass down a particular path, greater the quantity of pheromone deposited. The strength of this marker increases the probability that other ants will travel along the same path [77]. The deposited pheromone also evaporates as a function of time; the degradation rate is faster on longer paths than on shorter ones. Consequently, after a number of tours to and from the nest, there is a higher concentration of pheromone on shorter paths than on the longer ones.

Figure 2.5 depicts an example of ants finding the shortest path around an obstacle when foraging for food.

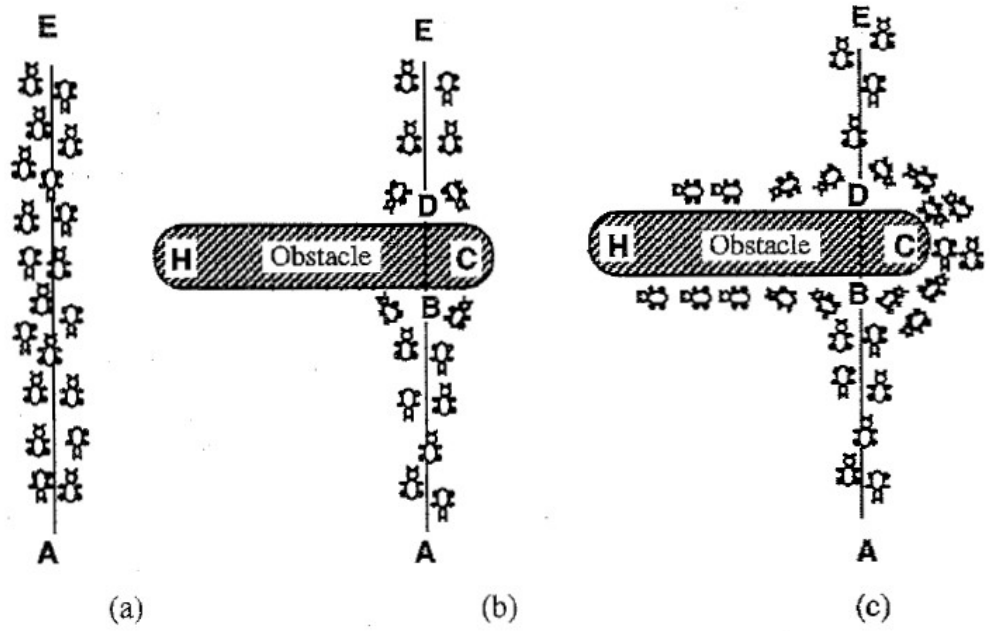

Figure 2.5 - Example of ants finding the shortest path around an obstacle when foraging for food. The source of the picture is [76].

In ACO, artificial ants work as simple computer agents within a network to probabilistically build solutions that exploit two types of information to guide their movement [76]:

1. Heuristic information on the problem being solved, which provides a greedy constructive heuristic.
2. Artificial pheromone trail information, which mimics real pheromones deposited by ants. This changes dynamically over time and is based on the number of tours undertaken by the ants.

The Ant System was the first ant algorithm proposed by Dorigo et al. 1996 [76]. Algorithmic variants have emerged since the early days of the initial research into ACO, which have resulted in performance improvements in comparison with the original algorithm. These variants include the Ant Colony System [78], the Max-Min Ant System [79], the Rank-based Ant System [80] and the Elitist Ant System [76].

The Ant System (AS) was initially applied to the Travelling Salesman Problem (TSP) [78], but has been used to solve other combinatorial problems such as the Quadratic Assignment Problem [81] and the Job-Shop Scheduling problem [76]. For a connected and undirected graph with *n* vertices (nodes) and *e* edges, ants generate a complete tour by choosing nodes using a probabilistic state transition rule. Once every ant has created their tour a global pheromone update is applied. A fraction of the pheromone evaporates on all edges, and each ant deposits an amount of pheromone on the edges as part of its tour in proportion to the quality of the tour. Edges which belong to better quality tours will receive greater pheromone.

There are three ways in which a pheromone trail can be updated in the AS algorithm. Trails can be categorized as belonging to one of three types: *ant-density*, *ant-quantity*, and *ant-cycle* [76] [82]. In ant-density, the pheromone quantity Q is deposited on the edge (r, s) whenever an ant moves from node *r* to node *s*. In ant-quantity, ants deposit a pheromone quantity Q over the distance (r, s) while building a solution. However, in ant-cycle ants deposit pheromone only after they have constructed a complete tour. Experiments have shown that the performance of ant-cycle is superior to that of the other two pheromone depositing methods based on the quality of their tour.

The Ant System achieved good results for small TSP problems, finding optimal or near-optimal solutions for up to 30 cities [78]. However, for larger problems the AS algorithm tends to achieve sub-optimal solutions, and in terms of computation time it is inefficient when solving such problems. The Elitist Ant System was the first improvement of the Ant System; the solution of the global best ant has more "weight" in contributing to the pheromone trails than any other ant. Over the years several extensions and improvements over AS have emerged, such as the Ant Colony System [83].

The Ant Colony System (ACS) was introduced by Dorigo et al., 1997 [78] and differs from the Ant System in 3 ways: (i) the state transition rule provides a direct way to achieve a balance between exploration of new edges and exploitation of *a priori* and accumulated knowledge about the problem, (ii) the *global pheromone updating rule* is only applied to edges that belong to the best ant tour, and (iii) while ants are constructing a solution a *local pheromone updating rule* (a.k.a. the local updating rule)

is applied. Informally, in the ACS m ants are initially positioned on n cities chosen according to some initialization rule (e.g. randomly). Each ant builds a tour (i.e. a feasible solution to the TSP) by repeatedly applying a stochastic greedy rule (in this case, the state transition rule). While constructing its tour an ant also modifies the amount of pheromone on the visited edges by applying the local updating rule. Once all ants have completed their tour, the amount of pheromone on the edges is modified again by applying the global updating rule. As was the case in the Ant System, ants are guided in building their tours by both heuristic information (they prefer to choose short edges) and by pheromone information. An edge with a high amount of pheromone is taken as a very desirable choice. The pheromone updating rules are designed with a tendency to deposit more pheromone on edges that should be visited by future ants [78].

Similarly to the evolutionary approach, ant colony optimization strategies are able to generate high quality solutions with a quite reduced computational effort, when compared to traditional ILP methods. Anghinolfi et al. [16] solved randomly-generated single-objective optimization problems with both ILP- and ACO-based approaches, and produced a statistically valid test set to prove the above idea. The solutions found by ACO are on average worse only by a factor of 1% with a computational time that is more than 50% faster. These results suggest the use of the ACO algorithm for large instances when acceptable sub-optimal results are needed in a short time, otherwise the proposed IP model can be exploited to generate higher quality solutions [16]. Similar results can be found for multi-objectives problems, as shown in Ghoseiri et al. [48].

In the original form, the ant colony optimization finds only one path (from the nest to the food). To successfully apply it to the transportation network problem, it is required to improve for multi-source and multi-destination cases. Musa et al. [30] proposed such an improvement to solve a problem of cross-docking network optimization. In the transportation problem of cross-docking network, loads are transferred from origins (suppliers) to destinations (retailers) through cross-docking facilities, without storing them in a distribution center (DC). The purpose is to reduce warehouses to purely trans-shipment centers where receiving and shipping are its only functions, and therefore, reduce the transportation and inventory costs. The network in [30] is comprised of three layers: suppliers, DC, and retailers, and a complete ant tour is represented by a bi-

dimensional matrix from each supplier to each retailer. The value in a cell indicates whether to sending the load directly or transit it through a DC. At each iteration, the ant finds the least costly shipment routes from all suppliers to all retailers, and it updates the matrix accordingly. For large problem instances (where the goal was switched from finding the optimal solution to finding a feasible lower bound and a time limit was implemented), the algorithm proposed in [30] often outperformed Branch-and-Bound of the LINGO solver.

Modifications on the ant colony algorithm allows it to be applied to multi-objective optimization problems as well. Utama et al. [42] proposed an implementation of the ACS for optimizing a palm oil based bioenergy supply chain. In such application, product quality, environmental factors, and service level are key performance measurements of the supply chain and they have been considered as single objectives of the problem. In addition, the authors have also considered minimization of distance and costs. The authors in [42] designed a set of fuzzy rules to assign a weight to the single objectives.

When dealing with dangerous goods, only considering one traditional objective in routing planning, such as the shortest route or the lowest cost, is no longer sufficient [38]. Xiang et al. [38] developed a multi-objective model for the route planning of dangerous goods. The performance measurements considered as single goal are total travelling time, accident probability and population exposure risk. The most commonly used method is to present relative importance as weighting factors for these objectives, and to produce a single value to evaluate solutions [38]. However, although this method is easy to use, one of the biggest drawbacks is that numerically quantifying the weights is a difficult task [38]. The authors in [38] implemented a variant of the ACS to search for Pareto solutions. The multi-goal objective function is based on the MINMAX method presented in Balling et al. [84]. At each optimization iteration, each single objective value is compared with the one of all visited solutions. The goal becomes maximizing the improvement of all single objective values. If all objectives of one solution are worse than those of another one, the value of the new objective function is always less than or equal to 1. On the contrary, if the value is always more than 1, this solution is a non-dominated one. A non-dominated solution belongs to the Pareto frontier or set of Pareto solutions.

F. Particle Swarm Optimization

Particle Swarm Optimization (PSO) was first introduced as a method for optimization of continuous nonlinear functions by Kennedy et al. [85]. The method was discovered through simulation of a simplified social model used to interpret and understand the movement of organisms in bird flocks or fish schools. The particle swarm optimizer began as a simulation of a simplified social milieu where agents were thought of as collision-proof birds. At each iteration a loop in the algorithm determines for each agent which other agent was its nearest neighbour, then assigned that agent's velocity to the agent in focus. Essentially this simple rule created a synchrony of movement. A random perturbation of the velocity of a given element was introduced in order to avoid the swarm settling down on an anonymous, unchanging direction. Moreover, the optimization function provides a force which guides the swarm to the best position found by one member. Velocity is dynamically adjusted based on the direction the element is moving relative to the position of the best known solution. Similarly to the Evolutionary Strategy or the Ant Colony Optimization, the algorithm focused the search in promising portions of the search space, and a stochastic component guarantees the avoidance of local minima.

As for the ES and ACO, the particle swarm optimization algorithm has been adapted and successfully applied to many supply chain optimization problems. Huang et al. [41] presented a variation of PSO to solve a resilience optimization problem of a four-party logistics (4PL) network. A 4PL network is a more robust design than the more common 3PL. The components of a 3PL supply chain are as usual suppliers, intermediate facilities or distribution centres, and demand zones. Let us assume that original facilities or 3PLs become unavailable due to natural disaster or human factors. In this case, new facilities or 3PLs must serve instead, resulting in an increment in costs for transportation and processing which can be regarded as the "loss". A 4PL design defines redundancy connections between elements in the network, and redundant resources. The potential for loss reduction caused by disruption is greater in a 4PL design. In the PSO implementation of [41], an element of the swarm is a list of utilized network links and represents a complete solution. The probability of disruptive events is estimated and the cost of the system when disruption occurs is compared with the cost

of the system under nominal conditions. The overall objective is to minimize the total cost both under nominal and under disruptions scenarios.

Che et al. [39] proposed a PSO variation to optimize the production and distribution planning of a multi-echelon supply chain based on more than one objective. The authors in [39] integrated cost and time criteria, and simultaneously considered multiple products, production loss, transportation loss, quantity discount, production capacity, and starting-operation quantity. The PSO proposed in [39] introduces a novel problem-specific heuristic to implement the disturbance mechanism.

Zhao et al. [44] defined a hybrid PSO approach for the optimization problem of an agri-food supply chain. In the context of agri-food production and distribution, reducing the production and transportation costs is critical, as limited shelf life, demand and price variability of agri-food products significantly increase the complexity of managing such a supply chain. Due to the NP-hardness of such type of problems and the large sized problems in the real world, meta-heuristics alone may require a significant computational effort to reach high quality solutions [44]. The authors in [44] extended the PSO algorithm with a local search mechanism to enhance the exploration strategy of the optimizer. As a result, the proposed algorithm finds high quality solutions much faster.

2.2.4 Hyper-Heuristics

A key drawback of current (meta-)heuristics is that state-of-the-art approaches for real-world problems tend to represent bespoke problem-specific methods which are expensive to develop and maintain [86]. Search techniques and meta-heuristic methods in particular have been proven successful when applied to real-world optimization problems, however, it is still difficult to apply them to new problems or instances of similar problems. Many state-of-the-art meta-heuristic developments are too *problem-specific* or too *knowledge-intensive* to be implemented in inexpensive, easy-to-use computational systems [87]. Difficulties related to meta-heuristics implementation arise mainly from the significant range of parameter or algorithm choices involved when using this type of approach and the lack of guidance on how to select them [86]. In addition, the scientific community's level of understanding of why different heuristics

work effectively (or not) in different situations does not facilitate simple choices of which approach to use in which situation [86].

Hyper-heuristic is an emerging methodology in search and optimization which has been developed with the goal of developing algorithms that are more general and easily applicable to a wide range of problem domains. Hyper-heuristics take on the challenge of automating the design and tuning of heuristics methods to solve hard computational search problems. The most recent definition is from Burke et al. [86]: *a search method or learning mechanism for selecting or generating heuristics to solve computational search problems*.

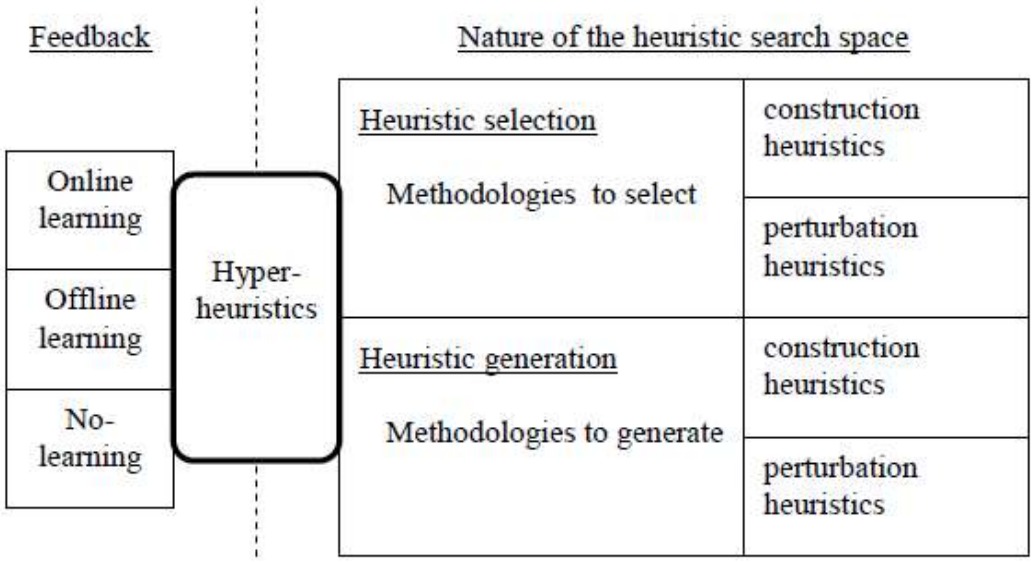

Figure 2.6 – Hyper-heuristics classification from E. Burke et al. [88].

Hyper-heuristics raise the level of generality at which optimization systems can operate and they are typically based on a set of low-level, easy-to-implement heuristics. Meta-heuristics act on the solution space, whereas hyper-heuristics act on the methods space.

Burke et al. [86] in 2013 surveyed the state of the art of hyper-heuristics and identified three high-level properties of such methods:

1. A hyper-heuristic is a higher level heuristic that manage a set of low-level heuristics.

2. It searches for a good method to solve the problem rather than for a good solution.
3. And it uses only limited problem-specific information.

The last property is considered to be the crucial one when designing a hyper-heuristic. These type of methods often use machine learning techniques to learn from and adapt to the currently addressed problem or instance. The typical classification is: heuristic *selection* and heuristic *generation*. A hyper-heuristic is classified as heuristic selection when the method mainly consists of choosing the most promising heuristic from a pre-determined set. Heuristic generation-based methods select components of existing heuristics to build a new one.

Zheng et al. [89] addressed the problem of emergency railway transportation planning. In disaster relief operations, huge amounts of relief supplies need to be transported from a variety of supply centers to the disaster-affected areas in a timely and effective manner, in order to alleviate the suffering and damage as much as possible [89]. Having advantages such as high capacity, less dependence on weather conditions, punctuality, and safety, railway transportation is particularly suitable for large-scale and long-distance freight transportation. The efficiency of emergency railway transportation has a great impact on the success of disaster relief operations. Zheng et al. [89] applied several state-of-the-art evolutionary algorithms to a variety of problem instances, but the solutions found were not satisfactory. The authors integrated a set of individual heuristics operators into a hyper-heuristic framework. The developed optimization process performs a stochastic search on the low-level heuristics by using feedback on their performance. The hyper-heuristic methods defined in [89] consists of the selection of evolutionary operators to be applied during the optimization process. At the beginning, the operators are selected randomly. The performance of the operators is recorded during the search and their probability of being applied is changed accordingly. After a few iterations, the evolutionary algorithm will utilize those operators which seems to be more suited to the given problem instance. This approach resulted in a higher overall performance on different instances.

2.2.5 Parameters Tuning

Most of the algorithms for tackling computationally hard optimization problems described in the sections above depend on a number of parameters which influence their behaviour [90]. The performance of such parametrized algorithms depends strongly on the particular values of the numerical and categorical parameters, and the appropriate setting of these parameters is itself a difficult optimization problem [90]. Meta-heuristics specifically are characterized by a large number of parameters. According to Birattari [91], a meta-heuristic is not properly an algorithm but rather a set of concepts that serve as guidelines for tackling an optimization problem. It is convenient to look at a meta-heuristic as at an algorithmic template that needs to be instantiated to yield a fully functioning algorithm [91]. The importance of tuning is generally recognized by the research community. However, in the vast majority of the cases, meta-heuristics are tuned by hand in a trial-and-error procedure guided by some rules of thumb [91].

According to Birattari [91], the trial-and-error approach presents many drawbacks, two of the most noticeable being:

1. In the context of large-scale industrial applications, a trial-and-error approach is extremely time consuming, labour-intensive, and requires the attention of a skilled practitioner, somebody well acquainted with the optimization algorithm.
2. In the context of research applications, such approach may very well invalidate any conclusions drawn from the experimental comparison of different algorithms.

Birattari [91] presented a detailed analysis of the research work which specifically address the problem of tuning meta-heuristics. The authors in [91] highlighted three major limitations that most of the work referred share, these are as follows:

1. Most of the studies about meta-heuristics tuning lacked a clear definition of the tuning problem itself.
2. Most researches fail to notice that tuning has always to be conceived with respect to a specific class of instances.
3. Finally, the lack of a precise statement on which specific figure of merit the tuning process is expected to optimize.

Birattari [91] designed an algorithm for the automated configuration of parameters which overcome these limitations, called F-Race. The defined algorithm belongs to the class of racing approaches from the area of machine learning. The basic idea is to streamline the evaluation of the candidate parameter configurations and to drop during the evaluation process those that appear less promising. Intuitively, less configurations are tested than for a brute force approach, and, therefore, the computational complexity of the algorithm is lower. The algorithm relies on the definition of a process to generate a stream of configurations and a metric to evaluate the expected performance of a candidate configuration using a finite number of experiments.

Birattari et al. [90] presented an extension of F-Race, capable of handling continuous parameters, named I/F-Race. Both of F-Race and I/F-Race are offline-tuning algorithms, as they produce one parameter configuration per problem set and apply it to each problem instance. The problem of online tuning is quite different from the offline variant, and it is typically based on some machine learning technique, often belonging to the reinforcement learning literature [91]. The key idea is to dynamically adapt a subset of the parameters of the optimization algorithm while performing the optimization itself. Online tuning allows the optimization process to be more flexible and better handle problem instances which differs significantly from each other. Such an approach is particularly suited to solving large and complex instances. Despite online tuning not being the main topic of the work in [91], Birattari [91] presented an analysis of the most relevant work for the topic of online tuning.

One of the key parameter of meta-heuristics is the termination condition. Depending on the definition, the termination condition may be represented as a discreet, continuous, or categorical parameter. While analysing the literature for the problems of automatic parameter tuning and optimal termination condition configuration, a lack of consideration for the latter problem emerged. According to related research, even work focused on parameter tuning rarely attempts to address or include the termination condition in their list of parameters. As a matter of fact, Dorigo et al. [92] reported that, for all meta-heuristics, there is no general termination criterion, and, in practice, a number of rules of thumb are employed.

The work presented in chapter 5 introduces a simple method to automatically set the termination condition of the ant colony optimization algorithm. As for F-Race [91] and I/F-Race [90], the algorithm is based on a supervised learning approach and requires a training set. However, the method differs from the approach presented in [90] and in [91] as a configuration is assigned to each problem instance independently. The termination condition considered is the optimal amount of iteration of the search process. A significant advantage of the algorithm presented in chapter 5 is the low overhead required in the training step. Each run of the optimization during the training step takes a nominal amount of iteration (e.g. one thousand iterations). Knowledge and characteristics of the problem instance are gathered during the search. If a similar instance occurs and strong evidence exists that the number of iterations can be reduced is present, then the termination condition is set accordingly for the new instance. During the training step, there is no need to run the search multiple times on the same instance to test different configurations. As a result, the training step can be performed as fast as a nominal optimization, and experimental results show that the optimization process is significantly faster when solving new unseen instances.

Of course, the algorithm presented in chapter 5 relies on the definition of the termination condition. As future work, the same ideas and principles could be applied to other parameters.

2.3 Summary

This chapter presented a general description of the problem of supply chain optimization and an overview of the optimization techniques most commonly employed. Throughout the chapter, references to research works and surveys that offer more in-depth analysis to specific aspects of the problem have been provided. Moreover, the limitations and research opportunities that emerge from such works have been highlighted and how they have been addressed in the present work has been described.

Regarding the design of supply chain models for optimization, a noticeable trend consists of selecting cost minimization as a performance measure of the supply chain over profit maximization. This limits the effectiveness of the optimization process as the

perspective of cost minimization is often too narrow and many key aspects of the supply chain are not considered. Similarly, in-transit inventory and transportation mode characteristics are often neglected when modelling supply chains.

When designing the model of Caterpillar's supply chain, these undesirable trends and limitations have been taken into consideration. The following chapter, chapter 3, depicts the model for Caterpillar's supply chain and works as an example on how to address and overcome such shortcomings.

The current chapter also began with a discussion about multi-objective optimization. A more comprehensive analysis is presented in chapter 4. During the literature review, it appeared that most of the methods to solve the problem of multi-objective supply chain optimization are very specific to either the problem or to the optimization algorithm. In order to apply a multi-goal analysis to Caterpillar's supply chain, it is needed an understand of what the general-purpose options were and how they would perform on Caterpillar's specific instance. Chapter 4 is the result of this investigation.

Despite the problem of supply chain optimization being well established, the review highlighted how many different aspects and variations characterized practical applications. Defining the model of a real-world supply chain is still a non-trivial task that the application of the general theory does not completely solve. Practical knowledge and experience of the specific business is still required to identify its key characteristics, and an understanding of the mathematical formalism is necessary to model them. Similarly, the state-of-the-art algorithms for real-world supply chain optimization tend to be expensive to develop and maintain, as they tend to be problem specific. Chapters 5 and 6 discuss the work done with the high-level goal of increasing the automatization of the supply chain optimization process. Specifically, chapter 5 defines a method to automatically set and optimize the termination condition of the optimization algorithm. The method not only improves the performance of the algorithm, but also reduces the effort necessary to apply the algorithm to the specific problem. Chapter 6 introduces a system to assist in designing the optimization model via the application of a data mining approach.

3 CATERPILLAR'S SUPPLY CHAIN

Caterpillar Inc. is the world leader in manufacture of construction and mining equipment, diesel and natural gas engines, industrial gas turbines and diesel-electric locomotives. The current revenue of Caterpillar is of the order of tens of billions and they sell products and parts via a worldwide dealer network. Caterpillar sells more than 3 million products and 700,000 parts in more than 20 countries around the world every year. They operate with more than 3,000 suppliers and 3,000 dealerships and their logistics operations alone are worth more than 60 million dollars per year.

The supply chain of Caterpillar is made of two key echelons: (i) the manufacturing and distribution to assembly points of components or parts, and (ii) the shipment of final products or machines to the dealerships. Both echelons are a four-layer network as shown in Figure 2.1. The first tier in the supply network for parts consists of the set of component factories. The second and third tiers are respectively outbound and inbound shipping ports. The last tier represents the product manufacturers and assembly points. The second key echelon of Caterpillar's supply chain starts from the tier of machine assembly points. The following two tiers are again outbound and inbound shipping ports

and the nodes in the last tier represents the dealerships and customers. Figure 3.1 shows a graphical representation of Caterpillar's supply chain based on the two echelons.

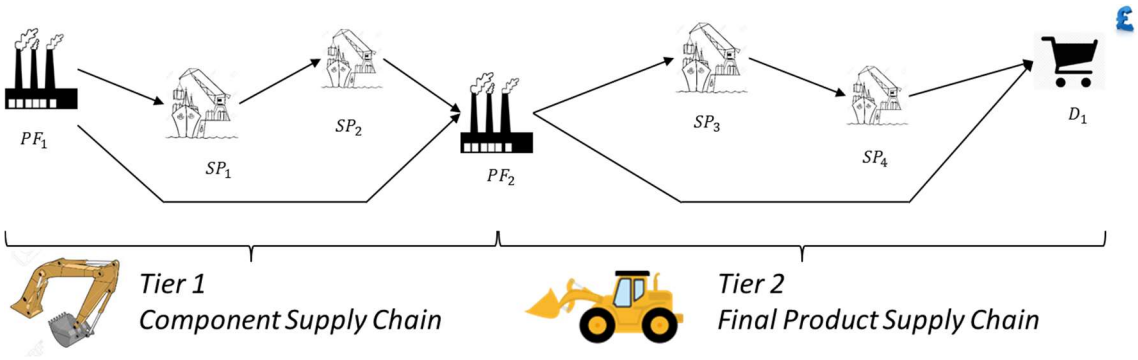

Figure 3.1 - Graphical representation of Caterpillar's supply chain.

The preferred transportation modes at Caterpillar are trucks and freighter ships. Occasionally, some products or parts are shipped on planes and trains. Machines are moved on flatbeds when on land and shipped on what is called "roll-on-roll-off" (RoRo) freighter ships when shipping across the ocean. If the size and weight allow it, components are grouped and shipped together in vans on land and containers on freights. Caterpillar is one of the largest freight movers in the world. They do not own the fleet of freights; however, they retain a significant leverage on the fleet owners and usually negotiate transportation pricing based on current demand and future commitment. Containerized freights are typically faster and less costly than roll-on-roll-off ships. Experiments are in progress to model the possibility of disassembling a machine at an outbound port and then ship it inside a container. Based on preliminary analysis, it is likely that the reduced transportation time and cost of such mode of transport may make up for the additional cost and time of disassembling and reassembling the machine at the ports.

The links in the supply network which connect component manufacturers and shipping ports are connections on land where components are moved via trucks. Similarly, the links between receiving ports and assembly points, between assembly points and ports, and ports and dealers are on land. Moreover, if component factories, assembly points, and dealerships are located on the same continent or region a direct land link is present, which allows products to avoid going through the shipping ports.

The nodes and connections of Caterpillar's network originated from a division of the regions Caterpillar operates in as shown in Figure 3.2. The Earth surface has been divided into hexagons of the same size. Hexagons containing locations involved in the supply process have been marked with key identifiers. For instance, in Figure 3.2 hexagons with a yellow identifier represents factory locations, light blue identifiers code the location of shipping ports. The purpose of such division was to reduce the level of details when building Caterpillar's model and abstract from scenarios which are not particularly meaningful when taking a global perspective. They provided a standard size for the hexagons, facilities which fall into the same hexagon are modelled as one node in the network representation and transportation aspects between such facilities are not considered in the optimization process.

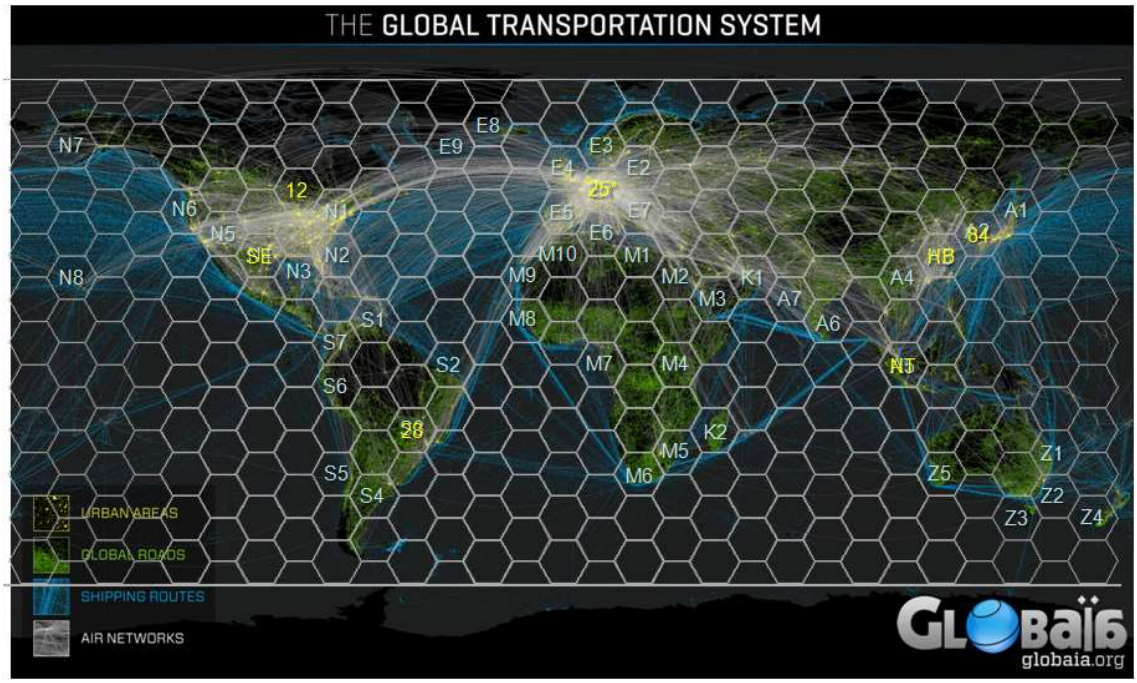

Figure 3.2 - Representation of Caterpillar's Distribution Network.

The production facilities for both components and complete products have an associated maximum production capacity and a unit-based production cost. Factories where machines are assembled from a set of components also have an estimated assembly time for each type of machine and the variability expected on it. In the business of manufacturing large and complex objects, factories are typically expensive to build and operate. Retooling a factory to produce a different product is a significant investment

and even shutting down factories may impact the viability of the company. All factories in the supply chain have an operating cost also referred to as period cost, which is independent from the quantity of product produced as long as the factory remains the property of the company.

Links between the nodes have an associated transportation cost which is at least a function of the distance and time required to travel across the link. Many more factors affect the transportation costs, e.g. energy price, inventory costs and international trade factors. Sections 3.2.1, 3.2.2, 3.2.3, and 3.2.4 present a detail model for these additional factors.

Caterpillar's preferred sales model is through dealerships. Each product has a price tag which may change based on the region in which the dealership is located (e.g. the same products sold in North America have a different price to Asia).

The performance measures of greatest interest to Caterpillar's Logistics division are profit, transportation/inventory time and resilience. Contrary to the trends highlighted in section 2.2, maximization of profit is preferred over cost minimization. Profit maximization is the most comprehensive measure as it not only accounts for transportation cost, but also considers sale prices, inventory costs, international trade factors and more. Of even higher significance is the goal of resilience maximization. A resilient supply network is a network that is not susceptible to possible disruptive events. Disruptive events for a supply chain are for instance the loss of a factory or a shipping port due to some natural disaster which struck an area or the loss of connection links between nodes due to a fault in service from a transportation company (e.g. the ships or the trucks on one route brake down, or employees go on strike). If the resilience level of the supply chain is high, it is likely that its performance would not be significantly affected in the case of a disruptive event (e.g. the profit loss would be low). Sections 3.1, 3.2, and 3.3 define the mathematical models used to optimize Caterpillar's Supply Chain according to the performance measures of travel time, profit, and resilience respectively.

Section 3.4 introduces the simple simulation system that has been implemented to model the uncertainty of the transportation costs. Section 3.5 describes the optimization algorithm employed, the Ant Colony Optimization, and discusses implementation

details. Finally, section 3.6 depicts the results of the experiments performed to test the optimization process.

3.1 Transportation Time Minimization

Transportation time is quite significant when optimizing a supply chain as it is typically proportional to costs, inventory requirements and service level. Links that take longer to cross have higher transportation costs and make more challenging to retain a high service level. In order to meet the customer demand at the requested time, the schedule of production must account for the time required to travel across the supply chain. Longer times means transportation constraints have higher impact on the production schedule. Inventory must be placed appropriately on the network such that customers demand can be met in time.

Modelling transportation time minimization is quite straight forward as it does not require significant knowledge of the mechanics of the supply chain. Such a model was the first one implemented for Caterpillar's supply chain. Starting from the model described by equations from (2.1) to (2.4), the information required are the locations of production sources, shipping ports and dealerships to compute the transportation times for each possible route in the network. The result is a set $d \in \mathbb{R}^{n \times m}$ of times which define the cost coefficients of the objective function. Equations (3.1), (3.2), (3.3), and (3.4) define the model for the minimization of transportation time.

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} x_{ij} \tag{3.1}$$

$$s.t.: \quad \sum_{j=1}^{m} x_{ij} \leq S_i \qquad \forall i \in [0, n]. \text{ Capacity constraints} \tag{3.2}$$

$$\sum_{i=1}^{n} x_{ij} = D_j \qquad \forall j \in [0, m]. \text{ Demand constraints} \tag{3.3}$$

$$x \in \mathbb{R}_{+}^{n \times m},\ c \in \mathbb{R}_{+}^{n \times m},\ S \in \mathbb{R}_{+}^{n},\ D \in \mathbb{R}_{+}^{m} \qquad \text{Domains} \tag{3.4}$$

3.2 Profit Maximization

Profit is the most comprehensive measure for understanding the performance of a supply chain from a business stand point. Improving the efficiency of the distribution plan or reducing the costs of the production process will result in an increased profit. Considering more dimensions of the supply chain, the optimization of a profit maximization model will result in a more realistic solution of greater significance to the managers. For instance, it is reasonable to depict a situation where the closest production facility to a dealership is significantly more expensive than one located further away. Lower production costs might compensate for the higher transportation costs. Such a scenario is even more likely if production and transportation commitment discounts are in place. Moreover, on the international marketplace, difference in government regulations and taxations could make it more compelling to manufacture and ship the product outside the country location of the dealership. Typically, such regulations affect the sale price and production cost. A model considering only transportation costs would be blind to this scenario.

Caterpillar's model for profit maximization includes the following factors:

- Sale prices at the dealership.
- Production costs and assembly costs.
- Production facilities operational costs (period cost).
- Inventory costs per unit of product and time to be held.
- Energy costs based on the mode of transports.
- International trade factors (tariff costs).

All above mentioned costs are per unit of product, with the exception of period costs which are per operational factory. Equations (3.5), (3.6), (3.7), and (3.8) define the model for profit maximization. $sp \in \mathbb{R}^m$ is the set of sale prices for each dealership j, $pc \in \mathbb{R}^n$ is the set of costs associated to each production facility i, and $tc \in \mathbb{R}^{n \times m}$ is the set of transportation costs for each route in the supply chain from manufacturer i to dealership j. Production costs, assembly costs, and period costs are added into pc. Sale prices are weighted on tariff costs in sp. And, finally, inventory, energy costs, and discount for commitment are included in the transportation cost calculation tc.

$$max \sum_{i=1}^{n} \sum_{j=1}^{m} (sp_j - (pc_i + tc_{ij})) \cdot x_{ij} \quad (3.5)$$

$$s.t.: \ \sum_{j=1}^{m} x_{ij} \leq S_i \qquad \forall i \in [0, n]. \text{ Capacity constraints} \quad (3.6)$$

$$\sum_{i=1}^{n} x_{ij} = D_j \qquad \forall j \in [0, m]. \text{ Demand constraints} \quad (3.7)$$

$x \in \mathbb{R}_+^{n \times m}, sp \in \mathbb{R}_+^m, pc \in \mathbb{R}_+^n, tc \in \mathbb{R}_+^{n \times m}, S \in \mathbb{R}_+^n, D \in \mathbb{R}_+^m$ Domains (3.8)

3.2.1 Energy Costs

The typical distribution plan for Caterpillar would span one year of demand. Usually, distribution plans are outlined for each product separately. The energy price changes quite significantly over the course of one year and it is extremely difficult to predict. Therefore, the optimization process must be flexible to accommodate such high level of variability.

The nominal energy cost when the distribution plan is delineated is from the grade of crude oil pricing given by the West Texas Intermediate benchmark. For the next 12 months, a prediction system is used to adjust and compensate possible energy price variations. The result of a Poisson distribution is taken over the expected value of energy cost for each of the following months. Figure 3.3 shows the result of the projection for the years 2013 and 2014. Equation (3.9) define the energy price E for a given month t. $E(0)$ is the initial month, and z is the percentile value for the Poisson distribution. σ_{E_1} is the standard deviation defining the negative confidence interval of the prediction, and s is the "exchange" parameter to obtain the standard deviation for the positive confidence interval $\sigma_{E_2} = s * \sigma_{E_1}$. Transportation costs in the objective function tc_{ij} are multiplied by the energy price $E(t)$ in any given month.

$$E(t) = E(0) * e^{z * s \sigma_{E_1} \sqrt{\frac{t}{12}}} \quad (3.9)$$

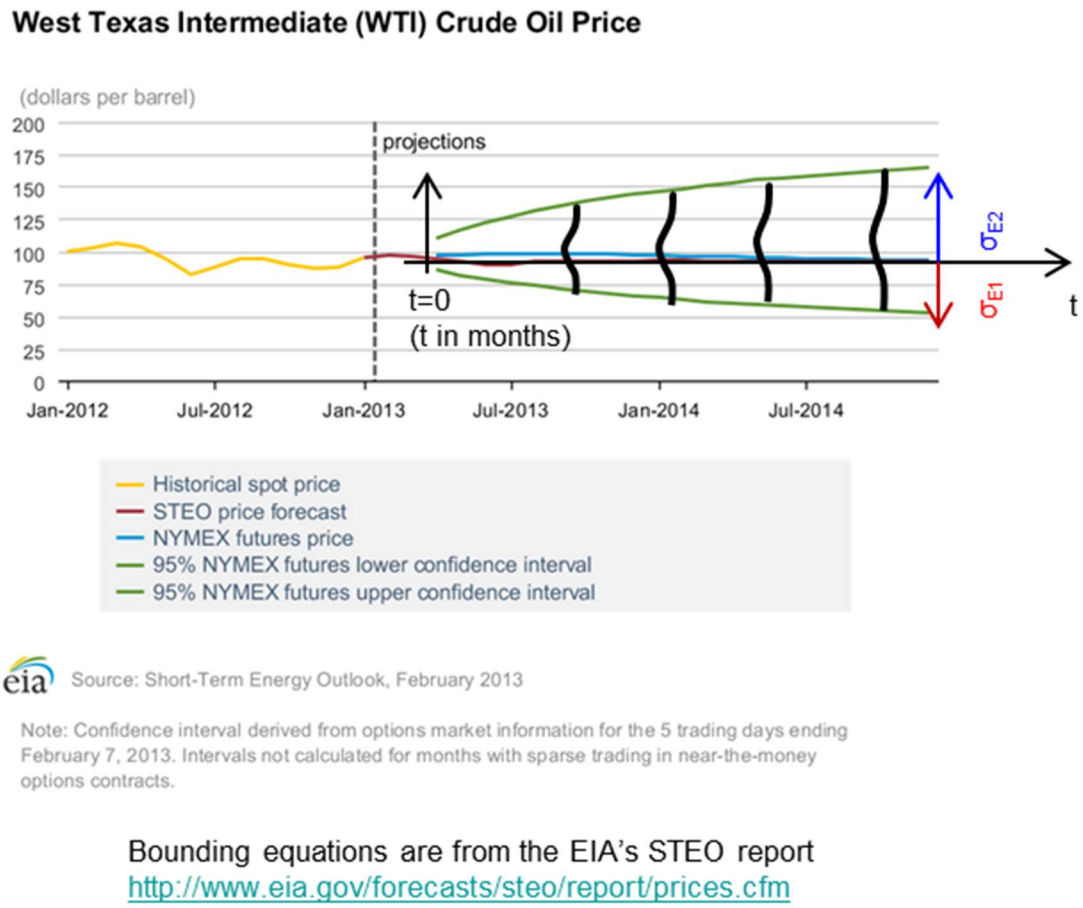

Figure 3.3 - Energy cost projection from West Texas Intermediate (WTI) Crude Oil Price

3.2.2 Lane Commitment Discount

As one of the largest ocean freight movers in the world, Caterpillar is able to negotiate transportation prices with the fleet owners based on current demand and future commitments. Most of the fleet owners grant Caterpillar discounts on transportation costs if they commit to a good portion of the upcoming year to purchase their service and ship product on the same ocean lane. In order for a distribution plan to be meaningful, such discounts must be considered in the profit calculation. Discount rates may vary based on the service provider; however, they are typically a function of the time of commitment and unit of product. Equation (3.10) outlines the model to approximate the discount mechanism, given the link between two ports SP_k and SP_l, the month t for which the discount is calculated, and the year to which the calculated

distribution plan belongs. T is the duration of the commitment in months, m physical characteristics of the product shipped (e.g. cubic volume of a machine), k the minimum lane usage price (in US dollars), and v is the amount of product committed. $E(t)$ is the energy price for the month t as defined by eq. (3.9). The parameters α, β, and γ tune the weight of commitment on the cost calculation and describes the sensitivity to different aspects of the model: specifically, α describes the sensitivity to commitment, γ to the volume, and β to something in between consistency and volume.

v is the value of the decision variable x in the optimization problem (3.5)-(3.8). The resulting objective function becomes cubic in x and the optimization process not only needs to search in the space for possible ways to ship products from factories to customers, but also for how long to commit to a set of port-to-port links. A new dimension is added to the problem and the optimizer must keep track of the lane usage across twelve distribution plans.

$$OC_{SP_k SP_l}(t) = \left(\alpha \left(\frac{12 - T}{12} \right)^2 + \beta \left(\frac{12 - T}{12} \right) m \cdot v + \gamma (m \cdot v)^2 + k + E(t) \right) \cdot m \cdot v \quad (3.10)$$

Table 3.1 summarizes the parameters used for the calculation of the lane commitment discount.

Table 3.1 – Parameters for the lane commitment discount calculation.

Parameter Symbols	Parameter Meanings	Values
α	Sensitivity to commitment	14.33457
β	Cross sensitivity between consistency and volume	0
γ	Sensitivity to volume	-2.39E-04
k	Minimum lane usage	9.35
m	Physical characteristics of the product (cubic volume of a machine)	90

3.2.3 Inventory Calculation

Correct placement of inventory is fundamental to retaining a high service level and mitigate the risk of shortfalls. Inventory used to meet the customer demand within a time window is called Cycle Stock; the inventory to overcome shortfalls and operate the business according to the plan is called Safety Stock. Keeping inventory however entails an additional cost to the distribution plan. The current optimization process at

Caterpillar models only cycle stock. The time interval for which inventory must be held at any given point in the network is determined based on the service level set for the distribution plan and the time required for products to be shipped from production facilities to dealerships. Service level is a function of promised time, which defines the time window when the customer is expecting to receive the order. A high service level means a good percentage of orders have been delivered on time.

Given a promised time $pt \in \mathbb{N}^+$, which is usually measured in days, inventory is needed if the time required to travel from the production facility to the dealership is greater than the promised time. Inventory may be kept at the production facilities, at the shipping ports and at the dealerships. In order to minimize the costs, the inventory should be stored as close to the manufacturers as possible. The costs of inventory are from the cycle stock (the inventory kept at the nodes in the supply chain) and from the product "trapped" in transit on the network.

Given the path defined by the nodes PF_i, PF_k, SP_l, SP_m, and D_j on the supply network as shown in Figure 3.1, let $tt(PF_i, PF_k)$, $tt(PF_k, SP_l)$, $tt(SP_l, SP_m)$, and $tt(SP_m, D_j)$ be the transportation times required to travel from one node to the other. The path has a direct link between component factory PF_i and assembly point PF_k. There are five possible locations where to hold inventory, which are the five nodes in the path. The inventory must be close enough to D_j in order to meet the promised time, and has to be as close to the production facility PF_i as possible in order to reduce the cost. The amount of inventory in days may be determined starting from the end of the path D_j and moving backwards towards PF_i. If the promise time is greater than $tt(SP_m, D_j)$ then no inventory is required on D_j as the demand can be met on time shipping from SP_m. On the contrary, if $pt < tt(SP_m, D_j)$ then the days of inventory required on D_j are $tt(SP_m, D_j) - pt$. The process can be repeated while moving towards the beginning of the path with the caveat that the inventory time possibly already allocated must be subtracted from the total transit time. For instance, moving on to SP_l, the total transit time now is $tt(SP_l, SP_m) + tt(SP_m, D_j)$. If $pt \geq tt(SP_l, SP_m) + tt(SP_m, D_j) - id(D_j)$ then no inventory is needed on SP_m, where $id(D_j)$ is the possible inventory days computed in the previous step. Otherwise, if $pt < tt(SP_l, SP_m) + tt(SP_m, D_j) - id(D_j)$

then the inventory days required on SP_m is $tt(SP_m, D_j) - id(D_j) - pt$. Algorithm 3.1 summarizes the steps to compute the required inventory days for a given path in Caterpillar's supply chain.

Figure 3.4 and Figure 3.5 summarize the algorithm to compute the required inventory days with two flowcharts. Figure 3.5 describes a subroutine utilized in the flowchart in Figure 3.4. The purpose of the flowcharts is to describe the idea of the algorithm. All the actual calculations and details are defined in Algorithm 3.1.

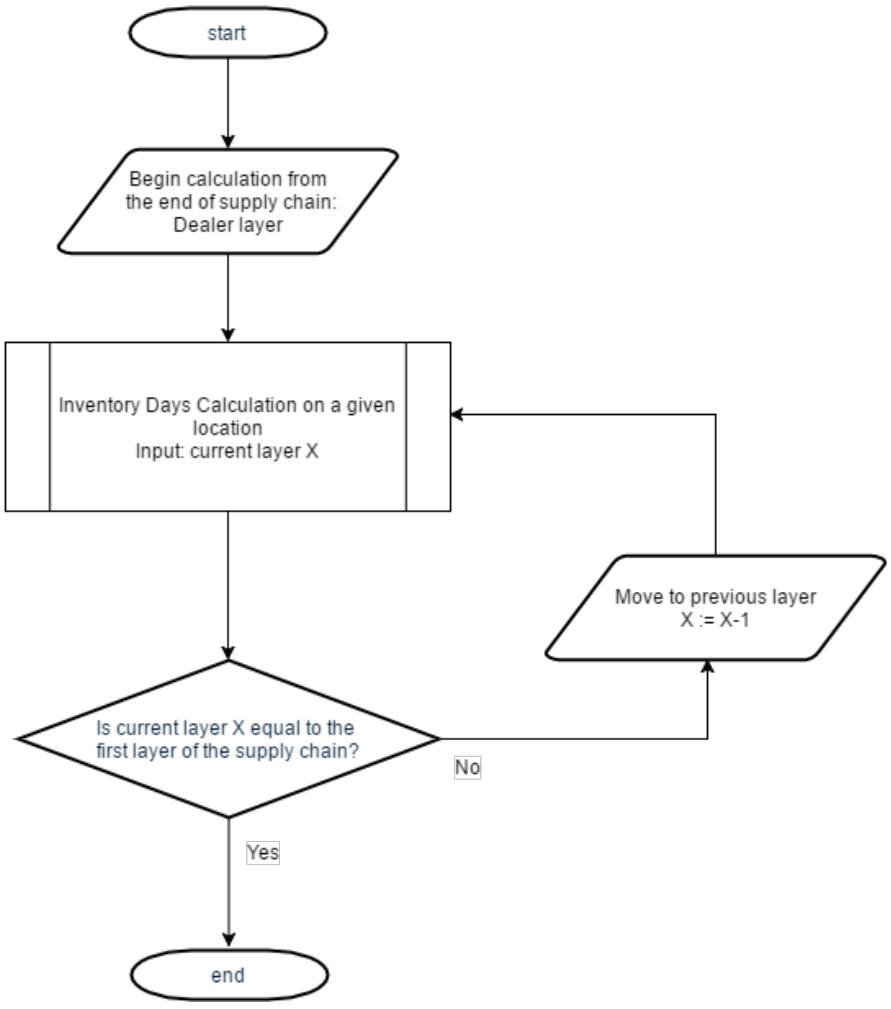

Figure 3.4 – Flowchart for the algorithm to compute the required inventory days. The subroutine for the inventory calculation on a given location is in Figure 3.5.

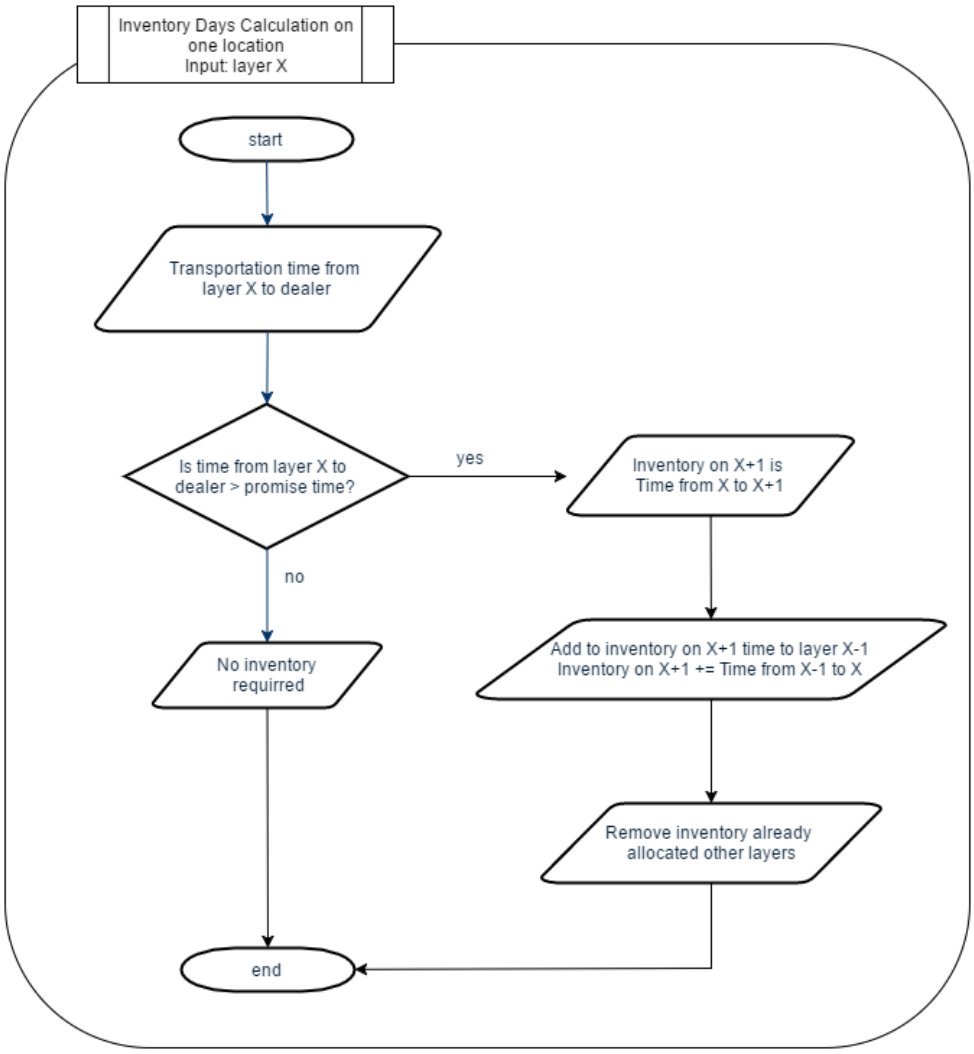

Figure 3.5 – Flowchart for the algorithm to compute the required days of inventory on a given location of the supply chain. The exact number of days for inventory is defined in Algorithm 3.1.

Algorithm for Inventory Days calculation

Require: Promise time $pt \in \mathbb{N}^+$. Path in the supply network defined by the nodes PF_i, PF_k, SP_l, SP_m, and D_j. Transportation times between consecutive nodes $tt(PF_i, PF_k)$, $tt(PF_k, SP_l)$, $tt(SP_l, SP_m)$, and $tt(SP_m, D_j)$.

1. **if** $pt \geq tt(SP_m, D_j)$ ▷ Inventory days at D_j
2. **then** $id(D_j) \leftarrow 0$
3. **else** $id(D_j) \leftarrow tt(SP_m, D_j) - pt$
4. **if** $pt \geq tt(SP_l, SP_m) + tt(SP_m, D_j) - id(D_j)$ ▷ Inventory days at SP_m

5. **then** $id(SP_m) \leftarrow 0$
6. **else** $id(SP_m) \leftarrow tt(SP_l, SP_m) + tt(SP_m, D_j) - id(D_j) - pt$
7. **if** $pt \geq tt(PF_k, SP_l) + tt(SP_l, SP_m) + tt(SP_m, D_j) - id(SP_m)$ ▷ Inventory days at SP_l
8. **then** $id(SP_l) \leftarrow 0$
9. **else** $id(SP_l) \leftarrow tt(PF_k, SP_l) + tt(SP_l, SP_m) + tt(SP_m, D_j) - id(SP_m) - pt$
10. **if** $pt \geq tt(PF_i, PF_k) + tt(PF_k, SP_l) + tt(SP_l, SP_m) + tt(SP_m, D_j) - id(SP_l)$ ▷ Inventory days at PF_k
11. **then** $id(PF_k) \leftarrow 0$
12. **else** $id(PF_k) \leftarrow tt(PF_i, PF_k) + tt(PF_k, SP_l) + tt(SP_l, SP_m) + tt(SP_m, D_j) - id(SP_l) - pt$
13. **if** $pt \geq tt(PF_i, PF_k) + tt(PF_k, SP_l) + tt(SP_l, SP_m) + tt(SP_m, D_j) - id(PF_k)$ ▷ Inventory days at PF_i
14. **then** $id(PF_i) \leftarrow 0$
15. **else** $id(PF_i) \leftarrow tt(PF_i, PF_k) + tt(PF_k, SP_l) + tt(SP_l, SP_m) + tt(SP_m, D_j) - id(PF_k) - pt$

return $id(PF_i), id(PF_k), id(SP_l), id(SP_m), id(D_j)$

Algorithm 3.1 – Pseudo code for calculation of inventory days required.

The days of inventory must then be converted into monetary value in order to be considered in the profit calculation. The cost of inventory is proportional to the transportation cost and the production cost per unit of product. The cost at each location is also weighted on the minimum acceptable rate of return, or hurdle rate. In this domain, hurdle rate is used as a synonym for cost of capital. Algorithm 3.2 shows the main components for computing the inventory carrying cost on one path. Let $tc(N_i, N_j) \in \mathbb{R}$ be the transportation cost for shipping one unit of product from node N_i to node N_j, and $pc_k \in \mathbb{R}$ be the production cost at manufacturer k. Let $ic(N_i) \in \mathbb{R}$ be the inventory holding cost at node N_i, and $ic(N_i, N_j) \in \mathbb{R}$ be the cost of inventory "trapped" between nodes N_i and N_j. Let $hr \in \mathbb{R}$ be the hurdle rate.

Algorithm for Inventory Costs calculation

Require: Path in the supply network defined by the nodes PF_i, PF_k, SP_l, SP_m, and D_j. Transportation costs between consecutive nodes $tc(PF_i, PF_k)$, $tc(PF_k, SP_l)$, $tc(SP_l, SP_m)$, and $tc(SP_m, D_j)$. Inventory days at the network nodes $id(PF_i)$, $id(PF_k)$, $id(SP_l)$, $id(SP_m)$, and $id(D_j)$. Production costs pc_i, and pc_k. The hurdle rate hr.

1. ▷ Inventory holding costs on nodes
2. $ic(PF_i) \leftarrow pc_i \cdot hr \cdot id(PF_i)$
3. $ic(PF_k) \leftarrow (pc_i + tc(PF_i, PF_k) + pc_k) \cdot hr \cdot id(PF_k)$
4. $ic(SP_l) \leftarrow \big(pc_i + tc(PF_i, PF_k) + pc_k + tc(PF_k, SP_l)\big) \cdot hr \cdot id(SP_l)$
5. $ic(SP_m) \leftarrow \big(pc_i + tc(PF_i, PF_k) + pc_k + tc(PF_k, SP_l) + tc(SP_l, SP_m)\big) \cdot hr \cdot id(SP_m)$
6. $ic(D_j) \leftarrow \Big(pc_i + tc(PF_i, PF_k) + pc_k + tc(PF_k, SP_l) + tc(SP_l, SP_m) + tc\big(SP_m, D_j\big)\Big) \cdot hr \cdot id(D_j)$
7. ▷ Inventory carrying costs on links
8. $ic(PF_i, PF_k) \leftarrow pc_i \cdot hr \cdot id(PF_i, PF_k)$
9. $ic(PF_k, SP_l) \leftarrow (pc_i + tc(PF_i, PF_k) + pc_k) \cdot hr \cdot id(PF_k, SP_l)$
10. $ic(SP_l, SP_m) \leftarrow \big(pc_i + tc(PF_i, PF_k) + pc_k + tc(PF_k, SP_l)\big) \cdot hr \cdot id(SP_l, SP_m)$
11. $ic(SP_m, D_j) \leftarrow \big(pc_i + tc(PF_i, PF_k) + pc_k + tc(PF_k, SP_l) + tc(SP_l, SP_m)\big) \cdot hr \cdot id(SP_m, D_j)$

return $ic(PF_i)$, $ic(PF_k)$, $ic(SP_l)$, $ic(SP_m)$, $ic(D_j)$, $ic(PF_i, PF_k)$, $ic(PF_k, SP_l)$, $ic(SP_l, SP_m)$, $ic(SP_m, D_j)$.

Algorithm 3.2 – Pseudo code for inventory costs calculation given the days of inventory.

3.2.4 International Trade Factors

Operating in an international market scenario both for selling and manufacturing, Caterpillar's logistics operations are affected significantly by international trade policies. International trade policies are instruments that national governments may use to regulate globalization effects. Typically, governments set tariffs and trade barriers on

import goods to protect the local marketplace, domestic employment standards, and customers. Since tariff is a tax, the local government will see increased revenue on import goods, and domestic industries will benefit from a reduction in competition. However, additional tax means an increase in sale price; therefore, customers and foreign companies are likely to be negatively affected. Foreign companies may occasionally avoid high import taxation by opening a production or assembly centre within that country.

Caterpillar's logistics division maintain a map of tariff levels between countries where their production facilities and dealerships are located. Each value between pairs of different countries determines the percentage of sale revenue to be paid as tax to the local government per unit of product sold. In the model for profit maximization, the total sale is therefore reduced by the amount of tariff applied.

3.3 Resilience Maximization

One interpretation for the problem of resilience maximization consists of measuring the potential loss occurring in case where a disruptive event incapacitates a node or a link in the supply chain network. The risk or the value of the loss may be mitigated by minimizing the volume going through each node and link. The ideal scenario consists of all products being shipped as evenly as possible on all nodes and links. The average quantity of product going through one element in the network should be low, and the standard deviation as close to zero as possible. No component of the network will handle most or all of the volume shipped, and, at any given point, if such element becomes non-operational, only the smallest amount of product as possible will be lost.

The optimization model becomes minimizing the percentage of total demand that passes through the "busiest" node or edge on the network, such that under disruption, only a small part of the monthly demand is affected. Given a complete distribution plan, the network resilience is the maximum of four terms: the percentage volume "trapped" at the dealerships, at the production facilities and assembly points, at the shipping ports, and at the transportation lane.

Let $v(N_i) \in \mathbb{N}^+$ and $v(N_i, N_j) \in \mathbb{N}^+$ be the volume of product travelling through the node N_i and the link connecting the nodes N_i and N_j respectively, for a given distribution plan. Let $I \in \mathbb{N}^+$, $J \in \mathbb{N}^+$, and $L \in \mathbb{N}^+$ be the number of manufacturing facilities, the number of dealerships, and the number of shipping ports respectively. Let $n \in \mathbb{N}^+$ be the number of nodes in the network. The four components for the resilience calculation are defined as in eq. (3.11), (3.12), (3.13), and (3.14).

The resilience for the i-th manufacturer is defined as follows:

$$\omega_i = \frac{v(PF_i)}{\sum_{k=0}^{I} v(PF_k)} \tag{3.11}$$

The resilience of the j-th dealership is:

$$\omega_j = \frac{v(D_j)}{\sum_{k=0}^{J} v(D_k)} \tag{3.12}$$

The resilience of the l-th shipping port is:

$$\omega_l = \frac{v(SP_j)}{\sum_{k=0}^{L} v(SP_k)} \tag{3.13}$$

The resilience of the transportation lane between nodes N_i and N_j is:

$$\omega_{ij} = \frac{v(N_i, N_j)}{\sum_{m=0}^{n} \sum_{k=0}^{n} v(N_k, N_m)} \tag{3.14}$$

The total resilience level of the distribution plan is the maximum of (3.11), (3.12), (3.13), and (3.14). In the present form, the problem is a minimization one, as the goal is to minimize the percentage volume going through the busiest element in the supply chain network. The busiest element in the supply chain is defined by equation (**3.15**).

$$max(\omega_i, \omega_j, \omega_l, \omega_{ij}) \tag{3.15}$$

The resilience level of the network is the opposite of the busiest node. For instance, if the busiest node has 98% of the traffic, according to Caterpillar's definition the network is only 2% resilient. This can be achieved by taking as an objective function one minus

equation (3.17). Equation (**3.16**) defines the final objective function for the problem of resilience maximization.

$$\max\left(1 - max\left(\omega_i, \omega_j, \omega_l, \omega_{ij}\right)\right) \tag{3.16}$$

The monetary value for the potential loss may be derived by weighting the resilience level of the busiest element with the total cost of the products going through such element. Such resilience representation has the advantage that resilience-weighted costs can be accurately represented. Product held at the dealership is more expensive that the same product before it has been shipped from the production facility, as the transportation cost weighs in the product total cost when it arrives at the dealership. The model represents the fact that the product cost is directly proportional to its position on the supply chain network.

3.4 Simulation of Transportation Costs

In the context of supply chains, simulation allows to reproduce and test different decision-making alternatives on more possible foreseeable scenarios, in order to ascertain in advance the level of optimality and robustness of a given strategy [93]. Trends such as globalization, heavy reliance on transportation and communication infrastructures, and lean manufacturing have led to an increase in the vulnerability of supply networks [94]. While having a large, complex supply chain may make a company more susceptible to disruptions, it can also act as an advantage if the network is utilized to mitigate those disruptions [95]. Typically, companies are concerned with maintaining very high level of customer service, even if disruption occurs. A simulation model allows to realistically test various customer behaviours in cases of disruptive events while evaluating the trade-off between service level and product availability (inventory investments drive the marginal profit) [95]. Risk profiles for the locations and connections in the supply chain are usually developed using Monte Carlo simulation, and the flow of material and network interactions are modelled using discrete-event simulation [95].

In the context of global supply chains, collection and exchange of information is critical and is often a challenge. In particular, Caterpillar has low control on the transportation costs, as it is not the owner of the network. Wrong or inaccurate information on transportation prices may affect the validity of the solution produced by the optimization process. In order to reduce the uncertainty, a simple simulation system has been implemented. A Monte Carlo simulation based on a triangular distribution is used to generate an expected value for the transportation costs. A triangular distribution is chosen because it is particularly suited to populations where the relationship between variables is known, but data is limited. The probability density function of the triangular distribution is:

$$P(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & for\ a \le x \le c \\ \frac{2(b-x)}{(b-a)(b-c)} & for\ c < x \le b \end{cases}, \tag{3.17}$$

which is based on the knowledge of minimum a, maximum b, and a likely value (mode) c, and x is in the range $x \in [a, b]$. The distribution function is:

$$D(x) = \begin{cases} \frac{(x-a)^2}{(b-a)(c-a)} & for\ a \le x \le c \\ 1 - \frac{(b-x)^2}{(b-a)(b-c)} & for\ c < x \le b \end{cases}, \tag{3.18}$$

A simulation step consists of generating a random number u according to the uniform distribution (3.18) in the interval $[0,1]$. The random variate x defined in (3.19) is a triangular distribution.

$$x = \begin{cases} a + \sqrt{u(b-a)(c-a)} & for\ 0 < u < f(c) \\ b - \sqrt{(1-u)(b-a)(b-)} & for\ f(c) \le u < 1 \end{cases}, \tag{3.19}$$

where $f(c) = (c-a)/(b-a)$. Figure 3.6 depicts the plot of the probability density function and of the cumulative distribution function.

The expected value of the transportation costs is generated according to (3.19). The actual transportation cost may be selected according to the percentile. Varying the percentile changes the scale of the costs in the studied scenario. Lower percentile means that the costs are supposed to be less than the expected value; therefore, the scenario is more optimistic. Higher percentile defines a scenario where it is more difficult to achieve high marginal profit.

A similar system is used to analyse different inventory scenarios as well. The amount of inventory days required varies with the target service level. Higher service level requires maintaining higher product availability. The inventory days may be generated with the same Monte Carlo simulation. The percentile corresponds to the targeted service level.

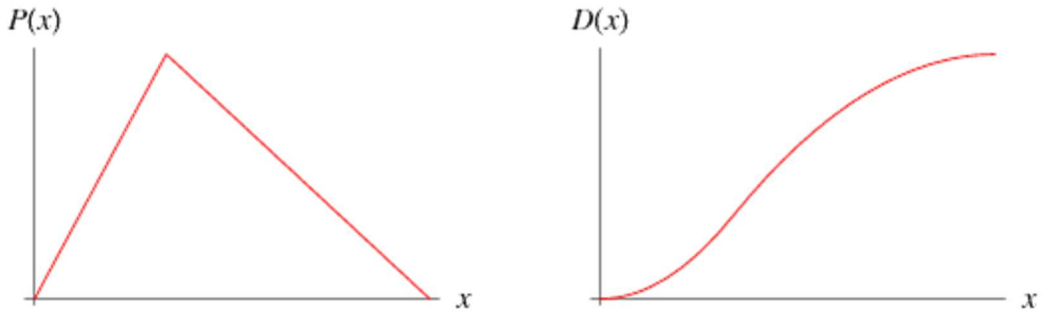

Figure 3.6 – Probability density function and cumulative distribution function of the triangular distribution.

Table 3.2 depicts the parameters used for the Monte Carlo simulation. The variance is used to determine the minimum and maximum variation allowed, namely a and b from equations (3.17)-(3.19).

Table 3.2 – Parameters for the Monte Carlo simulation on transportation costs and inventory levels. The variance is used to determine the minimum and maximum variation allowed, namely a and b from equations (3.17)-(3.19).

Parameter	Value
Number of simulations	10,000.00
Variance	10%
Percentile (service level)	95%

3.5 Ant Colony Optimization for Caterpillar's Supply Chain

The models described in the previous sections are not all linear. The model for lane commitment discount makes the objective function for the problem of profit maximization a polynomial function of degree 3. The objective function for the problem of resilience maximization essentially consists of taking the maximum of the minimum of four sets. Such function is non-linear. Depending on the functions, it is possible to

reduce a non-linear function to a linear one. Typically, the dimensionality of the function domain is increased (i.e. the number of input variables) and the function is projected onto a linear space. If the optimization model is a system of linear equations, linear programming techniques can be employed as optimization algorithm. The details, the advantages, and the disadvantages of linear programming are discussed in section 2.2.1. For the problem of Caterpillar's supply chain, the opted optimization algorithm is a meta-heuristic approach. The industrial partner of this project required that the optimization process would be as flexible as possible on the long term and would handle well increasing complexity. Given such requirement and the non-linearity of some of the models, a meta-heuristic appeared to be the most promising approach. The meta-heuristic algorithm implemented is the Ant Colony System (ACS) as defined in [78]. The ACS definition requires the implementation of three rules or strategies:

1. The state transition rule to drive the search of the ants.
2. The global pheromone updating rule to focus the search on the most promising solution space portion.
3. The local pheromone updating rule to force the ants to explore a larger portion of the solution space.

Informally, the ACS implementation works as follows: a colony releases a set number of ants. Each ant builds a complete distribution plan, if it can find one. The ants are guided in the search by both the heuristic information and by the pheromone level. The rules for pheromone update are designed such that ants choose routes that are more desirable (e.g. more inexpensive). While search for the solution, the ant uses the state transition rule to decide whether to choose the best next route or to select a random one. The ant also modifies the amount of pheromone on visited routes by applying the local pheromone update rule, increasing the amount of pheromone on such routes. Following ants are most likely to select routes with a low amount of pheromone and they will avoid already visited routes. Once all the ants have completed the search, the colony compares the solutions and increases the pheromone only on the routes belonging to the best solution found so far. The colony releases the ants again and repeats the process for a set number of iterations.

Let us recall that $m \in \mathbb{N}^+$ is the number of manufacturers and $n \in \mathbb{N}^+$ is the number of dealership locations in the network. Let $i \in [0, m]$ be a production facility and $j \in [0, n]$ a dealer. Given the ant k, the probability distribution with which the ant k at the production source i move to the dealer j is defined as follows:

$$p_k(i,j) = \begin{cases} \frac{[\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}}{\sum_{u \in J_k(i)} [\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}} & if\ s \in J_k(i) \\ 0 & otherwise \end{cases}, \tag{3.20}$$

where $\tau(i,j)$ is the pheromone value deposited on route (i,j), and $\eta(i,j)$ is the heuristic information. $J_k(i)$ is the set of neighboring nodes that remain to be visited by ant k at node i, $\alpha > 0$ is a parameter which determines the relative importance of pheromone information, and $\beta > 0$ is a parameter which determines the relative importance of heuristic information.

From the probability distribution given in equation (3.20), the state transition rule is:

$$s = \begin{cases} \underset{u \in J_k(i)}{\arg\max} \left\{ [\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta} \right\} & if\ q \leq q_0 \\ S & biased\ exploration \end{cases}, \tag{3.21}$$

where q is a random number uniformly distributed in [0, 1], q_0 is a parameter ($0 \leq q_0 \leq 1$) indicating the relative weighting of exploitation versus exploration, and S is a random variable selected according to the probability distribution given in (3.20).

Only the ant that produced the best solution within the same iteration deposits pheromone. Let $C(k)$ be a measure of ant k's solution based on the objective function. Let ρ be the pheromone decay parameter in the range: $0 < \rho < 1$. Given the best solution found so far v^*, the global pheromone updating rule is defined as follows:

$$\tau(i,j) = (1 - \rho) \cdot \tau(i,j) + \rho \cdot \Delta\tau(i,j), \tag{3.22}$$

where $\Delta\tau_k(i,j)$ is defined as:

$$\Delta\tau_k(i,j) = \begin{cases} C(k) & if\ (i,j) \in v^* \\ 0 & otherwise \end{cases}. \tag{3.23}$$

As an ant constructs a tour, the pheromone level on visited edges is changed by applying the local pheromone updating rule:

$$\tau(i,j) = (1-\rho)\cdot\tau(i,j) + \rho\cdot\tau_0, \tag{3.24}$$

where ρ is the pheromone decay parameter in the range: $0 < \rho < 1$ and τ_0 is the initial pheromone level. The effect of the local update is to decrease the pheromone level on visited edges which make them less desirable to subsequent ants. This increases the exploration of the search space within iterations [83]. Without this decrease, the majority of ants would search within a narrow neighborhood of the best previous tour, which is likely to result in a sub-optimal solution [78].

Table 3.3 summarizes the parameters in the definition above and set the values used in all numerical experiments, unless stated otherwise. The configuration of parameters below is from [78]. Other configurations have been randomly generated and tested before the deployment of the system. However, the configuration from [78] resulted to be the best performing one for all the experiments.

Table 3.3 - Ant Colony System set of parameters for all tested problem instances. These parameters are from [78].

Parameter	Value
Number of Ants	20
Maximum N° of Iterations	1,000
Pheromone Evaporation Rate (ρ)	0.1
Weight on Pheromone Information (α)	1
Weight on Heuristic Information (β)	20
Exploitation to Exploration Ratio ($Q0$)	0.9

3.5.1 Vogel's Approximation Method of Allocation

Vogel's Approximation Method of Allocation (VAM) is a heuristic method for solving the transportation problem [96]. It deals with determining a cost plan for transporting a single unit from several sources to a number of end customers. VAM is based on the concept of a penalty cost, which is defined as the difference between the lowest and

next lowest cost cell in a row or column of a cost matrix describing the alternatives for each source-demand solution [97]. Cost can be interpreted as the transportation cost, distance traveled or the time required to transport a unit from a supplier to a customer.

The general principles of VAM are embedded within the ACS implementation to determine better starting solutions for the transportation network optimization problem. The process has been implemented as follows:

1. For each dealer, the shortest/cheapest route to each manufacturer is computed.
2. The difference between the shortest/cheapest route to one manufacturer and the next shortest/cheapest route to another manufacturer is calculated as the penalty cost.
3. Ants build feasible solutions by firstly assigning dealers with higher penalty costs to manufacturers with shorter/cheaper routes as a mean of optimizing the objective function.

3.5.2 Parallel Implementation

When facing complex optimization problems, parallel computing techniques are usually applied to improve their efficiency, allowing population-based metaheuristics to achieve high quality results in reasonable execution times, even when tackling hard-to-solve optimization problems [98]. Dorigo et al. [92] first suggested the application of parallel computing techniques to enhance both the ACO search and its computational efficiency. Parallel implementations of ACO algorithms have become popular in the last decade in order to improve the efficiency. By splitting the population into several processing elements, parallel implementations of metaheuristics allow reaching high quality results in a reasonable execution time, even when facing hard-to-solve optimization problems [98]. The classic proposals of parallel ACOs focused on traditional supercomputers and clusters of workstations. Nowadays, the novel emergent parallel computing architectures such as multicore processors, graphics processing units (GPUs), and grid environments provide new opportunities to apply parallel computing techniques to improve the ACO search results and to lower the required computation times [98]. Pedemonte et al. [98] presented a survey of the current state-of-the-art of parallel ACO implementations and introduced a new taxonomy to classify the parallel models. The

authors used two main criteria related to the population organization to discriminate the categories in the taxonomy: the number of colonies and the cooperation. The taxonomy clusters the parallelization strategies into the following five categories:

1. **Master-slave model**. This category applies a hierarchical parallel model, where a master process manages the global information (i.e. pheromone matrix, best-so-far solution, etc.) and it also controls a group of slave processes that perform subordinated tasks. The model includes three level of granularity: coarse, medium, and fine grain. The granularity level defines the type of operations delegated to the slave agent. At the highest level, the tasks of the slave may correspond to one or more ants, and they comprise building, improving and/or evaluating one or more full solutions, and communicating back the result to the master. At the lowest level, the slave processes single components used to construct solutions, and frequent communication between the master and the slaves is usually required. The slave never solves the problem to its entirety.
2. **Cellular model**. The search space is structured in small neighbourhoods, and a different colony is assigned the task of finding the optimal solution in its neighbourhood.
3. **Parallel independent runs model**. A set of colonies explore the solutions space simultaneously and the search processes are executed on a set of processor. The executions are completely independent and the colonies have no interaction with one another.
4. **Multi-colony model**. A set of colonies explore the solution space simultaneously, each one using their own pheromone matrix. The colonies periodically cooperate and exchange information.
5. **Hybrid model**. This category includes those proposals that feature characteristics from more than one parallel model.

The most common metrics to measure the performance of parallel algorithms are the *speedup* and the *computational efficiency*. The speedup evaluates how much faster a parallel algorithm is than a corresponding sequential algorithm. The computational efficiency is the normalized value of the speedup, regarding the number of processors used to execute a parallel algorithm. Let T_1 be the execution time of the sequential

algorithm and T_m be the execution time of the parallel version using m processors. Equations (3.25) and (3.26) define the speedup and the efficiency respectively.

$$S_m = \frac{T_1}{T_m}, \tag{3.25}$$

$$E_m = \frac{S_m}{m}. \tag{3.26}$$

When $S_m < m$ the speedup is said to be sublinear. The speedup is linear when $S_m = m$, and superlinear when $S_m > m$. The ideal case for a parallel algorithm is to achieve linear speedup, although the most common situation is to achieve sublinear speedup values due to the times required to communicate and synchronize the parallel processes [98].

The ACO implementation for solving Caterpillar's supply chain optimization problem belongs to the third category: parallel independent runs model. A set of colonies using different parameters are concurrently executed on a set of processors. When all colonies have met their termination conditions, all found solutions are compared to each other, and the best one is kept. The parallel implementation has been developed with OpenMP [99] and the C++ latest standard. The parallel model implemented introduces very low overhead as all colonies run independently and no concurrent communication is required, and it is very flexible with respect to the number of cores available. The system has been deployed both on standard workstations and a high performance computing platform. The number of cores on the standard workstations varies from 4 to 8. A node on the high performance computing platform has 128 cores.

3.6 Experiments

The model and optimization algorithms described above have been tested on a dataset provided by Caterpillar for a medium size excavator, and two randomly generated datasets. The transportation network is shared by all three datasets and consists of 200 dealerships locations, 40 production facilities and assembly points, and 68 shipping ports. The connections between nodes in the network are determined by real-world

transportation routes. The demand and capacity in Caterpillar's problem is from a twelve months period from January 2015 to December 2015. The medium size excavator has been broken down into ten main components. Caterpillar's products are typically made of tens of thousands of components which are produced and shipped from all around the world. However, most of those components are inexpensive and do not significantly contribute to the cost of the machine. Frequently, about 20% of the components affect approximatively 80% of the total cost of the machine. Modelling thousands of components would significantly increase the overall complexity of the system, not just the optimization process, but also the processes for data collection and results interpretation. The ten symbolic components described in the dataset are:

1. Main engine;
2. Front bucket;
3. Hydraulic system and motor for travel operations;
4. Hydraulic system and motor for bucket operations;
5. Main pumps for the hydraulic system;
6. Main valves for the hydraulic system;
7. Cylinders for boom, stick and bucket movements;
8. Operator station – cab group;
9. Undercarriage – track group;
10. Main structures – lower and upper frames.

The results of the optimization of this dataset are reported in Table 3.5. The units for the rows and columns reporting the objective values have been purposely omitted due to commercial sensitivity of such data. Table 3.8 and Table 3.11 depict the results of the optimization for the two randomly generated problems. In both instances, the number of dealers, production facilities and shipping ports is the same as in the original problem; it is only the demand figures, the production capacities, the transportation times and costs and the sale prices that have been randomly generated. In the first problem, the figures have been generated according to a normal distribution with the same mean and standard deviation as in the original dataset (e.g. the demand figures have the same mean and standard deviation as those found in the original problem). The figures for the second problem are randomly generated in an interval between 0 and an upper limit which is a random increase over the maximum value in the original data, according to a

negative exponential distribution. The three datasets have been published on Figshare (figshare.com), an online digital repository. The original Caterpillar's dataset is published in [4]. The dataset generated according to a uniform distribution is in [5] and the last dataset is in [6]. Table 3.4 depicts the main characteristics of the three datasets. The analysis on network complexity is presented in Table 2.2, and is also applied to the considered problems. The comparison between the analysis presented in Table 2.2 and the problems addressed in this work is reported in Figure 3.7.

Table 3.4 – Analysis of the network complexity of the 3 datasets employed for the experiments presented in this chapter. The complexity analysis has been defined in section 2.2. The results may be compared with the values in Table 2.2.

	No. of Nodes	No. of Edges	No. of Sub-Graphs	Beta-index	No. of Cycles	Total Demand	Total Components Capacity	Total Machine Capacity
Dataset [4]	2,945.00	135,111.00	1.00	45.88	132,167.00	1,129.00	2,899,971.00	1,425.00
Dataset [5]	2,945.00	135,111.00	1.00	45.88	132,167.00	1,593.00	2,899,971.00	1,694.00
Dataset [6]	2,945.00	135,111.00	1.00	45.88	132,167.00	20,093.00	2,899,971.00	21,369.00

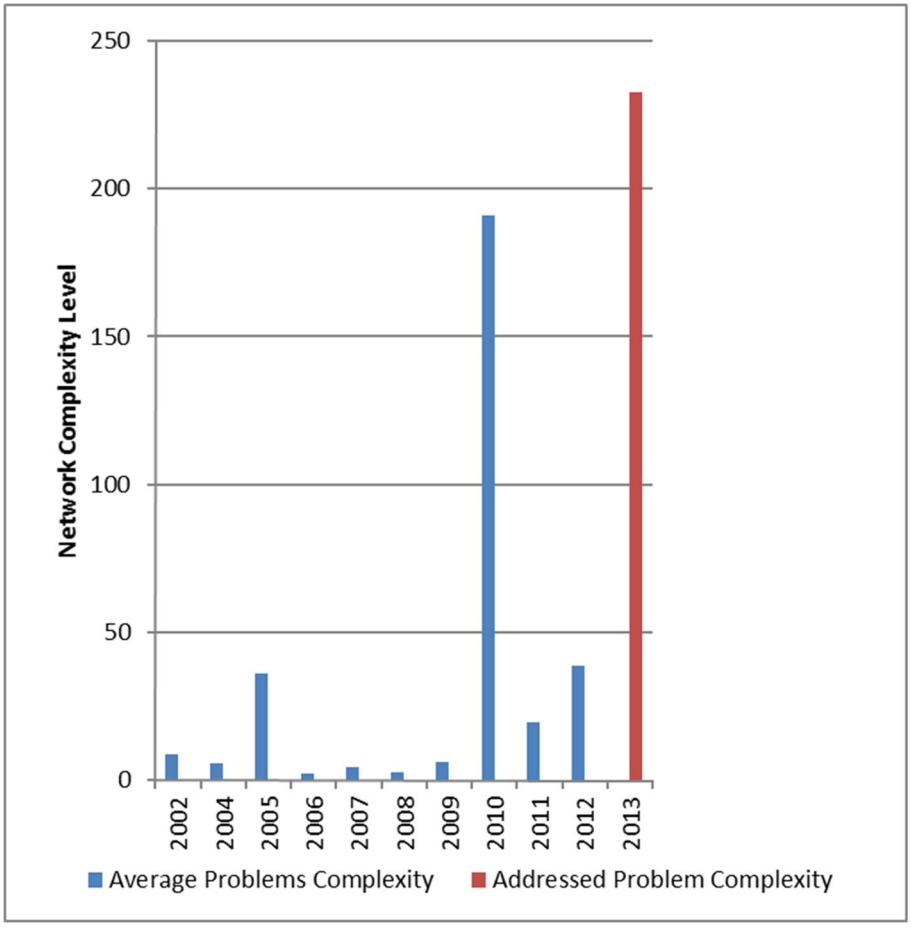

Figure 3.7 - Average complexity of transportation network problems addressed in the last decade. Features considered in measuring the problem complexity are number of nodes in the transportation network, number of edges, number of sub-graphs, and length of the whole network, the number of cycles in the graph and its level of connectivity. The value for 2013 indicates the complexity of the problem addressed in this work. Source of the analysis is Ogunbanwo et al. [1].

All the single experiments reported in Table 3.5, Table 3.8, and Table 3.11 are the results of the average over 10 runs of the optimization for the given objective and parameter configuration. The standard deviation has been reported as well. All tests have been performed on a Linux server, with an AMD Opteron 6140 800 MHz 16 core processor and 64 GB of RAM. To demonstrate the effectiveness of the optimization algorithm, four groups of runs of the optimization have been carried out with four different objective functions: profit maximization, transportation time minimization, transportation costs minimization, and resilience level maximization. The four solutions

have been measured against the remaining three objective functions. The best solution for a given goal is expected to be produced when an objective function is directly used in the optimization.

Table 3.6 and Table 3.7 measure respectively the percentage difference and percentage error between such measurements of a given solution. The percentage difference is defined as $\boldsymbol{d} = \frac{|x_1 - x_2|}{\left(\frac{x_1 + x_2}{2}\right)}$ where $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the two measurements to compare. The percentage error is $\boldsymbol{e} = \frac{|x_2 - x_1|}{x_1}$. The percentage difference is often used to compare values pertaining to the same property or characteristic and both values are calculated using different methods. The percentage error is used to compare a given measurement to the known or accepted value. Table 3.9 and Table 3.10 report the same analysis for the first randomly generated problem ([5]), Table 3.12 and Table 3.13 for the second random problem ([6]).

Table 3.5 – Results of the optimization performed on the dataset in [4]. Each experiment is the result of the average over 10 runs of the optimization for the given objective and parameter configuration. The standard deviation is reported alongside each result. Performance measures for the optimization are maximization of profit, transportation time and costs minimization, and network resilience maximization. The four distribution plans produced are measured against the remaining objectives. The runtimes of the experiments are based on a Linux server, with an AMD Opteron 6140 800 MHz 16 cores processor and 64 GB of RAM.

	Profit	Transportation Time (s)	Transportation Cost	Resilience (%)	Running Time (s)
Profit (max)	101,247,174.47	529,809.91	9,544,245.33	93.7500%	59
Standard Deviation	18,496.27	74.45	18,577.35	0.0000%	0.894427191
Transportation Time (min)	63,119,932.77	99,831.49	3,222,379.73	95.5682%	42.3
Standard Deviation	38,799.17	491.19	2,553.32	0.2425%	0.640312424
Transportation Cost (min)	63,077,829.61	101,021.18	3,174,611.82	93.6364%	45.7
Standard Deviation	2,616.81	377.88	50.79	0.1515%	0.640312424
Resilience (max)	63,231,835.10	103,392.81	3,220,912.33	98.8636%	43.1
Standard Deviation	17,466.77	621.99	983.24	1.2591%	0.3

Table 3.6 - Percentage difference between each pair of performance measure reported in Table 3.5. Given two performance measures x_1 and x_2 of a distribution plan, the percentage difference is defined as $d = \frac{|x_1 - x_2|}{\left(\frac{x_1 + x_2}{2}\right)}$.

Percentage Difference	Profit	Time	Cost	Resilience
Profit	0	1.36578826	1.001604694	0.053097345
Transportation Time	0.463927879	0	0.014934491	0.033898305
Transportation Cost	0.464559184	0.01184639	0	0.054309327
Resilience	0.46225156	0.035048125	0.014479036	0

Table 3.7 - Percentage error between each pair of performance measure reported in Table 3.5. Given two performance measures x_1 and x_2 of a distribution plan, the percentage error is defined as $e = \frac{|x_2 - x_1|}{x_1}$.

Percentage Error	Profit	Time	Cost	Resilience
Profit	0	4.307041877	2.006429093	0.051724138
Transportation Time	0.376575859	0	0.01504685	0.033333333
Transportation Cost	0.376991704	0.011916976	0	0.052873563
Resilience	0.37547062	0.035673266	0.014584621	0

Table 3.8 - Results of the optimization performed on the dataset in [5]. Each experiment is the result of the average over 10 runs of the optimization for the given objective and parameter configuration. The standard deviation is reported alongside each result. Performance measures for the optimization are maximization of profit, transportation time and costs minimization, and network resilience maximization. The four distribution plans produced are measured against the remaining objectives. The runtimes of the experiments are based on a Linux server, with an AMD Opteron 6140 800MHz 16 cores processor and 64 GB of RAM.

	Profit	Transportation Time (s)	Transportation Cost	Resilience (%)	Running Time (s)
Profit (max)	145,774,697.02	810,379.35	19,498,373.65	94.1288%	81.5
Standard Deviation	22,766.40	540.14	36,492.18	0.0000%	0.5
Transportation Time (min)	91,383,160.69	144,411.85	4,440,285.50	95.5682%	42.3
Standard Deviation	74,114.77	1,054.73	4,375.83	0.2425%	0.458257569
Transportation Cost (min)	91,299,308.82	145,683.05	4,433,667.42	93.5985%	45
Standard Deviation	3,099.76	408.12	33.01	0.1136%	0.447213595
Resilience (max)	91,583,318.45	150,875.64	4,490,177.08	98.8636%	42.7
Standard Deviation	21,741.70	688.33	720.15	1.2591%	0.458257569

Table 3.9 - Percentage difference between each pair of performance measure reported in Table 3.8. Given two performance measures x_1 and x_2 of a distribution plan, the percentage difference is defined as $d = \frac{|x_1 - x_2|}{\left(\frac{x_1 + x_2}{2}\right)}$.

Percentage Difference	Profit	Time	Cost	Resilience
Profit	0	1.39500134	1.258957076	0.049067713
Transportation Time	0.458694785	0	0.001491574	0.033898305
Transportation Cost	0.459564413	0.008764042	0	0.054713639
Resilience	0.456621433	0.043779641	0.012664867	0

Table 3.10 - Percentage error between each pair of performance measure reported in Table 3.8. Given two performance measures x_1 and x_2 of a distribution plan, the percentage error is defined as $e = \frac{|x_2 - x_1|}{x_1}$.

Percentage Error	Profit	Time	Cost	Resilience
Profit	0	4.611584891	3.397797983	4.7893%
Transportation Time	0.373120558	0	0.001492687	0.033333333
Transportation Cost	0.373695774	0.008802615	0	0.053256705
Resilience	0.371747496	0.044759416	0.012745578	0

Table 3.11 - Results of the optimization performed on the dataset in [6]. Each experiment is the result of the average over 10 runs of the optimization for the given objective and parameter configuration. The standard deviation is reported alongside each result. Performance measures for the optimization are maximization of profit, transportation time and costs minimization, and network resilience maximization. The four distribution plans produced are measured against the remaining objectives. The runtimes of the experiments are based on a Linux server, with an AMD Opteron 6140 800MHz 16 cores processor and 64 GB of RAM.

	Profit	Transportation Time (s)	Transportation Cost	Resilience (%)	Running Time (s)
Profit (max)	859,562,297.85	10,518,862.90	417,732,276.52	94.5076%	104.5
Standard Deviation	1,106,300.99	12,458.76	1,719,479.11	0.2934%	1.5
Transportation Time (min)	410,217,795.38	2,335,651.42	134,270,885.96	99.2424%	80.3
Standard Deviation	2,765,770.35	15,287.62	1,055,933.06	0.4791%	0.458257569
Transportation Cost (min)	430,072,213.94	2,559,278.28	128,819,951.87	95.5303%	85.2
Standard Deviation	901,755.32	14,431.10	400,214.36	0.0928%	0.4
Resilience (max)	419,856,111.17	2,759,706.46	141,709,089.55	99.2424%	81.7
Standard Deviation	5,073,481.34	39,583.02	1,316,285.25	0.8299%	0.458257569

Table 3.12 - Percentage difference between each pair of performance measure reported in Table 3.11. Given two performance measures x_1 and x_2 of a distribution plan, the percentage difference is defined as $d = \frac{|x_1 - x_2|}{\left(\frac{x_1 + x_2}{2}\right)}$.

Percentage Difference	Profit	Time	Cost	Resilience
Profit	0	1.273204304	1.057217626	0.048875855
Transportation Time	0.707751688	0	0.041437658	2.01701E-13
Transportation Cost	0.666064811	0.091370816	0	0.038117464
Resilience	0.687353228	0.166447596	0.095288385	0

Table 3.13 - Percentage error between each pair of performance measure reported in Table 3.11. Given two performance measures x_1 and x_2 of a distribution plan, the percentage error is defined as $e = \frac{|x_2 - x_1|}{x_1}$.

Percentage Error	Profit	Time	Cost	Resilience
Profit	0	3.503609916	2.242760694	0.047709924
Transportation Time	0.522759669	0	0.042314362	2.01701E-13
Transportation Cost	0.49966138	0.095744964	0	0.03740458
Resilience	0.511546618	0.181557501	0.100055446	0

Figure 3.8 shows the costs breakdown for the solution of the problem of profit maximization. The values are reported as fractions of the total revenue. As shown, for the original Caterpillar's problem, the solution achieved a profit of 8% over the revenue for the considered year. The last series in the plot is the average of the components costs. The production cost is by far the factor that most affect the profit. Considering that the optimizer may choose between many routes but only few production sources, such result suggests the optimization algorithm is quite effective in making the right choices. It is worth noticing the international trade costs affect significantly the overall profit and have a higher impact than the inventory costs. Figure 3.9 and Figure 3.10 report the costs breakdown for the profit maximization experiments for the two randomly generated datasets, [5] and [6] respectively.

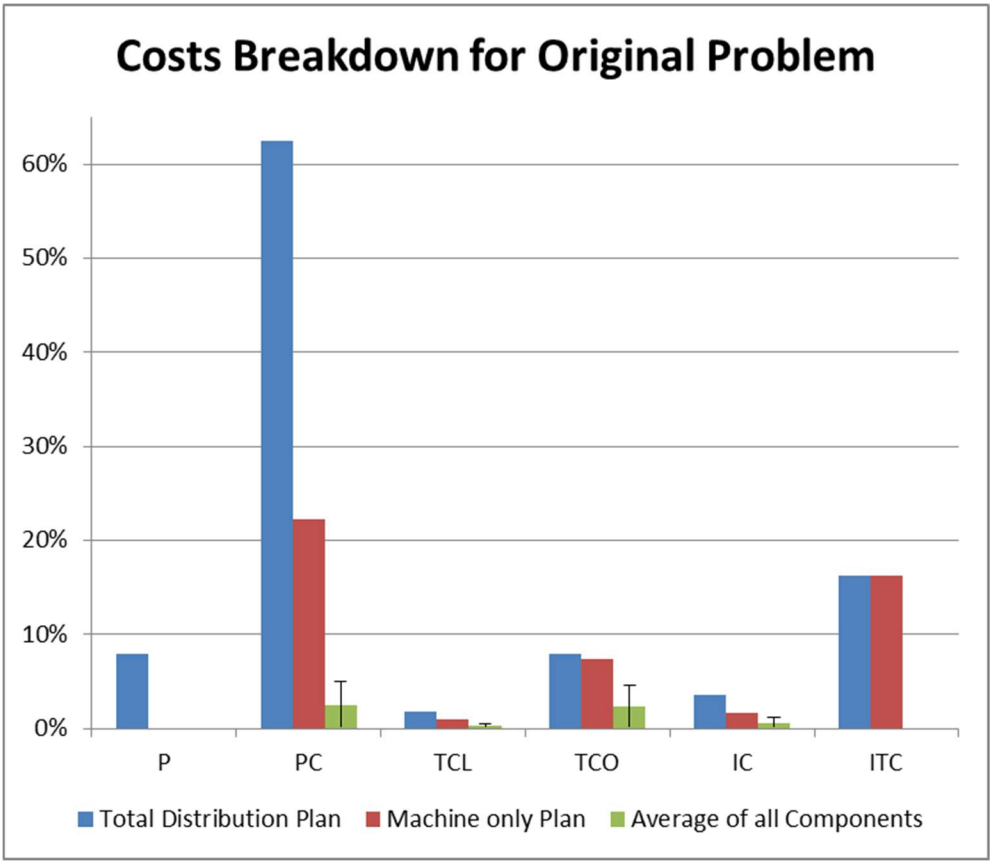

Figure 3.8 - Costs breakdown for the profit calculation reported in Table 3.5. The overall profit and costs are represented as percentage of the total sale. *P* stands for *profit*, *PC* for *production cost*, *TCL* for *transportation costs over land*, *TCO* for *transportation costs over ocean*, *IC* for *inventory costs*, and *ITC* for *international tariff costs*. The costs analyzed are from the distribution plan produced when optimizing for profit maximization. The achieved profit is over 8% of the total sale revenue and the largest cost is from machine production. International tariff cost is also quite significant.

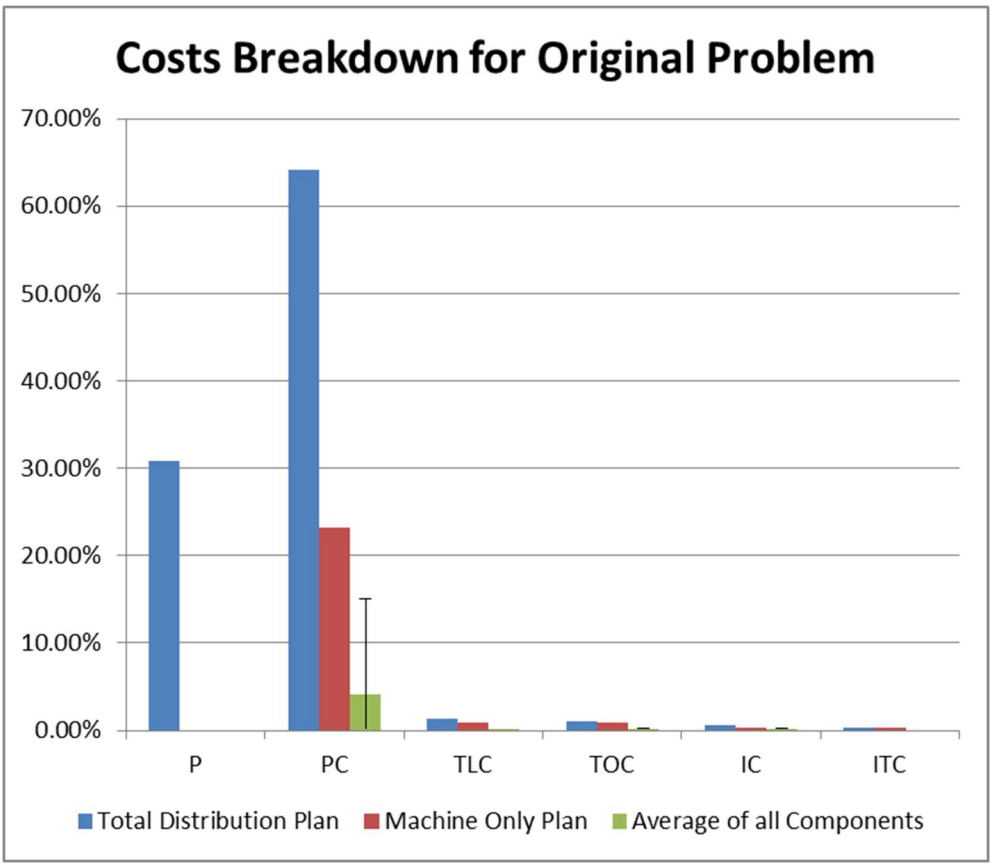

Figure 3.9 - Costs breakdown for the profit calculation reported in Table 3.8. The overall profit and costs are represented as percentage of the total sale. *P* stands for *profit*, *PC* for *production cost*, *TCL* for *transportation costs over land*, *TCO* for *transportation costs over ocean*, *IC* for *inventory costs*, and *ITC* for *international tariff costs*. The costs analyzed are from the distribution plan produced when optimizing for profit maximization. The achieved profit is slightly over 30% of the total sale revenue and the largest cost is from the production cost.

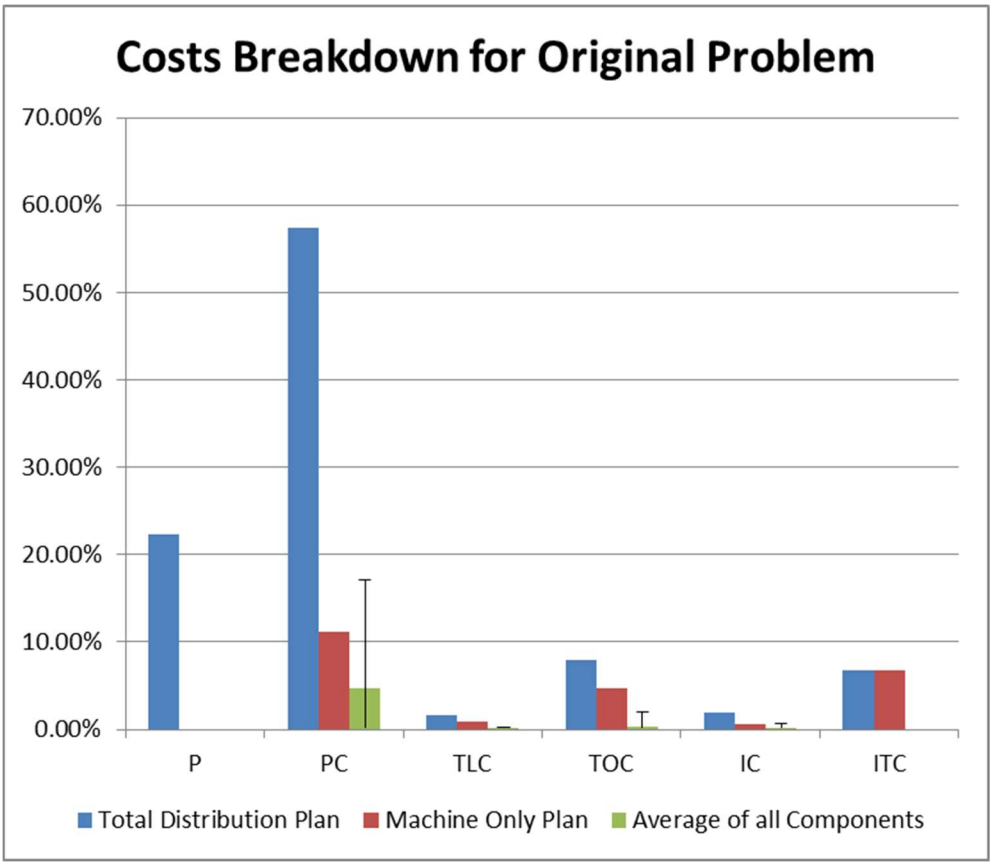

Figure 3.10 - Costs breakdown for the profit calculation reported in Table 3.11. The overall profit and costs are represented as percentage of the total sale. *P* stands for *profit*, *PC* for *production cost*, *TCL* for *transportation costs over land*, *TCO* for *transportation costs over ocean*, *IC* for *inventory costs*, and *ITC* for *international tariff costs*. The costs analyzed are from the distribution plan produced when optimizing for profit maximization. The achieved profit is over 20% of the total sale revenue and the largest cost is from the production cost.

The improvements of the parallel implementation are shown in Table 3.14. The runtimes show the result of the average of 10 runs on a Linux server. The values indicated as S_m and E_m are the *speedup* and the *computational efficiency* respectively defined by equations (3.25) and (3.26). The speedup evaluates how much faster a parallel algorithm is than a corresponding sequential algorithm. The computational efficiency is the normalized value of the speedup, regarding the number of processors used to execute a parallel algorithm. The gain improvement for 1,000 iterations on 16

cores was a 80% reduction of the runtime consistently for all the goals and datasets. The problem of profit maximization has a higher requirement of runtime. This is explained by the fact that the production cost of the suppliers is involved in the performance calculation. The decision is now finding the most inexpensive route to the supplier with the lowest production cost. Given the suppliers limited capacity, the order of the dealerships with which the capacity is allocated significantly affect the quality of the solution.

The optimization system presented in the current chapter has been deployed in Caterpillar's production environment. At the time of writing, the system is being used to design distribution plans for more than 7,000 products. Most of such distribution plans are less expensive than the ones implemented previous to the utilization of this system. Specifically, they have reduced transportation and inventory costs. Overall, the system improved Caterpillar's marginal profit on such products of a factor of 4.6% on average.

Table 3.14 - Average speedups of the ACS with OpenMP multi-core implementation. The runtimes are based on 10 experiments run on a Linux server, with an AMD Opteron 6140 800 MHz 16 cores processor and 64 GB of RAM. The parameters are as in Table 3.3. The values indicated as S_m and E_m are the *speedup* and the *computational efficiency* respectively. The speedup evaluates how much faster a parallel algorithm is than a corresponding sequential algorithm. The computational efficiency is the normalized value of the speedup, regarding the number of processors used to execute a parallel algorithm. Equations (3.25) and (3.26) define the speedup and the efficiency respectively.

		Transportation Time (s)	Transportation Cost (s)	Profit (s)	Resilience (s)
Dataset [4]	**AVG 1 core**	215.05879	245.3068875	300.7726	245.325001
	STD 1 core	1.78951109	3.178486019	3.744533	1.33965912
	AVG 16 cores	42.3	45.7	59	43.1
	STD 16 cores	0.64031242	0.640312424	0.894427	0.3
	S_m	5.08413214	5.367765591	5.09784	5.69199538
	E_m	0.31775826	0.335485349	0.318615	0.35574971
Dataset [5]	**AVG 1 core**	214.308642	244.8488318	298.6202	244.53973
	STD 1 core	1.84355266	2.822339254	3.41482	1.6310094
	AV 16 cores	42.3	45	81.5	42.7
	STD 16 cores	0.45825757	0.447213595	0.5	0.45825757
	S_m	5.06639816	5.441085151	3.664052	5.72692577
	E_m	0.31664989	0.340067822	0.229003	0.35793286
Dataset [6]	**AVG 1 core**	405.423752	471.0532426	554.313	453.850783
	STD 1 core	3.91480429	5.346803807	4.800868	2.48547111
	AVG 16 cores	80.3	85.2	104.5	81.7
	STD 16 cores	0.45825757	0.4	1.5	0.45825757
	S_m	5.04886366	5.528793927	5.304431	5.55508914
	E_m	0.31555398	0.34554962	0.331527	0.34719307

3.7 Summary

This chapter introduced a detailed description of the problem of supply chain optimization for Caterpillar. The chapter started with a general description of Caterpillar's Global Supply Chain for a medium sized excavator. Three mathematical models based on three performance metrics were presented to clearly define the optimization problems. The definition of the models addresses most of the shortcomings highlighted in chapter 3 and work as a reference for real-world global supply chain.

The chapter continues with a description of the simulation system used to analyse different scenarios of transportation cost and inventory levels. The costs and inventory levels are generated according to a Monte Carlo simulation with a triangular distribution.

The optimization algorithm chosen is the Ant Colony System and the theoretical characteristics as well as the implementation details are discussed in the current chapter. A heuristic method named Vogel's Approximation has been employed to improve the initialization of the algorithm. To reduce the runtime requirements of the optimization, a parallel variant of the Ant Colony System has been implemented through the multiprocessing platform OpenMP.

The last section of the chapter summarizes the main experiments run to test the optimization system. The tested performance measures for the optimization are maximization of profit, transportation time and costs minimization, and network resilience maximization. The four distribution plans produced were measured against the remaining objectives to demonstrate the effectiveness of the optimization process. The solutions of the profit maximization problem have been broken down to better display the behaviour of the algorithm on the most comprehensive model.

In the last set of experiments, the running times of the single-core implementation have been compared against the parallel version of the ACS.

The discussed optimization system is now used to design the distribution plans of more than 7,000 products at Caterpillar. The system improved Caterpillar's marginal profit on such products of a factor of 4.6% on average.

4 COMPOSITE GOAL METHODS FOR SUPPLY CHAIN OPTIMIZATION

This chapter is based on the results published in [2].

When optimizing transportation networks, several criteria can be used as the optimization goal, such as the shortest distance traveled, minimum inventory, minimum transportation cost and highest network resilience. In the case of industry based applications, it is often advantageous to simultaneously consider several of these goals with a view to developing a model that more accurately represents the operation of the actual business. Defining a mathematical model that incorporates the perspective of more than one criterion in itself is not a simple task and often involves the definition of complex non-linear models. Moreover, the goals of such criteria may well be mutually exclusive and result in the definition of a multi-goal model that is not or not always achievable in practice.

A simple way to handle the multi-objective optimization problem is to construct a composite objective function that is the weighted sum of the conflicting objectives [21]. In the literature this technique is also referred to as the *preference-based strategy* and is

the approach most often adopted in academic studies. The preference-based strategy is a trade-off that reduces a multi-goal approach to a single-goal optimization problem. However, in reality as a solution this trade-off has proved to be very sensitive to the relative preferences assigned to the goals [21] and in practice it is difficult for practitioners, even those familiar with the problem domain to precisely and accurately select such weightings [100].

The current chapter identifies the principal alternative methods for use in multi-objective optimization when applied to the solution of real-world supply chain optimization problems. The aim of this work is to identify and test those multi-objective optimization techniques that better address the complexities of Caterpillar's operating environment.

In the following sections, four generic strategies are described, which are used to optimize multi-goal problem scenarios and formally present seven implementations of these strategies. The methods have been designed and implemented with a view to solving the supply chain optimization problem reported in chapter 3.

In sections 4.1 and 4.2 the background to this work is presented and previous work on multi-goal optimization is introduced. In section 4.3 the methods used to combine single-goal optimization problems are formally described. In section 4.4 the outcome of the numerical experiments undertaken to verify and test the effectiveness of the proposed methods are outlined.

4.1 Motivations and Related Work

A robust solution to the multi-goal optimization problem is of particular interest to real-world applications where several optimization objectives are commonly involved. Multi-goal problems usually do not have a single 'best' solution, but are characterized by a set of solutions that are superior to others when considering all objectives [101]. This set is referred to as the Pareto set or as the non-dominated solution [101]. This multiplicity of solutions can be explained by the fact that individual objectives are often in conflict [101]. For example, Altiparmak et al. [34] defined three objectives for the transportation network optimization problem: the total cost, the total satisfied customer demand and the equity of the capacity utilization ratio for each production source. The

authors then implement a genetic algorithm to find the set of Pareto-optimal solutions. A similar example is presented in Yagmahan et al. [102] for the flow shop scheduling problem. The multi-objective function in this instance consists of minimizing the distance between the values of all the single-objective functions.

According to the following review it appears that most of the solutions proposed for multi-objective optimization problems are either specific to the kind of problem or to the kind of technique used to determine the optimal solution. In general, four generic solution strategies are used to solve multi-objective optimization problems.

The first strategy is called Goal Synthesis and requires the definition of a mathematical model which includes all the single-goal problems. This category is also referred to as the *preference-based strategy* [21]. The model defines one search space which is a sub-space of the intersection of the single-goal problem search spaces. The best composite solution is then sought on this space along one path. The solution found is feasible for each single-goal problem separately, but it is not necessarily the optimal one. Applying this strategy is no different from solving any other optimization problem: firstly, a mathematical model is defined and then an optimal solution is sought using an appropriate optimization algorithm. However, there is no guarantee that the intersection of the single-goal problems exists or that the definition of such a multi-goal model is even possible.

The second strategy is called Superposition and in contrast to the previous method does not require the definition of a multi-objective problem model. Firstly, a solution is computed for each of the single-goal problems and then a combination of them are taken as the multi-goal solution. The applicability of this strategy relies on the definition of a combination operator. Again it is possible that the combination of the single-goal solutions is empty and a feasible solution does not exist. Das et al. [103] designed a method based on this strategy to solve generic non-linear multi-objective optimization problems.

The third strategy is called Incremental Solving. Here each single-goal problem is solved sequentially in accordance with a predefined order, and the starting exploration point of the i-th problem is the solution or stopping point of the $(i-1)$-th problem. The solution for the multi-goal problem depends on the order used to solve the single-goal

problems. Boudahri et al. [17] adopted this strategy to optimize an agricultural products supply chain.

The final strategy is called Exploration and is based on a 'brute force' approach. Firstly, a large number of feasible solutions are generated for each single-goal problem and then the multi-goal solution is taken as the solution that represents the 'best' compromise for the set of single-goal problems. Applying this strategy should always lead to a solution, provided a feasible solution exists for at least one of the single-goal problems. In common with many brute force approaches the cost of producing a quality solution is computational expensive. Bevilacqua et al. [29] adopted this strategy to solve a generic distribution network and employed a genetic algorithm to improve the generation of solutions.

Figure 4.1 shows a visual representation of these four general strategies from the perspective of the search space and optimization steps.

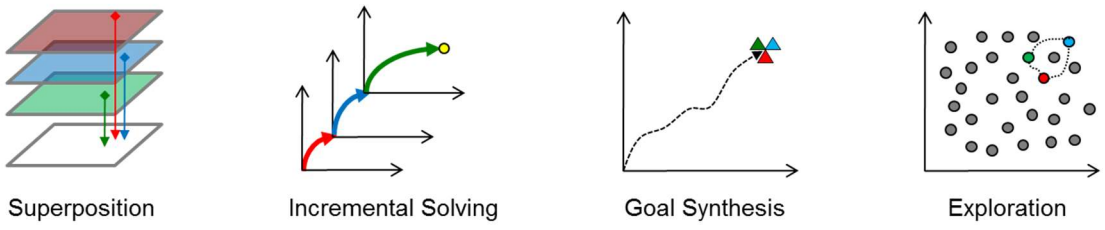

Figure 4.1 – Visual representation of the four general strategies described in this work for the problem of multi-objective optimization.

Aslam et al. [21] and Ogunbanwo et al. [1] provide extensive reviews of the work undertaken for the problem of transportation network optimization. The works presented in such reviews have been analyzed and categorized based on the reported methods with respect to those developed to solve multi-objective optimization problems. Table 4.1 and Figure 4.2 show the results of that analysis. It is possible to notice that in recent years the Goal Synthesis strategy is the dominant method used. Nevertheless, despite its popularity this work will show that it may not necessarily be the best choice when solving real-world supply chain optimization problems.

As will be discussed in the following sections, the method used in this work to solve the current specific real-world optimization problem is the Ant Colony System algorithm

[78]. Garcia-Martinez et al. [104] analyzed several ant colony optimization variants for multi-goal optimization and presented a taxonomy for them. The authors also performed an empirical analysis for the traveling salesman problem and compared their results with two other well-known multi-objective genetic algorithms. It is worth noting that a prerequisite of such analysis is to define a multi-goal model to generate the Pareto optimal frontier. Once again, the authors proposed a model that simultaneously considers all optimization goals (i.e. goal synthesis). This indicates a preference for the goal synthesis strategy over the use of alternatives.

Table 4.1 - Objectives investigated and strategies used in existing approaches for solving several multi-objective optimization problems in the area of operations research. SCO is *supply chain optimization*, MO is *multi-objective*, GS is *goal synthesis*, EX is *exploration*, IS is *incremental solving*, SP is *superposition*, and ACO is *ant colony optimization*.

Author (year)	Multi-objective method	Description
Altiparmak et al., 2006, [34]	GS	SCO for minimum transit time and minimum transportation costs
Bevilacqua et al., 2012, [29]	EX	SCO for minimum transit time and minimum transportation costs
Boudahri et al., 2011, [17]	IS	SCO for minimum travelled distance and minimum transportation costs
Cardona-Valdes et al., 2011, [105]	GS	SCO for minimum transit time and minimum transportation costs
Che Z. et al., 2010, [49]	GS	SCO for minimum transit time, minimum transportation costs, and maximum product quality
Che Z., 2012, [39]	GS	SCO for minimum transit time and minimum transportation costs
Chen C. et al., 2007, [61]	GS	SCO for minimum transit time and minimum transportation costs
Cintron et al., 2010, [106]	GS	SCO for minimum travelled distance, minimum transportation costs, maximum service level, and maximum product quality
Ding et al. 2004, [62]	GS	SCO for minimum transit time and minimum transportation costs
Ding et al., 2009, [8]	GS	SCO for maximum service level and minimum transportation costs

Ghoseiri et al., 2010, [48]	GS	SCO for minimum travelled distance and minimum transportation costs
Huang et al., 2011, [41]	GS	SCO for minimum transportation costs and maximum network resilience
Kamali et al., 2011, [107]	GS	SCO for minimum transportation costs and maximum service level
Liang, 2008, [108]	GS	SCO for minimum travelled distance, minimum transit time, and minimum transportation costs
Lin et al., 2008, [55]	GS	SCO for minimum transit time, minimum transportation costs, and maximum service level
Sadjady et al., 2012, [40]	GS	SCO for minimum travelled distance, minimum transportation costs, and minimum transit time
Utama et al., 2011, [42]	GS	SCO for minimum travelled distance, minimum transportation costs, maximum service level, maximum product quality, and minimum environmental impact
Wang, 2009, [50]	GS	SCO for minimum transportation costs and maximum network resilience
Yeh et al., 2011, [19]	GS	SCO for minimum transportation costs, minimum transit time, minimum environmental impact, and maximum product quality
Chen et al., 2004, [109]	GS	SCO for minimum transportation costs, maximum service level, and maximum network resilience
Sabri et al., 2000, [110]	IS	SCO for minimum travelled distance, minimum transportation costs, maximum network resilience, maximum service level and maximum product quality
Joines et al., 2002, [111]	GS	SCO for minimum transportation costs and maximum service level
Wang et al., 2011, [112]	GS	SCO for minimum transportation costs and minimum environmental impact
Torabi et al., 2008, [113]	GS	SCO for minimum transportation costs and maximum product quality
Amid et al., 2011, [114]	GS	Multi-goal supplier selection optimization
Wang et al., 2004, [115]	GS + SP	Multi-goal supplier selection optimization. GS is used to have a MO model and SP to determine the weights
Weber et al. 1993, [116]	GS	Multi-goal supplier selection optimization
Liu et al., 2000, [117]	GS	Multi-goal supplier selection optimization with goal synthesis for combination of 23 goals / factors

Kumar et al., 2004, [118]	GS	Multi-goal supplier selection optimization
Leung et al., 2007, [119]	GS	Trade-off between robustness and effectiveness of solution for multi-site production planning optimization problem
Yildiz et al., 2009, [120]	EX	Hybrid hill climbing optimization for manufacturing optimization with goals of minimizing the mass of the brake and minimizing the stopping time
Chaharsooghi et al., 2008, [121]	GS + EX	Efficient multi-goal ACO for multi-objective resource allocation problem. GS is used to have a MO model and EX to efficiently explore the Pareto optimal frontier
McMullen et al. 2006, [122]	GS + EX	Assembly line balancing optimization for the goals of crew size optimization, system utilization, jobs scheduling, and system design costs. GS is used to have a MO model and EX to efficiently explore the Pareto optimal frontier
Das et al., 1998, [103]	SP	New multi-purpose method for generating the Pareto optimal points

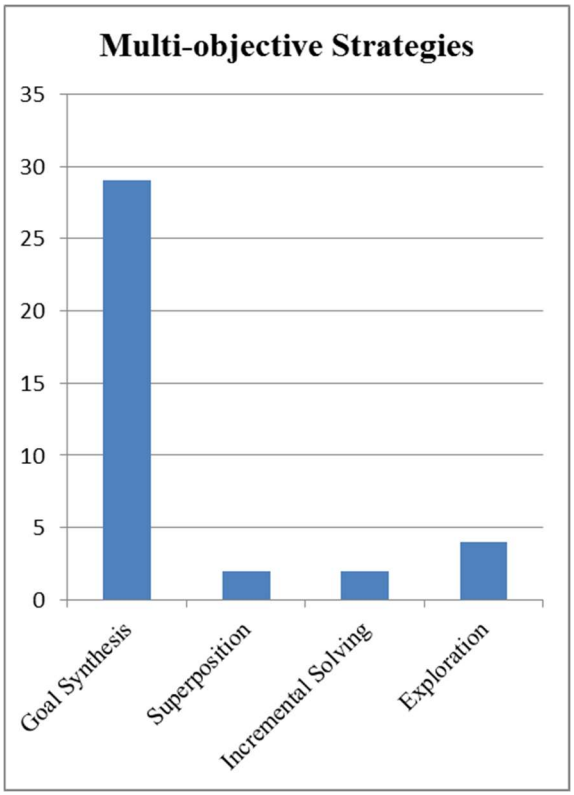

Figure 4.2 - Summary of the multi-objective strategies presented in Table 1 for solving multi-objective transportation network optimization problems.

4.2 Multi-objective Supply Chain Optimization Definition

A supply chain optimization problem may be expressed in terms of a minimization objective function, a set of variables and a set of constraints over these variables, regardless of the goal type (functions having to be maximized may be multiplied by -1). Given a vector of variables $x \in \mathbb{R}^n$ and a vector of cost coefficients $c \in \mathbb{R}^n$, a supply chain optimization problem may be defined as:

$$v^* = \min \{c^T x | Ax = b \ \wedge x \geq 0\}, \tag{4.1}$$

where $A \in \mathbb{R}^{m \times n}$ is a matrix of coefficients, $b \in \mathbb{R}^m$ is a vector of coefficients and $v^* \in \mathbb{R}^n$ is a vector of assignments for the variables x such that the value of the objective function $c^T x$ is minimum. The matrix A and the vector b define the constraints over the decision variables x and define the problem search space. Therefore, a supply chain optimization problem is defined by the tuple $lp \coloneqq (x, c, A, b, v)$. A multi-goal optimization problem is a set of tuples representing single-goal optimization problems:

$$LP(x, A, b) = \left\{(x, c, A, b, v) \middle| \exists c \in \mathbb{R}^{|x|} \wedge \exists v \in \mathbb{R}^{|x|}\right\}, \tag{4.2}$$

where the vector of variables $x \in \mathbb{R}^n$ and the set of coefficients A and b are the same for all the single-goal problems.

Let us recall that the variables x define the number of products to send on a given network route. The coefficients c usually depend on the goal and are typically information associated with a given route on the network (e.g. having to optimize for minimum transportation cost, $c_i \in c$ is the cost to send products via route i). Typically, the constraints defined by A and b are the constraints placed on production capacity and customer demand. The solution v is a distribution plan for the network.

4.3 Composite Goal Methods

Seven different means to solve the multi-objective optimization problem are now presented and described. These methods are a formalization of the four generic strategies described above in section 4.1.

Given a vector of variables $x \in \mathbb{R}^n$, a vector of coefficients $b \in \mathbb{R}^m$ and a coefficient matrix $A \in \mathbb{R}^{m \times n}$, let $S \coloneqq (x, A, b)$ be the tuple defining the problem search space. Recall from equation (4.2) that $LP(x, A, b)$ is the set of single-goal problems or the multi-goal optimization problem to be solved. The set LP is defined in section 4.2. For simplicity, whenever there is no ambiguity, LP has to be viewed as a synonym for $LP(x, A, b)$.

Let us define the projection operators on the search space S and on a given optimization problem $lp \in LP$ as $\pi_x(S) = x$, $\pi_A(S) = A$, $\pi_b(S) = b$, $\pi_c(lp) = c$ and $\pi_v(lp) = v$.

In order to improve readability, whenever there is no ambiguity, x, A and b is written instead of $\pi_x(S)$, $\pi_A(S)$ and $\pi_b(S)$ respectively. Similarly, c^j and v^j is written instead of $\pi_c(lp_j)$ and $\pi_v(lp_j)$.

The proposed methods require a function to solve the optimization problem. Here the Ant Colony Optimization algorithm described in section 3.5 above is used. The methods defined below in 4.3.1 to 4.3.6 are completely independent of this choice. The method defined in 4.3.7 is a specialization of the Ant Colony Optimization algorithm for solving multi-goal problems and it is used solely for the purpose of comparison.

Let ACS be the function representing the ant colony solver

$$\begin{aligned} ACS &: LP \to \mathbb{R}^n, \\ lp &\mapsto v, \end{aligned} \tag{4.3}$$

where $lp \in LP$ is the optimization problem to be solved and $v \in \mathbb{R}^n$ is a feasible solution to the problem lp.

In several of the methods described below, a reduction function is employed to narrow the problem search space, given a partial solution. Given an optimization problem $lp_j \in LP$, the reduction function is defined as:

$$\begin{aligned} red &: \mathbb{R}^n \to \mathbb{R}^m, \\ v^j &\mapsto b - A \cdot v^j. \end{aligned} \tag{4.4}$$

Application of the reduction function has the effect of reducing the production capacity and the customer demand by the amount of product already sent through the network.

The performance of a solution is the value of the objective function and is defined as the sum of the values of the vector v weighted by the cost coefficients c. Given an optimization problem $lp_j \in LP$, let p be the function that measures the performance of a solution:

$$p: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R},$$
$$v^j, c^j \mapsto \sum_{i=1}^{n} v_i^j \cdot c_i^j \tag{4.5}$$

4.3.1 Δ-Unification (DU)

The first method is based on the *goal synthesis* strategy. The first step consists of finding a solution for each single-goal problem. This provides an estimation of the optimal solution for each of the single-goal problem. A new optimization problem is then defined, whose goal is to minimize the difference between the current solution and the worst performing single-goal solution. Let of_{diff} be the objective function of such a problem:

$$of_{diff}(x) = \min_{lp_j \in LP} (|p(v^j, c^j) - p(x, c^j)|/p(v^j, c^j)), \tag{4.6}$$

where x are the variables of the optimization problem.

The new optimization problem is defined as:

$$lp_{diff} := \min \{of_{diff}(x) | Ax = b \ \wedge x \geq 0\}. \tag{4.7}$$

Solving the optimization problem lp_{diff} causes the solver to walk through the solution space along the intersection of the solution surfaces. Algorithm 4.1 shows the pseudo code for the above procedure.

Algorithm "Δ-Unification"

Require: Set of optimization problems LP, Problem space S

1. **for all** $lp_j \in LP$ **do** ▷ Solve each single-goal problem
2. $v^j \leftarrow ACS(lp_j)$
3. end for
4. ▷ An estimation of the optimal solution for each single-goal problem is established
5. $v^{\Delta} \leftarrow ACS(lp_{diff})$

return v^{Δ}

Algorithm 4.1 - Pseudo code for the procedure *Δ-Unification*

4.3.2 Weighted Frontier Walk (WFW)

The second method is also based on the *goal synthesis* strategy and involves the definition of a multi-goal problem whose objective function consists of a weighted combination of the single-goal problems. Let $w \in \mathbb{R}^{|LP|}$ be a vector of weights, where each weight is associated with a single-goal problem of the set LP. Let of_{wfw} be the objective function of such a problem:

$$of_{wfw}(x) = \min_{lp_j \in LP}(w_j \cdot |p(v^j, c^j) - p(x, c^j)|/p(v^j, c^j)). \tag{4.8}$$

The objective function of_{wfw} is similar to the objective function defined for the Δ-Unification method in section 4.3.1. In scenarios where the weights w are all equal, then this method is equivalent to the Δ-Unification method described in the previous section (see 4.3.1).

The new multi-goal optimization problem is defined as:

$$lp_{wfw} \coloneqq \min \{of_{wfw}(x) | Ax = b \ \wedge x \geq 0\}. \tag{4.9}$$

Algorithm 4.2 shows the pseudo code for the above procedure.

Algorithm "Weighted Frontier Walk"

Require: Set of optimization problems LP, Problem space S, Vector of weights w

1. **for all** $lp_j \in LP$ **do** ▷Solve each single-goal problem
2. $v^j \leftarrow ACS(lp_j)$
3. end for
4. ▷ An estimation of the optimal solution for each single-goal problem is established
5. $v^{wfw} \leftarrow ACS(lp_{wfw})$

return v^{wfw}

Algorithm 4.2 - Pseudo code for the procedure *Weighted Frontier Walk*

4.3.3 Iterative Superposition (IS)

The third method is based on the idea of *superposition*. A complete solution is first required for each of the single-goal problems and then the solution to the multi-goal problem is taken as combination of them. The combination is computed as the minimum intersection of the distribution plans. Let v^{int} be the result of the minimum intersection of the distribution plans. Each element $v_i^{int} \in v^{int}$ is defined as:

$$v_i^{int} = \min_{lp_j \in LP}(v_i^j). \tag{4.10}$$

The vector result of the minimum intersection is then used to reduce the problem space of each single-goal problem, by applying the reduction function defined above in equation (4.10).

It is unlikely that the first solution will satisfy all the required demands. As such the solution for the multi-goal problem is initialized as a vector of zeros of dimension $|x| = n$, $v^{mg} = \mathbf{0}_n$, and the intersection is added at each step $v^{mg} = v^{mg} + v^{int}$.

The procedure is repeated until such time as the demands are satisfied; a solution in the reduced space is computed for each single-goal problem, and the solution to the multi-goal problem is once again the minimum intersection. The pseudo code for this procedure is shown in Algorithm 4.3.

During the first step of the procedure, a complete solution is found for each single-goal problem. The solution to the multi-goal problem is then generated from the individual single-goal problem solutions. Uniformly from each single-goal solution v^j, iteratively the best elements of v^j are taken and added to the multi-goal solution v^{mg}, until such time as all demands are satisfied. It should be noted that the possibility exists such that the intersection of the solutions is empty i.e. $v^{int} = \mathbf{0}$; in such instances the reduction function will not modify the search space and the procedure itself may not converge.

Algorithm "Iterative Superposition"

Require: Set of optimization problems *LP*, Problem space *S*

1. ▷ Initialize multi-goal solution to zero
2. $v^{mg} \leftarrow \mathbf{0}_n$
3. repeat
4. **for all** $lp_j \in LP$ **do** ▷Solve each single-goal problem
5. $v^j \leftarrow ACS(lp_j)$
6. end for
7. ▷ Compute solutions intersection
8. **for all** $v_i^j \in v^j$ **do**
9. $v_i^{int} \leftarrow \min_{lp_j \in LP}(v_i^j)$
10. end for
11. ▷ Reduce the problem space
12. $b \leftarrow red(v^{int})$
13. ▷ Add the intersection to the multi-goal solution
14. $v^{mg} \leftarrow v^{mg} + v^{int}$
15. **if** $v^{int} = \mathbf{0}$ **then** ▷ The intersection is null
16. Complete v^{mg} with the best elements from $v^j \; \forall lp_j \in LP$
17. end if
18. **until** all demands are satisfied
19. **return** v^{mg}

Algorithm 4.3 - Pseudo code for the procedure *Iterative Superposition*

4.3.4 Incremental Solving via Tuning (IT)

The fourth method is based on the *incremental solving* strategy. The procedure starts by solving one of the single-goal problems and then iteratively adjusts the solution to increase its performance according to the remaining single-goal problems. The solution is adjusted by eliminating elements, the X elements that have the greatest negative impact on the current problem solution are eliminated, where $X \in \mathbb{R}$ and $X \leq |x|$. Once every single-goal problem has been considered, the problem space is reduced and the process is repeated until such time as all demands are satisfied.

Let ni be the function used to find an element in a given vector $v \in \mathbb{R}^n$ that has the greatest negative impact on the performance of a given optimization problem $lp \in LP$.

$$\begin{gathered} ni: \mathbb{R}^n \times LP \to \mathbb{N}^+, \\ v, lp \mapsto \underset{i \in [0,|v|]}{\operatorname{argmax}} p(v - \left(I_{|v|} \cdot v_i\right)_{[0,|v|],i}, \pi_c(lp)). \end{gathered} \tag{4.11}$$

The pseudo code for this procedure is shown in Algorithm 4.4.

The method requires that a single-goal optimization problem is set as the starting point and the results are dependent on the order in which single-goal problems are solved. It follows that the procedure should be run on all possible single-goal problem orderings as part of a complete analysis.

Algorithm "Incremental Solving via Tuning"

Require: Set of optimization problems LP, Problem space S, Number of elements to neglect X, Starting optimization problem $lp_j \in LP$

1. ▷ Initialize multi-goal solution to zero
2. $v^{mg} \leftarrow \mathbf{0}_n$
3. repeat
4. $\quad v^j \leftarrow ACS(lp_j)$
5. $\quad$ ▷ Adjust the solution by removing the X elements with greatest negative impact on the remaining single-goal problems
6. $\quad$ **for** $i \leftarrow 0$ **to** $X/(|LP| - 1)$ **do**
7. $\quad\quad$ **for all** $lp_k \in LP \setminus lp_j$ **do**

8. $v^{j}_{ni(v^j, lp_k)} \leftarrow 0$
9. **end for**
10. end for
11. $b \leftarrow red(v^j)$
12. $v^{mg} \leftarrow v^{mg} + v^j$
13. **until** all demands are satisfied
14. **return** v^{mg}

Algorithm 4.4 - Pseudo code for the procedure *Incremental Solving via Tuning*

4.3.5 Incremental Solving via Retention (IR)

The fifth method is based on the i*ncremental solving* strategy and in reality is a variation on the Incremental Solving via Tuning method described in the section 4.3.4. Again the procedure consists of solving each single-goal problem in sequence, but on this occasion rather than eliminating the elements with the greatest negative impact on the performance of the remaining problems, on this occasion the Y elements that have contributed the most to the performance of the current problem are retained, where $Y \in \mathbb{R}$ and $Y \leq |x|$.

Let hi be the function used to find the element of a given vector $v \in \mathbb{R}^n$ that has the greatest positive impact on the performance of the given optimization problem $lp \in LP$.

$$\begin{aligned} hi &: \mathbb{R}^n \times LP \rightarrow \mathbb{N}^+, \\ v, lp &\mapsto \underset{i \in [0, |v|]}{\operatorname{argmax}} (v_i \cdot (\pi_c(lp))_i). \end{aligned} \tag{4.12}$$

The pseudo code for this procedure is shown in Algorithm 4.5.

Algorithm "Incremental Solving via Retention"

Require: Set of optimization problems LP, Problem space S, Number of elements to retain Y

1. ▷ Initialize multi-goal solution to zero
2. $v^{mg} \leftarrow \mathbf{0}_n$
3. ▷ Initialize counter of remaining elements that may be retained
4. $r \leftarrow n$

5. repeat
6. **for all** $lp_j \in LP$ **do** ▷ Sequentially solve for each single-goal problem
7. $v^j \leftarrow ACS(lp_j)$
8. **if** $r \leq Y$ **then** ▷ All elements have to be retained
9. return $v^{mg} + v^j$
10. end if
11. ▷ Adjust the solution by retaining the Y elements with greatest positive impact on the current single-goal problem
12. $v^k \leftarrow \mathbf{0}_n$
13. **for** $i \leftarrow 0$ **to** Y **do**
14. $v^k_{hi(v^j - v^k, lp_j)} \leftarrow v^j_{hi(v^j - v^k, lp_j)}$
15. **end for**
16. $r \leftarrow r - Y$
17. $b \leftarrow red(v^k)$
18. $v^{mg} \leftarrow v^{mg} + v^k$
19. end for
20. **until** all demands are satisfied
21. **return** v^{mg}

Algorithm 4.5 - Pseudo code for the procedure *Incremental Solving via Retention*

4.3.6 Taguchi QLF-based Approach

The idea behind the sixth method originates from the theory of *Robust Engineering* and *Taguchi's Quality Loss Function (QLF)* [123] and can be classified as a *goal synthesis* based strategy. Taguchi's quality loss function encodes a penalty term for deviations from a particular target. Here let *L* be a loss function for each single-goal problem $lp \in LP$, in the form of:

$$\begin{aligned} L_{lp}&: \mathbb{R} \to \mathbb{R}, \\ t' &\mapsto k\,(t' - t)^2, \end{aligned} \tag{4.13}$$

where $t \in \mathbb{R}$ is the value of the target solution for the problem lp, $t' \in \mathbb{R}$ is the evaluation for another proposed design and $k \in R$ is the loss coefficient in terms of deviation from the target metric. The function computes the penalty, the loss for deviating from the target. Given an optimization problem $lp \in LP$, the value of t is an estimation of the optimum solution to the problem, which may be computed by applying the Ant Colony Solver ACS:

$$t = p\big(ACS(lp), \pi_c(lp)\big). \tag{4.14}$$

It results a new multi-goal optimization problem based on the loss function L. Let of_{qlf} be the objective function of such a problem:

$$of_{qlf}(x) = \sum_{lp_j \in LP} L_{lp_j}(p(x, \pi_c(lp_j))). \tag{4.15}$$

The new multi-goal problem consists of minimizing the total deviation loss from the best known solutions of the single-goal problems:

$$lp_{qlf} \coloneqq \min \{of_{qlf}(x) \big| Ax = b \ \wedge x \geq 0\}. \tag{4.16}$$

The pseudo code for this procedure is shown in Algorithm 4.6.

Algorithm "Taguchi QLF-based Approach"

Require: Set of optimization problems LP, Problem space S

1. **for all** $lp_j \in LP$ **do** ▷Solve each single-goal problem
2. $v^j \leftarrow ACS(lp_j)$
3. end for
4. ▷ An estimation of the optimal solution for each single-goal problem is established
5. $v^{qlf} \leftarrow ACS(lp_{qlf})$
6. **return** v^{qlf}

Algorithm 4.6 - Pseudo code for the procedure *Taguchi QLF-based Approach*

4.3.7 ACO-specific Multi-Goal Method

The last and final method differs from those previously described in that it is specific to the Ant Colony Optimization algorithm; the main idea here is to improve the global

pheromone updating strategy in order to simultaneously consider more than one goal. Given a solution generated by the solver, the level of pheromone deposited is increased in accordance with the performance improvement in each single-goal problem. A generated solution receives a full pheromone update if and only if it is an improved solution for each of the single-goal problems. The method may be classified as belonging to the *goal synthesis* category, despite the fact that it does not formally involve the definition of a multi-goal problem. This classification as *goal synthesis* is justified on the basis that the method employs exploration of the search space by taking into account more than one goal at a time as it walks along the intersection of the single-goal problem spaces.

As stated in Dorigo et al. [78], the original global pheromone rule is defined as follow:

$$\tau(r,s) = (1-\rho)\cdot\tau(r,s) + \rho\cdot\Delta\tau(r,s), \tag{4.17}$$

where (r,s) is an edge of the ant tour or a route in the network, $\tau(r,s)$ is the pheromone value deposited on the edge, ρ is the decay parameter $\rho \in [0,1]$, and $\Delta\tau(r,s)$ is a measure of the improvement in the solution.

Given a solution $v \in R^n$ for the problem $lp \in LP$, let δ be the function to measure the increase applied to the pheromone level:

$$\delta: \mathbb{R}^n \to \mathbb{R},$$
$$v \mapsto \sum_{lp\in LP} \frac{1}{|LP|}\cdot\left\lfloor\frac{p(v,\pi_c(lp))}{p(v^*,\pi_c(lp))}\right\rfloor, \tag{4.18}$$

where $v^* \in \mathbb{R}^n$ is the best known solution for the problem lp. The improved pheromone update strategy maybe stated as:

$$\tau(r,s) = \big((1-\rho)\cdot\tau(r,s) + \rho\cdot\Delta\tau(r,s)\big)\cdot\delta(v). \tag{4.19}$$

Let ACS_{mo} be the variant of the Ant Colony Optimization algorithm based on such a global pheromone update strategy.

The pseudo code for this procedure is shown in Algorithm 4.7.

The procedure requires selecting a single-goal problem $lp_k \in LP$ to be used by the solver as the main problem to solve. The advantage of using a single-goal problem

instead of defining a multi-goal one is that the procedure should find a feasible solution for lp_k even if the intersection of the single-goal problem search spaces is empty. Although it is not unreasonable to expect that the solution will be strongly influenced by the goal of the problem lp_k.

Algorithm "ACO-specific Multi-Goal Method"

Require: Set of optimization problems LP, Problem space S, Single-goal problem used by the AC_{mo} variant $lp_k \in LP$

1. **for all** $lp_j \in LP$ **do** ▷Solve each single-goal problem
2. $v^j \leftarrow ACS(lp_j)$
3. end for
4. ▷ An estimation of the optimal solution for each single-goal problem is established
5. $v^{mo} \leftarrow ACS_{mo}(lp_k)$
6. **return** v^{mo}

Algorithm 4.7 - Pseudo code for the procedure *ACO-specific Multi-Goal Method*

4.4 Experiments

As in section 3.6, each of the proposed methods has been tested on a set of 4 single-goal optimization problems: for maximum profit, for minimum transportation cost, for minimum transportation time, and for maximum network resilience. The dataset use for the experiments is the one provided by Caterpillar, published in [4].

The single-goal problems have first been solved to define a baseline against which the performance of the proposed composite goal methods can be compared. The solutions produced by the methods have been evaluated according to the single-goal objectives. Table 4.2 shows the percent difference between the performance of the single-goal problems and the performance of the combination methods. The incremental methods (section 4.3.4 and 4.3.5) have been run on all the possible orders of the single-goal problems and the method Weighted Frontier Walk (section 4.3.2) has been run on a set of 16 weight combinations. The combinations of weights have been generated according to a Monte Carlo Sampling strategy.

As alluded to before, the method used to solve the optimization problems is the Ant Colony System algorithm as defined in section Ant Colony Optimization for Caterpillar's Supply Chain. The parameters used for the test cases are reported in Table 3.3. Table 4.2 shows the runtime results for the experiments.

Table 4.2 - Percentage difference between single-goal problems and composite goal methods performances. WFW a-b-c-d stands for Weighted Frontier Walk and a, b, c, and d are the percentage weights assigned to the single goals. IT g1-g2-g3-g4 and IR g1-g2-g3-g4 stand for Incremental Solving via Tuning and Incremental Solving via Retention respectively and g1-g2-g3-g4 defines the order used to solve the single goal problems. P stands for maximum profit, T stands for minimum transit time, R stands for highest resilience, and C stands for minimum transportation cost. The combinations of weights for the WFW have been generated according to a Monte Carlo Sampling strategy.

	Profit (%)	Time (%)	Cost (%)	Resilience (%)	Running time (s)
WFW 52-87-62-35	1.56%	156.62%	1.57%	4.89%	1008
WFW 1-14-36-84	1.62%	173.94%	1.64%	5.05%	1022
WFW 74-92-43-81	2.08%	191.25%	2.17%	5.05%	1022
WFW 94-97-33-25	1.65%	159.74%	1.68%	5.22%	1008
WFW 0-92-11-50	1.49%	165.82%	1.49%	5.24%	1022
WFW 63-28-60-2	1.45%	166.37%	1.45%	5.37%	1022
WFW 2-28-32-7	1.81%	180.15%	1.86%	4.42%	1022
WFW 54-88-21-57	1.59%	159.68%	1.60%	5.06%	1008
WFW 59-54-86-77	2.08%	149.90%	2.16%	4.09%	1022
WFW 92-5-11-64	1.48%	164.61%	1.48%	5.20%	1022
WFW 69-85-60-0	1.45%	156.10%	1.44%	5.20%	1022
WFW 22-94-90-56	1.90%	175.05%	1.96%	4.70%	1022
WFW 29-50-31-2	1.52%	175.51%	1.53%	5.11%	1022
WFW 57-40-75-28	1.48%	177.30%	1.48%	4.83%	1008
WFW 1-31-7-75	1.89%	164.00%	1.95%	4.56%	1022
WFW 100-100-100-100	1.46%	160.06%	1.46%	5.22%	1022

Taguchi QLF	5.21%	193.63%	100.98%	4.24%	3234
IS	37.34%	1.92%	42.28%	1.86%	616
IT T-P-R-C	36.99%	1.56%	41.88%	1.67%	924
IT C-R-P-T	1.01%	128.43%	1.05%	5.83%	910
IT C-R-T-P	1.01%	128.43%	1.05%	5.83%	910
IT C-P-R-T	1.01%	128.22%	1.05%	5.83%	910
IT C-P-T-R	1.01%	128.22%	1.05%	5.83%	924
IT C-T-R-P	1.01%	128.43%	1.05%	5.83%	910
IT C-T-P-R	1.01%	128.22%	1.05%	5.83%	308
IT R-C-P-T	49.16%	353.93%	55.72%	2.36%	308
IT R-C-T-P	49.00%	363.48%	55.53%	2.51%	322
IT R-P-C-T	51.56%	389.74%	58.45%	3.25%	308
IT R-P-T-C	49.38%	359.98%	55.97%	2.90%	308
IT R-T-C-P	50.33%	383.10%	57.06%	2.74%	308
IT R-T-P-C	49.12%	364.35%	55.67%	2.65%	910
IT P-C-R-T	1.04%	117.83%	1.11%	5.89%	924
IT P-C-T-R	1.04%	117.83%	1.11%	5.89%	910
IT P-R-C-T	1.04%	117.83%	1.11%	5.89%	910
IT P-R-T-C	1.03%	118.73%	1.10%	5.39%	910
IT P-T-C-R	1.03%	118.73%	1.10%	5.39%	924
IT P-T-R-C	1.03%	118.73%	1.10%	5.39%	616
IT T-C-R-P	36.70%	1.04%	41.55%	1.24%	630
IT T-C-P-R	36.70%	1.04%	41.55%	1.24%	616
IT T-R-C-P	36.70%	1.04%	41.55%	1.24%	630
IT T-R-P-C	36.99%	1.56%	41.88%	1.67%	616
IT T-P-C-R	36.99%	1.56%	41.88%	1.67%	5684
IR T-P-R-C	24.86%	88.69%	28.08%	1.71%	6356
IR C-R-P-T	9.85%	154.35%	11.00%	3.48%	6076
IR C-R-T-P	5.13%	161.40%	5.63%	2.53%	6398
IR C-P-R-T	7.44%	158.35%	8.26%	2.68%	6468
IR C-P-T-R	6.81%	146.43%	7.54%	2.89%	6300
IR C-T-R-P	10.01%	158.84%	11.19%	1.51%	6412

IR C-T-P-R	6.64%	146.32%	7.36%	2.80%	5236
IR R-C-P-T	14.73%	151.08%	16.55%	3.79%	5138
IR R-C-T-P	16.43%	150.41%	18.49%	3.04%	5208
IR R-P-C-T	14.79%	145.78%	16.63%	3.16%	5152
IR R-P-T-C	14.20%	135.92%	15.95%	2.76%	5096
IR R-T-C-P	15.26%	145.97%	17.16%	3.27%	5068
IR R-T-P-C	13.94%	157.32%	15.65%	2.87%	5544
IR P-C-R-T	2.05%	120.04%	2.13%	4.83%	5782
IR P-C-T-R	2.00%	119.60%	2.08%	4.75%	5418
IR P-R-C-T	2.36%	122.07%	2.48%	4.92%	5362
IR P-R-T-C	2.38%	119.70%	2.51%	4.73%	5544
IR P-T-C-R	1.89%	118.72%	1.94%	4.79%	5446
IR P-T-R-C	2.31%	118.91%	2.42%	4.52%	5754
IR T-C-R-P	26.60%	78.76%	30.06%	1.20%	5838
IR T-C-P-R	24.78%	65.05%	27.99%	1.15%	5558
IR T-R-C-P	26.24%	85.80%	29.64%	1.47%	5474
IR T-R-P-C	24.64%	88.11%	27.83%	1.48%	5810
IR T-P-C-R	24.47%	89.50%	27.63%	1.14%	6358
ACO-Specific	3.01%	183.73%	3.22%	5.70%	3234
DU	1.30%	167.10%	1.28%	5.78%	840

When comparing optimization methods for multi-goal problems, it is usually difficult to rank one approach over another in absolute terms. Ideally, the best method should produce a solution with the same performance as those produced when optimizing for each single goal, but in practice this is difficult to achieve. The results not only depend on the definition of the multi-goal method, but also on the properties of the single-goal problems. For example, problems might conflict or be mutually exclusive.

In the case of supply chain optimization, one common denominator could be profit: most of the metrics such as transportation time and network resilience can be monetized. However, in real business environments profit alone may not always be the dominant factor, distribution plans that yield lower profit, but offer greater value with

respect to other metrics may be preferred. For instance, resilience implies risk, some companies are more averse to risk taking than the impact on profit alone would imply. Total inventory carrying costs equate to cash flow and/or funds tied up in the business that cannot otherwise be invested elsewhere; when trading volumes are low cash flow may become more important than pure profit.

However, one possible evaluation scenario would be to calculate the relative performance of methods by ranking each method by goal and then combine the ranked position of a method on each goal by summing its position on the different goals. Table 4.3 shows the result of the ranking procedure.

Table 4.3 - Composite goal methods ranking. The methods have been ranked base on their relative performance. Lower number means a higher position in the rank, hence higher performance. In order to improve visualization, the cells have been color coded in a scale from red to green. Higher performing methods are color-coded green, lower performing ones are color-coded red.

	Profit	Time	Cost	Resilience	Sum	Rank of sum
IS	65	8	64	16	153	51
IT T-P-R-C	61	5	60	12	138	31
IT C-R-P-T	2	31	5	64	102	8
IT C-R-T-P	2	31	5	64	102	8
IT C-P-R-T	5	28	2	64	99	5
IT C-P-T-R	5	28	2	64	99	5
IT C-T-R-P	2	31	5	64	102	8
IT C-T-P-R	5	28	2	64	99	5
IT R-C-P-T	68	66	67	17	218	68
IT R-C-T-P	66	68	65	18	217	67
IT R-P-C-T	72	72	71	30	245	72
IT R-P-T-C	69	67	68	27	231	70
IT R-T-C-P	71	71	70	22	234	71
IT R-T-P-C	67	69	66	20	222	69
IT P-C-R-T	11	15	12	70	108	11
IT P-C-T-R	11	15	12	70	108	11

IT P-R-C-T	11	15	12	70	108	11
IT P-R-T-C	8	19	9	57	93	1
IT P-T-C-R	8	19	9	57	93	1
IT P-T-R-C	8	19	9	57	93	1
IT T-C-R-P	58	2	57	6	123	18
IT T-C-P-R	58	2	57	6	123	18
IT T-R-C-P	58	2	57	6	123	18
IT T-R-P-C	61	5	60	12	138	31
IT T-P-C-R	61	5	60	12	138	31
IR T-P-R-C	55	13	54	15	137	30
IR C-R-P-T	44	43	43	32	162	57
IR C-R-T-P	39	52	39	19	149	45
IR C-P-R-T	43	47	42	21	153	51
IR C-P-T-R	42	39	41	26	148	43
IR C-T-R-P	45	48	44	11	148	43
IR C-T-P-R	41	38	40	24	143	37
IR R-C-P-T	48	42	47	33	170	61
IR R-C-T-P	51	41	50	28	170	61
IR R-P-C-T	49	36	48	29	162	57
IR R-P-T-C	47	34	46	23	150	47
IR R-T-C-P	50	37	49	31	167	60
IR R-T-P-C	46	46	45	25	162	57
IR P-C-R-T	32	25	32	43	132	28
IR P-C-T-R	31	23	31	41	126	21
IR P-R-C-T	36	26	36	46	144	39
IR P-R-T-C	37	24	37	40	138	31
IR P-T-C-R	28	18	28	42	116	15
IR P-T-R-C	35	22	35	37	129	25
IR T-C-R-P	57	10	56	5	128	24
IR T-C-P-R	54	9	53	4	120	16
IR T-R-C-P	56	11	55	9	131	27

IR T-R-P-C	53	12	52	10	127	22
IR T-P-C-R	52	14	51	3	120	16
ACO-Specific	38	63	38	62	201	64
DU	15	57	15	63	150	47
WFW 52-87-62-35	23	45	23	45	136	29
WFW 1-14-36-84	25	58	25	47	155	54
WFW 74-92-43-81	34	64	34	48	180	63
WFW 94-97-33-25	26	50	26	54	156	55
WFW 0-92-11-50	21	55	21	55	152	49
WFW 63-28-60-2	17	56	17	56	146	40
WFW 2-28-32-7	27	62	27	36	152	49
WFW 54-88-21-57	24	49	24	49	146	40
WFW 59-54-86-77	33	40	33	34	140	35
WFW 92-5-11-64	20	54	20	52	146	40
WFW 69-85-60-0	16	44	16	51	127	22
WFW 22-94-90-56	30	59	30	39	158	56
WFW 29-50-31-2	22	60	22	50	154	53
WFW 57-40-75-28	19	61	19	44	143	37
WFW 1-31-7-75	29	53	29	38	149	45
WFW 100-100-100-100	18	51	18	53	140	35
Taguchi QLF	40	65	72	35	212	66

From Table 4.3, it is possible to observe that methods based on the incremental solving strategy (i.e. Incremental Solving by Tuning 4.3.4 and Incremental Solving by Retention 4.3.5) are positioned in the top ranking. While a ranking approach is appropriate in creating a discrete ordering, it does not necessarily convey information about the relative relationships between the experiments/goals.

Figure 4.3 shows an alternative visual representation of the data from Table 4.2. Figure 4.3 shows that there is no consistent difference between the methods that belong to the same generic strategy (as defined in section 4.1). For instance, methods based on goal synthesis, such as *Weighted Frontier Walk*, *Delta-Unification*, *Taguchi QLF-based* and

ACO-specific all produce distribution plans with similar performances on the single-goal problems (i.e. very low difference for profit, slightly higher for resilience and higher still for time). Similarly for methods based on incremental solving strategy, such as *Incremental Solving by Tuning* and *Incremental Solving by Retention* have produce distribution plans whose performance is dependent on the order in which the single-goal problems have been solved. In that the first single-goal problem to be solved has greatest influence on the overall solution. For example, by taking the *Incremental Solving by Tuning* method and running it in the order Resilience-Time-Profit-Cost produces a distribution plan that is not dissimilar to the performance produced for the order Resilience-Profit-Time-Cost, but is completely different from that produced for the order Profit-Time-Resilience-Cost.

Arguably the *Iterative Superposition* method is the best one. The distribution plan found performs well for each single-goal problem (where all the percent differences are below 50%) and the gap that exists between single-goal performances is not as large in comparison to the other methods. On occasions, the Incremental Solving by Tuning and Incremental Solving by Retention methods produce similar results, but they exhibit the drawback of having a dependency on the order used to solve the single-goal problems and on the number of elements to be eliminated or retained.

The methods described in this chapter have also been tested on the two randomly generated problems in [5] and [6].

Figure 4.4 and Figure 4.5 show the percent difference between the performance of the single-goal problems and the performance of the combination methods for the two randomly generated problems, [5] and [6] respectively.

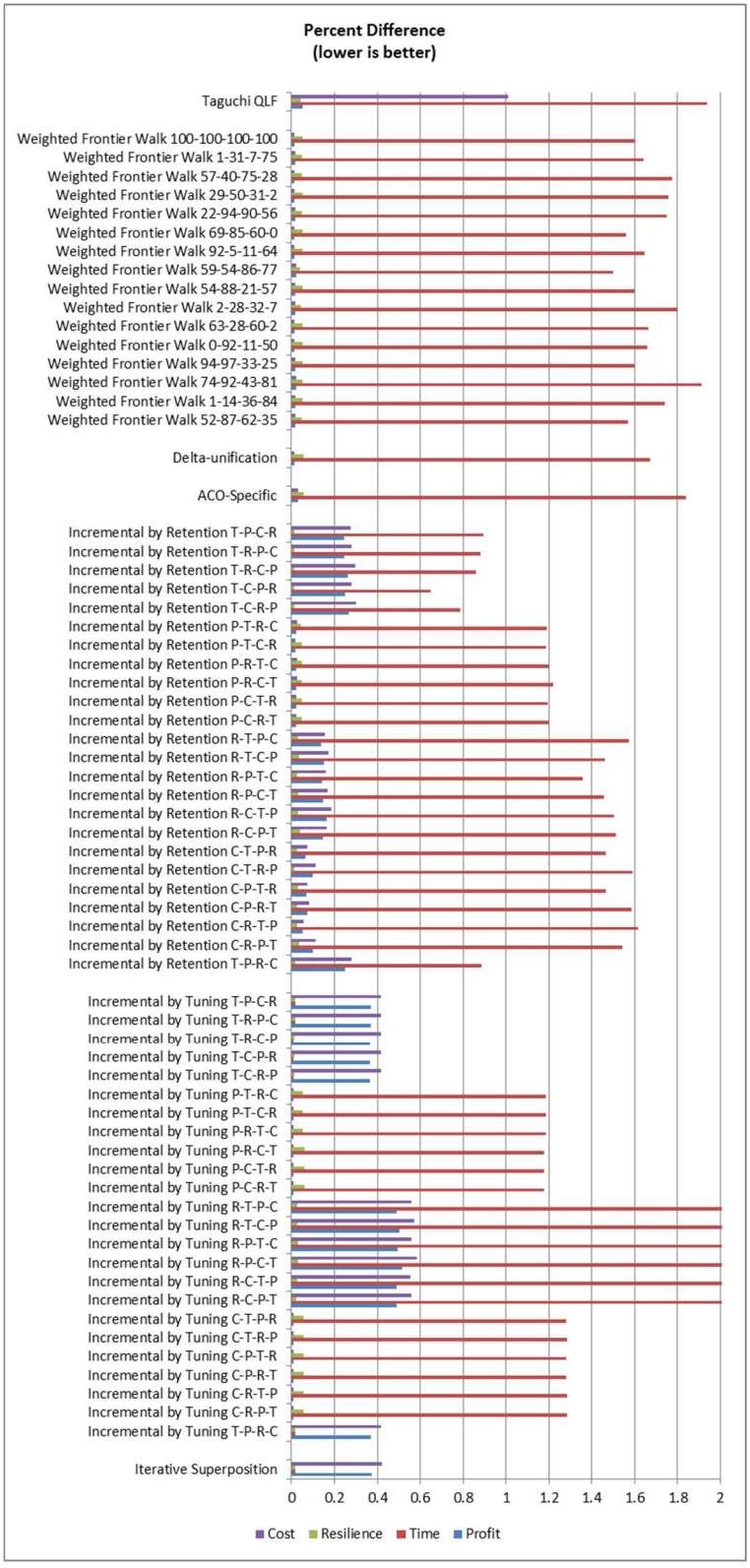

Figure 4.3 - Visual representation of percentage difference between single-goal problems and composite goal methods performances for the original problem in [4].

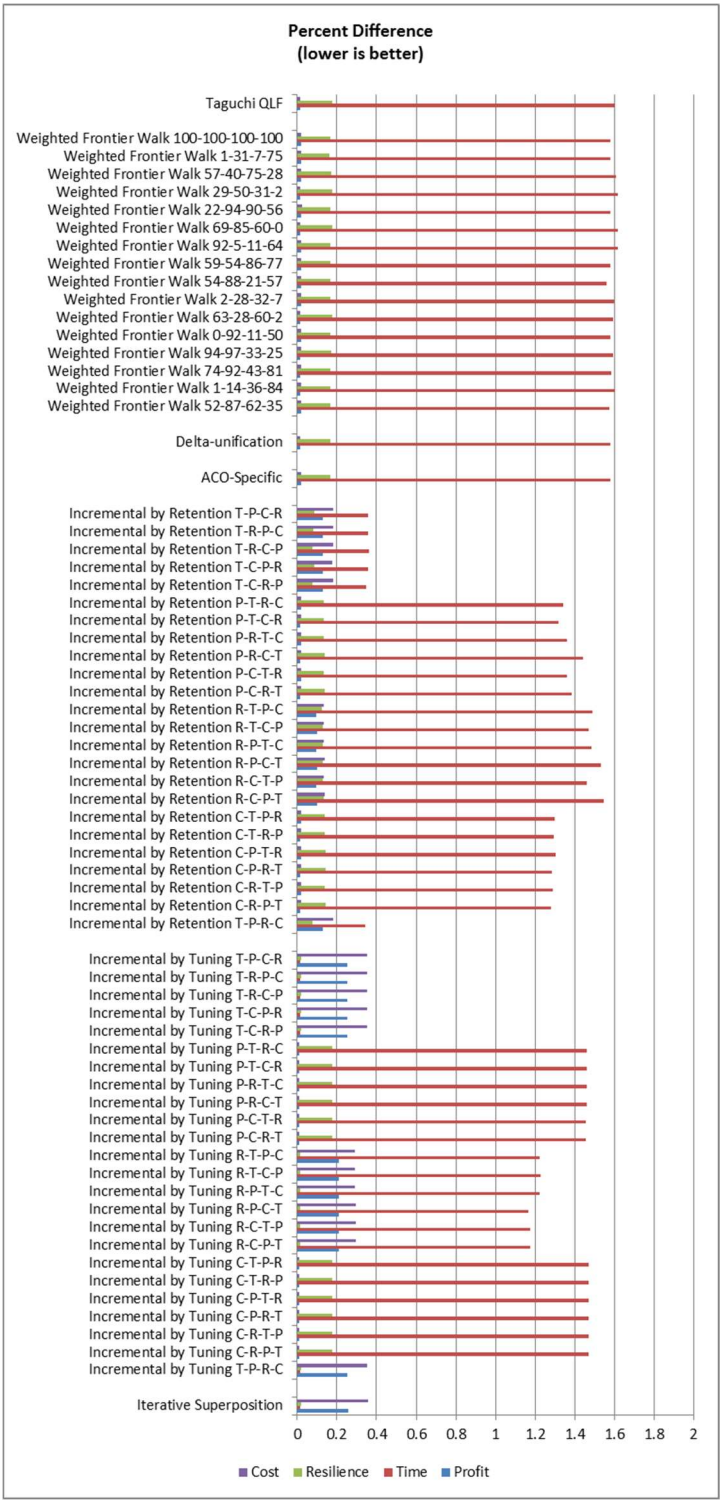

Figure 4.4 - Visual representation of percentage difference between single-goal problems and composite goal methods performances. The problem is randomly generated according to a normal distribution with mean and standard deviation as in the original dataset. The dataset is in [5].

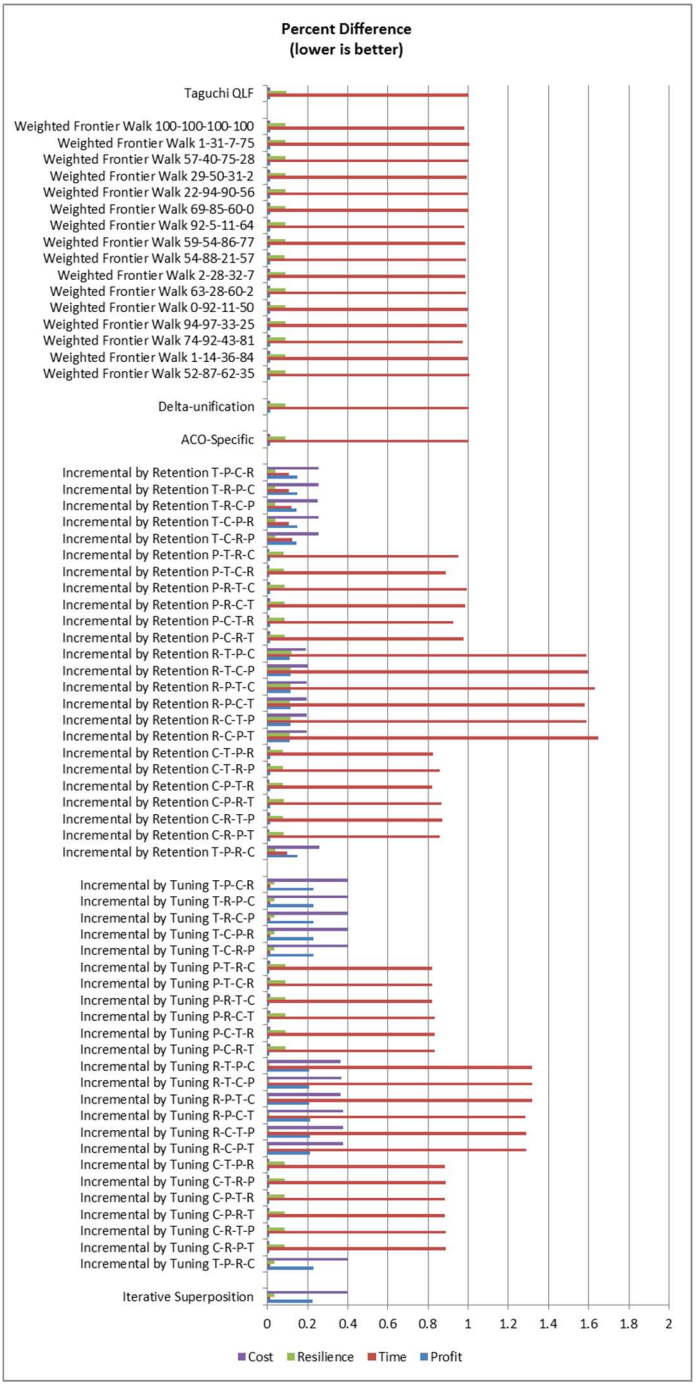

Figure 4.5 - Visual representation of percentage difference between single-goal problems and composite goal methods performances. The problem is randomly generated where the figures are an interval between 0 and an upper limit which is a random increase over the maximum value in the original data, according to a negative exponential distribution. The dataset is in [6].

The Iterative Superposition method has again proved to have the best performance for both randomly generated problems.

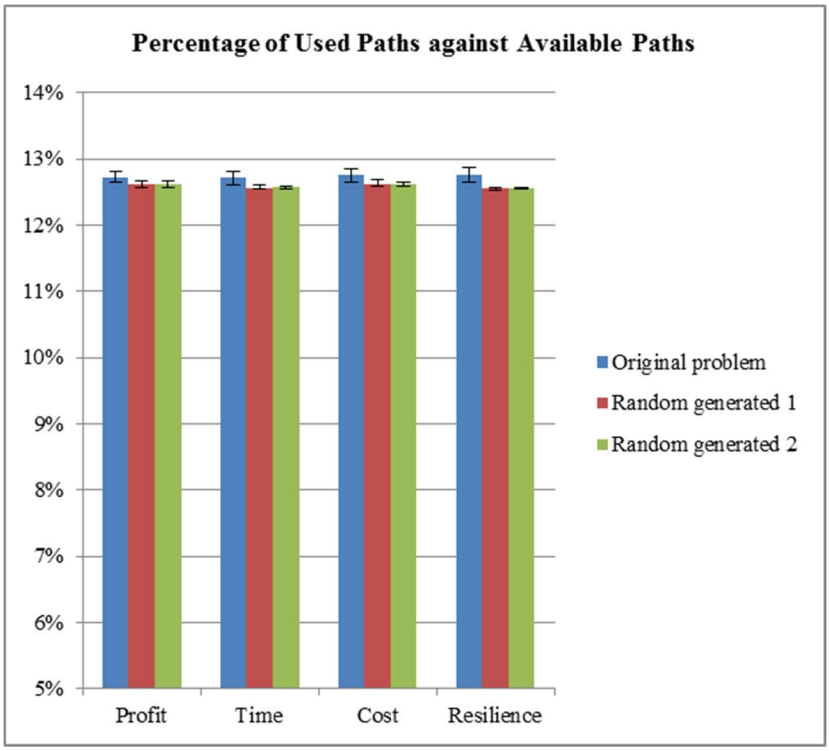

Figure 4.6 - Mean percentage of used paths against all available paths for all data sets and for all single-goal problems. The data are averaged through a period of 12 months, which each month presents a different dealer demand.

The network used for testing presents a very large number of possible paths. However, Caterpillar's business is characterized by having a relative low monthly demand for any given type of product. On average, over a period of 12 months, 12% of all possible routes are used each month to satisfy the demand (see Figure 4.6). It is reasonable to expect that due to the large number of possible routes, the problem of maximizing the network resilience is the easiest to solve. The demand may be spread evenly on the network. Moreover, solution for the problems of profit maximization and transportation costs minimization may be expected to overlap as they involve similar economic aspects. This may not be the case for the problem of travelled time minimization. It is possible that even the most expensive production source may be chosen to satisfy a portion of the demand, provided the production facility is closely located in proximity to the dealers. Since ocean lane discounts for lanes with high shipping commitments are

considered, the most expensive production source is unlikely to be considered for both the profit maximization and costs minimization goals. The solver would choose an inexpensive production facility, even if its location is not the closet to the dealership, thereby increasing transportation cost.

The results from running the multi-goal optimization methods on the three datasets confirm these expectations. All solutions had low differences with respect to resilience, for profit maximization and cost minimization the differences were similar. The quality of the solution in term of time minimization is inversely proportional to the goals for profit maximization and cost minimization.

As previously discussed, the Iterative Superposition strategy was shown to be one of the best methods to solve a multi-objective optimization problem. Not only such strategy was capable of finding high quality solutions for all four goals simultaneously, but also it is the only strategy that is independent from the initial configuration or additional parameters. One of the main differences between the four multi-goal strategies relates to the information used to find a combined solution. The goal synthesis strategy only uses the heuristic information from the set of single-goal problems; the solution is a combination of this heuristic information. In this context, the heuristic information refers to the information held on the routes in the network which guides the solver in building the distribution plan. For instance, if the goal is to maximize the profit arising from a distribution plan, then the heuristic information is most likely to be the transportation cost for the given routes. Figure 4.7 and Figure 4.8 show the heuristic information matrix associated with the distribution network for the problems of cost minimization and travelled time minimization respectively. Figure 4.7 and Figure 4.8 only report the heuristic information for one month and the routes represented are only the ones connecting active production sources and dealers. Production sources and dealers are said to be active if they have positive, non-zero capacity and demand for the given month.

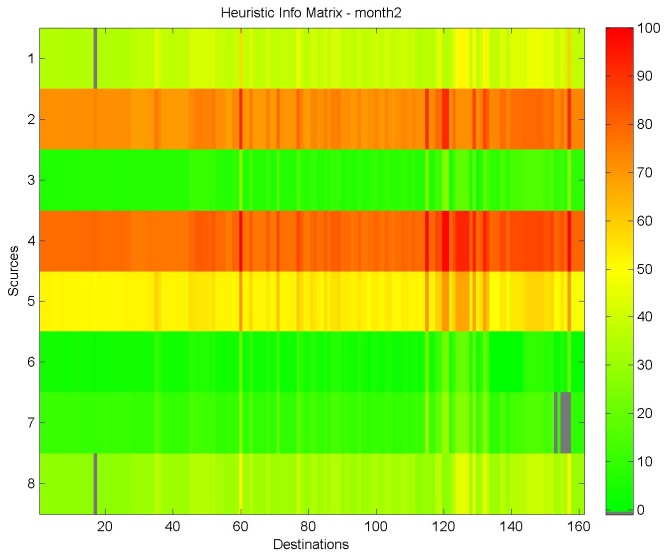

Figure 4.7 - Heuristic information matrix for the problem of minimization of transportation cost for all possible routes from sources to destinations. The color scale goes from green as most profitable route to red as least profitable route. Gray routes are non-connected routes. Only active routes for the given month are displayed. A route is said active if the production sources and dealer has positive non-zero capacity and demand respectively.

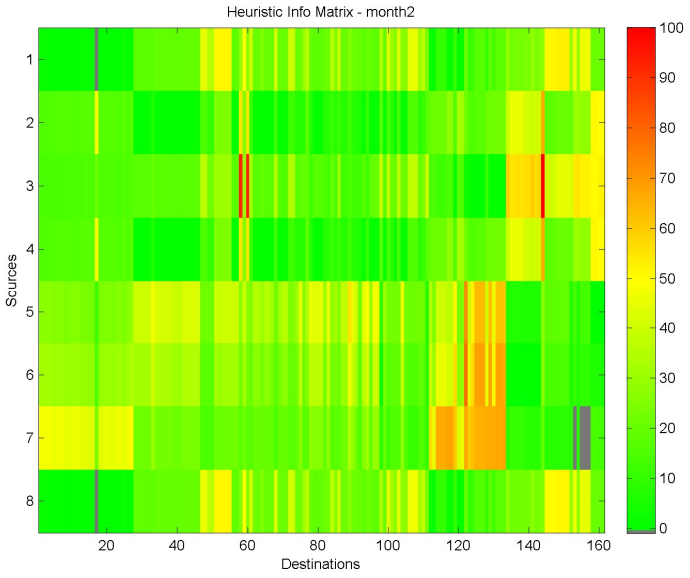

Figure 4.8 - Heuristic information matrix for the problem of minimization of travelled time for all possible routes from sources to destinations. The color scale

goes from green as most profitable route to red as least profitable route. Gray routes are non-connected routes. Only active routes for the given month are displayed. A route is said active if the production sources and dealer has positive non-zero capacity and demand respectively.

The incremental strategy makes use of heuristic information combined with a partial distribution plan that is optimal for one of the single-goal problems. Figure 4.9 and Figure 4.10 shows a visualization of the distribution plan for the problems of cost minimization and travelled time minimization respectively (as for Figure 4.7 and Figure 4.8, only active routes are shown). Recall that the incremental strategy consists of building a solution for one objective, retaining or removing part of it, and then solving the remaining part in accordance with the next objective.

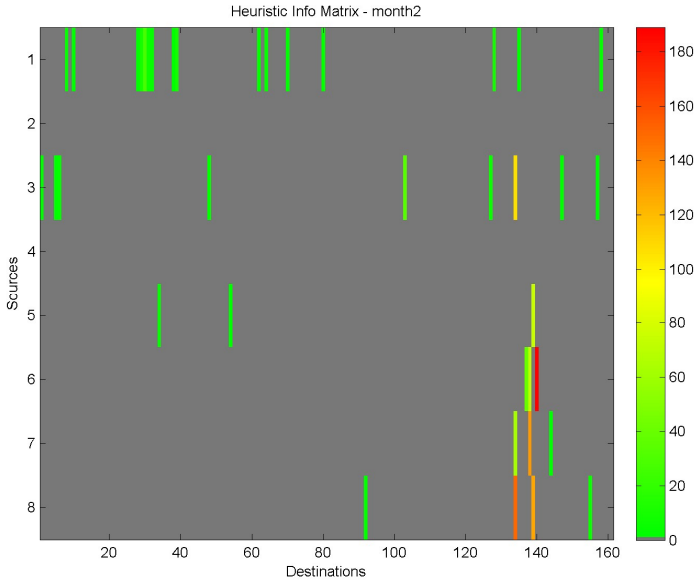

Figure 4.9 - Distribution plan for the problem of transportation cost minimization. The color scale goes from green as route with only one machine sent through, to red for highly used routes. Gray routes are not used. Only active routes for the given month are displayed. A route is said active if the production sources and dealer has positive non-zero capacity and demand respectively.

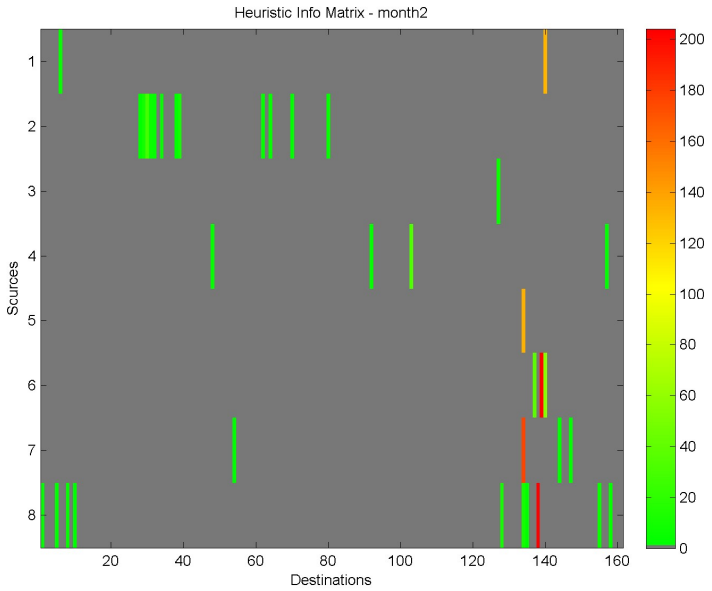

Figure 4.10 - Distribution plan for the problem of travelled time minimization. The color scale goes from green as route with only one machine sent through, to red for highly used routes. Gray routes are not used. Only active routes for the given month are displayed. A route is said active if the production sources and dealer has positive non-zero capacity and demand respectively.

Whereas the superposition strategy makes only use of the optimal solution for each of the single-goal problems, the solution consists of finding the best solution for each single-goal problem separately, and then using the optimal distribution plans to build a solution for the multi-goal problem. Having the optimal solutions, then the complete heuristic information matrices may appear to be 'noisy' and it stands to reason that combining such solutions may for the majority of cases be the better approach. From a search space perspective, a strategy that works with optimal solutions as its inputs can be expected to produce a multi-goal solution that is closer to all single-goal ones. When starting from the intersection of the heuristic information matrices there is no guaranties that the solution which is the closest to all the single-goal ones will be the one resulting from the intersection of the search spaces and could be significantly different from the search spaces of the single-goal problems.

Future work may consist of a theoretical and formal analysis of the different characteristics and behaviors of the discussed strategies to confirm the previous hypothesis.

4.5 Discussions

The motivation and rational for undertaking this work was to highlight and provide a better understanding of the body of work in the literature relating to multi-goal analysis of transportation network optimization. Clearly from the review undertaken it was evident that more work could be done to enhance knowledge and foster understanding in certain areas of the topic.

Firstly, the literature review of work relating to multi-goal optimization could be extended. While the current problem focuses on supply chain optimization, it is apparent that such work is applicable to and could be extended into others areas of operational research. This would provide an opportunity to better understand which multi-goal optimization methods are preferred and why. It is possible that the current bias towards one specific method for multi-goal optimization is the result of existing software availability. If the current generations of optimization tools do not provide implementation that address multi-goal strategies, practically it is advantageous to adopt the goal synthesis approach and define a multi-goal model, which may be the input to the optimization tool. As future work, the literature could be extended to include information relating to optimization tool implementations and their capabilities.

Moreover, the methods presented here should be tested using a different optimization algorithm and on a differing dataset. While the Ant Colony System (ACS) is a very well established and accepted optimization algorithm, there remains the possibility that the results could be biased by undefined behaviors particular to the ACS algorithm. While testing the hypothesis on two randomly generated problems strengthens the result, it would nevertheless be interesting to test the outcomes on independent datasets, or even a different problem.

Finally, a theoretical analysis of the different characteristics and behaviors that pertain to the discussed strategies is needed to better understand the reasons why and under what circumstances some strategies consistently perform better than others.

4.6 Summary

The aim of this work was to identify the most promising multi-objective optimization techniques available for solving real-world 'industrial' supply chain optimization problems. The state of art for multi-objective optimization have been reviewed and four generic strategies have been identified, which are referred to as *goal synthesis*, *superposition*, *incremental solving* and *exploration*. Seven instances of these four strategies have been implemented. The preferred approach from analysis and review of the current literature would appear to be the construction of a model that combines single optimization goals. However, the experiment using goal combination methods produced low quality solutions in comparison to those produced by other strategies. In particular, the *superposition* strategy proved to be the most promising solution found, performing well across all single-goal problems and having the additional advantage that it is not dependent on the solution ordering or on the weightings assigned to individual single objectives.

The work presented here has aided in the development of a more accurate optimization model for the business of the industrial partners and has helped in the identification of optimization methods that are capable of producing high quality distribution plans. This work will serve as a reference on multi-objective methods for real-world 'industrial' supply chain optimization problems.

5 Improving Ant Colony Optimization Performance through Prediction of Best Termination Condition

This chapter is based on the results published in [3].

In the last decade or so, meta-heuristics approaches have been successfully applied to many NP-hard optimization problems. These approaches are popular due to their adaptability and application potential across differing problem domains. They are known for their ability to find high-quality solutions to the most complex of combinatorial optimization problems. Equally, they are also known for their high computational complexity. As discussed in section 2.2.5, the successful application of meta-heuristics is also paramount to the optimal tuning of their many parameters. The performance of such parametrized algorithms depends strongly on the particular values of the parameters, and the appropriate setting of these parameters is itself a difficult

optimization problem [90]. The termination condition is one of the key parameter of the meta-heuristic approaches, a parameter which highly influences the quality of the solution found and the computational requirements. This work addresses the task of improving the runtime complexity of the Ant Colony System (ACS) when applied to a real-world supply chain optimization problem. The improvement is achieved by adopting a machine learning approach. Given an unseen problem instance, the best termination point for the optimization process is predicted by analyzing its behavior on previously solved instances which are the most similar to the current one. Figure 5.1 shows a graphical representation of the proposed system.

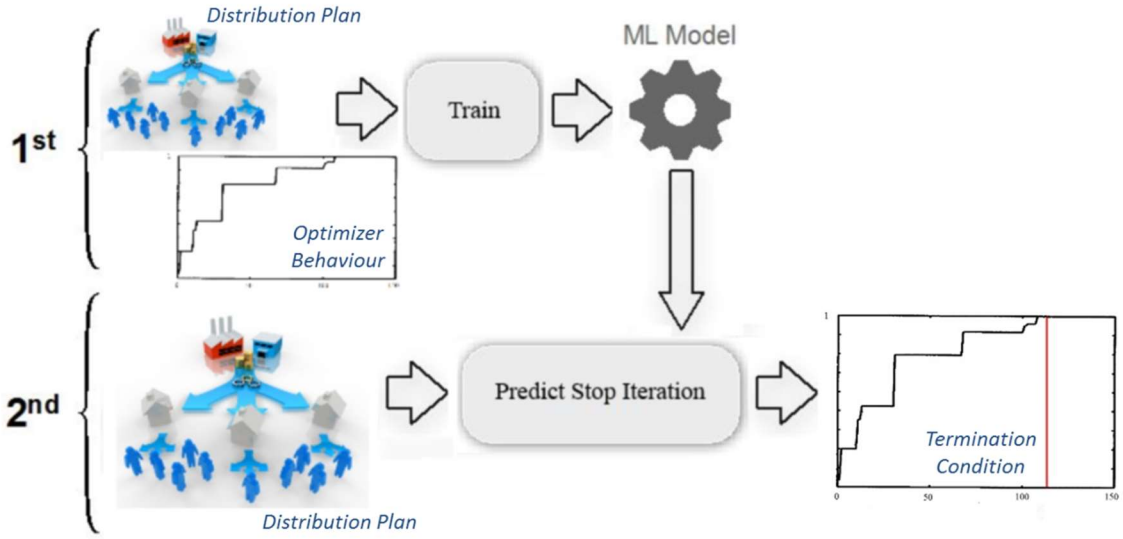

Figure 5.1 - Graphical representation of the proposed system for the improvement of the runtime performance of the Ant Colony System.

This chapter is structured as follows: section 5.1 explains the motivations and analyze related work. Section 5.2 presents the fitness landscape analysis used to gather information on the optimizer behavior. In section 5.3 the features used to characterize different problem instances are defined. Two differing class definitions are presented in the section. Such definitions are based on knowledge from the fitness landscape analysis and allow the setting of a termination criterion for a given problem instance. In section 5.4 the results of the optimization experiments carried out using the proposed method are analyzed. Section 5.5 draws the conclusions and discuss future research directions.

5.1 Motivations and Related Work

According to Arisha et al. [124] the most frequently adopted approaches used when solving supply chain optimization problems are *gradient-based methods*, *metamodel-based methods*, *statistical methods* and *random search / metaheuristics*. Arisha et al. [124] also discusses the limitations of traditional techniques such as linear programming, integer programming and mixed-integer programming when handling the inherent interdependencies found in the current generation of supply chain networks. A review conducted by Ogunbanwo et al. [1] identified a trend towards the use of meta-heuristic approaches as the solution basis for solving transportation networks problems. The most common approaches include (Multi Objective) Genetic Algorithm, Ant Colony Optimization (ACO), and Swarm Particle Optimization. The ACS is a variation of the ACO and is defined in Dorigo et al. [78]. A successful application to the problem of transportation network optimization may be found in Musa et al. [30].

In the context of Ant Colony System and Meta-heuristics approaches, much work has been done in reducing the runtime requirements of the methods. Typically, the most common approaches either employ methods to reduce the search steps and arrive more quickly at higher quality solutions or they exploit parallelization / hardware acceleration techniques. Tseng et al. [125] presented a novel method to generally speed up the Ant Colony Optimization (ACO) for the Travelling Salesman Problem (TSP), by reducing redundant steps in its search. Pedemonte et al. [98] present a survey of recent advances in the parallel implementation of Ant Colony Optimization. This work approached the problem of runtime reduction by focusing on the optimal setting of termination criteria to minimize the runtime required for a given instance.

According to Dorigo et al. [92], for all meta-heuristics, there is no general termination criterion. In practice, a number of rules of thumb have been used: the maximum CPU time elapsed, the maximum number of solutions generated, the percentage deviation from an optimum lower/upper bound, and the maximum number of iterations without improvement in solution quality are examples of such rules [92]. Lv et al. [126] analyzed recent reviews of Ant Colony Optimization applications with a view to answer the questions “how to evaluate improvement?” and “what are the termination conditions?”. However, their survey did not provide concrete answers, they found that

all termination criteria are described with vague phrases, such as "no improvement is possible", or "termination conditions are met" [126]. More generally, Lv et al. [126] considered some of the earlier fundamental work on meta-heuristics without finding a consensus about termination criteria. More recently, Zhang et al. [127] analyzed the approximate termination condition for the ACO applied to TSP. They found that many of the termination conditions are only used in experimentation and are often too difficult or uneconomic for deployment in solving practical problems [127].

The approach taken in this work learns from the behavior of the optimization process on previous problem instances in setting the termination criteria. It was observed that in many instances the optimization algorithm finds the best solution early in its search and then stalls, continuing the search for many more iterations without finding a better solution. This phenomenon is referred to as the *stalling effect*, Stomeo et al. [128] state: "The problem of stalling effect in fitness functions is related to the non-improvement of the fitness values during the evolutionary process". Figure 5.2 shows an example of the stalling effect. In this work, it is sought the relationship between the problem characteristics and the performance of the optimization process, with the intent of predicting how the solver will perform on a given instance and set the termination criteria to minimize the solver search time.

In the course of the chapter, it is shown how this method answers the concerns raised by Lv et al. [126] and Zhang et al. [127]. A definite procedure to evaluate improvement is provided, which is adopted to set proper termination conditions. Using well-known machine learning algorithms for prediction and existing and open source libraries for the implementation, the difficulty of adoption of the proposed system is kept low, making it economic for deployment in real-world application. Complexity wise, both the learning and prediction steps do not significantly affect the performance: the learning step, which is required to be performed only once, is expected to be fast due to the small number of features involved, and including the prediction step into the optimization process will significantly reduce the time requirements as the termination criteria are dynamically set to the optimum of each instance.

Understanding the relationship that exists between the problem instances and the optimization algorithm has led to improvements in the optimization process. Smith-

miles et al. [129] used a knowledge discovery approach to seek insight into the relationship between the Scheduling Problem structure and the effectiveness of heuristics. Rules from a decision tree were used to select the best heuristic from a portfolio. Similar work has been undertaken in Smith-miles et al. [130] for the Traveling Salesman Problem. Here, it is presented a similar approach where instead of using the acquired knowledge to select the most promising algorithm from a portfolio, it is used to improve the performance of the current one.

5.2 Fitness Landscape Analysis

Fitness landscape analysis [131] provides a vivid metaphor of the search space as perceived by an optimization process [132]. Metaphors of a landscape are commonly used to aid the understanding of heuristic search methods when solving combinatorial optimization problems. Furthermore, the concept has been shown to be useful for understanding the behavior of combinatorial optimization algorithms, and can help in predicting their performance [133].

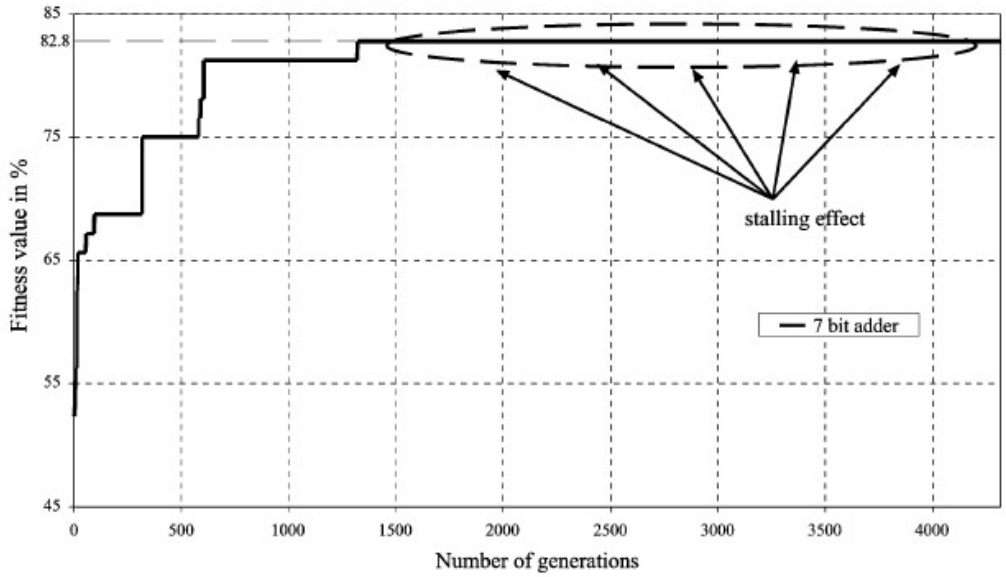

Figure 5.2 - The stalling effect in fitness function analysis refers to the phenomenon where the fitness values do not improve during most of the optimization process. Source of the figure is Stomeo et al. [128].

Given a vector of variables $x \in \mathbb{R}^n$ and a vector of cost coefficients $c \in \mathbb{R}^n$, let us recall the definition of an optimization problem as:

$$v^* = \min \{c^T x | Ax = b \ \wedge x \geq 0\}, \tag{5.1}$$

where $A \in \mathbb{R}^{m \times n}$ is a matrix of coefficients, $b \in \mathbb{R}^m$ is a vector of coefficients and $v^* \in \mathbb{R}^n$ is a vector of assignments for the variables x such that the value of the objective function $c^T x$ is minimum. The matrix A and the vector b define the constraints over the decision variables x and define the problem search space. Therefore, an optimization problem is defined by the tuple $lp \coloneqq (c, A, b)$.

The fitness landscape of an optimization problem lp is the tuple $fl \coloneqq (S, f, d)$, where $S(A, b) = \{v \in \mathbb{R}^n | Av = b\}$ is the set of all possible solutions, $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ is the fitness function defined as $f: v \mapsto c^T v$ and $d: R^n \times R^n \longrightarrow R$ is the distance between two feasible solutions. In Evolutionary Computation, for binary coded problems, the distance measure is usually the Hamming distance between bit strings [133]. For problems where the solution is a vector of real number, the Euclidean distance may be applied.

Usually, the fitness landscape is interpreted as a graph $G = \{V, E\}$ with vertex set $V = S$ and edge set $E = \{(v, v') \in S \times S | d(s, s') = d_{min}\}$ with d_{min} denoting the minimum distance between two points in the search space [133]. Such interpretation allows for effective analysis and visualization of the search space. However, for the purpose of this work, the interest lays in analyzing how the search for the optimal solution evolves over time and in predicting the best termination point based on instance features. Let us define the search process or *walk* on a landscape [134] as the t -tuple $\Gamma = (v_0, v_1, \dots, v_{t-1})$ being the sequence of visited solutions during the search/optimization process. The fitness landscape analysis adopted in this work is, therefore, the sequence of fitness function evaluations at each iteration:

$$\Phi_\alpha = \big(f(v_0), f(v_1), \dots, f(v_{t-1})\big). \tag{5.2}$$

The performance of the search process may be measured as the number of iterations required to find the optimal solution.

Let us define the speed of the search process and its acceleration, respectively as the improvement of the best known solution over the first one and the rate of change in the speed. Given the iteration $i \in (0, t)$, the speed of the optimization process for the tuple Φ is:

$$\begin{aligned} s&: \mathbb{Z}^+ \to \mathbb{R}, \\ i &\mapsto (\Phi(i) - \Phi(0))/i, \end{aligned} \tag{5.3}$$

and the acceleration is:

$$\begin{aligned} a&: \mathbb{Z}^+ \to \mathbb{R}, \\ i &\mapsto \big(s_\Phi(i) - s_\Phi(0)\big)/i. \end{aligned} \tag{5.4}$$

Such definitions of speed and acceleration describe the rate of improvement of the best known solution at any given iteration. Figure 5.3 shows an example of the result of this analysis.

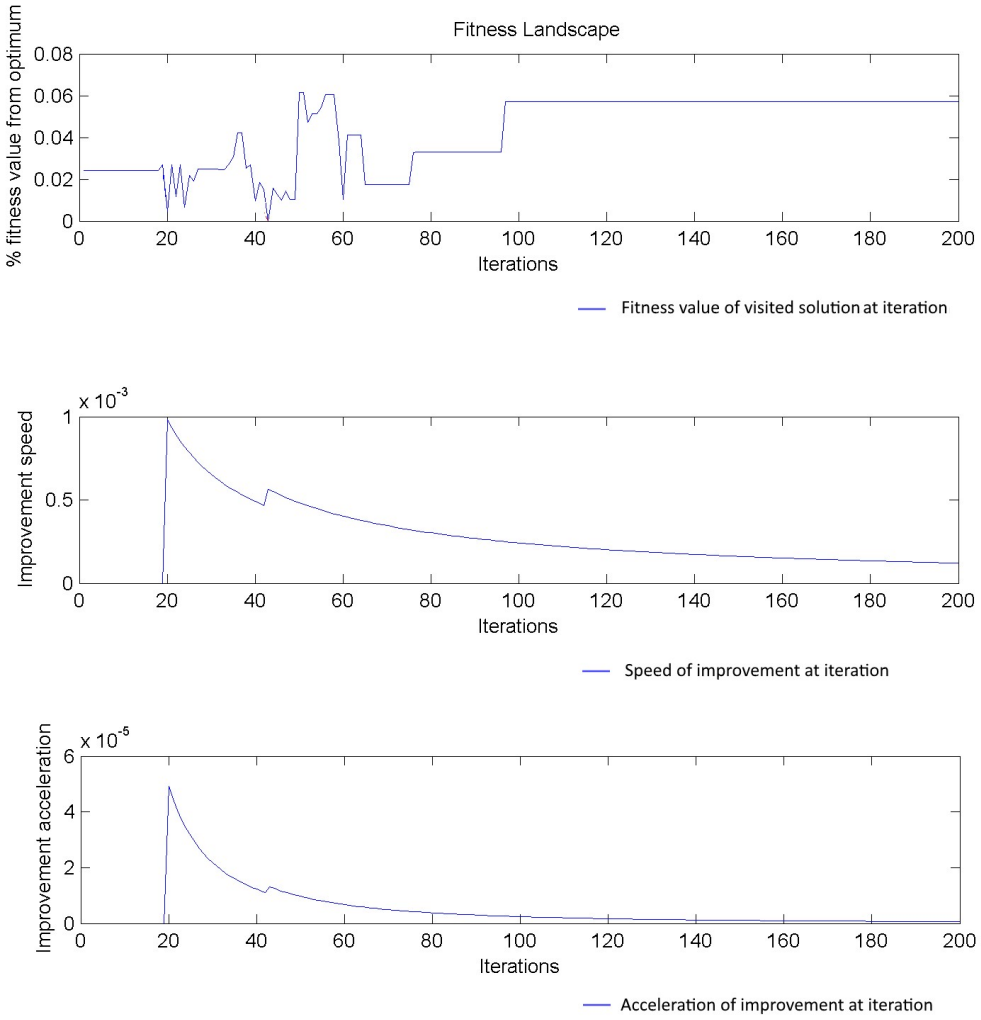

Figure 5.3 - Example of fitness landscape analysis as defined in Eq. (5.2) with speed and acceleration improvement. The definition of speed and acceleration is respectively in Eq. (5.3) and (5.4).

As a variation, the fitness landscape analysis can be modified to include the topology of the search space by considering the mean pair-wise distance of visited solutions at any given iteration. The updated definition of fitness landscape analysis would be as follows:

$$\Phi_{\beta} = \big(p(0), p(1), \dots, p(t-1)\big) \tag{5.5}$$

where let p be the function that measure the mean pair-wise distance of the visited solutions:

$$\mathrm{p}: \mathbb{Z}^+ \to \mathbb{R},$$
$$i \mapsto \frac{\sum_{k\in[0,i]} \sum_{j\in[0,k)} d(v_k, v_j)}{i * (i-1)/2}, \tag{5.6}$$

and consequently the standard deviation on the pair-wise distance is:

$$\mathrm{p_{sd}}: \mathbb{Z}^+ \to \mathbb{R},$$
$$i \mapsto \sqrt{\frac{\sum_{k\in[0,i]} \sum_{j\in[0,k)} \left(d(v_k, v_j) - p(v_k)\right)^2}{i * (i-1)/2}}, \tag{5.7}$$

Figure 5.4 shows an example of the modified fitness landscape analysis. The figure shows a significant increase in the standard deviation of the visited solutions around the 100th iteration. It is difficult to predict the reasons of this phenomenon as the results depend on the optimization algorithm behavior and the search space landscape. However, the most likely reason for this is due to the application of random moves. During the first 100th iterations, the optimization algorithm mostly performed local searches in the neighborhood of the starting solutions. The algorithm did not perform any (or very few) random moves. All the solutions at this point are expected to be ferly close to the starting one. Around the 100th, the algorithm performed a few random moves. The algorithm is now searching in a different portion of the search space. This portion may be either far or different from the starting one. The feasible solutions found in this new portion are likely to be significantly different from the ones found before the random steps.

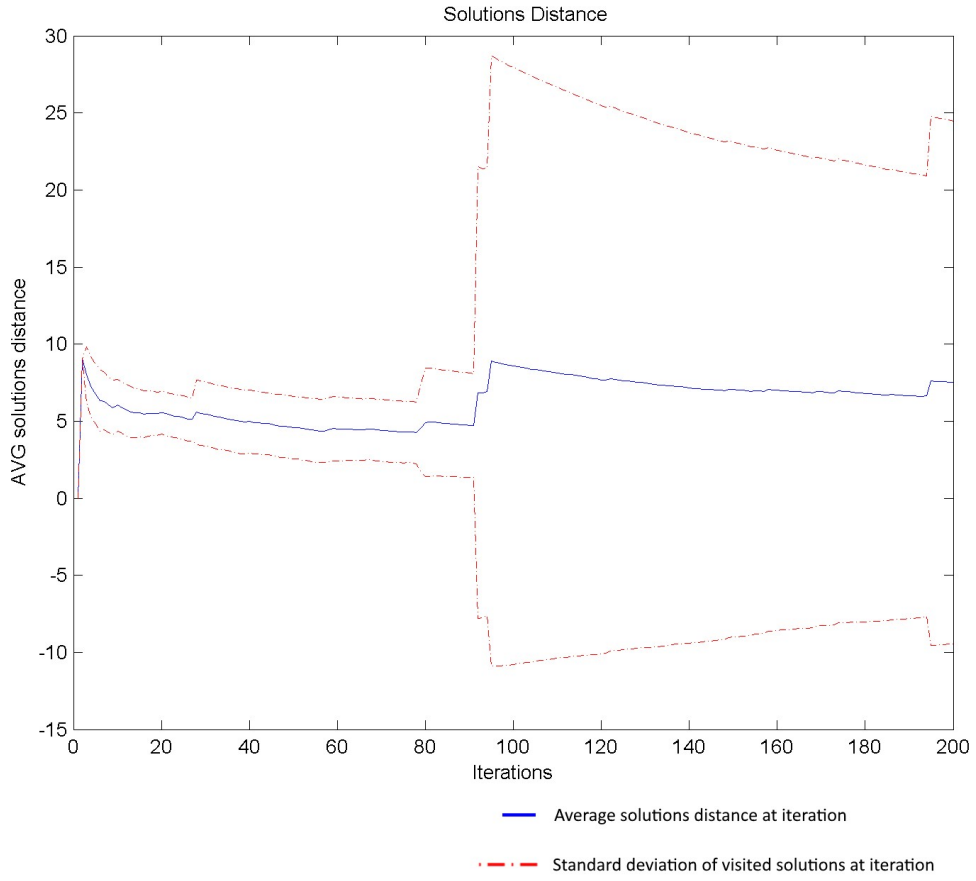

Figure 5.4 - Example of fitness landscape analysis as defined in Eq. (5.5). The solid line is the mean pair-wise distance of the visited solutions as defined in Eq. (5.6) and the dash-dot lines are the standard deviation on such mean as in Eq. (5.7).

5.3 Features of a Supply Chain Optimization Problem

The optimization of supply chains commonly consists of finding the best route to send products from a set of suppliers to a set of customers/dealers. As a generic problem, supply chain optimization is defined by a set of suppliers, a set of dealers and a distribution network. Each supplier is associated with a production capacity and cost, each dealer has a product demand which may vary over time and the distribution network is defined in terms of transportation times and costs between network nodes. Solutions to such problem are usually sought by the application of mathematical programming and artificial intelligence techniques. The model of the problem addressed

in this chapter is as defined in chapter 3. The optimization algorithm implemented is the Ant Colony System as defined in section 3.5.

The following sections, 5.3.1 and 5.3.2, characterize features of the problem instances and propose two class definitions related to the solver behavior. The purpose of the class definitions is to provide an understanding of the complexity of a given instance by considering the behavior of the optimization algorithm. These class definitions allow the termination condition to be set according to the difficulty level of the instance. These are mostly related to the maximum number of iterations or to the maximum number of visited solutions. As described above, this work focuses on termination condition to address the *stalling effect* problem and improving the time complexity of the optimization process over a given set of instances. However, it is reasonable to assume the same principle may very well be adopted to set others parameters. Arguably, for example a more difficult instance might require a higher number of ants or a lower exploitation to exploration ratio.

5.3.1 Problem Features

Supply chain optimization problems usually differ in their demand, the production capacity and some details of the distribution network.

The features adopted to summarize variations in demand and the production capacities are:

- *Percentage of active dealers*. The total number of dealers is known from the definition of the full distribution network. Instances with more active dealers typically will be more difficult to solve and probably require more iterations.
- *Mean and standard deviation of the demand*. Such statistics briefly summarize the distribution of the demand throughout the network.
- *Mean and standard deviation of the capacity*. As with the demand, this feature describes the distribution of the capacity throughout the network.
- *Mean and standard deviation of the capacity per demand*. The purpose here is to measure how much capacity is available on average to satisfy the demand of a given dealer.

- *Ratio of total demand to total production capacity*. This feature is a generalization of the previous one.

The features used to describe the distribution network are:

- *The ratio of production sources to dealers*. This highlights how many production sources are available to satisfy a given dealer's demand.
- *The total number of connections between production sources and dealers*. This describes the level of connectivity in the underlying network.
- *Mean and standard deviation of the values in the heuristic information matrix*. In this context, the heuristic information refers to the information held on the routes in the network which guide the solver in building the distribution plan. For instance, if the goal is to maximizing the profit of a distribution plan, then the heuristic information is likely to be the transportation cost on the routes. Such a feature should distinguish between instances with different variations in transportation costs. Instances with uniformly distributed costs are likely to be easier to solve as small variations in the distribution plan will not fundamentally affect the overall profit.

5.3.2 Class Definition

As the intent of this work is to reduce the task of finding the best termination condition to a classification problem, the following class definitions need to be discrete and preferably of a nominal type. A discretization of the class features has been achieved by applying a simple clustering on their values using instances of the training set. Three obvious class values $\{easy, medium, hard\}$ may be produced from the application of K-means [135] with 3 clusters.

5.3.2.1 Fitness Function Values Through Iterations

The first class definition is based on the fitness landscape analysis as define in Eq. (5.2). A smaller sample containing a sequence of fitness values is considered for each problem instance. The number of samples is 10% of the total number of iterations and the sampling rate is quadratic so that more iterations at the beginning of the search process are considered and hence more details are collected prior to the

best solution being found. These sequences are the input to the clustering step and the output centroid are themselves sequences of fitness values. The speed and acceleration of the centroid sequences are measured as in Eq. (5.3) and (5.4). Figure 5.5 shows an example depicting the result of clustering the sequences of fitness function values. The termination condition is the average value of the following criteria:

- The first iteration when the best solution is found, $\left(argm \quad_{i\in[0,t)}\Phi_\alpha(i)\right)_0$.
- The highest iteration when the speed of change falls below the average speed. Let I be the set of iteration indexes where the speed is closer to the average value $I = arg \quad_{i\in[0,t)}\left(s_{\Phi_\alpha}(i) \cong (\sum_{j\in[0.t)} s_{\Phi_\alpha}(j))/t\right)$. The iteration of termination is $I_{|I|}$.
- The highest iteration when the acceleration falls below to the average acceleration. Similarly to the step above the set I is defined as $I = argm \quad_{i\in[0,t)}\left(a_{\Phi_\alpha}(i) \cong (\sum_{j\in[0.t)} a_{\Phi_\alpha}(j))/t\right)$ and the iteration of termination $I_{|I|}$.

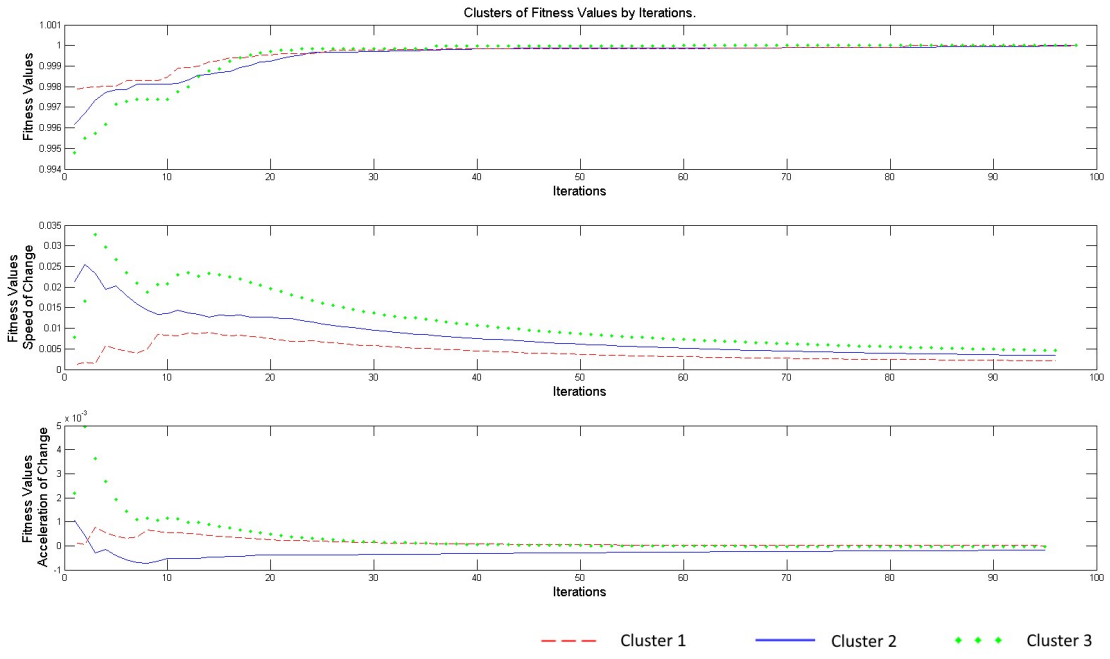

Figure 5.5 - Centroids result of the clustering of the fitness function analysis based on the definition in Eq. (5.2). The fitness function values are normalized for visualization purposes. The speed and acceleration of the resulting centroids is also measured according to Eq. (5.3) and (5.4).

5.3.2.2 Pair-wise Distance Between Visited Solutions

The definition of the second class is based on the fitness landscape analysis as defined in Eq. (5.5). The purpose of this definition is to avoid visiting solutions that are the same or very close to each other. In almost all practical applications, the optimization process is stopped after a finite number of search operations, regardless of whether the optimal solution has been found or not. An approximation to the optimal solution is generally acceptable provided the quality is reasonably high. Arguably, if one of the main concerns is reducing the computational time, then one may be willing to accept lower quality solutions. This definition attempts to terminate the optimization process as soon as the difference between visited solutions does not significantly improve the quality of the found solution; that is the tested solutions are not very different from each other and those perturbations do not lead to an improvement in the solution. As for the previous class definition, for each instance, the sequence of fitness function values is sampled according to a quadratic

rate. These sequences are the input to the clustering step and an example of centroids is shown in Figure 5.6. Again, the termination condition is the average value of the following criteria:

- The first iteration when the pair-wise distance between the visited solutions is the highest, $\left(argma\ _{i\in[0,t)}\Phi_\beta(i)\right)_0$.
- The highest iteration when the speed of change falls below the average speed. The iteration of termination is $I_{|I|}$ where the set I is defined as $I = argma\ _{i\in[0,t)}\left(s_{\Phi_\beta}(i) \cong (\sum_{j\in[0.t)} s_{\Phi_\beta}(j))/t\right)$.
- The highest iteration when the acceleration falls below to the average acceleration. Similarly to the step above, the termination iteration is $I_{|I|}$ where the set I is defined as $I = argma\ _{i\in[0,t)}\left(a_{\Phi_\beta}(i) \cong (\sum_{j\in[0.t)} a_{\Phi_\beta}(j))/t\right)$.

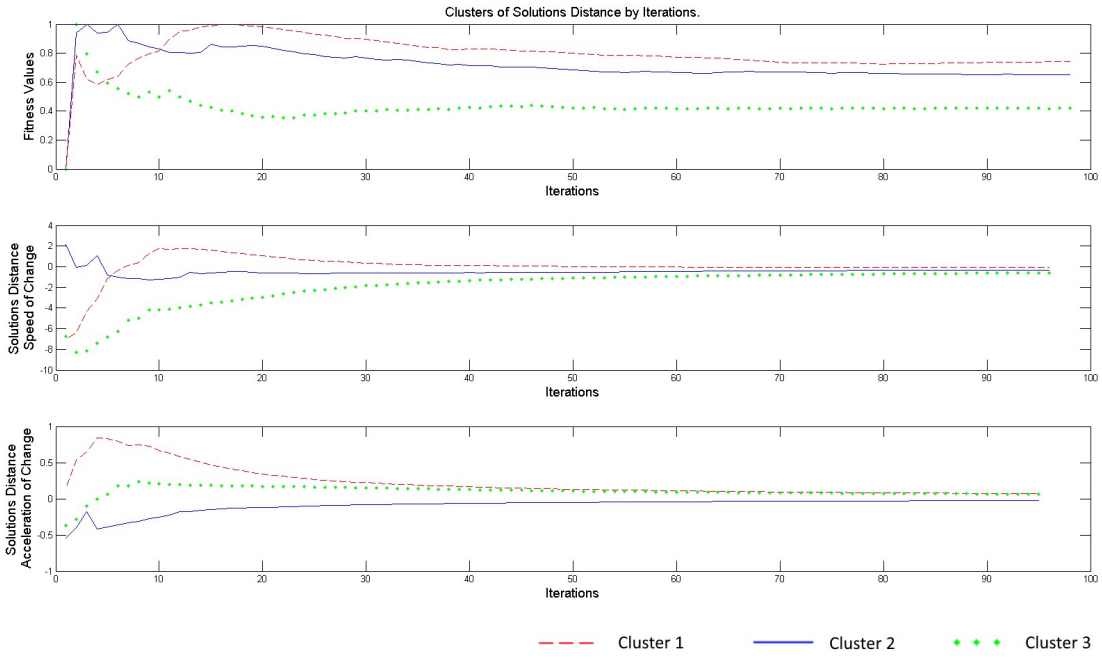

Figure 5.6 - Centroids result of the clustering of the fitness landscape analysis based on the definition in Eq. (5.5), (5.6), and (5.7). The speed and acceleration of the resulting centroids is also measured.

5.4 Numerical Experiments

The two class definitions described in 5.3.2 have been tested on the dataset provided by Caterpillar published in [4]. To address the machine learning needs of this work, it has been adopted the data mining framework Weka [136] to implement and test several classification systems. The implemented classification systems are *0R*, *1R*, *Naïve Bayes*, *Bayes Network*, *J48*, *Random Forest* and *SMO with a polynomial kernel of degree 3*, all of which have been used to build a classification committee. Witten et al [137] in their book describe the theory and implementation details of all these algorithms. The algorithm *0R* is often used to set a baseline for classification accuracy, as its predicted class is simply the most frequent one. As the amount of data available for training and testing is limited, the performance of each single model has been assessed through a 10-fold cross validation scheme. In Table 5.1, the classification accuracy of the single models is shown. The method presented in this chapter also has been tested on the two

randomly generated problems in [5] and [6]. Table 5.2 shows the accuracy measures for the first generated problem, whereas Table 5.3 shows them for the second.

Table 5.1 - Classification accuracy on the problem in [4] for 1000 iterations and K-means with 3 clusters to define the class. The 0R system provides a baseline for the classification problem. The average and standard deviation do not include 0R. 0R stands for ZeroR, the baseline classifier, 1R [138] for OneR and SMO [139] for sequential minimum optimization, the algorithm for training support vector machine.

	0R (baseline) (%)	1R (%)	Naïve Bayes (%)	Bayes Net (%)	J48 (%)	Random Forest (%)	SMO Poly3 (%)	Average (%)	STD (%)
Transportation Time									
5.3.2.1	64.7%	63.9%	63.2%	61.8%	63.9%	64.7%	63.9%	63.2%	61.8%
5.3.2.2	76.4%	61.8%	64.6%	64.6%	65.3%	76.4%	61.8%	64.6%	64.6%
Transportation Cost									
5.3.2.1	55.6%	69.4%	72.9%	75.7%	75.7%	55.6%	69.4%	72.9%	75.7%
5.3.2.2	63.9%	71.5%	75.0%	77.1%	76.4%	63.9%	71.5%	75.0%	77.1%
Profit									
5.3.2.1	45.8%	83.3%	79.2%	83.3%	91.7%	45.8%	83.3%	79.2%	83.3%
5.3.2.2	41.7%	83.3%	75.0%	95.8%	87.5%	41.7%	83.3%	75.0%	95.8%
Resilience									
5.3.2.1	84.7%	86.8%	86.8%	88.2%	86.8%	84.7%	86.8%	86.8%	88.2%
5.3.2.2	83.3%	97.2%	93.8%	93.1%	95.8%	83.3%	97.2%	93.8%	93.1%

Table 5.2 - Classification accuracy on the problem in [5] for 1000 iterations and K-means with 3 clusters to define the class. The average and standard deviation do not include 0R. 0R stands for ZeroR, the baseline classifier, 1R [138] for OneR and SMO [139] for sequential minimum optimization, the algorithm for training support vector machine.

	0R (baseline) (%)	1R (%)	Naïve Bayes (%)	Bayes Net (%)	J48 (%)	Random Forest (%)	SMO Poly3 (%)	Average (%)	STD (%)
Transportation Time									
5.3.2.1	63.9%	70.1%	69.4%	50.0%	66.7%	68.8%	63.2%	64.6%	6.5%
5.3.2.2	71.5%	69.4%	69.4%	52.8%	66.0%	73.6%	69.4%	67.5%	6.4%
Transportation Cost									
5.3.2.1	67.4%	73.6%	70.8%	68.8%	78.5%	76.4%	66.0%	71.6%	4.4%
5.3.2.2	67.4%	71.5%	75.0%	72.9%	77.1%	74.3%	66.0%	72.0%	3.8%
Profit									
5.3.2.1	37.5%	100.0%	95.8%	95.8%	87.5%	100.0%	87.5%	94.4%	5.2%
5.3.2.2	37.5%	100.0%	95.8%	95.8%	87.5%	100.0%	87.5%	94.4%	5.2%
Resilience									
5.3.2.1	91.7%	66.7%	91.7%	91.7%	91.7%	91.7%	83.3%	86.9%	8.7%
5.3.2.2	94.4%	93.8%	95.1%	92.4%	91.7%	93.1%	95.1%	93.7%	1.3%

Table 5.3 - Classification accuracy on the problem in [6] for 1000 iterations and K-means with 3 clusters to define the class. The average and standard deviation do not include 0R. 0R stands for ZeroR, the baseline classifier, 1R [138] for OneR and SMO [139] for sequential minimum optimization, the algorithm for training support vector machine.

	0R (baseline) (%)	1R (%)	Naïve Bayes (%)	Bayes Net (%)	J48 (%)	Random Forest (%)	SMO Poly3 (%)	Average (%)	STD (%)
Transportation Time									
5.3.2.1	70.1%	54.9%	68.1%	68.8%	74.3%	68.1%	68.8%	67.6%	5.6%
5.3.2.2	66.7%	50.0%	70.1%	62.5%	68.1%	63.9%	64.6%	63.7%	6.1%
Transportation Cost									
5.3.2.1	45.1%	55.6%	61.1%	51.4%	58.3%	56.3%	45.1%	53.3%	5.8%
5.3.2.2	48.6%	48.6%	53.5%	41.7%	52.1%	46.5%	48.6%	48.5%	3.5%
Profit									
5.3.2.1	62.5%	83.3%	75.0%	100.0%	87.5%	87.5%	83.3%	86.1%	7.4%
5.3.2.2	70.8%	87.5%	87.5%	95.8%	92.0%	92.0%	92.0%	91.0%	2.9%
Resilience									
5.3.2.1	66.7%	66.7%	66.7%	33.3%	83.3%	66.7%	66.7%	64.3%	13.9%
5.3.2.2	87.5%	88.2%	91.0%	86.1%	88.9%	84.0%	87.5%	87.6%	2.0%

5.4.1 Performance Improvements

This section compares the performance of the original optimization process with the one developed in this chapter and deploy it to predict the best stopping iteration for each given instance. The metrics for the experiment are the performance measure of the given objective and computation time required to find the distribution plan. Table 5.4 show the performance improvements when the classification system is adopted to predict the best stopping iteration for each given instance on the problem in [4], Table 5.5 and Table 5.6 for the problems in [5] and [6] respectively. The Δ difference in Table 5.4, Table 5.5, and Table 5.6 for both the optimization performance and the runtime is calculated as $\left(\frac{|x_a - x_e|}{x_e}\right) * 100$, where x_e is the expected value (the result of the original

optimization) and x_a is the actual one (the result of the proposed algorithm). As shown by the experiments below, the application of both the class definitions from 5.3.2.1 and 5.3.2.2 consistently reduced runtime of about a third or more, while the found solutions have no significant difference. The runtime reduction is consistent with the reduction of the quality of the solution. The values in Table 5.4, Table 5.5, and Table 5.6 differ from Table 3.5, Table 3.8, and Table 3.11 as the results presented in the latter experiments are based on the average over 10 runs of the optimization.

Table 5.4 - Performance improvements when the classification system is adopted to predict the best stopping iteration for each given instance on the problem in [4]. The values for the regular optimization differ from Table 3.5 as the results presented in the latter experiment are based on the average over 10 runs of the optimization. The units for "Optimization Performance" column depend on the objective type.

	Optimization Performance	Δ Opt. Performance (%)	Runtime (s)	Δ runtime (%)
Transportation Time				
Regular opt	95,733.03	-	216.9601	-
5.3.2.1	96,017.19	0.297%	73.38092	66.18%
5.3.2.2	95,925.64	0.201%	75.09549	65.39%
Transportation Cost				
Regular opt	3,173,897.62	-	251.6311	-
5.3.2.1	3,174,502.60	0.019%	86.76324	65.52%
5.3.2.2	3,174,251.33	0.011%	88.31299	64.90%
Profit				
Regular opt	101,247,174.47	-	311.0951	-
5.3.2.1	101,231,928.30	0.015%	127.2448	59.10%
5.3.2.2	101,093,944.10	0.151%	102.0328	67.20%
Resilience				
Regular opt	0.988636364	-	249.888	-
5.3.2.1	0.977272727	1.149%	2.540268	98.98%
5.3.2.2	0.984848485	0.383%	22.38805	91.04%

Table 5.5 - Performance improvements when the classification system is adopted to predict the best stopping iteration for each given instance on the problem in [5]. The values for the regular optimization differ from Table 3.8 as the results presented in the latter experiments are based on the average over 10 runs of the optimization. The units for "Optimization Performance" column depend on the objective type.

	Optimization Performance	Δ Opt. Performance (%)	Runtime (s)	Δ runtime (%)
Transportation Time				
Regular opt	136,987.58	-	218.9261	-
5.3.2.1	137,526.00	0.393%	74.42885	66.00%
5.3.2.2	137,400.50	0.301%	74.09663	66.15%
Transportation Cost				
Regular opt	4,488,605.36	-	247.634	-
5.3.2.1	4,494,763.13	0.137%	87.16367	64.80%
5.3.2.2	4,495,143.24	0.146%	88.49257	64.26%
Profit				
Regular opt	147,373,460.30	-	305.3214	-
5.3.2.1	147,370,169.78	0.002%	114.3661	62.54%
5.3.2.2	145,986,178.56	0.941%	107.311	64.85%
Resilience				
Regular opt	0.992424242	-	251.8403	-
5.3.2.1	0.977272727	1.527%	2.576935	98.98%
5.3.2.2	0.986742424	0.573%	17.0009	93.25%

Table 5.6 - Performance improvements when the classification system is adopted to predict the best stopping iteration for each given instance on the problem in [6]. The values of the regular optimization differ from Table 3.11 as the results presented in the latter experiments are based on the average over 10 runs of the optimization. The units for "Optimization Performance" column depend on the objective type.

	Optimization Performance	Δ Opt. Performance (%)	Runtime (s)	Δ runtime (%)
Transportation Time				
Regular opt	2,345,846.86	-	417.1356	-
5.3.2.1	2,456,623.74	4.722%	4.233668	98.99%
5.3.2.2	2,453,835.26	4.603%	4.733023	98.87%
Transportation Cost				
Regular opt	127,427,904.27	-	479.3158	-
5.3.2.1	128,520,322.02	0.857%	158.5022	66.93%
5.3.2.2	127,657,131.79	0.180%	163.1223	65.97%
Profit				
Regular opt	861,800,648.54	-	560.5053	-
5.3.2.1	860,957,855.61	0.098%	365.6538	34.76%
5.3.2.2	859,298,923.34	0.290%	334.6041	40.30%
Resilience				
Regular opt	0.996212121	-	470.437	-
5.3.2.1	0.988636364	0.760%	4.812747	98.98%
5.3.2.2	0.992424242	0.380%	5.4566	98.84%

5.5 Summary

The aim of this work was to make improvements in the practical time complexity for the Ant Colony System when applied to a real-world supply chain optimization problem. The starting observation was that in many instances the optimization algorithm finds the best solution early in its search and then stalls, effectively searching over many more iterations without finding a better solution. This is referred to as the phenomenon of *stalling effect*. This work postulates that if the onset of the stalling

effect could be predicted, then for that given instance the search can be terminated with the ensuing benefit that the overall optimization process might very well require less time to find a solution of equal or comparable quality.

The approach presented learns from the behavior of the optimization process itself on past instances in setting the termination criteria. A relationship between specific characteristics of the problem and the performance of the optimization process is sought. The relationship is used to predict how the solver will perform on a given instance and to set the termination criteria such that the time spent by the solver in its search is minimized.

A fitness landscape analysis has been performed to understand the behavior of the optimization process. Two class definitions have been proposed to capture the behavior of the process, classify the problem instances and predict the best termination iteration. Features not related to the optimization process have been used to characterize different problem instances. Several classical machine learning classification algorithms have been employed to learn the relationship between problem instances and termination classes.

The proposed algorithm has been tested on a real-world supply chain, plus two random generated problems. The runtime of the Ant Colony System has been reduced to 72.35% on average (with a standard deviation of 18.16%) in all the experiments, while the overall difference in the quality of the solutions was deemed acceptable.

As future work, more features of the problem will be investigated, in order to gain a better understanding of the factors that make the greatest contribution to the performance of the optimization process. Moreover, it is needed to improve the class definitions or the classification system such that the found solutions to all intent and purposes are no difference to the original. Finally, as anticipated, a variation of this approach could be employed to set other parameters to their appropriate ‘best’ value. Future work will investigate the effectiveness of the method in setting other parameters and determining the impact of their relationship with the problem features.

6 AUTOMATIC MODEL DEFINITION FOR SUPPLY CHAIN OPTIMIZATION

This chapter has been submitted for publication as short paper in IEEE Transactions on Evolutionary Computation.

Optimization is one of the key components of any decision support system. It allows to automatically explore all possible scenarios and to discover the best one. In the context of supply chain management, optimization is particularly effective as it allows to simultaneously consider the many aspects and factors of the distribution network. The typical process of implementing a generic optimization system consists of three main tasks: data collection and preparation, model definition, and optimization algorithm implementation. Figure 1.2 shows the sequence of tasks necessary to deploy an optimization system. A key requirement in this sequence is knowledge and understanding of the logistics operations and experience in designing mathematical models. Despite the fact that many supply chain models have been proposed over the years, implementing such models in specific cases is still a challenging task, which typically is performed manually.

In this chapter, it is addressed the task of increasing the automatization of implementing an optimization system, and, specifically, the design of the optimization model. For this work, a scenario is envisaged where a supply chain has been in operation for quite some time, but an automatic optimization system is not as yet in place. The data generated by the management of the supply chain contains insight about the model. In particular, distribution plans implemented in the past may provide useful information. For instance, they could highlight how just a few production sources or routes are more preferable than others by considering the frequency of their usage in the implemented solutions. Regression analysis is used to "mine" such information and automatically create an approximate objective function. The objective function is then integrated into a general supply chain optimization model and used to evaluate new unseen solutions. Figure 6.1 depicts the new process of implementing an optimization system. It is shown that this system produces a model that closely approximates the original one, and requires significantly lower effort.

This chapter is structured as follows: section 6.1 explains the motivations and analyze related work. Section 6.2 presents the regression analysis used to approximate the optimization model starting from existing solutions. In section 6.3 the main steps of the system are described and it is shown how the approximated model is used during the optimization. Section 6.4 presents the results of the optimization experiments performed using the approximated model. In section 6.5 conclusions are drawn and future research directions are discussed.

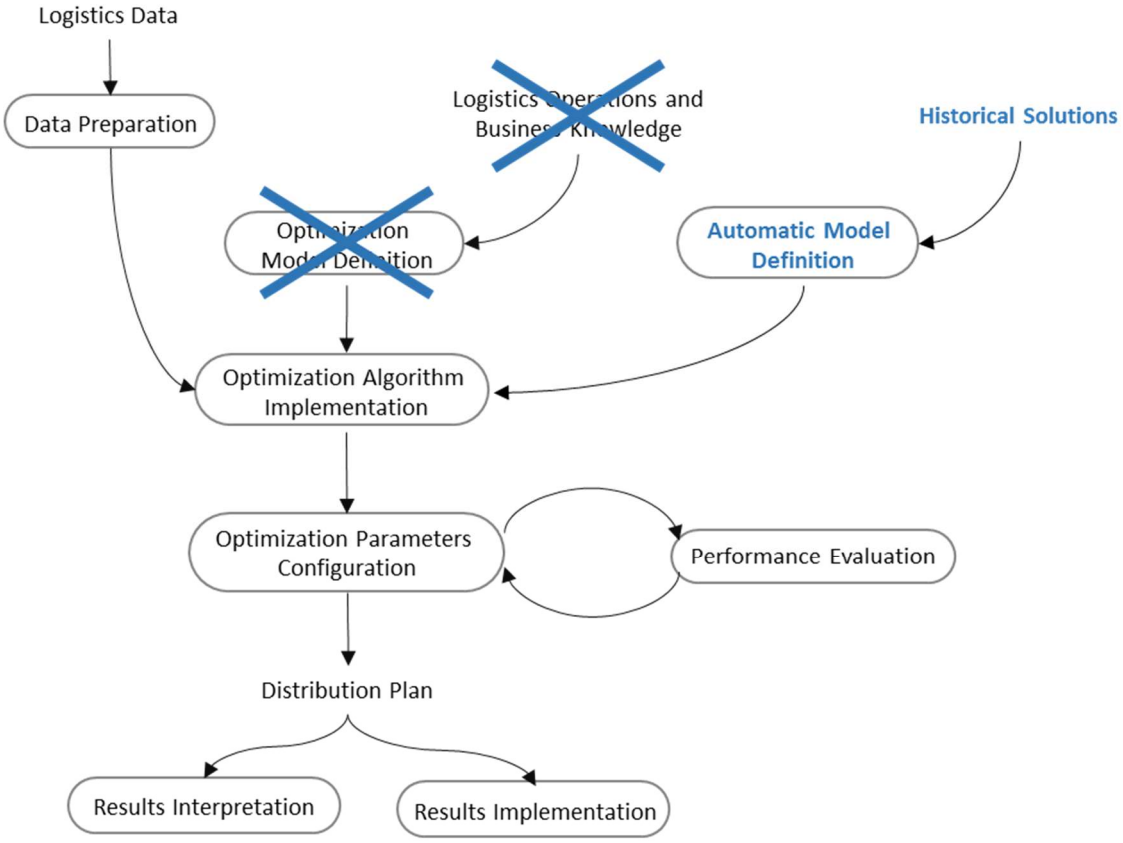

Figure 6.1 – Process of implementing an optimization system highlighting the differences and improvements made possible by the idea introduced in this chapter. Specifically, in-depth knowledge of the logistics operations and the business is not required any longer. Moreover, the system presented in this chapter aids the definition of the optimization model and reduces the effort and time required. Given a set of manually generated historical solutions, a good approximation of the model is automatically generated from it.

6.1 Motivations and Related Work

The application of optimization processes as an aid to decision makers when analyzing and managing a supply chain has proven to be considerably successful in a range of scenarios. In the academic community, many optimization models and algorithms have been designed. Nevertheless, implementing and deploying a new optimization process is still a complex task, which requires significant knowledge of the business and substantial development time, effort and experience. Arguably, the three main steps that

must be completed sequentially in order to successfully deploy an optimization system are as follows:

1. Data and detailed information about the supply chain must be collected and properly formatted.
2. A formal description of the supply chain must be designed to describe the relationships between the main factors of the optimization problem. Typically, a mathematical model which consists of an optimization function, a set of decision variables, and a set of constraints over them are typically used.
3. An optimization algorithm may be employed to effectively explore the search space defined by the optimization model and find the optimal solution to the problem.

Each step requires expertise from different areas. The first step requires the cooperation of supply chain managers and the employees responsible for data collection and maintenance.

The second step is possibly the most complex as it is necessary to have a full understanding of the business process and experience in designing mathematical models to accurately represent it. Typically, an operations research and a management engineering background are desirable. Cope et al. [140] investigated the challenges of designing models for supply chain simulation. Similarly to optimization, simulation requires skills and a scientific background for its implementation, which are vital for this methodology to deliver value to the company adopting it [140]. There are several practices that rely on simulation modeling for strategic and operational decision-making. These practices require hiring simulation engineers, building internal simulation teams, or contract consultants. They are different in terms of budget, time to implement, and returns on investment [140]. Cope et al. [140] analyzed several generic simulation models from a wide range of decision problems. A generic model is applicable to a large set of problems or instances and yet is sufficiently accurate to distinguish between critical performance criteria. Employing a generic model eliminates major portions of the upfront model design process. Generic models are bug free, they have been code optimized for fast run times, and they can be consistently applied throughout the corporation [140]. A model can be constructed that is generic and

reusable with a reasonably small set of unique components. If properly defined, a reusable model is often more accurate and efficient than new models individually constructed for each application scenario [140]. Cope et al. [140] designed an ontology for the supply chain problem that includes two sets of elements:

1. Elements to define the supply chain (e.g. production facilities, material flows, inventory and storage locations, etc.).
2. Elements to describe the logic of the simulation model.

The first set supports defining the supply chain configuration. The second set is used in describing the logic of the simulation. The simulation model is then automatically generated from the descriptions provided.

Simulation and optimization are often used in conjunction to provide a complete decision support system. However, they address two quite different types of analysis, which answer two different questions. Simulation answers the question "what if", whereas optimization the question "what is the best". They require a different "information" view of the problem and, therefore, require a different model representation. A simulation model describes the probability and occurrence of events in the network. An optimization model defines the costs of the logistics operations in the network. The ontology designed by Cope et al. [140] may be adopted to define the optimization model. The first set of elements of the ontology may help support the definition of the supply network. The authors in particular tested the following scenarios:

1. Add a warehouse or distribution center.
2. Vary demand, add/remove customer.
3. Modify supplier to include more detail.
4. Add a new supplier.
5. Vary inventory strategy.

All the above scenarios are applicable in the definition of a supply chain optimization model. The ontology is used to define the supply network. However, it cannot define the logic of the optimization model using it. The system presented in this chapter starts

from a generic definition of the supply network and generates an optimization model from a training set of feasible distribution plans.

Regarding the third task for the deployment of an optimization system, two broad sets of techniques are often employed to find the optimal solution: exact analytical methods and artificial intelligence search algorithms. An example of an analytical method is linear programming, and one for artificial intelligence searching is the Ant Colony System. Linear programming takes advantage of mathematical properties of the problem to quickly explore the search space. Linear programming methods have proven to be quite efficient when solving real-world optimization problems. However, they require the model to satisfy such mathematical properties. Designing the model such that these properties are satisfied is challenging and requires an understanding of difficult mathematical concepts. For a more detailed discussion see section 2.2.1.

Most artificial intelligence search algorithms are advertised as being general purpose. However, in many scenarios, such algorithms require the implementation of specific heuristics in order to perform correctly. Even if no specific heuristics are required, an important characteristic which defines these methods is the large number of parameters required. Understanding and testing the most relevant parameter configurations and their variants may be very resource intensive. Burke et al. [87] argued that the practical impact of heuristic methods and other search techniques in commercial and industrial organizations has not been as great as might have been expected some years ago. Many state-of-the-art heuristics are too problem-specific or too knowledge-intensive to be implemented in an in-expensive, easy-to-use computer systems [87]. More recently, Burke et al. [86] confirmed that the successful application of search methods in real-world computational problems is often threatened by the significant numbers of parameters, algorithm variants, and the lack of guidance in selecting them.

From the previous considerations it appears that the second and third steps are affected by similar limitations: the lack of automatic tools to aid their implementation. The work presented in this chapter focus on the second step for the deployment of an optimization system. Specifically, the aim is to reduce the effort in designing an optimization model. The algorithm proposed in chapter 5 partially addresses the third step. However, the development of more efficient and complete set of tools could be future work.

6.2 Regression Analysis for Model Estimation

Let us recall the general definition of an optimization problem is $v^* = \min\{f(x) | Ax = b \wedge x \geq 0\}$, where $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function and the system of equations $Ax = b$ define the search space. An optimization problem is therefore described by the tuple $OP := (f, A, b)$. The space of feasible solutions is $S(A, b) = \{v \in \mathbb{R}^n | Av = b \wedge v \geq 0\}$. The optimization process may then be defined as a function from the optimization problem $op \in OP$ to a solution $v \in S(A, b)$ belonging to the solution space: $ACS: OP \rightarrow \mathbb{R}^n$, $op \mapsto v$. For more details, see section 4.2.

The definition of an objective function and the search space must closely capture the operational characteristics of the business for the result of the optimization to be meaningful. In the context of the presented system, it is assumed that an automatic optimization and decision support system has not as yet been implemented for a particular business process. As such, a formal mathematical definition of the optimization problem is not available. The objective function f is unknown and outlining a detailed definition would require significant effort and the involvement of business experts. It is also assumed that the considered business process has been operated manually for some time and a set of feasible solutions are available. Let $FS := \{(v, \gamma) | f(v) = \gamma \wedge v \in S\}$ be the set of pairs of feasible solutions adopted in the past and their performance evaluation. In the context of supply chains, many businesses do not employ optimization systems to manage the entirety of their logistics operations. Decisions on how to ship products are made at a local level. It is reasonable to assume that if the business has sufficient history, the overall distribution plans do not violate any constraints, all shipping decisions are plausible and of acceptable quality. A system capable of quickly capturing the "good sense" intrinsic design of such manually-generated solutions, may help reducing the time required to formally define an optimization model.

The purpose of the presented system is to produce an approximation function $h: \mathbb{R}^n \rightarrow \mathbb{R}$ of f such that the error generated when h is used to compute the performance of a solution is minimum, $\epsilon_h = \min_{\text{h}}\{|f(v) - h(v)||v \in FS\}$. Such a function h may be obtained through a machine learning-based approach. Equation (6.1) depicts the

definition of h, where $v \in \mathbb{R}^n$ is a feasible solution and $pm \in \mathbb{R}$ is the value of performance measure for the solution.

$$\begin{aligned} h&: \mathbb{R}^n \to \mathbb{R}, \\ v &\mapsto pm, \end{aligned} \tag{6.1}$$

The solutions in FS are the elements of the training set. Each element in the training set is classified according to their performance evaluation. As the output variables belong to a continuous domain, a regression analysis is performed. Regression analysis is a statistical tool to understand and investigate the relationship between variables. For the problem of supply chain optimization, the input variables define the amount of product shipped on selected routes. The output variable is the performance of the distribution plan. A typical approach in carrying out a regression analysis is to use the linear regression method. In linear regression, the output variable is assumed to be a linear combination of the input parameters. Let x_0 and x_1 be two independent input variables, and y_0 and y_1 the output dependent variables, the linear regression model is defined as below:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon, \ i \in \{0,1\}, \tag{6.2}$$

where β_0 and $\beta_1$are the parameters which define the relationship between input and output variables. ϵ is the error estimate. When applied to the problem of supply chain optimization, x_i are the possible routes in the network and y_i the performance of the distribution plan. During the learning step, the parameters β_j may be estimated using the least square method. The least square method consists of minimizing the difference $e_i = y_i - \hat{y}_i$ between the values of y_i and the estimates $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$, where $\hat{\beta}_j$ are the estimates for the parameters β_j. Eq. (6.3) depicts the least square method:

$$\min \sum_{i=1}^{n} e_i^2. \tag{6.3}$$

For the simple case of eq. (6.2), the values of the estimates $\hat{\beta}_j$ are:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}\,, \tag{6.4}$$

$$\hat{\beta}_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \tag{6.5}$$

where $\bar{x}$ and $\bar{y}$ are the mean values of the respective variables.

The resulting function $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \epsilon,\ i \in \{0,1\}$ of the least square method replaces the objective function of the supply chain optimization problem. When the relationship between input and output variables is not linear, alternative methods to linear regression and least square may be employed. Support Vector Machines (SVM) are often used for regression analysis. As shown in section 6.4, a SVM has been use to build the regression model and the types of kernel tested are: linear, polynomial of degree two and three, and a radial basis function. In machine learning, the Gaussian Process is also applied in regression analysis. In a Gaussian Process regression model, every input point is associated with a normally distributed random variable. Liu et al. [141] designed a Gaussian Process to build a surrogate model of a computationally expensive optimization problem. Given a small number of evaluations of the objective function, the authors built an approximation or a surrogate to the objective function. The surrogate model is less expensive to evaluate and is used to assist the evolutionary algorithm in finding the optimal solution to a set of test functions for optimization (e.g. the Ackley function [142], or the Griewank function [143]). Figure 6.2 shows a simple example of Gaussian Process-based regression.

Despite being expensive to evaluate, knowing the optimization function allows the training set to be sampled more uniformly. A more uniform input set increases the effectiveness of the regression analysis and reduces the approximation error. The system presented in this chapter undergoes one further step: the optimization model is unknown and a surrogate must be built from previously designed solutions.

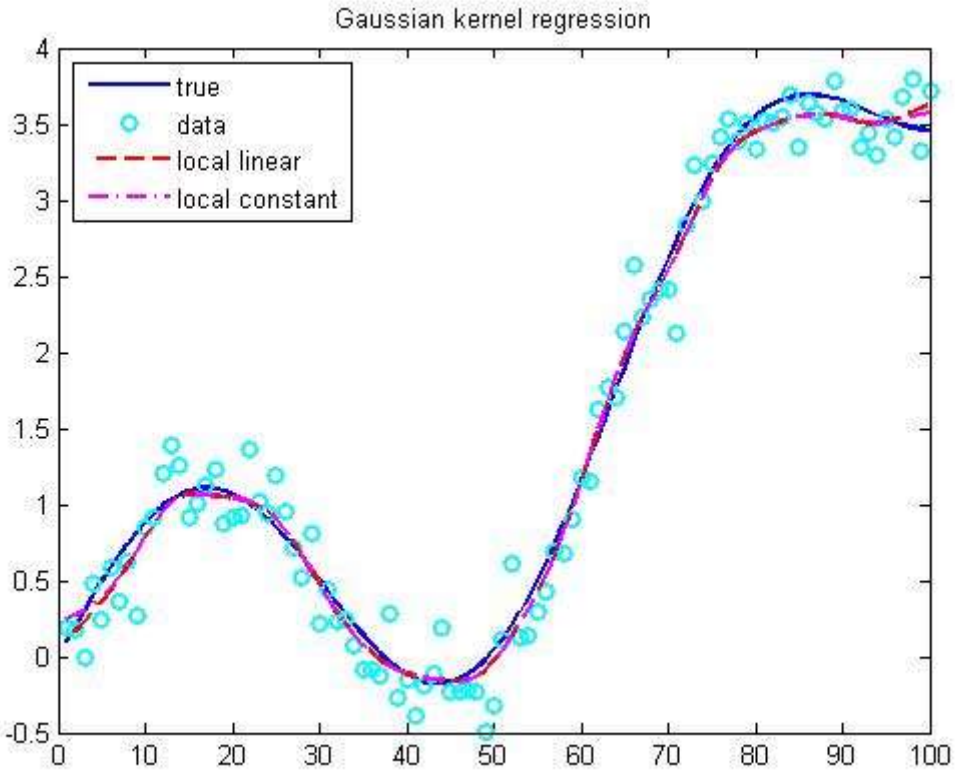

Figure 6.2 - Example of regression analysis of a continuous function.

6.3 Approximated Optimization Model

As anticipated, the goal of the system proposed in the current chapter is to aid the design of the optimization model. For the application of this system, it is assumed that the supply chain has been operational for some time. An automatic optimization system however is not in place. The supply chain has been managed manually, and some solutions or distribution plans exist and are available. For a supply chain optimization problem, the model describes how shipping products on selected routes affects the overall performance of the distribution plan. The model allows the costs of different routes to be compared such that the most inexpensive (according to some performance metric) may be utilized. Manually generated solutions may provide insights on the impact of using some routes. Arguably, if a path from a production source to a shipping port is overwhelmingly selected against many others, it is likely that shipping products on that lane is preferable or less costly, and it would also be part of the optimal solution. The proposed system consists of mining the solutions to discover such insights, if

present. The output is an approximation of the model which is then employed in the optimization step.

The system consists of two main steps:

1. Learning step: the approximated optimization model is defined from a training set of feasible solutions.
2. Optimization step: the approximated model is used to aid the optimization.

Figure 6.3 depicts a simple graphical representation of the system. The inputs of the first step are a basic definition of the supply network, and a set of feasible solutions with their respective evaluation. The supply network is constructed from the ontology defined by Cope [140]. The network consists of a set of suppliers, connected to distribution centers and customers' locations. The output of the first step is the function h as defined in the section 6.2 above (see equation (6.1)). Equations (6.6)-(6.9) define the new optimization model with the approximated objective function.

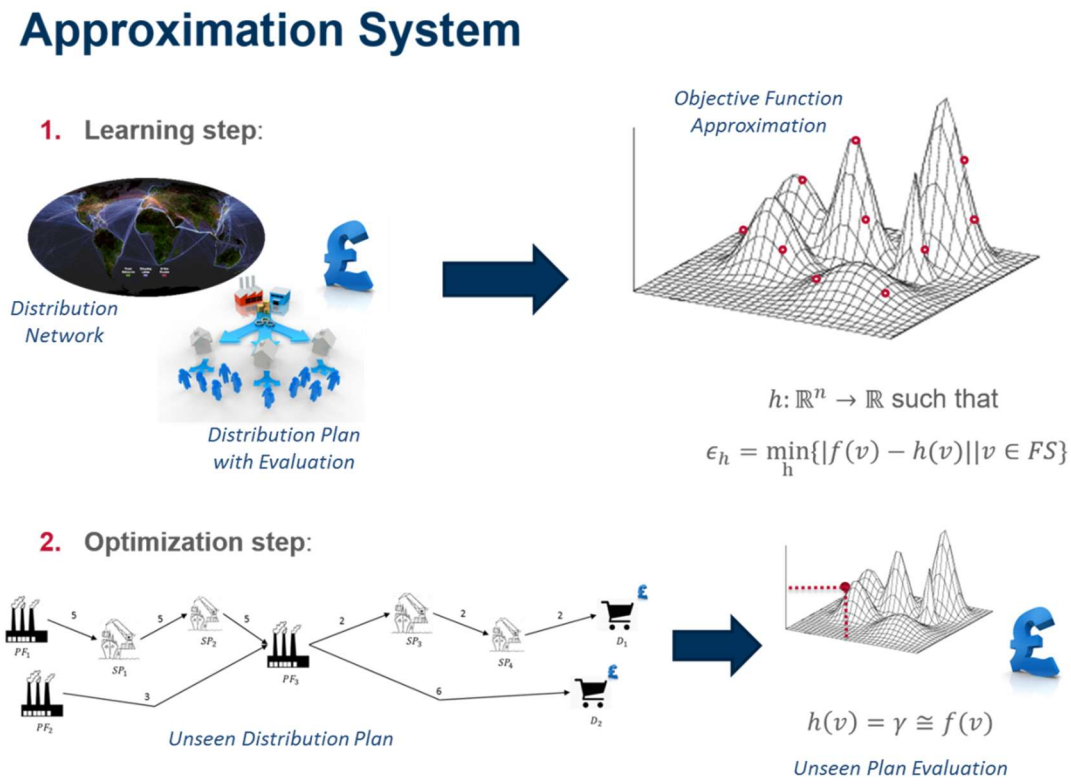

Figure 6.3 - Graphical representation of the system for the approximation of the optimization model starting from manually generated distribution plans.

$$min\, h(x) \qquad \text{Objective function} \qquad (6.6)$$

$$s.t.: \ \sum_{j=1}^{n} x_{ij} \leq S_i \qquad \forall i \in [0, m]. \text{ Capacity constraints} \qquad (6.7)$$

$$\sum_{i=1}^{n} x_{ij} = D_j \qquad \forall j \in [0, n]. \text{ Demand constraints} \qquad (6.8)$$

$$x \in \mathbb{N}_{+}^{n \times m},\ c \in \mathbb{R}_{+}^{n \times m},\ S \in \mathbb{R}_{+}^{m},\ D \in \mathbb{R}_{+}^{n} \qquad \text{Domains} \qquad (6.9)$$

The input to the second step is the generated model and a possible solution built by the optimization algorithm. The optimization algorithm would then evaluate the solution with the generated model.

If a partial definition of the optimization model is available, the proposed system may also be used to increase the accuracy of the optimization. The objective function would combine the information gathered from the manually generated solutions and the partially defined model. In the context of a global supply chain, it is reasonable to assume that basic transportation information is known. The transportation costs between nodes of the network are strictly related to the distance between them, and a function of it can be used as a reasonably good approximation. Moreover, companies undoubtedly have great control and awareness of their production costs and sale prices. With such information, defining a model using equations (2.1) to (2.4) is relatively effortless. Equation (6.10) depicts the objective function in such a scenario, where $x_{ij} \in \mathbb{N}^{+}$ is the route from production source i to dealership j, $tc_{ij} \in \mathbb{R}^{+}$ is the approximated transportation cost, $pc_i \in \mathbb{R}^{+}$ is the production cost, and $sp_j \in \mathbb{R}^{+}$ is the sale price.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} (sp_j - (pc_i + tc_{ij}))\ \cdot x_{ij}, \qquad (6.10)$$

The model learnt from the actual solutions provides the missing information (e.g. inventory costs, international factors, etc.). Section 6.4 presents the results of numerical experiments carried out on a problem of profit maximization. Two sets of experiments have been conducted. In the first set, the only information used for the optimization is that gathered from the solutions in the training set. The second set uses a simple approximation of transportation costs, and actual production costs and sale prices. The

basic optimization model from equation (6.10) is employed to add more solutions to the training set. Additional solutions are generated through the Monte Carlo Sampling (MCS) strategy.

The main limitations of the system are in the diversity of the solutions in the training set and the quality of the regression model. If h does not accurately represent the search space, the solution produced by the optimization process are likely to be of low quality. However, the system is not intended to completely replace the use of a manually designed optimization model. The purpose is to reduce the time required to produce an exact optimization model, or possibly demonstrate the ability of an optimization algorithm on a specific problem without the need for the step of first defining the optimization model.

6.4 Numerical Experiments

The proposed system has been tested on the problem of profit maximization as described in section 3.2 and on two sets of experiments. The remaining optimization problems addressed in this work are not considered, such as transportation time/cost minimization, and resilience maximization. If basic transportation information (i.e. transportation distances/time, production costs and sale prices) are available, then designing the optimization models should be quite effortless. The models defined by the equations (3.1)-(3.4) and (3.11)-(3.14) are not very different from the standard supply chain optimization model in equations (2.1)-(2.4). Collecting accurate transportation information would be most effective way to improve the accuracy of the model.

For the first set of experiments, the model used is only generated from the application of regression analysis to the training set of historical solutions. No additional information about the supply chain is considered. The model is as defined by equations (6.6)-(6.9). In the second set, the training set is extended with additional distribution plans. The distribution plans have been generated using a Monte Carlo Sampling (MCS) strategy. The function used to evaluate and classify the new solutions is defined in equation (6.10).

The datasets used are again the one provided by Caterpillar and the two randomly generated ones. They are published in [4], [5], and [6]. These datasets have demand

values for one year only. The training set for the regression analysis is made up of solutions for 8 months of the year (i.e. 66% of the 12 months). In order to increase the size of the training set, a new dataset has been generated with 10 times as many months. The dataset may be interpreted as covering the last ten years or as ten different products. This dataset has been publish on figshare in [7].

For the regression analysis, it has been adopted the machine learning framework Weka [136]. Weka provides several different regression algorithm implementations. The tested regression algorithms are as follows:

- Linear Regression (LR).
- Support Vector Machine with linear kernel (SVR).
- Support Vector Machine with polynomial kernel of degree 2 (SVR 2).
- Support Vector Machine with polynomial kernel of degree 3 (SVR 3).
- Support Vector Machine with a Radial Basis Function kernel (SVR RBF).
- Gaussian Process with linear kernel (GP).
- Gaussian Process with polynomial kernel of degree 2 (GP 2).
- Gaussian Process with polynomial kernel of degree 3 (GP 3).
- Gaussian Process with a Radial Basis Function kernel (GP RBF).

The performance of such regression algorithms has been assessed through a 10-fold cross validation scheme.

Table 6.1 summarizes the accuracy of the regression analysis. The regression algorithm used to create the approximated model is the Gaussian Process with the Radial Basis Function (RBF) kernel as designed by B. Liu et al. [141]. Only one algorithm has been selected to reduce the overhead on the runtime requirements of the optimization as low as possible.

Table 6.1 – Regression analysis accuracy. The regression algorithms are Linear Regression (LR), Support Vector Machine for Regression with linear kernel (SVR), with polynomial kernel of degree 2 (SVR 2) and 3 (SVR) and a Radial Basis Function kernel (SVR RBF), and a Gaussian Process with linear kernel (GP), polynomial kernel of degree 2 (GP 2) and 3 (GP 3), and a Radial Basis Function kernel (GP RBF). Each measurement is the result of a 10-fold cross validation on the training set.

	LR (%)	SVR (%)	SVR 2 (%)	SVR 3 (%)	SVR RBF (%)	GP (%)	GP 2 (%)	GP 3 (%)	GP RBF (%)
Dataset [4]	97.94%	97.70%	95.82%	92.93%	94.74%	93.23%	94.16%	93.37%	97.96%
Dataset [5]	97.19%	98.25%	97.23%	95.66%	94.38%	92.26%	94.43%	95.09%	98.26%
Dataset [6]	97.98%	98.16%	97.20%	89.68%	94.85%	93.15%	95.60%	95.55%	98.38%
Dataset [7]	98.41%	98.86%	98.84%	98.74%	94.87%	98.66%	98.70%	98.50%	98.83%

The optimization algorithm employed is the Ant Colony System as defined in section 3.5. The configuration used is the same as discussed in section 3.5 with the only exception being the number of iterations, which was increased to five thousand. More iterations during the colony search are allowed and a larger number of random moves are granted. Since the available information is less accurate, the optimization process should be allowed to explore a larger portion of the search space.

Table 6.2 reports the results of the optimization performed on the four datasets. The first row for each dataset named MOD-CAT corresponds to the optimization of Caterpillar's models as defined in chapter 3. MOD-70 refers to the simplified model for the profit maximization problem as defined by equation (6.10). MOD-61 stands for the model generated by the application of the regression analysis only on the training set as defined by equation (6.1). MOD-70-EXT refers to the experiments with the training set extended through sampling. Despite the regression being accurate, the solutions of the optimization from models MOD-61 and MOD-70-EXT are of lower quality than the original optimization. Moreover, the solutions appear to be close to the ones produce by the base line MOD-70. These results are currently being investigated. As future work, the objective is to improve the solutions produced by MOD-61 and MOD-70-EXT. for The Δ difference in Table 6.2 both the optimization performance and the runtime is calculated as $\left(\frac{|x_a - x_e|}{x_e}\right) * 100$, where x_e is the expected value (the result of the original

optimization) and x_a is the actual one (the result of the proposed algorithm). Figure 6.4 shows the results of the regression analysis for the dataset in [4]. The regression algorithm used is the Gaussian Process with a Radial Basis Function as kernel. Figure 6.5 shows the results of the regression on the extended training set created in MOD-70-EXT. Figure 6.6 shows the results for the regression analysis for the dataset in [7].

Table 6.2 – Results of the optimization on the four problems. The first row for each dataset named MOD-CAT corresponds to the optimization of the Caterpillar's models as defined in chapter 3. MOD-70 refers to the simplified model for the profit maximization problem defined by equation (6.10). MOD-61 stands for the model generated by the only application of the regression analysis on the training set defined by equation (6.1). MOD-70-EXT refers to the experiments with the training set extended through sampling. The units for "Optimization Performance" column depend on the objective type.

		Optimization Performance	Δ Opt. Performance (%)	Runtime (s)	Δ runtime (%)
Dataset [4]	**MOD-CAT**	101,247,174.47	-	59.00	-
	MOD-70	82,182,432.18	18.830%	62	5.09%
	MOD-61	83,077,978.53	17.945%	1672	2733.90%
	MOD-70-EXT	83,945,386.77	17.089%	1838	3015.25%
Dataset [5]	**MOD-CAT**	147,373,460.30	-	81.5	-
	MOD-70	108,276,791.06	26.529%	66	19.02%
	MOD-61	112,098,498.83	23.936%	1681	1962.58%
	MOD-70-EXT	116,967,007.75	20.632%	1835	2151.53%
Dataset [6]	**MOD-CAT**	861,800,648.54	-	104.5	-
	MOD-70	582,936,900.88	32.358%	98	6.22%
	MOD-61	541,216,236.52	37.199%	2064	1875.12%
	MOD-70-EXT	574,853,463.70	33.296%	2283	2084.69%
Dataset [7]	**MOD-CAT**	1,867,248,467.67	-	3458	-
	MOD-70	1,031,973,384.32	44.733%	2394	30.77%
	MOD-61	961,549,461.85	48.505%	18652	439.39%
	MOD-70-EXT	961,892,766.28	48.486%	21520	522.33%

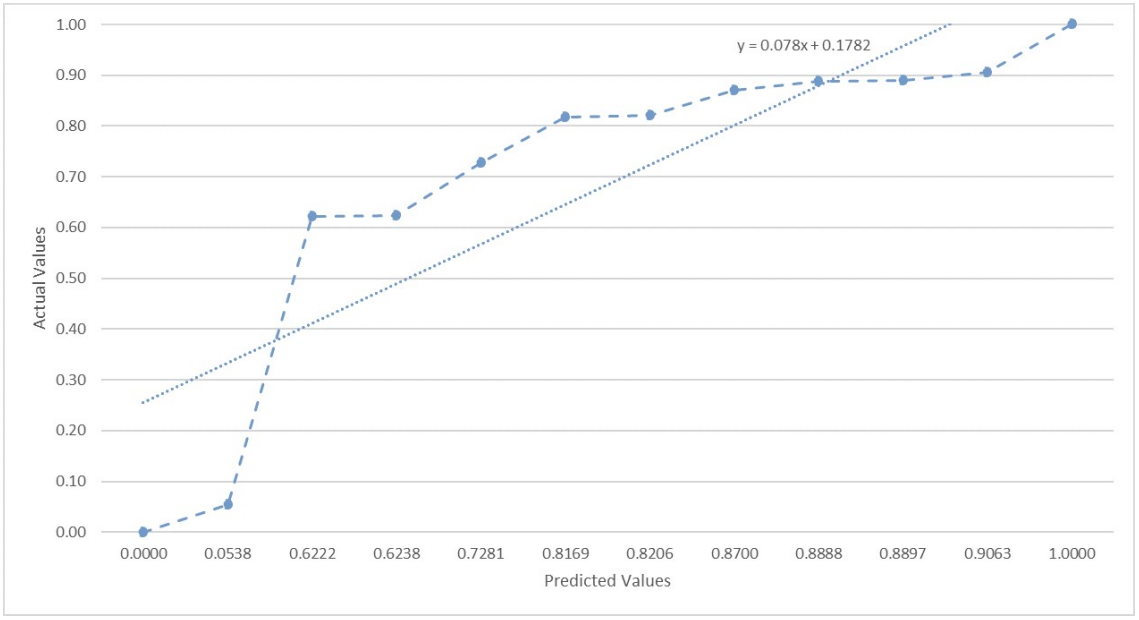

Figure 6.4 – Results of the regression on the dataset in [4]. The regression algorithm used to make the prediction is the Gaussian Process with a Radial Basis Function kernel.

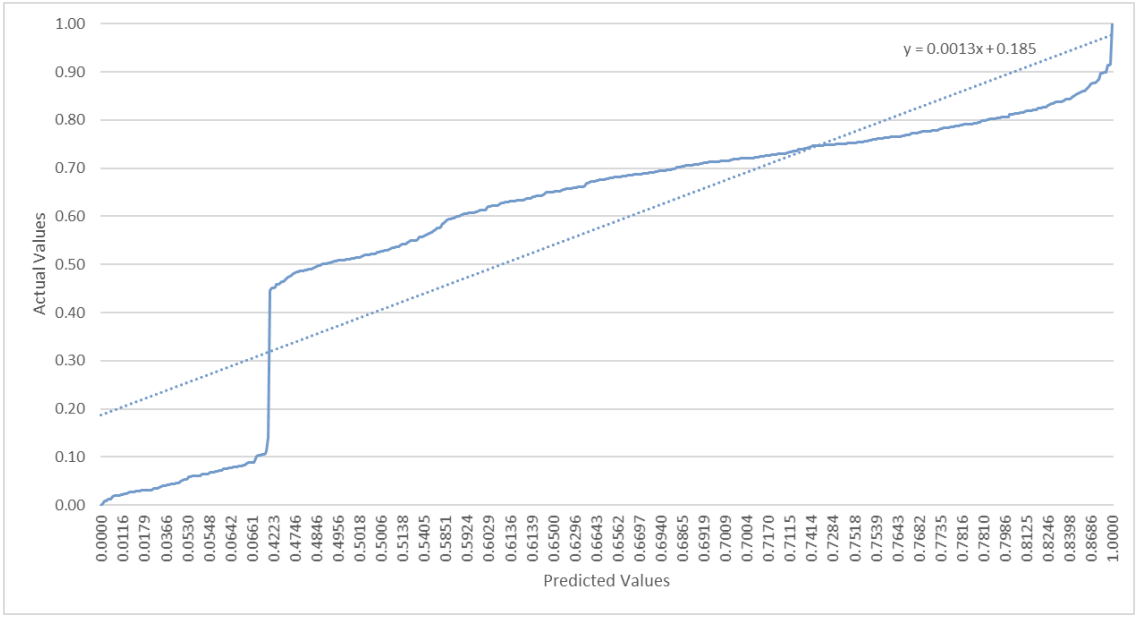

Figure 6.5 - Results of the regression on the dataset in [4]. The regression algorithm used to make the prediction is the Gaussian Process with a Radial Basis Function kernel. The training and test sets have been extended with the sampling of solutions generated with a basic supply chain optimization model. The sampling was performed according to Monte Carlo sampling strategy.

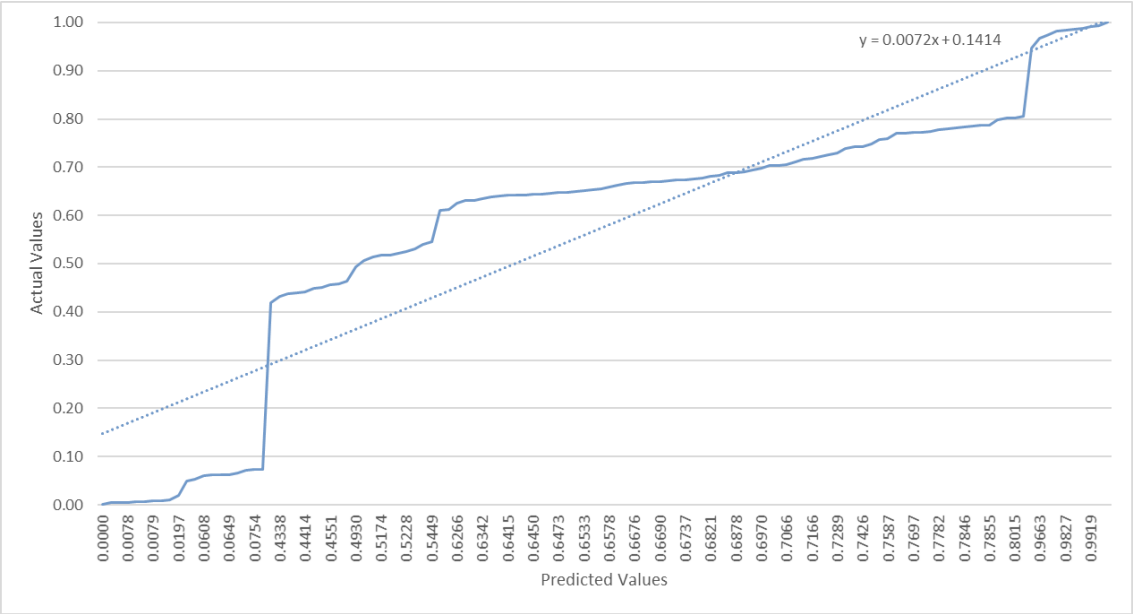

Figure 6.6 - Results of the regression on the dataset in [7]. The regression algorithm used to make the prediction is the Gaussian Process with a Radial Basis Function kernel.

Proving the effectiveness of the proposed system in reducing the effort of designing an optimization model is not trivial.

Table 6.3 reports the resources required in developing the optimization system for Caterpillar. The requirements are measure in term of personnel involved, development time and man-week. The values are compared with the resources required in implementing the approach described in this chapter. Such data are based on the current project experience and might vary in other context. It is estimated that the application of the proposed algorithm could have saved almost 1,000 man-weeks to complete the project.

Table 6.3 – Report of the resources required to develop an optimization process. The requirements are measured in term of development time and personnel involved. The data for the regular optimization are from the current project experience of developing the system for Caterpillar. The data for the proposed system are from the development of the algorithm discussed in this chapter.

Task	Development time	Personnel involved	Man-week	Δ man-week
Regular Optimization				
Data collection	6 months	2	52 weeks	-
Model definition	1.5 years	10+	780 weeks	-
Algorithm implementation	1 years	3	156 weeks	-
Proposed System				
Data collection	2 weeks	2	4 weeks	48 weeks
Training set collection	2 weeks	2	4 weeks	776 weeks
Algorithm implementation	3 months	1	12 weeks	144 weeks
Overall improvement	**96 fewer weeks**	**10 fewer people**	**968 fewer man-weeks**	

From the description above, it is possible however to imagine a general-purpose software system capable of taking as input a set of tables describing distribution plans as a training set, and the basic production information (i.e. sale prices and production costs). The software would implement the regression analysis, a general supply chain optimization model, and the Ant Colony System. Given a new set of demand, the software would be able to find a high quality distribution plan in a matter of minutes. Such software is independent of the supply chain analysed and, therefore, its application should be effortless and efficient.

6.5 Summary

The aim of this work was to reduce the effort of designing an optimization model. While developing the optimization system for Caterpillar's Global Supply Chain, it appeared abundantly clear that there is a shortage of tools to aid in the design of an optimization model. The task is mostly performed manually and requires a significant involvement of personnel who have at least an operations research and management engineering background. In the last couple of decades, several models for different

types of supply chains have been proposed. Nevertheless, even adapting such models to the specific case at hand requires significant knowledge of the business and substantial development time, effort, and experience.

This chapter proposes a system capable of automatically instantiating a generic supply chain model by "mining" the knowledge in the business data. It is assumed that the supply chain has been operational for some time and it has been managed manually. Regression analysis is used to approximate the objective function of the optimization model. The training data were a set of distribution plans and their performance measurement. The result of the regression analysis is a function from a feasible distribution plan to its performance value, and it may be used to evaluate new unseen distribution plans.

The system has been tested on Caterpillar's Supply Chain optimization problem. The Ant Colony System is used as previously defined to find near-to-optimum solutions. The regression analysis produced accurate objective functions. However, the optimization when using such objective function did not produce solutions of similar quality to the original ones. Generic and easy-to-retrieve information were used to build a basic supply chain optimization model. Such a model sets the baseline. All the solutions found by the proposed system are close to the baseline. Ideally, the system should find solutions of much higher quality. These results suggest that using a generic optimization model might be more effective than the system proposed. The reasons why the optimization step does not perform as expected are being investigated. The high accuracy of the regression analysis would suggest different results. The intent is to analyze the search space defined by the approximated model. It is necessary to perform a more accurate comparison between the original model and the one result of the regression analysis.

As additional future work, it will be investigated the application of this system to different types of optimization problems to gain a better understanding of the effectiveness of the approach. Moreover, it would be important to formalize the kind of information readily available in different companies, data that could be employed to automate the process of model design.

The project industrial partners at Caterpillar expressed an interest in the system proposed. They suggested that such an approach could also be useful to measure the quality of the actual designed model. During the development process, they encountered a problem with the data and the information provided by other teams at Caterpillar. In one test case, the demand figures and sale prices available were significantly skewed from the real ones. They belonged to a different product than the one stated. When the results of the optimization were presented, they did not match the expectation of the team responsible for managing the product. The solutions were actually of low quality. The proposed system could be used to measure the accuracy of the designed model. If the solutions of the optimization are significantly different from the past distribution plans, it could mean the presence of a mistake in either the data or the model definition. As future work, this application will be investigated and a test case will be created.

7 CONCLUSIONS AND FUTURE WORK

The primary aim of this work was to increase the level of automation when developing an optimization system. The project focused specifically on the problem of supply chain optimization. Three key areas have been identified where significant improvement could be made. A set of strategies have been developed to reduce the required expert interaction to complete tasks in such areas. The three main aspects of the problem that have been considered are as follows:

1. Standardization of Multi-Objective Optimization techniques and investigation of the relationship between MOO strategies and problem instances.
2. Improvement of the optimization algorithm usability via the definition of an automatic termination condition tuning strategy.
3. Automatization of the task of optimization model design.

The methods developed to achieve the above mentioned objectives have been adopted to implement an optimization system for the problem of Caterpillar's global supply chain optimization. All the algorithms have been tested on datasets provided by the logistics department of Caterpillar.

The following sections outline the achievements of this work. A distinction is made between scientific results and achievements for the company.

7.1 Scientific Results

Of great interest to both research and the industrial community is multi-objective optimization. Multi-objective optimization allows the implementation of a more significant and realistic analysis. However, most proposed methods are very specific to either the problem at hand or to the optimization algorithm used. As for the first contribution of this thesis, a review of the work done in the area of multi-objective optimization has been presented and four general-purpose strategies identified. Such strategies have been described in great details. The review provides more standardization in the area of multi-objective optimization. Experiments have been run on Caterpillar's dataset and two additional test cases. The discussion of the results outlines insights about the relationship between MOO strategies and problem instances. The analysis has aided in the development of more accurate optimization models. This work will serve as a reference on multi-objective methods for real-world 'industrial' supply chain optimization problems.

A significant limitation of meta-heuristic algorithms such as the Ant Colony System is their large number of parameters. As for the second main contribution of this work, a novel algorithm has been presented to improve the usability of the Ant Colony System when optimizing the problem of supply chain. The proposed algorithm is based on a machine learning approach and consists of learning the behaviour of the optimization process on a set of test instances. The gathered knowledge is then used to set the best termination condition for an unseen instance. This approach was capable of significantly reducing the time required to find high quality solutions. In most experiments, the runtime of the optimization system has been reduced by 60% or more. The algorithm improves the usability of the ACS as well by automatically tuning one of its most important parameters. After a step of training, applying the optimization algorithm to different instances requires less effort as the algorithm will partially adapt to the new problems. This algorithm will serve as an approach to utilize machine learning-based approaches to improve optimization algorithms usability.

Despite the problem of supply chain optimization being well-established, the development of the system for Caterpillar highlighted how much effort is still required to deploy an optimization system on a new instance of the problem. In the final part of

this work, a system is proposed to aid the design of optimization models and therefore reduce the effort and resources required to implement an optimization process. The system is again based on a machine learning approach. The objective is to automatically create good approximation of Caterpillar's model, starting from available information. The system performs quite well, however, it is weaker than the initial analysis would suggest. In particular, the approximated models seem to be very close to the originals. However, when they are employed in the optimization, the solutions found are closer to the baseline rather than the target. As the baseline is from a generic supply chain model, it is reasonable to argue that the contribution of this new system is not meaningful. Nevertheless, the idea is quite promising and it was well received by the industrial partners. There is quite a remarkable potential for improving the automatization in implementing an optimization process. It has been estimated that the application of the proposed algorithm could have saved almost 1,000 man-weeks to complete the project. As the first step to address this challenge, the discrepancy between the regression accuracy and the optimization results is being investigated.

7.2 Industrial Results

A final contribution of this thesis is from an industrial stand point and it is the development of an optimization system for Caterpillar's global supply chain. This thesis has described the process of developing the optimization system and has highlighted the challenges and research opportunities identified while undertaking this work. The most noticeable set of opportunities are concerned with the definition of the optimization models. It has been pointed out how most of the work proposed in this area has limitations and exhibits undesirable trends. Throughout the definition of Caterpillar's model, it has been explained how such limitations have been addressed and overcame. The model for Caterpillar's scenario may well work as a reference for real-world global supply chain optimization.

The designed optimization model has been proven to accurately describe Caterpillar's logistics operation and the optimization algorithm to find high quality distribution plans in a fairly short time. The system has been deployed into Caterpillar's production environment and it is now used to design the distribution plans of more than 7,000

products. Some internal testing has shown that the system improves Caterpillar's marginal profit on such products of a factor by 4.6% on average.

7.3 Future Work

Throughout this work, possible future work and research directions have been suggested and discussed.

Regarding the multi-objective analysis, the bias towards one specific multi-objective method highlighted in the review is probably due to the availability of software. As future work, it would be interesting to extend the literature review into other areas of operational research and to include information relating to the existing optimization tools. Moreover, a theoretical analysis of the characteristics of the presented strategies is desirable. A deeper understanding is likely to expose the reasons why and under what circumstances some strategies should be preferred over others.

The method proposed for improving the computational requirements of the Ant Colony System yielded compelling results. It would be of interest to apply the same principal to the automatic tuning of other parameters. Moreover, it would be useful to investigate more features of the problem with the goal of understanding which factors mostly affect the performance of the optimization process.

Finally, more tools to aid in the design and development of the optimization processes should be proposed. Increasing the standardization and automatization of this task would be beneficial to all. For that purpose, it would be useful to formalize the different scenarios and the information available for the application of the concept of automatic model definition.

8 REFERENCES

[1] A. Ogunbanwo, A. Williamson, M. Veluscek, R. Izsak, T. Kalganova, and P. Broomhead, "Transportation Network Optimization," *Encycl. Bus. Anal. Optim.*, pp. 2570–2583, Feb. 2014.

[2] M. Veluscek, T. Kalganova, P. Broomhead, and A. Grichnik, "Composite goal methods for transportation network optimization," *Expert Syst. Appl.*, Dec. 2014.

[3] M. Veluscek, T. Kalganova, and P. Broomhead, "Improving ant colony optimization performance through prediction of best termination condition," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, 2015, pp. 2394–2402.

[4] M. Veluscek and T. Kalganova, "Caterpillar Global Supply Chain - Logistics Data for a Medium Size Excavator," Nov. 2015.

[5] M. Veluscek and T. Kalganova, "Global Supply Chain - Uniformly Random Logistics Data from a Caterpillar's Test Case," Nov. 2015.

[6] M. Veluscek and T. Kalganova, "Global Supply Chain - Random Logistics Data from a Caterpillar's Test Case," Nov. 2015.

[7] M. Veluscek, "Mining Company's Global Supply Chain - Logistics Data for a Medium Size Excavator - Extended Dataset," 2016.

[8] H. Ding, L. Benyoucef, and X. Xie, "Stochastic multi-objective production-distribution network design using simulation-based optimization," *Int. J. Prod. Res.*, vol. 47, no. 2, pp. 479–505, Jan. 2009.

[9] R. Ganeshan, "Managing supply chain inventories: A multiple retailer, one warehouse, multiple supplier model," *Int. J. Prod. Econ.*, vol. 59, no. 1–3, pp. 341–354, Mar. 1999.

[10] G. M. Magableh, M. D. Rossetti, and S. Mason, "Modeling and analysis of a generic cross-docking facility," in *Proceedings of the 2005 Winter Simulation Conference*, 2005, pp. 1613–1620.

[11] B. M. Beamon, “Supply chain design and analysis: Models and methods,” *Int. J. Prod. Econ.*, vol. 55, no. 3, pp. 281–294, Aug. 1998.

[12] L. Zhang and Y. Zhou, “Study on Distribution Network Optimization of Compound Fertilizer Supply Chain,” in *2009 International Conference on Management and Service Science*, 2009, pp. 1–6.

[13] L. Juntao and L. Bingwu, “Study of Logistics Distribution Network Planning Based on Heuristic Algorithm,” in *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, 2011, vol. 1, pp. 1129–1133.

[14] L. Wen and F. Meng, “Optimization of Logistics Distribution Network Based on Improved Particle Swarm Optimization,” *2008 Fourth Int. Conf. Nat. Comput.*, pp. 659–663, 2008.

[15] Y. Li, Q. Su, and T. Li, “The Optimization and Integration of the Transportation and Inventory Cost Based on Time in Supply Chain Logistics System,” in *2010 International Conference on Logistics Engineering and Intelligent Transportation Systems*, 2010, pp. 1–4.

[16] D. Anghinolfi, M. Paolucci, S. Sacone, and S. Siri, “Integer programming and ant colony optimization for planning intermodal freight transportation operations,” *2011 IEEE Int. Conf. Autom. Sci. Eng.*, pp. 214–219, Aug. 2011.

[17] F. Boudahri, Z. Sari, F. Maliki, and M. Bennekrouf, “Design and optimization of the supply chain of agri-foods: Application distribution network of chicken meat,” *2011 Int. Conf. Commun. Comput. Control Appl.*, pp. 1–6, Mar. 2011.

[18] C. M. Parveen, A. R. P. Kumar, and T. V. V. L. Narasimha Rao, “Integration of lean and green supply chain - Impact on manufacturing firms in improving environmental efficiencies,” in *International Conference on Green technology and environmental Conservation (GTEC-2011)*, 2011, pp. 143–147.

[19] W.-C. Yeh and M.-C. Chuang, “Using multi-objective genetic algorithm for partner selection in green supply chain problems,” *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4244–4253, Apr. 2011.

[20] L. Du, J. Wu, and F. Hu, “Logistics network design and optimization of closed-loop supply chain based on mixed integer nonlinear programming model,” *2009 ISECS Int. Colloq. Comput. Commun. Control. Manag.*, pp. 414–417, Aug. 2009.

[21] T. Aslam and A. H. C. Ng, “Multi-objective optimization for supply chain management: A literature review and new development,” in *Supply Chain Management and Information Systems (SCMIS), 2010 8th International Conference on*, 2010, pp. 1–8.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[23] L. E. Holloway, J. Li, and Y. Hu, “Towards modeling of resilience dynamics in manufacturing enterprises: Literature review and problem formulation,” in *2008 IEEE International Conference on Automation Science and Engineering*, 2008, pp. 279–284.

[24] K. Zhao, A. Kumar, T. P. Harrison, and J. Yen, "Analyzing the Resilience of Complex Supply Network Topologies Against Random and Targeted Disruptions," *IEEE Syst. J.*, vol. 5, no. 1, pp. 28–39, Mar. 2011.

[25] Y. Jiang, L. Zhao, and S. Sun, "A resilient strategy for meat-food supply chain network design," in *2009 IEEE International Conference on Industrial Engineering and Engineering Management*, 2009, pp. 1479–1483.

[26] W. H. Ip, "Evaluation and Analysis of Logistic Network Resilience With Application to Aircraft Servicing," *IEEE Syst. J.*, vol. 3, no. 2, pp. 166–173, Jun. 2009.

[27] G. B. Dantzig, "Linear Programming." p. 201, 1963.

[28] A. Goldberg and R. Tarjan, "Solving minimum-cost flow problems by successive approximation," *Proc. Ninet. Annu. ACM Symp. Theory Comput.*, 1987.

[29] V. Bevilacqua, N. Costantino, M. Dotoli, M. Falagario, F. Sciancalepore, P. Taylor, P. Bari, and V. R. David, "Strategic design and multi-objective optimisation of distribution networks based on genetic algorithms," *Int. J. Comput. Integr. Manuf.*, vol. 25, no. 12, pp. 1139–1150, Dec. 2012.

[30] R. Musa, J.-P. Arnaout, and H. Jung, "Ant colony optimization algorithm to solve for the transportation problem of cross-docking network," *Comput. Ind. Eng.*, vol. 59, no. 1, pp. 85–92, Aug. 2010.

[31] C. J. Vidal and M. Goetschalckx, "Strategic production-distribution models: A critical review with emphasis on global supply chain models," *Eur. J. Oper. Res.*, vol. 98, no. 1, pp. 1–18, Apr. 1997.

[32] H. J. Ko and G. W. Evans, "A genetic algorithm-based heuristic for the dynamic integrated forward/reverse logistics network for 3PLs," *Comput. Oper. Res.*, vol. 34, no. 2, pp. 346–366, Feb. 2007.

[33] L. K. Nozick and M. A. Turnquist, "A two-echelon inventory allocation and distribution center location analysis," *Transp. Res. Part E Logist. Transp. Rev.*, vol. 37, no. 6, pp. 425–441, Dec. 2001.

[34] F. Altiparmak, M. Gen, L. Lin, and T. Paksoy, "A genetic algorithm approach for multi-objective optimization of supply chain networks," *Comput. Ind. Eng.*, vol. 51, no. 1, pp. 196–215, Sep. 2006.

[35] J. Sadeghi, S. Sadeghi, and S. T. A. Niaki, "A hybrid vendor managed inventory and redundancy allocation optimization problem in supply chain management: An NSGA-II with tuned parameters," *Comput. Oper. Res.*, vol. 41, pp. 53–64, Jan. 2014.

[36] M. J. Meixell and V. B. Gargeya, "Global supply chain design: A literature review and critique," *Transp. Res. Part E Logist. Transp. Rev.*, vol. 41, no. 6, pp. 531–550, Nov. 2005.

[37] J. J. Bravo and C. J. Vidal, "Freight transportation function in supply chain optimization models: A critical review of recent trends," *Expert Syst. Appl.*, vol. 40, no. 17, pp. 6742–6757, Dec. 2013.

[38] Q. Xiang, H. Li, B. Huang, and R. Li, "Improved ant colony optimization for

multi-objective route planning of dangerous goods," *2012 8th Int. Conf. Nat. Comput.*, no. Icnc, pp. 772–776, May 2012.

[39] Z. H. Che, "A particle swarm optimization algorithm for solving unbalanced supply chain planning problems," *Appl. Soft Comput.*, vol. 12, no. 4, pp. 1279–1287, Apr. 2012.

[40] H. Sadjady and H. Davoudpour, "Two-echelon, multi-commodity supply chain network design with mode selection, lead-times and inventory costs," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1345–1354, Jul. 2012.

[41] M. Huang, R. Li, and X. Wang, "Network construction for fourth-party logistics based on resilience with using Particle Swarm Optimization," *2011 Chinese Control Decis. Conf.*, pp. 3924–3929, May 2011.

[42] D. N. D. Utama, T. Djatna, E. Hambali, Marimin, and D. Kusdiana, "Multi objectives fuzzy ant colony optimization of palm oil based bioenergy supply path searching," *Adv. Comput. Sci. Inf. Syst. (ICACSIS), 2011 Int. Conf.*, pp. 177–182, 2011.

[43] M. C. Georgiadis, P. Tsiakis, P. Longinidis, and M. K. Sofioglou, "Optimal design of supply chain networks under uncertain transient demand variations," *Omega*, vol. 39, no. 3, pp. 254–272, Jun. 2011.

[44] X. Zhao, J. Dou, and S. Studies, "A hybrid particle swarm optimization approach for design of agri-food supply chain network," *Serv. Oper. Logist. Informatics (SOLI), 2011 IEEE Int. Conf.*, no. Dc, pp. 162–167, 2011.

[45] N. Han and X. Ji, "Optimization of logistics distribution routing problem based on improved ant colony algorithm," *Mach. Learn. Cybern. (ICMLC), 2010 Int. Conf.*, no. July, pp. 11–14, 2010.

[46] J. Chen, J. Lu, and S. Qi, "Transportation network optimization of import crude oil in China based on minimum logistics cost," *Emerg. Manag. Manag. Sci. (ICEMMS), 2010 IEEE Int. Conf.*, pp. 335–338, 2010.

[47] Y.-H. Chang, "Adopting co-evolution and constraint-satisfaction concept on genetic algorithms to solve supply chain network design problems," *Expert Syst. Appl.*, vol. 37, no. 10, pp. 6919–6930, Oct. 2010.

[48] K. Ghoseiri and B. Nadjari, "An ant colony optimization algorithm for the bi-objective shortest path problem," *Appl. Soft Comput.*, vol. 10, no. 4, pp. 1237–1246, Sep. 2010.

[49] Z. H. Che and C. J. Chiang, "A modified Pareto genetic algorithm for multi-objective build-to-order supply chain planning with product assembly," *Adv. Eng. Softw.*, vol. 41, no. 7–8, pp. 1011–1022, Jul. 2010.

[50] H. S. Wang, "A two-phase ant colony algorithm for multi-echelon defective supply chain network design," *Eur. J. Oper. Res.*, vol. 192, no. 1, pp. 243–252, Jan. 2009.

[51] L. Lin, M. Gen, and X. Wang, "Integrated multistage logistics network design by using hybrid evolutionary algorithm," *Comput. Ind. Eng.*, vol. 56, no. 3, pp. 854–873, Apr. 2009.

[52] F. T. S. Chan and N. Kumar, "Effective allocation of customers to distribution

centres: A multiple ant colony optimization approach," *Robot. Comput. Integr. Manuf.*, vol. 25, no. 1, pp. 1–12, Feb. 2009.

[53] H. Mohammadi Bidhandi, R. Mohd. Yusuff, M. M. H. Megat Ahmad, and M. R. Abu Bakar, "Development of a new approach for deterministic supply chain network design," *Eur. J. Oper. Res.*, vol. 198, no. 1, pp. 121–128, Oct. 2009.

[54] H. C. W. Lau, T. M. Chan, W. T. Tsui, and G. T. S. Ho, "Cost Optimization of the Supply Chain Network Using Genetic Algorithms," *IEEE Trans. Knowl. Data Eng.*, no. c, pp. 1–36, 2009.

[55] Z. Lin and L. Wang, "Multi-Stage Partner Selection Based on Genetic-Ant Colony Algorithm in Agile Supply Chain Network," *2008 9th Int. Conf. Young Comput. Sci.*, pp. 1884–1889, Nov. 2008.

[56] X. Qin and X. Liu, "The integrated distribution network design problem with transportation economies-of-scale: Model and algorithm," *Serv. Oper. Logist. Informatics, 2008. IEEE/SOLI 2008. IEEE Int. Conf.*, no. 70702038, pp. 1454–1458, 2008.

[57] Y. Huang, Z. Qiu, and Q. Liu, "Supply chain network design based on fuzzy neural network and PSO," *Mach. Learn. Cybern. (ICMLC), 2010 Int. Conf.*, no. September, pp. 2189–2193, 2008.

[58] R. Z. Farahani and M. Elahipanah, "A genetic algorithm to optimize the total cost and service level for just-in-time distribution in a supply chain," *Int. J. Prod. Econ.*, vol. 111, no. 2, pp. 229–243, Feb. 2008.

[59] J. J. L. J. Caldeira, R. R. C. Azevedo, C. A. Silva, and J. M. C. Sousa, "Supply-chain management using ACO and beam-ACO algorithms," *Fuzzy Syst. Conf. 2007. FUZZ-IEEE 2007. IEEE Int. IEEE, 2007.*, pp. 1–6, Jun. 2007.

[60] B. Fu, X. Song, Z. Guo, and P. Zhang, "An optimization model for container transportation network with ACO approach," *Evol. Comput. 2007. CEC 2007. IEEE Congr.*, pp. 4768–4775, 2007.

[61] C.-L. Chen, T.-W. Yuan, and W.-C. Lee, "Multi-criteria fuzzy optimization for locating warehouses and distribution centers in a supply chain network," *J. Chinese Inst. Chem. Eng.*, vol. 38, no. 5–6, pp. 393–407, Sep. 2007.

[62] H. Ding, L. Benyoucef, and X. Xie, "A simulation-based optimization method for production-distribution network design," *Syst. Man Cybern. 2004 IEEE Int. Conf.*, pp. 4521–4526, 2004.

[63] Z. Ji, A. Chen, and K. Subprasom, "Finding multi-objective paths in stochastic networks: a simulation-based genetic algorithm approach," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, 2004, vol. 1, pp. 174–180.

[64] B. S. Pimentel, G. R. Mateus, and F. A. Almeida, "Stochastic capacity planning and dynamic network design," *Int. J. Prod. Econ.*, vol. 145, no. 1, pp. 139–149, Sep. 2013.

[65] B. Yu, "Optimizing Bus Transit Netwrok with Parallel Ant Colony Algorithm," *Proc. East. Asia Soc. Transp. Stud.*, vol. 5, pp. 374–389, 2005.

[66] L. Wang, S. K. Kowk, and W. H. Ip, “Design of an improved quantum-inspired evolutionary algorithm for a transportation problem in logistics systems,” *J. Intell. Manuf.*, vol. 23, no. 6, pp. 2227–2236, Jul. 2011.

[67] M. Hosseinzadeh and A. Branch, “An Optimization Model for Reverse Logistics Network under Stochastic Environment Using Genetic Algorithm,” *Int. J. Bus. Soc. Sci.*, vol. 3, no. 12, pp. 249–264, 2012.

[68] J.-F. Cordeau, F. Pasin, and M. M. Solomon, “An integrated model for logistics network design,” *Ann. Oper. Res.*, vol. 144, no. 1, pp. 59–82, May 2006.

[69] IBM, “CPLEX Optimizer.” [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[70] L. S. Inc., “LINGO.” [Online]. Available: http://www.lindo.com/index.php?option=com_content&view=article&id=16:new-features-for-lingo-110&catid=4:lingo.

[71] GNU, “GLPK (GNU Linear Programming Kit).” [Online]. Available: https://www.gnu.org/software/glpk/.

[72] M. T. Melo, S. Nickel, and F. Saldanha-da-Gama, “Facility location and supply chain management – A review,” *Eur. J. Oper. Res.*, vol. 196, no. 2, pp. 401–412, Jul. 2009.

[73] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization,” *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[74] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, “Hybrid metaheuristics in combinatorial optimization: A survey,” *Appl. Soft Comput.*, vol. 11, no. 6, pp. 4135–4151, Sep. 2011.

[75] M. S. Alam, M. M. Islam, X. Yao, and K. Murase, “Diversity Guided Evolutionary Programming: A novel approach for continuous optimization,” *Appl. Soft Comput.*, vol. 12, no. 6, pp. 1693–1707, Jun. 2012.

[76] M. Dorigo, V. Maniezzo, and a Colorni, “Ant system: optimization by a colony of cooperating agents.,” *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 26, no. 1, pp. 29–41, Jan. 1996.

[77] Y. Dong, J. Gu, and N. Li, “Combination of genetic algorithm and ant colony algorithm for distribution network planning,” *Proc. Sixth Int. Conf. Mach. Learn. Cybern. Hong Kong*, no. August, pp. 19–22, 2007.

[78] M. Dorigo and L. 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.

[79] T. Stutzle and H. Hoos, “MAX-MIN Ant System and local search for the traveling salesman problem,” *Proc. 1997 IEEE Int. Conf. Evol. Comput. (ICEC ’97)*, pp. 309–314, 1997.

[80] R. F. H. C. S. Bernd Bullnheimer, “A New Rank Based Version of the Ant System - A Computational Study.”

[81] V. Maniezzo and A. Colorni, “The ant system applied to the quadratic assignment problem,” *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 5, pp. 769–778, 1999.

[82] F. Zhao, "An improved ant colony optimization algorithm with embedded genetic algorithm for the traveling salesman problem," *2008 7th World Congr. Intell. Control Autom.*, pp. 7902–7906, 2008.

[83] C. Blum, "Ant colony optimization: Introduction and recent trends," *Phys. Life Rev.*, vol. 2, no. 4, pp. 353–373, Dec. 2005.

[84] R. J. Balling, J. T. Taber, M. R. Brown, and K. Day, "Multiobjective Urban Planning Using Genetic Algorithm," *J. Urban Plan. Dev.*, vol. 125, no. 2, pp. 1–61, Jun. 1999.

[85] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proc. ICNN'95 - Int. Conf. Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[86] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, Jul. 2013.

[87] E. Burke, G. Kendall, J. Newall, and E. Hart, "Hyper-heuristics: An emerging direction in modern search technology," *Handb. Metaheuristics*, 2003.

[88] E. K. E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," *Handb. Metaheuristics*, pp. 1–21, 2010.

[89] Y. Zheng, M.-X. Zhang, H. Ling, S.-Y. Chen, and S. Member, "Emergency railway transportation planning using a hyper-heuristic appraoch," *Ieee Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 321–329, Feb. 2015.

[90] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stutzle, "Automated Algorithm Tuning using F-races: Recent Developments," in *MIC 2009: The VIII Metaheuristics International Conference*, 2009, pp. 1–10.

[91] M. Birattari, "The Problem of Tuning Metaheuristics as Seen from a Machine Learning Approach," Universite Libre de Bruxelles, 2004.

[92] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[93] S. Terzi and S. Cavalieri, "Simulation in the supply chain context: a survey," *Comput. Ind.*, vol. 53, no. 1, pp. 3–16, Jan. 2004.

[94] L. a. Deleris and F. Erhun, "Risk management in supply networks using Monte-Carlo simulation," *Proc. Winter Simul. Conf. 2005.*, pp. 1643–1649, 2005.

[95] A. J. Schmitt and M. Singh, "Quantifying supply chain disruption risk using Monte Carlo and discrete-event simulation," pp. 1237–1248, Dec. 2009.

[96] S. Korukoğlu and S. Ballı2, "An Improved Vogel's Approximation Method For The Transportation Problem," *Math. Comput. Appl.*, vol. 16, no. 2, pp. 370–381, 2011.

[97] A. E. Samuel and M. Venkatachalapathy, "Modified Vogel ' s Approximation Method for Fuzzy Transportation Problems," vol. 5, no. 28, pp. 1367–1372, 2011.

[98] M. Pedemonte, S. Nesmachnow, and H. Cancela, "A survey on parallel ant

colony optimization," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5181–5197, Dec. 2011.

[99] O. A. R. B. (OpenMP ARB), "OpenMP." [Online]. Available: http://openmp.org/wp/.

[100] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliab. Eng. Syst. Saf.*, vol. 91, no. 9, pp. 992–1007, Sep. 2006.

[101] I. Alaya, C. Solnon, and K. Ghedira, "Ant Colony Optimization for Multi-Objective Optimization Problems," in *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, 2007, vol. 1, pp. 450–457.

[102] B. Yagmahan and M. M. Yenisey, "Ant colony optimization for multi-objective flow shop scheduling problem," *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 411–420, 2008.

[103] I. Das and J. E. Dennis, "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, Aug. 1998.

[104] C. García-Martínez, O. Cordón, and F. Herrera, "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP," *Eur. J. Oper. Res.*, vol. 180, no. 1, pp. 116–148, Jul. 2007.

[105] Y. Cardona-Valdés, A. Álvarez, and D. Ozdemir, "A bi-objective supply chain design problem with uncertainty," *Transp. Res. Part C Emerg. Technol.*, vol. 19, no. 5, pp. 821–832, Aug. 2011.

[106] A. Cintron, A. R. Ravindran, and J. A. Ventura, "Multi-criteria mathematical model for designing the distribution network of a consumer goods company," *Comput. Ind. Eng.*, vol. 58, no. 4, pp. 584–593, May 2010.

[107] A. Kamali, S. M. T. Fatemi Ghomi, and F. Jolai, "A multi-objective quantity discount and joint optimization model for coordination of a single-buyer multi-vendor supply chain," *Comput. Math. with Appl.*, vol. 62, no. 8, pp. 3251–3269, Oct. 2011.

[108] T.-F. Liang, "Fuzzy multi-objective production/distribution planning decisions with multi-product and multi-time period in a supply chain," *Comput. Ind. Eng.*, vol. 55, no. 3, pp. 676–694, Oct. 2008.

[109] C.-L. Chen and W.-C. Lee, "Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices," *Comput. Chem. Eng.*, vol. 28, no. 6–7, pp. 1131–1144, Jun. 2004.

[110] E. H. Sabri and B. M. Beamon, "A multi-objective approach to simultaneous strategic and operational planning in supply chain design," *Omega*, vol. 28, no. 5, pp. 581–598, Oct. 2000.

[111] J. A. Joines, R. E. King, M. G. Kay, and D. Gupta, "Supply chain multi-objective simulation optimization," in *Proceedings of the Winter Simulation Conference*, 2002, vol. 2, pp. 1306–1314.

[112] F. Wang, X. Lai, and N. Shi, "A multi-objective optimization for green supply chain network design," *Decis. Support Syst.*, vol. 51, no. 2, pp. 262–269, May

2011.

[113] S. A. Torabi and E. Hassini, "An interactive possibilistic programming approach for multiple objective supply chain master planning," *Fuzzy Sets Syst.*, vol. 159, no. 2, pp. 193–214, Jan. 2008.

[114] A. Amid, S. H. Ghodsypour, and C. O'Brien, "A weighted max–min model for fuzzy multi-objective supplier selection in a supply chain," *Int. J. Prod. Econ.*, vol. 131, no. 1, pp. 139–145, May 2011.

[115] G. Wang, S. H. Huang, and J. P. Dismukes, "Product-driven supply chain selection using integrated multi-criteria decision-making methodology," *Int. J. Prod. Econ.*, vol. 91, no. 1, pp. 1–15, Sep. 2004.

[116] C. A. Weber and J. R. Current, "A multiobjective approach to vendor selection," *Eur. J. Oper. Res.*, vol. 68, pp. 173–184, 1993.

[117] J. Liu, F. Ding, and V. Lall, "Using data envelopment analysis to compare suppliers for supplier selection and performance improvement," *Supply Chain Manag. An Int. J.*, vol. 5, no. 3, pp. 143–150, 2000.

[118] M. Kumar, P. Vrat, and R. Shankar, "A fuzzy goal programming approach for vendor selection problem in a supply chain," *Comput. Ind. Eng.*, vol. 46, no. 1, pp. 69–85, Mar. 2004.

[119] S. C. H. Leung, S. O. S. Tsang, W. L. Ng, and Y. Wu, "A robust optimization model for multi-site production planning problem in an uncertain environment," *Eur. J. Oper. Res.*, vol. 181, no. 1, pp. 224–238, Aug. 2007.

[120] A. R. Yıldız, "An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry," *J. Mater. Process. Technol.*, vol. 209, no. 6, pp. 2773–2780, Mar. 2009.

[121] S. K. Chaharsooghi and A. H. Meimand Kermani, "An effective ant colony optimization algorithm (ACO) for multi-objective resource allocation problem (MORAP)," *Appl. Math. Comput.*, vol. 200, no. 1, pp. 167–177, Jun. 2008.

[122] P. R. McMullen and P. Tarasewich, "Multi-objective assembly line balancing via a modified ant colony optimization technique," *Int. J. Prod. Res.*, vol. 44, no. 1, pp. 27–42, Jan. 2006.

[123] G. Taguchi, E. A. Elsayed, and T. C. Hsiang, *Quality Engineering in Production Systems (Mcgraw Hill Series in Industrial Engineering and Management Science)*. Mcgraw-Hill College, 1988.

[124] A. Arisha and W. Abo-Hamad, "Optimisation Methods in Supply Chain Applications: a Review," in *12th Annual Conference of the Irish Academy of Management, Galway Mayo Institute of Technology*, 2009.

[125] S.-P. Tseng, C.-W. Tsai, M.-C. Chiang, and C.-S. Yang, "A fast Ant Colony Optimization for traveling salesman problem," *IEEE Congr. Evol. Comput.*, pp. 1–6, Jul. 2010.

[126] Q. Lv and X. Xia, "Towards Termination Criteria of Ant Colony Optimization," in *Third International Conference on Natural Computation (ICNC 2007) Vol V*, 2007, vol. 5, pp. 276–282.

[127] Z. Zhang, Z. Feng, and Z. Ren, “Approximate termination condition analysis for ant colony optimization algorithm,” in *2010 8th World Congress on Intelligent Control and Automation*, 2010, no. 60875043, pp. 3211–3215.

[128] E. Stomeo, T. Kalganova, and C. Lambert, “Generalized disjunction decomposition for evolvable hardware.,” *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 36, no. 5, pp. 1024–43, Oct. 2006.

[129] K. A. Smith-miles, R. J. W. James, J. W. Giffin, and Y. Tu, “Understanding the Relationship between Scheduling Problem Structure and Heuristic Performance using Knowledge Discovery,” in *Learning and Intelligent Optimization*, 2009.

[130] K. Smith-miles, J. Van Hemert, and X. Y. Lim, “Understanding TSP Difficulty by Learning from Evolved Instances,” in *Learning and Intelligent Optimization*, 2010, pp. 266–280.

[131] S. Wright, *The roles of mutation, inbreeding, crossbreeding, and selection in evolution*. 1932.

[132] E. Pitzer, M. Affenzeller, A. Beham, and S. Wagner, “Comprehensive and Automatic Fitness Landscape Analysis Using HeuristicLab,” in *Computer Aided Systems Theory – EUROCAST 2011*, vol. 6927, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 424–431.

[133] B. Freisleben and P. Merz, “Fitness landscape analysis and memetic algorithms for the quadratic assignment problem,” *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 337–352, 2000.

[134] J. Humeau, A. Liefooghe, E.-G. Talbi, and S. Verel, “ParadisEO-MO: from fitness landscape analysis to efficient local search algorithms,” *J. Heuristics*, vol. 19, no. 6, pp. 881–915, Jun. 2013.

[135] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967, vol. 233, no. 233, pp. 281–297.

[136] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update.” 2009.

[137] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining Practical Machine Learning Tools and Techniques*, 3rd ed., vol. 40, no. 6. Elsevier Inc., 2001.

[138] R. C. Holte, “Very simple classification rules perform well on most commonly used datasets.,” *Mach. Learn.*, vol. 11, pp. 63–91, 1993.

[139] J. Platt, “Fast Training of Support Vector Machines using Sequential Minimal Optimization,” in *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998.

[140] D. Cope and M. S. Fayez, “Supply chain simulation modeling made easy: An innovative approach,” in *2007 Winter Simulation Conference*, 2007, pp. 1887–1896.

[141] B. Liu, Q. Zhang, and G. G. E. Gielen, “A Gaussian Process Surrogate Model Assisted Evolutionary Algorithm for Medium Scale Expensive Optimization

Problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 180–192, Apr. 2014.

[142] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, 2005, pp. 68–75.

[143] M. Locatelli, "A Note on the Griewank Test Function," *J. Glob. Optim.*, vol. 25, no. 2, pp. 169–174, 2003.

9 APPENDICES

APPENDIX A: CATERPILLAR'S DATASET

The following sections provide a preview of the datasets published in [4], [5], [6], and [7]. The main information of the datasets may be divided in the following sections:

1. Demand figures and sales prices.
2. Machine production details.
3. Components production details.
4. Transportation times and costs.

Each section shows the main tables with their respective headers and briefly describes the type of information contain in them. The actual data published is formatted as XML files. However, in order to improve visualization and readability, the data are visualized as regular tables.

The following data are reported here for visualization purposes only and may vary from the original datasets. For the actual data please see [4], [5], [6], and [7].

Demand Figures and Sales Prices

Demand figures define the objective of the optimization. They state how much product must be produced and then shipped to the customers. Table 9.1 depicts the demand figures for a subset of dealerships for the months from January 2015 to June 2015. Sales prices provide information to estimate the profit of a given distribution plan. Table 9.2 shows the sales prices for the five main regions of operation. Being a global supply chain, international trade factors also affect the profit of the distribution plan. In Caterpillar's model, tariff costs are considered. Table 9.3 defines the percentage of sale price to be paid as tariff tax.

Table 9.1 – Demand figures.

Region	Dealer Code	Jan-15	Feb-15	Mar-15	Apr-15	May-15	Jun-15
ADSD-N	B010	0	5	5	12	25	18
ADSD-N	B190	3	5	13	14	25	17
ADSD-N	B030	3	5	13	14	21	19
ADSD-N	B150	3	5	13	14	18	19
ADSD-N	B160	3	5	11	14	17	20
ADSD-N	B170	4	5	10	14	17	17

ADSD-N	B270	4	5	12	13	15	17
ADSD-N	B290	5	5	12	12	16	20
ADSD-N	B330	4	5	12	18	18	19
ADSD-N	B350	5	5	13	16	21	19
ADSD-N	B370	5	6	11	13	18	21

Table 9.2 – Regional sales prices.

Region	Price
APD	224,115
ADSD-N	186,390
ADSD-S	189,079
CJL	195,116
EAME	169,783

Table 9.3 – Tariff costs (part of international trade factors).

Demand	Country	Source-1	Source-2	Source-3	Source-4
B150	USA	0.09	0.09	0.09	0.09
B180	USA	0.05	0.05	0.05	0.05
B170	USA	0.05	0.05	0.05	0.05
B190	USA	0.05	0.05	0.05	0.05
B370	USA	0	0	0	0
B420	USA	0	0	0	0
B37A	USA	0	0	0	0
B01C	USA	0	0	0	0
B010	USA	0.038	0.038	0.038	0.038
D020	USA	0	0	0	0
D070	USA	0.05	0.05	0.05	0.05

Machine Production Details

Production capabilities of a supply chain may be summarized by production capacity and costs. Table 9.4 shows the production costs and capacities for the main production factories. Production factories also have a fixed cost, which is independent from the amount of product actually produced. In Caterpillar's supply chain, production facilities for the final product are practically assembly points, where components are put together

the make the machine. Table 9.5 provides the additional information of production. *Assembly Planned Days* refers to the estimate number of days required to assemble the given product. In order to increase resilience, the standard deviation of the assembly days is also considered as *Variability*. Fixed costs of operation of the factories are indicated in the rows *Additional Variable Costs* and *Period Costs*.

Table 9.4 – Final products production costs and capacities for the active production factories.

Source ID	Product Cost	Production Capacity
source-6	163,786.26	1250
source-17	117,758.26	80
source-21	150,132.16	95

Table 9.5 – Additional production factories and assembly points details.

Source ID	Source-6	Source-17	Source-21
Assembly Planned Days	7	3	3
Variability (d)	3	5	5
Additional Variable Costs (per machine)	$ 57,585.50	$ 19,604.70	$ 40,547.20
Period Cost Basis	$ 966,268.82	$ 351,843.59	$ 400,348.01

Components Production Details

Production capacity and costs of components also contribute to the production capabilities of a supply chain. Table 9.6 and Table 9.7 list respectively the production costs and capacity for each component and for each production factory. In Caterpillar's supply chain, a typical product is large enough that it cannot be shipped together with other products or parts. Components however are grouped together by physical characteristics and shipped into containers or trucks. Determining how to group components is itself an optimization problem and its solution has a significant effect on the transportation costs. Physical characteristics of components are fundamental for shipment planning. Table 9.8 summarizes the main physical characteristics of the components and Table 9.9 describes the preferred transportation modes for each component.

Table 9.6 – Main components production costs.

Component ID	Source-1	Source-2	Source-3	Source-4	Source-5	Source-6
0	11,995.6	5,754.9	1,169.3	5,746.1	7,449.1	487.5
1	971.2	384	596.1	363.6	578.2	169.1
2	240.1	656.7	262.8	417.8	387.2	334.8
3	15.8	14.2	23.9	16.9	10.4	5.1
4	323.8	224.3	256.4	103	7.2	102
5	13	8.2	12.8	5.6	12.9	1.2
6	4.9	21.4	3.8	6.8	0.4	6.9
7	58.4	14.8	18.7	19.4	30.5	26.4
8	226.5	375.5	35.1	111.8	83.4	0.7
9	311.6	131.4	172.5	73.2	22.2	47.2

Table 9.7 – Main components production capacity.

Component ID	Source-1	Source-2	Source-3	Source-4	Source-5	Source-6
0	0	0	0	0	0	75,300
1	0	0	0	0	0	53,698
2	0	0	0	0	0	67,210
3	0	0	0	0	0	75,646
4	0	0	0	0	0	72,320
5	0	0	0	0	0	78,574
6	0	0	0	0	0	77,181
7	0	0	0	0	0	68,556
8	0	0	85,362	0	0	0
9	0	0	0	0	0	0

Table 9.8 – Main components physical characteristics.

Component ID	Quantity	Length (m)	Width (m)	Height (m)	Cube (m^3)	Weight (kg)
0	2	1	1	1	1	1
1	2	1	1	1	1	1
2	4	1	1	1	1	1
3	5	1	1	1	1	1
4	1	1	1	1	1	1
5	3	1	1	1	1	1
6	4	1	1	1	1	1
7	1	1	1	1	1	1
8	3	1	1	1	1	1

9	3	1	1	1	1	1

Table 9.9 – Main components transportation mode and container shipping details.

Component ID	Cube per container	Weight per container	Ocean transportation type	Land transportation type
0	14	18	40ft	Van
1	6	12	40ft	Van
2	4	40	40ft	Van
3	15	5	40ft	Van
4	3	6	40ft	Van
5	30	27	40ft	Van
6	32	12	40ft	Van
7	9	9	40ft	Van
8	24	24	40ft	Van
9	18	6	40ft	Van

Transportation Times and Costs

Transportation times and costs are typically defined as a function of transportation mode and route characteristics. Caterpillar's products are moved on the supply chain via trucks and freighter ships. If physical characteristics of products allow it and the economics make sense, some parts or machines are shipped on planes and train. Machines are moved on flatbeds when on land and shipped on what is called "roll-on-roll-off" (RoRo) freighter ships when shipping across the ocean. Whenever possible, components are grouped together and shipped in vans on land and containers on freights.

Caterpillar's global supply chain is made of 7 main layers. However, the layers that are used to ship components overlap with the layers for the final machines. In practice, there are only 4 layers for which transportation data is needed for:

1. Direct layer from production source to dealership.
2. Layer from production source to outbound shipping port.
3. Layer from outbound to inbound shipping port.
4. And layer from inbound shipping port to dealership.

Table 9.10, Table 9.11, Table 9.12, and Table 9.13 define the transportation times for the layers respectively. Table 9.14, Table 9.15, Table 9.16, and Table 9.17 describe the transportation costs.

Table 9.10 – Transportation times from production source to dealership location.

Destination Node	Source-1	Source-2	Source-3	Source-4	Source-5	Source-6
B010	4.06974	7.62162	5.22379	8.24551	4.17566	3.78236
B190	10.854	6.42197	9.0116	8.82454	7.83565	5.75561
B030	5.61323	7.64382	7.70041	9.85946	8.96738	3.73861
B150	5.30468	5.64883	2.15986	7.48522	4.2821	8.66845
B160	3.74264	10.0067	10.1218	7.94127	8.80864	3.26534
B170	8.78881	3.47849	4.86383	1.50706	10.4481	6.49515
B270	3.90451	7.50545	4.59451	1.59816	2.23761	10.0117
B290	7.55878	6.42598	4.0699	1.58572	9.15707	5.77437
B330	3.70067	4.68993	1.43263	6.91886	2.58046	3.16803
B350	7.64162	2.55204	7.91842	3.17009	2.18223	6.32986

Table 9.11 – Transportation times from production factories to outbound shipping ports.

Destination Node	Source-1	Source-2	Source-3	Source-4	Source-5	Source-6
N1	3.54507	1.89881	3.46196	3.6225	4.6748	7.56861
N2	7.92096	2.36812	2.94542	6.57457	2.72099	2.26917
N3	7.68093	1.96888	4.5111	3.99685	7.88826	4.69922
N4	1.55081	7.35364	4.56697	3.25239	7.52668	2.06474
N5	6.74794	6.12468	4.86673	3.47597	6.60938	7.49959
N6	3.01729	4.21982	8.10721	3.21914	6.44154	8.02586
N7	-1	-1	-1	-1	-1	-1
N8	-1	-1	-1	-1	-1	-1
N9	-1	-1	-1	-1	-1	-1
N10	-1	-1	-1	-1	-1	-1

Table 9.12 – Transportation times from outbound to inbound shipping ports.

Destination Node	N1	N2	N3	N4	N5	N6
N1	14.8033	102.027	166.635	33.8044	65.4118	169.687
N2	31.5665	21.2685	105.724	154.703	32.176	16.3792
N3	78.9419	163.614	39.6482	149.079	155.004	83.3308
N4	20.0546	49.1769	87.5915	108.801	105.294	27.0837

N5	55.8399	171.998	131.226	98.9054	113.9	64.0896
N6	11.1455	175.661	38.4701	72.723	91.5475	159.359
N7	85.7257	52.8707	150.645	175.93	121.156	45.3769
N8	14.5901	87.9096	8.49463	91.3213	58.6445	89.2526
N9	38.783	129.446	98.2195	59.6631	50.3032	150.861
N10	126.83	143.187	67.6064	62.8232	102.514	158.318

Table 9.13 – Transportation times from inbound shipping ports to dealerships locations.

Destination Node	N1	N2	N3	N4	N5	N6
B010	1.49727	4.93789	3.809	3.82434	2.25245	4.95817
B190	3.03376	9.16689	5.76605	4.43143	4.13211	7.58552
B030	4.308	3.23758	5.10043	5.0796	9.01721	5.73284
B150	1.85051	8.9028	4.73734	4.89889	1.83582	8.1388
B160	7.75559	5.59983	4.57858	8.53952	5.83781	4.29524
B170	9.52327	8.40648	2.12765	9.21347	5.15428	2.56987
B270	8.5371	4.74001	8.45189	4.98876	4.05393	6.81623
B290	3.29554	9.4528	2.5517	6.88188	6.45237	10.3665
B330	3.31652	5.37618	2.19659	3.78417	6.70349	5.83165
B350	3.15408	7.05156	4.52	4.26511	8.22228	2.30685

Table 9.14 – Transportation costs from production sources to dealership locations.

Destination Node	Source-1	Source-2	Source-3	Source-4	Source-5	Source-6
B010	4525.11	2477.87	2082.75	3806.33	1230.48	3440.58
B190	2615.84	2913.6	4567.71	3418.09	5084.96	1689.55
B030	2140.6	3990.44	4035.01	4304.07	2536.83	3057.85
B150	4077.7	1498.25	4240.69	842.559	5018.36	2246.05
B160	4274.79	4203.25	1946.98	4401.34	1526.97	2453.65
B170	4147.33	953.298	1209.04	1234.3	2499.7	4634.01
B270	1655.93	2679.12	4810.75	4764.76	1056.84	3973.08
B290	1803.85	3436.87	2734.25	2781.27	2319.63	1113.65
B330	3624.87	3734.98	2266.81	3580.34	1575.27	4622.51
B350	4684	1944.2	758.678	4030.65	2497.16	3620.19

Table 9.15 – Transportation costs from production factories to outbound shipping ports.

Destination Node	Source-1	Source-2	Source-3	Source-4	Source-5	Source-6
N1	2474.16	3506.58	2936.73	2673.87	942.473	1548.29
N2	1619.03	2601.07	1022.76	1969.34	2089.75	844.754
N3	1107.89	2814.62	1464.58	1341.07	1731.05	1775.82
N4	2587.27	1238.36	3139.38	3355.82	782.082	1167.48
N5	2413.59	1471.18	3326.31	2180.23	1908.42	2562.81
N6	2494.21	2019.89	3606.71	3251.1	817.219	2289.99
N7	-1	-1	-1	-1	-1	-1
N8	-1	-1	-1	-1	-1	-1
N9	-1	-1	-1	-1	-1	-1
N10	-1	-1	-1	-1	-1	-1

Table 9.16 – Transportation costs from outbound to inbound shipping ports.

Destination Node	N1	N2	N3	N4	N5	N6
N1	114.655	14.8262	66.2312	151.061	30.9634	137.788
N2	148.145	58.0719	144.062	88.6322	41.3443	15.8672
N3	28.9933	12.6543	104.017	110.15	127.973	80.5178
N4	14.0721	113.599	24.6152	98.3273	59.2554	100.429
N5	118.587	29.61	77.1391	61.6619	127.996	23.0622
N6	152.009	60.9181	6.60373	123.202	121.055	130.809
N7	36.0686	65.1575	113.201	83.6919	128.993	23.5319
N8	115.196	19.8571	81.5588	81.8817	18.618	71.9524
N9	12.7877	26.9961	16.7746	141.845	112.906	65.247
N10	105.066	29.6316	76.2439	150.284	114.488	42.6324

Table 9.17 – Transportation costs from inbound shipping ports to dealerships locations.

Destination Node	N1	N2	N3	N4	N5	N6
B010	2610.38	2302.85	2907.72	1053.72	5015.88	4220.11
B190	3853.59	1822.57	3767.97	4913.58	4585.84	4004.53
B030	1451.9	3368.38	4778.63	3261.84	3303.1	4492.29
B150	4956.82	4419.19	4374.96	3253.17	3738.6	1918.83
B160	1209.74	2971.37	3076.7	3116.18	2011.79	4892.83
B170	3658.2	2953.2	3663.75	1990.42	3713.24	1485.73
B270	1853.54	1933.13	2381.86	1853.35	1001.1	771.394
B290	3674.54	3299.5	1938.32	5032.48	1418.62	2778.26
B330	3669.45	1742.36	1483.05	4456.75	4564.01	4756.18
B350	4602.37	4731.03	1556	3605.13	2914.83	4338.44