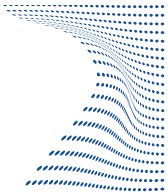
ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR



Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (XXXIII cycle)

Combinatorial Optimization Problems under Deterministic, Stochastic, and Dynamic Scheduling Decisions

Mina Roohnavazfar

* * * * * *

Supervisors

Prof. Roberto Tadei - Politecnico di Torino
Prof. Seyed Hamid Reza Pasandideh - Kharazmi University
Dr. Daniele Manerba (Co-supervisor) - Università degli Studi di Brescia

Doctoral Examination Committee:

Prof. Paolo Montuschi, Department of Control and Computer Engineering, Politecnico di Torino;
Prof. Paolo Brandimarte, Department of Mathematical Sciences, Politecnico di Torino;
Prof. Fariborz Jolai, Department of Industrial Engineering, Tehran University, Tehran, Iran;
Prof. Mehdi Seifbarghy, Department of Industrial Engineering, Alzahra University, Tehran, Iran;
Prof. Alireza Arshadikhamseh, Department of Industrial Engineering, Kharazmi University, Tehran, Iran;
Prof. Ali Ghodratnama, Department of Industrial Engineering, Kharazmi University, Tehran, Iran.

Politecnico di Torino
September 2021

I hereby declare that the contents and organisation of this dissertation constitute my own original work and do not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Mina Roohnavazfar

Turin, September 2021

Summary

Many essential and practical decision problems can be expressed as Combinatorial Optimization Problems (COPs), aiming to find the best of an exponentially large set of solutions. COPs occur in engineering, physics, mathematics, economics, management, commerce, social sciences, and even politics. In many of such problems, scheduling the involved operations at the right time is of critical importance (e.g., when to visit a customer for a delivery or to start processing a job on a machine, which is the order for addressing tasks inside a complex project, and so on).

According to the available knowledge on the problem and the length of its optimization horizon, a decision-maker addresses decisions which may range from the setting of targets for the entire enterprise (long- and medium-term), to detailed operations regarding day-by-day activities (short-term), up to specific actions to implement just as soon as particular events occur (e.g., a disruption). From this perspective, the decisions can be broadly classified into strategic/tactical, operational, and real-time decisions. Strategic decisions are significant choices influencing the whole business (or a major part of it), and tactical decisions relate to implementing strategic ones. Operational decisions are taken for the day-by-day operations of the enterprise and have a short-term horizon as they are implemented repetitively at lower levels of management. Finally, real-time decisions must be made immediately against new incomes of information on the process and, in general, relate to actions for adjusting the plan while it is executed.

COPs may explicitly or not consider data uncertainty. In deterministic settings, all the problem parameters are precisely known over the decision-making horizon and available in advance for decision-making. Conversely, in stochastic settings, the problem parameters are only partially known and described by random variables or distributions rather than approximated by a single value. Another settings, known as the dynamic one, involve additional information that may arrive during the decision-making process itself.

In this thesis, we consider deterministic, stochastic, and dynamic COPs containing critical scheduling decisions, which in turn involve possible inter-period relations (precedence, time-dependent parameters, time windows, and so on). Focusing on classical applications (such as optimal path problem, job scheduling, vehicle routing

problems) and different planning decisions over the optimization horizon, we will consider ad-hoc modeling and solving approach by deepening and sometimes pushing further modeling paradigms like Mixed-Integer Linear Programming (MILP), Stochastic Programming (SP), Discrete Choice Models (DCM), and Online Optimization (OO), as well as solving/algorithmic frameworks like Logic-Based Benders Decomposition (LBBD), Extreme Value Theory Deterministic Approximation (EVTDA), Reinforcement Learning (RL) and meta-heuristic algorithms such as Iterated Local Search (ILS) and Simulated Annealing (SA).

In the following, we briefly describe the deterministic, stochastic, and dynamic problems addressed, together with their peculiarities and the methodology used to approach them.

- Deterministic COPs

  Concerning deterministic COPs with operational scheduling decisions, we study some interesting variants of the Vehicle Routing Problem (VRP), which involve scheduling decisions over the time horizon specifying the customers' visit. Motivated from some real-life applications, such as package delivery or order-picker routing problems, we describe and model two VRP variants: the multi-trip single-vehicle routing problem with AND-type precedence constraints and vehicle routing problem with AND/OR type precedence constraints and time windows.

  In many real industrial or social service environments where a set of tasks or services are performed, partial orders known as Precedence Constraints (PCs) are defined to specify which nodes need to be visited before other ones. In logistics, some customers/target locations may have priorities over the others to be served/visited due to their interconnections, as in the Dial-A-Ride or Pickup and Delivery problems. For instance, in the Dial-A-Ride problem, the inhaul node is visited before the backhaul node. Again, the pickup node cannot be visited after the delivery node in the Pickup and Delivery problem. Such relations represent the so-called *conventional PCs*. In practice, however, there are many cases in which conventional PCs are not suitable, while alternative relations, such as AND-precedence, OR-precedence, SOFT-precedence, and S-precedence, are defined. In particular, an AND-precedence constraint requires that a node be reached only after visiting all of its defined predecessors. In contrast, an OR-precedence constraint requires a node to be reached if at least one of its predecessors has been met before. A problem containing both the above types of relations is said to have AND/OR-precedence constraints.

  An interesting example is the order picker routing problem. A picker walks or drives through a warehouse to collect the requested items and put them in a roll container considering fragility restrictions, stackability, shape, size, and weight. To prevent damage to the more fragile items, pickers cannot put heavy items on top of lighter ones that AND-PCs can ensure. Again,

due to the stackability feature, an item can be collected when at least one of its predecessors has been retrieved, represented by OR-PCs. Such physical features and preferred loading or unloading sequences (to avoid extra effort on sorting and packing the collected items at the end of the retrieving process) can be represented as AND/OR type PCs. Moreover, to deal with various storage and replenishment policies that describe the availability of items like food or perishable products, time window limitations are defined for retrieving the items in such an application.

The first proposed problem is optimally solved by an exact approach based on the Logic-Based Benders Decomposition algorithm. Instead, given its higher complexity, a hybrid meta-heuristic algorithm combining Iterated Local Search and Simulated Annealing approaches is developed for the second problem. For both problems, extensive computational experiments are conducted to evaluate the developed solution approaches in CPU time and solution quality.

- Stochastic COPs

  Many COPs which involve operational scheduling decisions are considered under uncertainty. In this thesis, we try to interpret such problems as multi-stage Random Utility Models. A decision-maker needs to choose alternatives among a set of choices stage by stage to achieve a sequence of optimal decisions eventually. Due to the complexity of most optimization problems, it is crucial to develop a multi-stage structure that would allow us to make decisions sequentially, considering several criteria and requirements that describe the problem. However, the stochasticity affecting the information for the future stages makes the problems very hard to solve even under this new perspective.

  However, for this kind of setting, an asymptotic Deterministic Approximation (DA) approach has been recently proposed by Tadei et al. [202], which computes the total expected utility (or cost) of the optimal sequence of alternatives. Unlike other paradigms for modeling multi-stage optimization problems under uncertainty (e.g., Stochastic Programming or Stochastic Dynamic Programming), the exact knowledge of the probability distribution of the random variables is not needed to derive the DA approach. According to this approach, the entire decision-making process collapses into a single-stage approximation without revealing the uncertainty of the random oscillations over time. This is particularly helpful also to overcome the curse of dimensionality linked to stochastic problems.

  In this thesis, the DA approach is validated through two different multi-stage decision-making processes, a very general optimization problem and a more specific and constrained one. In the first case, we address a generic optimal path problem in a multi-stage stochastic decision-making network. The objective can be defined as a maximization of choice utilities or minimization

of costs. This problem is modeled as sequential decision-making in a graph layered into stages to find the optimal path of choices in a recursive fashion. Apart from providing the approximated expected value of the path, we propose a way of deriving feasible solutions using a Nested Multinomial Logit model, which returns the choices probabilities at different stages. In the second case, we study a stochastic single-machine job scheduling problem that involves operational decisions under features of learning effect on processing times, sequence-dependent setup times, and machine configuration selection. The objective is to find the sequence of jobs and configurations to process each job to minimize the makespan. Here, the more constrained nature of the problem leads us first to derive a Mixed-Integer Non-Linear Programming model based on the concept of accessibility for the alternative decisions at each stage. Then, by defining a new, more flexible accessibility measure, we approximate and linearize such a model by reducing the problem to search for the shortest path throughout stages, which can be efficiently solved.

For both settings, the accuracy and efficiency of the proposed DA approach are experimentally proved on a large set of randomly generated instances. In the first application, the approximation approach is compared with the Expected Value Problem. At the same time, in the job scheduling case, classical two-stage and multi-stage Stochastic Programming models are applied to evaluate the performance of the proposed approximation. It is shown that the DA approach provides a powerful decision support tool that overcomes the computational burden of solving fat stochastic programs that depends on the number of scenarios considered.

- Dynamic COPs

  Differently from the static (deterministic or stochastic) case, in which all the information is fully known in advance, Dynamic COPs involve additional information that may arrive during the decision-making process (e.g., new orders, changes of available resources). This creates the need for an optimization that adapts while the situation changes on a real-time basis.

  In the thesis, we study the online single-machine job scheduling problem. Some recent developments suggest that Reinforcement Learning (RL) techniques can effectively deal with online scheduling issues. So, we apply four of the essential RL techniques, namely Q-learning, Sarsa, Watkins's Q(λ), and Sarsa(λ), to address the problem to minimize two different and widely used objective functions, i.e., the total tardiness and the total earliness plus tardiness of the jobs. Our main goal is to provide insights into how such techniques perform in the scheduling process under the different frequency of job arrivals and compare their performance with two other approaches. Namely, a random assignment (*Random*) which selects a job randomly and one of the most popular dispatching rules called the *earliest due date* (*EDD*) rule.

Acknowledgements

Firstly, I am extremely grateful to Professor Roberto Tadei for the continuous support of my research and his invaluable advice, patience, motivation, and immense knowledge.

I would like to extend my sincere gratitude to Professor Seyed Hamid Reza Pasandideh. His support and plentiful experience have encouraged me in my academic research.

Besides my supervisors, I would like to thank my co-supervisor, Doctor Daniele Manerba, who provided insights and expertise that greatly assisted me in the entire research. Without his precious support, it would not be possible to conduct this research.

I am also thankful to the Department of Control and Computer Engineering of Politecnico di Torino for providing me the opportunity to join with the Dual Ph.D. Program with the Kharazmi University of Tehran in Iran.

I would like to dedicate this thesis to my loving family

Their support, encouragement, and constant love have sustained me throughout my life. I will always appreciate all they have done.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation to deterministic, stochastic, and dynamic combinatorial optimization problems with scheduling decisions

Combinatorial Optimization Problems (COPs) are widespread in computer science and operations management areas. COPs are defined based on an objective function and logical constraints. A feasible solution of COPs, satisfying all the constraints, is composed of discrete decisions. Every decision may affect the total value of the objective function and the feasibility of the solution. The optimal solution is the best feasible one, distinguished based on the objective function value of candidates in a finite (and huge) discrete set. Depending on the problem, any COP can be represented as a minimization or maximization problem. One of the two formulations is often more natural, but algorithmically, minimization and maximization problems can be equivalently treated.

Of course, an exhaustive search to solve COPs, which consists of enumerating all possible solutions and choose the best one, is far to be suitable. In realistic situations, this approach becomes intractable due to the combinatorial nature of such problems. Several exact methods have been developed to provide an optimal solution to COPs, and, in some cases, efficient exact algorithms can optimize the problem in polynomial time. When efficient algorithms are not available for more complex situations, the problem can be mathematically formulated and then solved by a solver (e.g., Cplex, GAMS, Gurobi, AMPL, OPL, etc.). However, solving most COPs by the solvers is computationally demanding as the CPU time may grow exponentially.

Due to the inner complexity of most COPs (e.g., an NP-Hard problem) and being computationally challenging to provide a solution, other approaches like heuristic, metaheuristic, and approximation approaches can be advisable to apply instead of exact methods. While the availability of optimal solutions is necessary in some

cases, in many real applications, a good approximated solution obtained in acceptable CPU time is enough, especially for large-size instances of a COP.

Many COPs are concerned with scheduling decisions on various activities and resources, such as machines, airplanes, and people, to meet desired objectives. Such problems occur in almost all management fields (e.g., finance, marketing, production, scheduling, logistic, inventory control, and facility location). For example, most airlines need to determine crew schedules that minimize the total operating cost. Again, automotive manufacturers may want to determine the design of a fleet of cars that will maximize their market share. Or again, a flexible manufacturing facility needs to schedule the production for a plant without having much advance notice as to what parts will need to be produced that day.

Dealing with COPs implies different levels of decisions such as strategic (long-term), tactical (mid-term), operational (short-term). Let us consider the supply chain problem. Decisions on the location of manufacturers or distributors and defining transportation modes could be classified as strategic decisions. Production planning, selection of suppliers are examples of tactical decisions, while short-term decisions such as production scheduling, vehicle routing, etc., are considered operational decisions.

The usual assumption in COPs is that all the problem elements are precisely known, which provides a deterministic setting. Within this framework, it is assumed that the perfect information about the problem inputs is known and available in advance. In practice, the COPs may be expressed under uncertainty or dynamicity since the data may be partially known or changed dynamically over the entire decision horizon. Because of incomplete information or the dynamics of the environment, the outcome of any specific plan is uncertain. It is easy to understand that, in those cases, ignoring the parameter variability and uncertainty may lead to inferior or, even worse, non-feasible decisions.

It is well-known that explicitly addressing uncertainty in optimization problems generally increases the decision-making process's complexity and poses significant computational challenges. Therefore, it always makes sense to see whether it is possible to incorporate stochasticity in an approximated way, converting the stochastic model into a deterministic one and, if so, how accurate this approximation is. Stochastic Programming is one of the main existing paradigms in dealing with uncertain data assuming that the random input data follow probability distributions and pursues optimality in the average sense, adopting a risk-neutral perspective. However, it is not easy to measure the probabilistic distribution of input data in practice. Even if an estimate of such a distribution is available, many scenarios are necessarily needed to approximate it accurately. Unfortunately, as the number of scenarios increases, the problem becomes more complex since the number of decision variables and constraints grows.

In this spirit, three main parts corresponding to deterministic, stochastic and dynamic COPs are proposed in this thesis. In the first part, we aim to address

deterministic COPs motivated by two applications from Vehicle Routing Problems (VRPs). The objective of this part is twofold. The first one is describing multi-trip single-vehicle routing problem with AND-type precedence constraints, modeling and solving that using logic based benders decomposition algorithm. Our second objective is describing vehicle routing problem with AND/OR precedence constraints and time windows, mathematically formulating the problem and solving that through a developed hybrid algorithm.

The second part of the thesis is associated to the stochastic COPs focusing on solving multi-stage decision making problems under uncertainty. Two research problems are proposed and addressed using the asymptotic deterministic approximation approach. The first problem seeks an optimal path on a multi-stage stochastic decision network, while the second research deals with stochastic single machine scheduling problem considering learning effect on processing times, sequence-dependent setup times, and machine configuration selection, simultaneously.

In the third part, online single machine scheduling problem as an application in dynamic COPs is proposed and addressed using reinforcement learning approaches.

In the remainder of this chapter, we separately provide the aims and overviews of the proposed COPs under deterministic, stochastic, and dynamic settings.

1.2 Aims and overview on deterministic COPs

In this thesis, we study an extension of vehicle routing problems that involve inter-period relations such as AND-type precedence constraints, OR-type precedence constraints, time windows, etc. Our proposed generalization is necessary for problems in which the nodes associated with customers/target locations may have priorities over the others to be served/visited by the vehicles due to their interconnections in the Dial-A-Ride or Pickup and Delivery problems. In these problems, each node is visited after just one predecessor. For instance, in the Dial-A-Ride problem, for any backhaul node j, there is a particular inhaul node i where the relation $i < j$ must be met within a route. This is the so-called conventional precedence constraint (PS). In practice, there are cases in which conventional PCs are not suitable. So, alternative definitions, such as AND-precedence, OR-precedence, are defined. AND-type PCs are defined when a node has multiple predecessors met before, while the OR-type PCs mean that a node can be reached once at least one of its predecessors has been met before.

Our motivation for the proposed problem comes from real applications such as the order picker routing problem where a picker walks or drives through the warehouse to collect the requested items and put them in a roll container considering fragility restrictions, stackability, shape, size, and weight. Such physical features and preferred loading or unloading sequences (to avoid extra effort on sorting and

packing the collected items at the end of the retrieving process) can be represented as AND/OR PCs.

In this sense, two problems are proposed in deterministic circumstances described briefly in the following subsections.

1.2.1 Multi-trip single-vehicle routing problem with AND-type precedence constraints

The first problem studied in this thesis is the multi-trip single-vehicle routing problem with AND-type precedence constraints represented in Chapter 2. The problem aims to determine a sequence of trips required to visit all the nodes, assign nodes to the trips, and find the sequence of the nodes in each trip to minimize the total traveling cost. The nodes are related to AND-type precedence relations that must be respected within a route and among them by determining the order of trips.

To address the proposed problem, three mathematical formulations are developed. The performance of the models is experimentally investigated on a set of generated instances. Moreover, a decomposition algorithm based on the Logic-Based Benders Decomposition (LBBD) approach is developed, which partitions the original problem into an assignment master problem and sequencing sub-problems. A new extension of a valid optimality cut is developed, and its performance is experimentally investigated. The cuts are generated from the solutions of subproblems and added to the master problem to achieve faster convergence. Additionally, we presented a relaxed version of LBBD by defining a limit for optimality gap and CPU time in deriving master-problem solutions. In such a way, the algorithm's efficiency improves. It allows the algorithm to find a feasible solution to the original problem in less CPU time and larger instances.

The performance of proposed LBBD algorithms is evaluated and compared together through extensive computational experiments. The results show that the two exact LBBDs can solve most instances, while the relaxed version of LBBD can heuristically solve all the generated instances in a shorter computational time. The results derived from this research have been submitted to an international journal [171].

1.2.2 Vehicle routing problem with AND/OR precedence constraints and time windows

The second deterministic problem, described in Chapter 3, is the vehicle routing problem with time windows in which the nodes are related together through AND/OR precedence constraints. The proposed problem calls for determining optimal vehicles' optimal routes to minimize the total traveling and service time while serving all the customers.

To deal with the problem, we formulate it as a Mixed Integer Linear Programming (MILP) model to optimally solve small-sized instances by spending a lot of CPU time. Furthermore, a meta-heuristic algorithm based on hybridization of Iterated Local Search (ILS) and Simulated Annealing (SA) approaches is developed, which complements the advantages of both ILS and SA in a single optimization framework. The algorithm parameters are tuned using the Taguchi method to address the instances in a reasonable time effectively. Moreover, we extend the well-known Solomon's benchmark to generate test instances for the proposed problem. The computational result highlights the promising performance of our proposed algorithm in terms of CPU time and solution quality. The results derived from this research have been submitted to an international journal and Arxiv [172].

1.2.3 Thesis contributions on the proposed deterministic COPs

The first part of this research is devoted to the deterministic COPs motivated by applications from vehicle routing problems. This research contributes to the state-of-the-art in the following principal aspects:

- we describe and provide three mathematical formulations for the multi-trip single-vehicle routing problem with AND-precedence constraint. We conduct computational experiments to compare the performance of the models in terms of computational time and problem size;
- we develop and implement three algorithms based on the Logic-Based Benders Decomposition approach to solving the proposed problem. In the first LBBD, a recently proposed optimality cut in the literature is used. In the second LBBD, our new valid optimality cut is applied. We present a relaxed version of LBBD (the third one) by defining a limit for optimality gap and CPU time in deriving master-problem solutions. We perform computational experiments to assess the proposed LBBD algorithms;
- we describe and develop a model for the vehicle routing problem with AND/OR precedence constraints and time windows to optimally solve small-sized instances;
- we develop and implement a meta-heuristic algorithm based on Iterated Local Search (ILS) and Simulated Annealing (SA) approaches. We adjust the algorithm parameters using the Taguchi method and conduct computational experiments to compare and evaluate the proposed MILP model and the algorithm performances in dealing with different instances.

1.3 Aims and overview on stochastic COPs

In COPs, many settings can be interpreted as a multi-stage decision-making process in which decisions are made step by step to achieve an optimal sequence of choices eventually. Many applications, such as control systems, finance, logistics, network security, robotics, and traffic management, involve a sequential decision-making process as a more general decision structure. In such applications, decisions may be linked sequentially in nature.

Moreover, in many realistic situations, the decisions are taken under uncertainty without perfect information about the problem inputs. Incorporating uncertainty in the sequential decision-making process imposes significant computational complexity. The Stochastic Programming (SP) approach is one of the main existing paradigms in which uncertain parameters are expressed as random variables with known probability distributions. However, in practice, the probability distribution of random variables is not easy to measure. Even if an estimate of such a distribution is available, many scenarios are necessarily needed to approximate it accurately. Unfortunately, as the number of scenarios increases, the problem becomes more complex since the number of decision variables and constraints grows. So, it always makes sense to see whether it is possible to incorporate stochasticity in an approximated way, converting the stochastic model into a deterministic one and, if so, how accurate this approximation is.

A Deterministic Approximation (DA) approach has been recently proposed in [202] which deals with the multi-stage dynamic stochastic decision-making process. Knowing the probability distribution of random variables is not needed, and it overcomes the computational burden of solving fat stochastic programs.

In this spirit, the objective of this part of the thesis is threefold. First, we describe the asymptotic deterministic approximation approach [202] and some background information to derive it. Second, we aim to address the search of optimal paths in a multi-stage stochastic decision network as a general application of the DA approach. Third, we interpret the stochastic single-machine scheduling problem as a more specific application of the multi-stage random decision process dealt with by the DA approach.

1.3.1 Background on the Deterministic Approximation approach

As provided in Chapter 4, some information is needed before applying the deterministic approximation (DA) approach. The conceptual background knowledge includes a review of the Discrete Choice Models (DCMs), Random Utility Models (RUMs), the most common theoretical basis of DCMs, and the Multinomial Logit models. Moreover, the asymptotic approximation of the probability distribution of random utility of alternatives and the expected value of the total utility of the

decision process for the static case, which includes only one stage of decisions, are presented.

Then, the deterministic approximation derived for the multi-stage stochastic decision-making process is presented in detail. It is described that, in many real-life applications, the decision-maker is asked to solve several RUMs consecutively over multiple discrete stages aiming at maximizing (minimizing) the expected value of the total utility (cost). Since in such a structure, decisions are nested each other (i.e., the utility of an alternative at each stage is affected by the utilities of the selected alternatives in the subsequent stages), the decision process cannot be decomposed into distinct stages. In particular, it is shown that under appropriate assumptions and using some results of the Extreme Value Theory, a Gumbel distribution can asymptotically approximate the probability distribution of the best alternative utility and, in turn, the total utility of the process can be analytically derived. Moreover, the choice probability can be modeled as a Nested Multinomial Logit model.

1.3.2 Optimal paths in multi-stage stochastic decision networks

Our first problem in the stochastic COPs area is searching optimal paths in a multi-stage stochastic decision-making network, as presented in Chapter 5. The problem is the first application in the literature of the Deterministic Approximation approach, which validates this method's performance.

The optimal path problem is modeled as a sequential choice of nodes in a graph layered into stages to find the optimal path value recursively in this research. More precisely, the decision network includes several stages, and each stage contains a set of nodes. The decision-maker must select one node at each stage to eventually achieve an optimal path over the network. Incorporating uncertainty in the problem, it is assumed that the utility of each node is composed of a deterministic stage-dependent term and random term oscillations with an unknown probability distribution. Using the DA approach, the total utility of the optimal path can be approximately derived. Moreover, an optimal path solution can be heuristically obtained using a Nested Multinomial Logit model, representing the choice probability at the different stages. The accuracy and efficiency of the proposed method are experimentally proved on a large set of randomly generated instances.

The results derived from this research have been published in an international journal [173].

1.3.3 Stochastic single machine scheduling problem as a multi-stage dynamic random decision process

In Chapter 6, we study the stochastic single machine scheduling problem. The features of the learning effect on processing times, sequence-dependent setup times, and machine configuration selection are considered simultaneously. More precisely, the machine works under a set of configurations and requires stochastic sequence-dependent setup times to switch from one configuration to another. Moreover, the stochastic processing time of a job is a function of its position and the machine configuration. The objective is to find the sequence of jobs and choose a configuration to process each job to minimize the makespan.

We first propose classical two-stage and multi-stage Stochastic Programming formulations of the problem. Then, a multi-stage deterministic approximation approach is proposed to solve the problem by looking at the problem as a multi-stage dynamic random decision process. The method first derives a non-linear model based on the accessibility of alternative decisions at each stage. Then, it approximates and linearizes such a model by reducing the problem to the search for a shortest path throughout the stages. Extensive computational experiments are carried out on various sets of instances. We discuss and compare the results of applying stochastic models with those obtained by the deterministic approximation approach. This approach shows excellent performance compared to stochastic models both in terms of solution accuracy and computation time.

The results derived from this research have been published in an international journal [174].

1.3.4 Thesis contributions on the proposed stochastic COPs

The main objective of this part of research is to validate the Deterministic Approximation approach recently proposed by Tadei et al. [202] as a tool to efficiently deal with multi-stage stochastic decision-making problems in which the parameters are uncertain with an unknown probability distribution and being changed stage by stage over the entire decision stages. This part of the thesis contributes to the state-of-the-art in the following principal aspects:

- we provide the first concrete application in the literature of the deterministic asymptotic approximation approach to determine the total value (cost or utility) of an optimal (shortest or longest) path over a network with a multi-stage structure. The utility of alternatives is assumed to be a random variable with an unknown probability distribution and being changed dynamically over the stages of the decision horizon. The accuracy of the approach is tested versus benchmarks obtained by optimally solving the expected value problem over a great number of different instances;

- we derive heuristically optimal path solutions using a Nested Multinomial Logit model for the choice probability and investigating its quality in the proposed optimal path problem;
- we apply the deterministic approximation approach to address a single machine scheduling involving simultaneously stochastic sequence-dependent setup times and stochastic position-dependent processing times affected by learning effect;
- two-stage and multi-stage stochastic programming formulation are formally derived from modeling the proposed stochastic single machine job scheduling;
- approximated makespan, and optimal solutions of the proposed job scheduling problem are found by adapting the deterministic asymptotic approximation approach. This provides a powerful decision support tool that overcomes the computational burden of solving fat stochastic programs that depend on the number of scenarios considered;
- we provide a method to calibrate parameter β inside the deterministic approximation approach, which is critical for its accuracy.

1.4 Aims and overview on dynamic COPs

COPs can be divided into two categories, static or dynamic problems. In the static case, all the data are fully and deterministically known in advance. In contrast, dynamic settings involve real-time decisions based on up-to-date information regarding the system's state. In particular, a decision has to be taken because something relevant in the environment has changed. The decision-maker has to react to the change by making a new decision. For instance, in the dynamic job scheduling problem, the decision-maker has to assign new incoming jobs online. Also, when a machine breaks down, some jobs have to be reassigned. In these situations, an event corresponds to the new jobs arrival or the machines breakdown, and decisions on scheduling new jobs or re-scheduling some existing jobs need to be made.

Dynamic production scheduling is one of the most important aspects to address in many manufacturing companies. The manufacturing industries sometimes include a machine bottleneck, which affects, in some cases, all the jobs. Even manufacturing systems include multiple machines, improper usage of each machine can slow down the whole production process as each one represents a chain's primary block. Studies on single machine scheduling problems have been gaining importance for a long time since this bottleneck's management is crucial (see, [1], [111], [160]. In online scheduling, a decision-maker regularly schedules jobs over time, attempting to reach the overall best performance. Reinforcement Learning (RL) is a continuing and goal-directed learning paradigm, and it represents a promising

approach to deal with online scheduling as its potential has been revealed in several works (see, e.g., [74], [185], [235]).

In this spirit, this part of the thesis aims to address the online single-machine job scheduling problem using RL approaches. The research is described briefly in the following subsection.

1.4.1 Online single-machine scheduling problem

This thesis study the online single machine job scheduling problem with release dates, where preemption is allowed. The problem is to arrange the queue's jobs to minimize two different objective functions: total tardiness and the total earliness plus tardiness of the jobs. They are among the most widely used objectives in scheduling, focusing on meeting job due dates. In particular, the minimization of the second objective characterizes the Just-In-Time principle in production. Our motivation for this study comes from the Plastic and Rubber industrial project[1], transforming raw material into a final product, in which frequent occurrences of unexpected events call for dynamic and flexible scheduling.

To address the problem, we apply four of the essential RL techniques, namely Q-learning, Sarsa, Watkins's Q(λ), and Sarsa(λ) and compare their performance, under different operating conditions, with two other approaches: a random assignment (*Random*), which selects a job randomly, and one of the most popular dispatching rules called the *earliest due date* (*EDD*) rule. Furthermore, we propose some preliminary results obtained by using Deep Q Network (DQN), which utilizes the power of neural networks to approximate the value function.

The results derived from this research have been published in a book chapter [117].

1.4.2 Thesis contributions on the proposed dynamic COP

The final part of the thesis is devoted to the dynamic COP applying the online single machine scheduling problem. The research contributes the literature on the following aspects:

- we apply four RL approaches (namely Q-learning, Sarsa, Watkins's Q(λ), and Sarsa(λ)) to the online single machine scheduling problem under the different frequency of job arrivals to minimize two different objective functions, namely, total tardiness and the total earliness plus tardiness of the jobs;

[1]Plastic&Rubber 4.0. Piattaforma Tecnologica per la Fabbrica Intelligente (Technological Platform for Smart Factories), URL: https://www.openplast.it/en/homepage-en/

- we get insights on the compared methods (four RL approaches, random assignment and the *earliest due date* rule) and give practitioners suggestions on selecting the best method against the specific situation;
- we also approximate the value function using Deep Q Network (DQN), which utilizes the power of neural networks.

Chapter 2

Multi-trip single-vehicle routing problem with AND-type precedence constraints

2.1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) has been a research subject for many years due to the economic importance of the problem and its methodological challenges. The classical version of CVRP seeks an optimal nodes assignment to vehicles and optimizes the sequence of nodes within each route. To better represent real-world problems, additional attributes need to be considered leading to different variants of the CVRP. In some of them, the fleet comprises one single vehicle with limited capacity to perform multiple trips, like helicopters. An important reason for this gain of interest can be of enormous economic importance to some companies. The standard VRP assumes single use of vehicles over a planning period. However, in several contexts, like home delivery of fresh goods, the duration of the routes is short. Therefore they must be combined to form a complete workday. In applications where the vehicle capacity is small (electrical vehicles or small-sized vans) or the planning period is large, performing more than one route per vehicle may be the only practical solution. This is the so-called Multi-Trip Vehicle Routing Problem (MTVRP). The MTVRP has become increasingly important due to the advent of electronic services, like e-groceries, where customers can order goods online and have them delivered at home. In such applications, the vehicle is allowed for multiple trips to avoid an oversized fleet.

In many real industrial or social service environments where a set of tasks or

services are performed, partial orders known as Precedence Constraints (PCs) are defined that specify which nodes need to be visited before which other ones. The PCs play an essential role in a wide range of applications in various fields, such as the assembly industry, distribution of services, products, or passengers, construction projects, production scheduling, and maintenance support. In general cases, precedence constraints are represented by an arbitrary directed acyclic graph in which each arc corresponds to a precedence constraint between a pair of nodes.

In logistics, it is also tough to suppose that the customer nodes are visited independently. Some customers/target locations may have priorities over the others to be served/visited due to their interconnections, as in the Dial-A-Ride or Pickup and Delivery problems. In these problems, each node is visited at most after just one predecessor. For instance, in the Dial-A-Ride problem, for any backhaul node j, there is a particular inhaul node i where the relation $i < j$ (we refer to it as conventional precedence constraint) must be met within a route. In practice, there are cases in which the conventional PCs are not suitable. So, alternative definitions, such as AND precedence, OR precedence, and S-precedence, are defined. AND-type precedence constraints are defined when a node has multiple predecessors which have been met before. For example, given node i, a set of predecessors $\{j, k, l\}$ is considered which, consequently, result in a set of pairwise relations $\{(j < i), (k < i), (l < i)\}$.

A practical application of AND-type precedence constraints can be visiting patients by medical personal during the evacuation of a special needs population in anticipation of a disaster. In such a situation, some patients have priorities (due to their urgency or importance) to be visited before others before being transported to the shelters. Another application might be freight transportation by helicopter or ship, where some locations need to be reached after some others.

Let's consider a delivery problem in which a customer's order includes various items collected from different locations. In such a situation, the customer must be visited in a route only after (not necessarily immediately) visiting the locations of requested goods. This situation can also be seen in an order picker routing problem where a picker walks or drives through the warehouse to collect the requested items and put them in a roll container considering fragility restrictions, stackability, shape, size, and weight. For example, to prevent damage to light items, pickers are not allowed to put heavy items on top of light items. Such physical features and preferred loading or unloading sequences (to avoid extra effort on sorting and packing the collected items at the end of the retrieving process) can be represented as AND-type PCs that specify which items should be collected before a given one within a trip or among them.

To the best of our knowledge, there is no available research in the Literature of VRP in which the nodes are visited satisfying AND-type PCs along and among the trips. This chapter describes the Multi-Trip Capacitated Single-Vehicle Routing Problem with AND-type Precedence Constraints. The main objective is to find the

optimal sequence of routes with the minimum total traveling cost. The nodes are related to AND-type precedence relations that must be respected within a route and among them by determining the order of trips. To address the proposed problem, three mathematical formulations are developed. The performance of the models is experimentally investigated on a set of generated instances. Moreover, a decomposition algorithm based on the Logic-Based Benders Decomposition (LBBD) approach is developed, which partitions the original problem into an assignment master problem and sequencing sub-problems. A new extension of a valid optimality cut is developed, and its performance is experimentally investigated. The cuts are generated from the solutions of subproblems and added to the master problem to achieve faster convergence. We also propose a relaxed version of LBBD, which allows the algorithm to find a feasible solution in less CPU time and even larger instances.

The contribution of the research is threefold in this chapter: (i) considering AND-type precedence constraints in the multi-trip capacitated single-vehicle routing problem; (ii) three mixed-integer programming models are developed and experimentally compared on a set of generated instances; (iii) a logic-based benders decomposition algorithm with a new valid optimality cut and its relaxed version are designed and implemented capable of obtaining good solutions in terms of both quality and computational time.

This chapter is organized as follows. Section 2.2 is dedicated to the literature review of the problem. In Section 2.3, the proposed problem is described and three mathematical formulations are provided. Moreover, the computational results of the proposed models in solving a set of instances are represented in this part. In Section 2.4, we develop the LBBD algorithm to solve the problem. In Section 2.5, the computational results of the proposed algorithms are provided. Finally, we end up with a summary and some future research suggestions in Section 2.6.

2.2 Literature review

The Literature on the proposed problem can be distinguished into three main research areas. The first one deals with the precedence constraints, with the main focus on the routing problem. The second part discusses the multi-trip single-vehicle routing problem, while the last one includes the studies proposing the logic-based benders decomposition algorithm.

2.2.1 Problems with precedence constraints

In many real operational research problems where a set of tasks or services are performed, it is tough to suppose that the tasks are independent of each other. In principle, the relative order between a couple of tasks is represented as a pairwise relation named Precedence Constraint (PC). These relations make the problems

have a more comprehensive range of applications in various fields. Given nodes i and j, precedence constraint $i < j$, referred to a conventional precedence constraint, can be interpreted as activity i is performed before j. Different precedence constraints are defined in practice, such as 'AND', 'OR', 'S' types. The AND-type precedence constraint means that a node can be reached only after visiting all of its predecessors. The OR-type precedence constraint is defined when a node has multiple predecessors, and it can be reached when at least one of its predecessors has been met before. Under the S-type PC, a task can be started once all of its predecessors have been started. So, the task does not need to wait for the completion of its predecessors.

In the context of machine scheduling, Soft Precedence Constraints (SPC) have been defined by a bipartite network at a transshipment port [234]. The SPCs can be violated, but with a certain penalty. Goldwasser and Motwani [83] derive inapproximability results for a specific single-machine scheduling problem with AND/OR precedence constraints. Gillies and Liu [80] addressed single and parallel machine scheduling problems to meet deadlines considering different structures of AND precedence constraints. They proved NP-completeness of finding feasible schedules in many polynomially solvable settings with only AND-type precedence constraints. Moreover, they give priority-driven heuristic algorithms to minimize the completion time on a multiprocessor. Mohring et al. [138] provided some algorithms for the more general and complex model of AND/OR precedence constraints. They showed that feasibility and questions related to generalized transitivity could be solved using the same linear-time algorithm. Moreover, they discussed a natural generalization of AND/OR precedence constraints and prove that the same problems become NP-complete in this setting. Lee et al. [108] focused on flexible job-shop scheduling problems with AND/OR precedence constraints in the operations. They provided a MILP model able to find optimal solutions for small-sized instances. They also developed a heuristic algorithm that results in a good solution for the problem regardless of its size. Moreover, a schedule builder who always gives a feasible solution and genetic and tabu search algorithms based on the proposed schedule builder were presented. Van Den Akker et al. [216] developed a solution framework that provides feasible schedules to minimize the minimax type on a set of identical parallel machines subject to release dates, deadlines, AND/OR precedence constraints. They determined a high-quality lower bound by applying column generation to the LP relaxation. Agnetis et al. addressed some special cases of job shop, and flow shop scheduling problems with S-type precedence constraints [2]. They provided exact polynomial algorithms for a two-machine job shop and a two-machine flow shop, and an m-machine flow shop with two jobs.

In the context of routing problems, Moon et al. [140] addressed the Traveling Salesman Problem with Precedence Constraints (TSPPC), defined as pairwise relations between each couple of nodes. The PCs are represented as an order under which the nodes must be visited. A genetic algorithm that involves a topological

sort and a new crossover operation is proposed to solve the model. Mingozzi et al. [135] dealt with the TSP with time windows and precedence constraints using a dynamic programming approach. Fagerholt and Christiansen [67] considered a TSPPC with a time window to solve the bulk ship scheduling problem. The model is solved as a shortest path problem on a graph. Renaud et al. [165] proposed a heuristic model to solve the pickup and delivery TSP formulated as the TSPPC. Bredstrom and Ronnqvist [30] developed a mathematical model for the combined vehicle routing and scheduling problem with time windows. The sets of pairwise synchronization and precedence constraints are considered between customer visits, independently of the vehicles. Also, they described some real-world problems to emphasize the importance of the mentioned constraints, such as homecare staff scheduling, airline scheduling, and forest operations. Bockenhauer et al. [29] studied a variant of TSP in which a given subset of nodes are visited in a prescribed order in the computed Hamiltonian cycle. They presented a polynomial-time algorithm to solve the problem. Haddadene et al. [162] modeled a home health care structure as a variant of vehicle routing problem with time windows and timing constraints. Some patients ask for more than one visit simultaneously or in given priority order. A MILP model, a greedy heuristic, two local search strategies, and three metaheuristics are proposed to solve the problem. Recently, the task assignment problem for a team of heterogeneous vehicles has been investigated in which packages are delivered to a set of dispersed customers subject to precedence constraints. Using graph theory, a lower bound on the optimal time is constructed. Integrating with a topological sorting technique, several heuristic algorithms are developed to solve it [17].

A closely related case of the VRP with PCs is the Dial-A-Ride problem, which is an exhaustively studied problem. In this problem, the pairwise relations are inherently represented between pickup and delivery points within a route, i.e., for any backhaul node j, there is a particular inhaul node i where the PC ($i < j$) must be met within a route. For a comprehensive survey of the developed models, applications, and algorithms that address the Dial-A-Ride problems, the reader is referred to [93], [139], and [45].

The order-picking problem is one of the main applications of AND-type PCs in the context of routing problems. However, little works in order-picking problems have focused on PCs. Zulj et al. [239] considered the PCs in a warehouse of a German manufacturer of household products, where heavy items are not allowed to be stored on top of delicate items to prevent damage to the delicate items. To avoid the sorting effort at the end of the order-picking process, they propose a picker-routing strategy respecting the precedence constraints. An exact algorithm based on dynamic programming is used to evaluate the strategy and compared with the simple s-shape routing strategy. Dekker et al. [51] investigated combinations of storage assignment strategies and routing heuristics for a real case arising in a warehouse of a wholesaler of tools and garden equipment. A guideline has to

be considered indicating that fragile products have to be picked last. Matusiak et al. [134] presented a simulated annealing method to address the joint order batching and precedence-constrained picker-routing problem in a warehouse with multiple depots. The shortest path through the warehouse is determined using an exact algorithm. Chabot et al. [39] introduced the order-picking routing problem underweight, fragility, and category constraints. They propose capacity-index and two-index vehicle-flow formulations as well as four heuristics to solve the problem. Furthermore, a branch-and-cut algorithm is applied to solve the two mathematical models.

It should be noted that the proposed precedence relations in most picker routing problems (mentioned above) have been represented as a pre-specified sequence of nodes. However, in our proposed problem, AND-type PCs are defined under which the nodes assignments to trips and the sequence of nodes in each route are determined in the model.

2.2.2 Multi-Trip Single-Vehicle Routing Problem

In the Multi-Trip Vehicle Routing Problem (MTVRP), each vehicle can perform several trips during the planning horizon. In situations where the vehicle capacity is small or applications, such as home delivery of perishable goods like food, the duration of the routes is short. Therefore, the vehicles can travel several trips to complete a workday. In 1990, this problem was introduced by Fleischmann (FLE 1990). A survey on this problem can be found in [36] and [37] which give a unified view on mathematical formulations, exact and heuristic approaches, and variants of MTVRP.

In the literature, some papers dealt with multi-trip single-vehicle routing problems. Following, we refer to a few ones. Martinez-Salazar et al. [133] studied a customer-centric routing problem with multiple trips of a single vehicle to minimize the total waiting time of customers (latency). They proposed two mixed-integer formulations and a metaheuristic algorithm capable of obtaining good solutions in quality and computational time. Azi et al. [14] addressed a multi-trip single-vehicle routing considering time windows and deadline constraints. Considering that the time windows constraints may prevent serving all customers, the objective was to maximize the number of served customers and minimize the total distance. They proposed a method based on an elementary shortest path algorithm with resource constraints. Rivera et al. [168] worked on the multi-trip cumulative capacitated single-vehicle routing problem inspired by disaster logistics. The objective is the minimization of total arrival time. Two mixed-integer linear programming models, a low-based formulation and a set partitioning problem, are proposed for small instances. In contrast, an exact algorithm based on a resource-constrained shortest path problem is developed for the larger instances. Angel-Bello et al. [8] considered a routing problem with multiple uses of a single vehicle and service time in

demand points to minimize the sum of clients waiting time to receive service. They present and compare two mixed-integer formulations for this problem based on a multi-level network.

2.2.3 Logic-Based Benders Decomposition algorithm

The LBBD has mostly aroused interest among researchers in the planning and scheduling field. However, few papers in the vehicle routing area have focused on this approach. Here, we can imply some of the most cited studies that use this approach in various optimization problems. Implementation of the LBBD on parallel machine scheduling problem with sequence-dependent setup times and job availability intervals has yielded outstanding results in comparison with integer programming (IP), and constraint programming (CP) as proposed in [78]. A three-level LBBD solves the outpatient scheduling problem efficiently, which involves planning and scheduling decisions in [167]. The Multi-distributed operating room scheduling (DORS) problem is addressed using LBBD proposed by Roshanaei et al. [176]. The algorithm was enhanced using effective acceleration cuts led to faster convergence. Their proposed problem tackles the allocation of patients as well as scheduling of operating rooms. The study performed by Barzanji et al. [21] showed that the proposed LBBD algorithm has outstanding results over the existing heuristic-based search methods for the type-1 of integrated process planning and scheduling (IPPS) problem named flexible job shop scheduling with process plan flexibility problem. Simultaneous planning, lot sizing, and scheduling problem are studied by Martinez-Salazar et al. [133]. They proposed an integrated branch-and-check with a MIP heuristic based on the logic-based Benders platform. The LBBD is deployed effectively in the export dry bulk terminals problem, which involves two allocation problems: planning and scheduling parts (see, [214]). Recently, the LBBD was applied to address the heterogeneous fixed fleet vehicle routing problem with time windows with the aim of cost minimization [62]. Valid optimality and feasibility cuts were devised to guarantee the convergence of the algorithm. Extensive computational experiments illustrated the effectiveness of the suggested approach.

2.2.4 Summarizing reviewed literature

In this section, we provide Table 2.1 to summarize the related Literature. As shown, no available research in the Literature of vehicle routing and even picker routing problems considers AND-type PCs.

As mentioned before, the most related research in logistics includes dial-a-ride or pickup and delivery problems which only focus on the pairwise relations between a couple of nodes (inhaul and backhaul nodes) being satisfied within a route. However, we want to stress that assuming such relations, under which at most one

Table 2.1: Summarized literature reviewed in this section

Publication	Problem	Precedence Constraints	Solution Approach
[168]	multi-trip single-vehicle routing	-	exact algorithm
[14]	multi-trip single-vehicle routing	-	exact algorithm
[8]	multi-trip single-vehicle routing	-	Cplex solver
[133]	multi-trip single-vehicle routing	-	meta-heuristic algorithm
[216]	parallel machine scheduling	AND/OR type PCs	column generation
[234]	machine scheduling	Soft type PCs	approximation algorithms
[108]	flexible job-shop scheduling	AND/OR type PCs	heuristic algorithm
[80]	machine scheduling	AND type PCs	heuristic algorithms
[2]	machine scheduling	S type PCs	exact algorithms
[83]	assembly sequencing	AND/OR type PCs	-
[140]	travelling salesman problem	conventional PCs	genetic algorithm
[135]	travelling salesman problem	conventional PCs	dynamic programming
[165]	pickup and delivery TSP	conventional PCs	heuristic model
[30]	combined vehicle routing and scheduling	conventional PCs	-
[29]	travelling salesman problem	pre-specified sequence	heuristic
[162]	vehicle routing problem	given priority order	meta-heuristics and heuristic
[17]	assignment problem	conventional PCs	heuristic
[239]	order-picking	pre-specified sequence	dynamic programming
[51]	storage assignment and routing	pre-specified sequence	heuristics
[134]	order batching and picker-routing	pre-specified sequence	meta-heuristic
[39]	order-picking routing	pre-specified sequence	branch-and-cut and heuristics
[78]	Machine scheduling	-	LBBD
[167]	multi-modal outpatient scheduling	-	LBBD
[176]	operating room scheduling	-	LBBD
[21]	integrated planning and scheduling	-	LBBD
[133]	planning, lot sizing and scheduling	-	LBBD
[214]	planning and scheduling	-	LBBD
[62]	vehicle routing	-	LBBD
current research	**multi-trip single-vehicle routing**	**And type PCs**	**LBBD**

predecessor per node is defined which must be visited in the same route, can be restrictive in many real-life situations. In this regard, we propose the AND-type PCs represented on a directed acyclic graph, where given a node, a set of pair-wise relations are met within and among the trips. Thus, our research contributes to the Literature by describing, modeling, and providing solution approaches for Multi-Trip Capacitated Single-Vehicle Routing Problem with AND-type PCs.

2.3 Problem description and mathematical formulation

The proposed problem is a generalization of the CVRP in which the nodes are related to each other using AND-type precedence constraints. The fleet is composed of a single vehicle capable of traveling multiple trips per working day. The problem aims to determine a sequence of trips required to visit all the nodes, assign nodes to the trips, and find the sequence of the nodes in each trip to minimize the total traveling cost.

The proposed problem is defined on a directed acyclic graph $G = (N, A, c)$, where $N = \{0,1,2,\ldots,n\}$ is a set of geographically located nodes including the depot (node 0), and $A = \{(i,j) : i,j \in N, i \neq j\}$ is the set of arcs. For each arc (i,j), a travel cost c_{ij} is associated. The cost matrix is symmetric, i.e., $c_{ij} = c_{ji}, \forall i,j \in V, i \neq j$. Given node i, set AND_i is defined including the AND-type predecessors of node i. The proposed PCs are respected between nodes within and among the trips. Given PC $i < j$, if node i and j are, respectively, assigned to trips s and k, then trip s is done before starting trip k by the vehicle. In such a way, the PC $i < j$ is met among the trips. For example, let's consider the nodes and the AND-type precedence constraints as depicted in Figure 2.1. It can be seen that the vehicle travels three trips to visit all the nodes. According to the defined PCs, the set of AND-type predecessors of node r includes $\{g, c\}$. As depicted, both predecessors are visited before node r in the first trip of the vehicle. Concerning AND predecessors of node a, it can be noticed that both nodes k and m are visited before node a in such a way that node k is assigned to the same trip as node a, while node m has been visited in the previous trip. Finally, two nodes h and p are not allocated to the same trip as node u. However, the related PCS are met while both nodes are visited before node u in the previous trip.

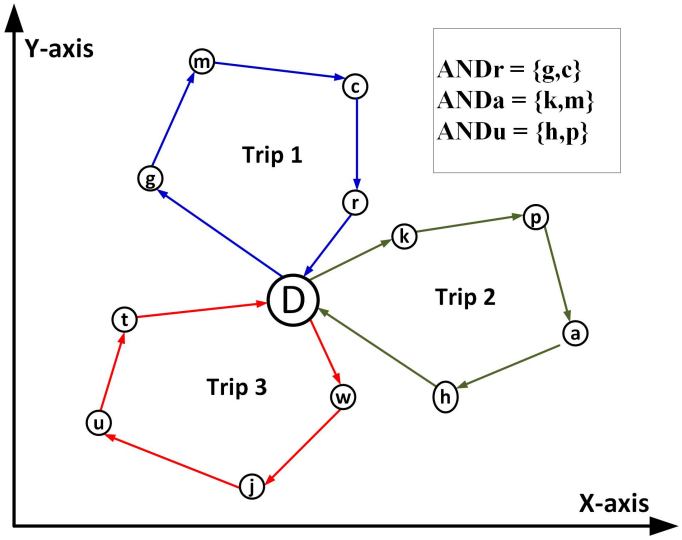


Figure 2.1: Illustration of an example with AND-type PCs respecting within and among the trips

Let us assume the following restrictions:

- Each trip starts and ends at the depot,
- Each node is visited once,
- Total demand of each route does not exceed the capacity of the vehicle,

- Number of trips cannot exceed the Maximum value p,
- PCs are met between nodes within and among the trips.

In the next subsections, three mathematical formulations are developed to address the proposed problem. Let us introduce the following notations and parameters:

- $N = \{0,1,2,...,n\}$: set of nodes including depot;
- $\acute{N} = \{0,1,...,n,n+1,...,n+p\}$: set of nodes including p dummy depots;
- c_{ij}: travelling cost from node i to node j;
- p: upper bound for the number of trips;
- Q: capacity of the vehicle;
- d_i: demand of node i;
- M: a big scalar;
- AND_i: set of predecessors of node $i \in N \setminus \{0\}$.

2.3.1 Three-index MIP model

In this section, the Three-index MIP model of the proposed problem is represented. Let us introduce following decision variables:

x_{ijr}: binary variable takes value 1 if arc (i,j) is linked in trip r, 0 otherwise;
s_{ir}: integer variable indicating the position of node i in trip r;
u_r: binary variable takes value 1 if the vehicle travels trip r, 0 otherwise.

Then, the problem can be stated as:

$$Z1 = \min \sum_{r=1}^{p} \sum_{\substack{j=0 \\ i \neq j}}^{n} \sum_{i=0}^{n} c_{ij} x_{ijr} \tag{2.1}$$

subject to

$$\sum_{j=1}^{n} x_{0jr} = u_r \qquad \forall r \in \{1,...,p\}, \tag{2.2}$$

$$\sum_{i=0, i \neq j}^{n} \sum_{r=1}^{p} x_{ijr} = 1 \qquad \forall j \in N \setminus \{0\}, \tag{2.3}$$

$$\sum_{i=0}^{n} x_{ijr} = \sum_{i=0}^{n} x_{jir} \qquad \forall j \in N \setminus \{0\}, \forall r \in \{1, ..., p\}, \tag{2.4}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{j=1}^{n} d_j x_{ijr} \leq Q.u_r \qquad \forall r \in \{1, ..., p\}, \tag{2.5}$$

$$u_r \leq u_{\acute{r}} \qquad \forall r, \acute{r} \in \{1, ..., p\}_{r>\acute{r}}, \tag{2.6}$$

$$\sum_{i=0}^{n} \sum_{k=1}^{r} x_{iek} \geq \sum_{i=0}^{n} x_{isr} \qquad \forall e \in AND_s, r \in \{1, ..., p\}, \tag{2.7}$$

$$s_{ir} + 1 - M(1 - x_{ijr}) \leq s_{jr} \qquad \forall i \neq j \in N, r \in \{1, ..., p\}, \tag{2.8}$$

$$s_{ir} + 1 \leq s_{jr} \qquad \forall i \in Pr_j, r \in \{1, ..., p\}, \tag{2.9}$$

$$0 \leq s_{ir} \leq \sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{j=1}^{n} x_{ijr} \qquad \forall i \in N, r \in \{1, ..., p\}, \tag{2.10}$$

$$s_{ir} \geq 0 \qquad \forall i \in N, r \in \{1, ..., p\}, \tag{2.11}$$

$$x_{ijr}, u_r \in \{0,1\} \qquad \forall i, j \in N, r \in \{1, ..., p\}. \tag{2.12}$$

Equation (2.1) is the objective function that minimizes the total traveling costs. Equations (2.2) state that each trip starts from the depot. Equations (2.3) represent that each node is visited exactly once in just one trip. Constraints (2.4) declare that the vehicle, in each trip, enters and leaves a node exactly once. Vehicle capacity is guaranteed by equations (2.5) in each trip. Using constraints (2.6), the sequence of vehicle trips is determined so that trip r cannot be done unless trip $r-1$ has been performed before. Precedence constraints are met in the nodes allocations to trips as restricted in (2.7). It indicates that every predecessor of node s in trip r are assigned to the same trip or its previous ones. Constraints (2.8), which link the position variable s_{ir} and the binary variable x_{ijr} together, insure the order of nodes in each trip. Constraints (2.9) enforce the precedence constraints among the nodes inside each route. Constraints (2.10) limit the upper bound of the position variables. This constraint, along with (2.8), prevents the model from creating subtours. Finally, constrains (2.11) and (2.12) define the integer and binary variables, respectively.

2.3.2 Two-index MIP model

In this section, the Two-index MIP model for the proposed problem is developed. To assign the nodes to the trips, dummy depots are used as separators between vertices. It means the nodes positioned between every two depots form a

trip. Therefore, the size of the set $N = \{0,1,2,...,n\}$ increases by the maximum number of trips which leads to set $\acute{N} = \{0,1,...,n,n+1,...,n+p\}$ with $n+1+p$ nodes. It should be noted that all the traveling costs for these dummy depots are assumed to be the same as the original depot, i.e., $c_{0j} = c_{n+rj}, r \in 1,2,...,p$. In this formulation, to respect the precedence constraints, continuous variables $Po_i, \forall i \in \acute{N}$ are introduced, which indicate the nodes' positions in the overall sequence of all the nodes (including dummy depots) in the set $\acute{N}$, no matter in which trip a node is visited. The proposed variables and developed Two-index MIP model are represented as follows:

y_{ij}: binary variable takes value 1 if arc (i,j) is linked, 0 otherwise;
Po_i: integer variable indicating the position of node $i \in \acute{N}$;
Cd_i: integer variable indicating the cumulative demand of the trip at node $i \in \acute{N}$.

Then, the problem can be stated as:

$$Z2 = \min \sum_{\substack{i=0 \\ i \neq j}}^{n+p} \sum_{j=0}^{n+p} c_{ij} y_{ij} \tag{2.13}$$

subject to

$$\sum_{i=0, j \neq i}^{n+p} y_{ij} = 1 \qquad \forall j \in \acute{N} \setminus \{0\}, \tag{2.14}$$

$$\sum_{j=1, j \neq i}^{n+p} y_{ij} = 1 \qquad \forall i \in \acute{N} \setminus \{n+p\}, \tag{2.15}$$

$$Cd_i + d_j - M(1 - y_{ij}) \leq Cd_j \qquad \forall i \neq j \in \acute{N}, \tag{2.16}$$

$$d_i \leq Cd_i \leq Q \qquad \forall i \in \acute{N}, \tag{2.17}$$

$$Po_j + M(1 - y_{ij}) \geq Po_i + 1 \qquad \forall i \neq j \in \acute{N}, \tag{2.18}$$

$$1 \leq Po_i \leq |\acute{N}| \qquad \forall i \in \acute{N}, \tag{2.19}$$

$$Po_j \geq Po_i + 1 \qquad \forall i \in AND_j, \tag{2.20}$$

$$Po_i \geq 0 \qquad \forall i \in \acute{N}, \tag{2.21}$$

$$Cd_i \geq 0 \qquad \forall i \in \acute{N}, \tag{2.22}$$

$$y_{ij} \in \{0,1\} \qquad \forall i,j \in \acute{N}. \tag{2.23}$$

As usual objective function (2.13) minimizes the total travelling costs. Equations (2.14) and (2.15) are the regular routing rules. Constraints (2.16) and (2.17), derived from the well-known Miller-Tucker-Zemlin formulation, ensure the routes connectivity and limit the number of nodes in a trip considering the vehicle capacity. Constraints (2.18), which relate the two types of variables, ensure the order of linked nodes in the overall sequence of the total nodes. The position value is bounded by restriction (2.19). The precedence constraints are satisfied both inside and among the trips simultaneously by imposing constraints (2.20) since the variables have no index associated to the trips. Finally, constrains (2.21)-(2.23) define the integer and binary variables, respectively.

2.3.3 Integrated assignment and sequencing MIP model

In this section, we represent an Integration model containing distinct assignment and sequencing decisions. The variables and formulation are presented as follows:

x_{ir}: binary variable takes value 1 if node $i \in N$ is assigned to trip r, 0 otherwise;
y_{ijr}: binary variable 1 if arc(i, j) is linked in trip r, 0 otherwise;
po_{ir}: integer variable indicating the position of node $i \in N$ in trip r;
u_r: binary variable takes value 1 if the vehicle travels trip r, 0 otherwise.

Since each trip starts from and ends at the depot, the first position of each trip is associated with the depot while the last position is dedicated to the node, which directly precedes the depot at the last linked arc(i,0) in the trip r.

Then, the problem can be stated as:

$$Z3 = \min \sum_{r=1}^{p} \sum_{\substack{j=0 \\ i \neq j}}^{n} \sum_{i=0}^{n} c_{ij} y_{ijr} \tag{2.24}$$

subject to

$$\sum_{r=1}^{p} x_{ir} = 1 \qquad \forall i \in N \setminus \{0\}, \tag{2.25}$$

$$\sum_{i=1}^{n} d_i x_{ir} \leq Q.u_r \qquad \forall r \in \{1, ..., p\}, \tag{2.26}$$

$$x_{jr} = \sum_{i=0, i \neq j}^{n} y_{ijr} \qquad \forall j \in N, r \in \{1, ..., p\}, \tag{2.27}$$

$$x_{ir} = \sum_{j=0, j \neq i}^{n} y_{ijr} \qquad \forall i \in N, r \in \{1, ..., p\}, \tag{2.28}$$

$$u_r \leq u_{\acute{r}} \qquad \forall r, \acute{r} \in \{1, ..., p\}_{r > \acute{r}}, \tag{2.29}$$

$$\sum_{k=1}^{r} x_{ek} \geq x_{sr} \qquad \forall e \in AND_s, r \in \{1, ..., p\}, \tag{2.30}$$

$$po_{jr} + M(1 - y_{ijr}) \geq po_{ir} + 1 \qquad \forall i \neq j \in N, r \in \{1, ..., p\}, \tag{2.31}$$

$$1 \leq po_{ir} \leq \sum_{j=0, j\neq i}^{n} x_{jr} \qquad \forall i \in N, r \in \{1, ..., p\}, \tag{2.32}$$

$$po_{jr} \geq po_{ir} + 1 - M(2 - x_{ir} + x_{jr}) \qquad \forall i \in Pr_j, r \in \{1, ..., p\}, \tag{2.33}$$

$$po_{jr} \geq 0 \qquad \forall j \in N, r \in \{1, ..., p\}, \tag{2.34}$$

$$y_{ijr}, x_{ir}, u_r \in \{0,1\} \qquad \forall i, j \in N, r \in \{1, ..., p\}. \tag{2.35}$$

The objective function is represented by equation (2.24). Equations (2.25) guarantee that each node is allocated to exactly one trip. The vehicle capacity is ensured in each trip by equations (2.26). Equations (2.27) and (2.28) relate the two assignment and sequencing variables in a way that each node must be entered and exited exactly once in each trip. Using constraints (2.29), the sequence of vehicle trips is determined so that trip r cannot be done unless trip $r-1$ has been performed before. Precedence constraints are met in the nodes allocation to the trips as restricted in (2.30). Constraints (2.31) relate the two types of the variables to ensure the nodes' positions for the linked arc in each trip. The position variables are limited by restrictions (2.32). Equation (2.33) imposes precedence constraints within each trip using assignment and position variables. Finally, constrains (2.34) and (2.35) define the Integer and binary variables, respectively.

In the next subsection, a set of experiments are conducted to evaluate and compare the performance of the three presented models on a set of generated instances.

2.3.4 Comparison of the three MIP formulations

In this section, the performance of the three developed models is evaluated through an extended set of small instances proposed by Martinez-Salazar et al. [133]. The instances having 10, 15, 20, 25, and 30 nodes are randomly generated from points with real coordinates using a uniform distribution in the range [0,100]. Rounded Euclidean distances are taken as travel costs between each pair of nodes. The maximum number of trips is fixed in 2 for 10-nodes instances and 3 for other sizes. Depending on the size of the instance, Q was set to 120,120, 140, 180, 200. Demand d_i is randomly assigned with a value of 10, 20, or 30 so that the sum of the demands is between $(p+1)Q$ and pQ to ensure feasibility. There are 5 instances for each value of n.

An upper triangular matrix without the diagonal called Precedence Matrix (PM) is developed to represent the AND-type precedence constraints. Each element of

PM denotes whether or not a precedence relation exists between the two corresponding nodes. If node s have AND-type predecessors $\{i, j, k\}$, then $PM_{is} = 1$, $PM_{js} = 1$, $PM_{ks} = 1$. If there is no PC between the two nodes, the corresponding element of the matrix is zero. The representation of precedence constraints as an upper triangular matrix ensures the feasibility of the relations.

To generate the PCs, we somehow adopt the scheme proposed by Derriesel and Monch [55] who addressed the parallel machines with sequence-dependent setup times, precedence constraints, and ready times. The precedence relations are inserted using the factor $\tau = \{0,0.4,0.8\}$ to evaluate PCs' impact by considering three different scales. Given a column, if a chosen random number from $U[0,1]$ is less than τ, we do not consider any predecessors for the node associated with that column. Otherwise, the number of predecessors is chosen according to $U[0, n - 1^{th}]$, where n^{th} is the number of that column. Then, the predecessors are randomly selected from the set of already generated nodes $\{1^{th}, ..., n - 1^{th}\}$.

As a result, the total number of instances is 75 which is the combinations of the number of nodes $n \in \{10,15,20,25,30\}$, the scale of precedence constraints $\tau \in \{0,0.4,0.8\}$, and 5 generated instances for each combination of n and τ. All the models and codes are executed by GAMS 24.1 on corei5 pc with 2.50 GHz CPU and 4 GB of RAM, and the time limit is set to 5400 seconds.

Table 2.2 presents the results of experiments for comparing the three models. Column 1 displays the number of nodes of the instances, while column 2 indicates the scale of precedence constraints. Minimum, maximum, and average CPU time (in seconds) are shown from columns 3 to 5 for the three-index model, from columns 8 to 10 for two-index, and from columns 13 to 15 for integrated one. Columns 6, 11, and 16 show the number of instances (out of 5) which can be optimally solved (denoted by opt). A dash in these columns means that the model could not find optimal solutions for non of the 5 instances in the time limit. Values in columns 7, 12, 17 are the percentage of average optimality gap (denoted by $\Delta(\%)$) over non-optimized instances.

The results highlight way better performance of the Two-index model than the other two ones in terms of CPU time, the number of optimally solved instances, and the gap in dealing with more than 20 nodes. As it can be observed, the three-index and integrated formulations can not reach an optimal solution in the time limit 5400 seconds in dealing with instances with more than 25 and 20 nodes, respectively. However, the two-index formulation can optimally solve all the instances, while the three-index and integrated models can find optimal solutions for, respectively, 52 and 35 out of 75 instances.

Not surprisingly, by increasing the instances sizes, the percentage of averaged optimality gap for non-optimized instances grows except for little discontinuities. However, different behavior is noticed as the scale of PCs increases. It seems, having more PCs does not necessarily lead to more complexity of the problem, and consequently, growing optimality gap and the number of non-optimized instances.

Despite the quality obtained by the models, we also want to point out its efficiency. As expected, the CPU time grows as the size of the problem increases for the three models. Moreover, the performance of the three models in dealing with small-sized instances (up to 15 nodes) is similar. In contrast, for larger instances, it is clear that the two-index formulation can solve all instances in far less time than the two other ones.

Taking into account the various values for the scale of PCs, it can be seen that the models in dealing with different sizes do not show an exactly similar trend. It seems increasing or decreasing the number of PCs can not necessarily, affect the problem complexity. In some cases, more PCs lead to less computational time, and in some other cases, the opposite is observed, which means that not the number of PCs but the structure of them on a graph may increase or decrease the problem complexity. This topic has also been addressed in [161].

Table 2.2: Computational results of the three proposed MIP models.

Instance		Three-index model					Two-index model					Integrated model				
n	τ	CPU time (s)			opt	Δ(%)	CPU time (s)			opt	Δ(%)	CPU time (s)			opt	Δ(%)
		Min	Max	Avg			Min	Max	Avg			Min	Max	Avg		
10	0	5	8	7	5	0	5	6	5	5	0	5	7	6	5	0
10	0.4	5	7	6	5	0	5	6	5	5	0	6	7	7	5	0
10	0.8	6	8	7	5	0	5	7	6	5	0	6	8	7	5	0
15	0	15	28	20	5	0	13	19	15	5	0	16	32	23	5	0
15	0.4	12	32	19	5	0	15	30	22	5	0	17	47	30	5	0
15	0.8	18	36	25	5	0	15	27	19	5	0	20	35	27	5	0
20	0	449	830	638	5	0	58	191	113	5	0	577	1005	849	4	8
20	0.4	315	961	746	4	12	66	219	181	5	0	490	490	490	1	33
20	0.8	388	827	673	5	0	73	327	202	5	0	-	5400	5400	-	28
25	0	3800	4729	4201	3	16	360	844	608	5	0	-	5400	5400	-	19
25	0.4	1931	3855	3128	4	22	392	797	575	5	0	-	5400	5400	-	18
25	0.8	3669	3669	3669	1	27	466	991	831	5	0	-	5400	5400	-	35
30	0	-	5400	5400	-	19	1377	2894	2159	5	0	-	5400	5400	-	23
30	0.4	-	5400	5400	-	35	1026	2603	1844	5	0	-	5400	5400	-	29
30	0.8	-	5400	5400	-	30	1369	3082	2762	5	0	-	5400	5400	-	36

2.4 Logic-Based Benders Decomposition algorithm

In this chapter, the Logic-Based Benders Decomposition algorithm is implemented to address the original problem using a two-phase method called cluster-first and route-second [71]. The LBBD decomposes the original problem into an assignment master problem. The nodes are allocated to several required trips and independent sequencing subproblems with the special structure of the traveling salesman problem considering AND-type PCs. At each iteration of the LBBD, the master problem is solved and provides a lower bound for the original problem since it is a relaxation of the original problem. After solving the MP and specifying

the node assignments to the trips, the subproblems (associated with the trips) are derived, providing an upper bound. Using the solution of the subproblems, the optimality cuts are developed to be added to the master problem. Such a method converges to the optimum if and only if the master problem solution improves the lower bound, and the added cuts exclude the current master problem solution. This procedure goes on until an optimal solution is reached to the original problem or an early stopping criterion is found.

In the master problem and subproblems, the decisions are taken while satisfying the AND-type PCs among the trips and within them. More precisely, the nodes are allocated to the trips in the assignment master problem so that the AND-type PCs between the couples of nodes are respected among the trips. In contrast, the PCs are met within each trip by the subproblem associated with that trip.

2.4.1 Master problem

The master problem aims at making decisions on the nodes' assignment to several required trips satisfying PCs among the trips. So, it involves the binary decision variables $x_{ir}, i \in N, r \in \{1, ..., p\}$ which takes value 1 if node i is assigned to trip r, and the binary decision variables $u_r, r \in \{1, ..., p\}$ which takes value 1 if the vehicle travels trip r. The proposed master-problem is derived from the Integrated model, defined by constraints (2.25-2.35), and represented as follows:

Then, the problem can be stated as:

$$\min Z_{Master} \tag{2.36}$$

subject to

$$\sum_{r=1}^{p} x_{ir} = 1 \qquad \forall i \in N \setminus \{0\}, \tag{2.37}$$

$$\sum_{i=1}^{N} d_i \times x_{ir} \leq Q.u_r \qquad \forall r \in \{1, ..., p\}, \tag{2.38}$$

$$u_r \leq u_{\acute{r}} \qquad \forall r, \acute{r} \in \{1, ..., p\}_{r>\acute{r}}, \tag{2.39}$$

$$\sum_{k=1}^{r} x_{ek} \geq x_{sr} \qquad \forall r \in \{1, ..., p\}, e \in Pr_s, \tag{2.40}$$

$$Sub\text{-}Problem\ Relation, \tag{2.41}$$

$$Optimality\ Cuts, \qquad \forall l \in Iterations, \tag{2.42}$$

$$x_{ir}, u_r \in \{0,1\} \qquad \forall r \in \{1, ..., p\}, i \in N. \tag{2.43}$$

As it can be seen, the MP involves constraints on the nodes allocation, the vehicle capacity, order of trips, AND-type PCs among the trips, and sub-problem relaxation and benders cuts.

Recently, research proposed by Cire et al. [44] has shown that including the master problem with a relation of subproblem can significantly decrease the number of iterations as well as the computational time of LBBD as it provides a tighter lower bound for the original problem. Thus, we propose the following variables and the relaxed subproblem:

$y_{ijr} \in$ [0,1]: continuous variable indicating the arc(i, j) in trip r;

σ_r: integer variable for the lower bound on the travelling cost of trip r.

Then

$$\min \sum_{r=1}^{p} \sigma_r \tag{2.44}$$

subject to

$$x_{jr} = \sum_{i=0, i\neq j}^{n} y_{ijr} \qquad \forall j \in N, r \in \{1, ..., p\}, \tag{2.45}$$

$$x_{ir} = \sum_{j=0, j\neq i}^{n} y_{ijr} \qquad \forall i \in N, r \in \{1, ..., p\}, \tag{2.46}$$

$$\sigma_r \geq \sum_{\substack{i=0 \\ i\neq j}}^{n} \sum_{j=0}^{n} c_{ij} y_{ijr} \qquad \forall r \in \{1, ..., p\}, \tag{2.47}$$

$$Z_{Master} \geq \sum_{r=1}^{p} \sigma_r \tag{2.48}$$

$$x_{ir} \in \{0,1\} \qquad \forall i \in N, r \in \{1, ..., p\}, \tag{2.49}$$

$$0 \leq y_{ijr} \leq 1 \qquad \forall i, j \in N, r \in \{1, ..., p\}. \tag{2.50}$$

It should be noted that continuous variables y_{ijr} are enforced to take binary values 0 or 1 due to the constraints (2.45) and (2.46) which are related to the binary assignment variables x_{ir}. Using constrains (2.47), (2.48), the lower bound for the cost of each trip and the value of master problem are computed, respectively. Also, constrains (2.49), (2.50) define the binary and continuous variables, respectively.

It should be noted that the MP solutions may not be globally feasible in terms of the sequence of nodes in the trips since no constraints are imposed to eliminate the possible sub-tours and the PCs among the nodes within each trip.

2.4.2 Sub-problem

When the sub-optimal solution of the MP determines the number of required trips and the nodes assigned to them, each sub-problem associated with each trip can be seen as a traveling salesman problem with AND-type PCs. At each iteration, the sum of subproblems values (associated with the trips) provides an upper bound for the original problem. To formulate the subproblem, Alt_r^l and nb_r^l are defined, respectively, indicating the cluster (set) of nodes assigned to trip r and its total number computed by the solution of the master problem in iteration l. The two parameters are updated as the algorithm proceeds. For each iteration l and each trip r (with $u_r = 1$), the sub-problem model and the variables are represented as follows:

y_{ij}: binary variable takes value 1 if arc (i, j) is linked, 0 otherwise;
po_i: continuous variable indicating the position of node $i \in N$.

$$Z^l_{SP(r)} = \min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} y_{ij} \tag{2.51}$$

subject to

$$\sum_{i=0}^{n} y_{ij} = 1 \qquad \forall j \in Alt_r^l, \tag{2.52}$$

$$\sum_{j=0}^{n} y_{ij} = 1 \qquad \forall i \in Alt_r^l, \tag{2.53}$$

$$po_j - po_i + M(1 - y_{ij}) \geq 1 \qquad \forall i, j \in Alt_r^l, \tag{2.54}$$

$$po_j \geq po_i + 1 \qquad i, j \in Alt_r^l, i \in AND_j, \tag{2.55}$$

$$1 \leq po_i \leq nb_r^l \qquad i \in Alt_r^l, \tag{2.56}$$

$$po_j \geq 0 \qquad \forall j \in N, \tag{2.57}$$

$$y_{ij} \in \{0,1\} \qquad \forall i, j \in N. \tag{2.58}$$

As usual objective function (2.51) minimizes the total travelling costs for trip r. Equations (2.52) and (2.53) are the regular routing rules. Constraints (2.54), which relate the two types of variables, ensure the order of linked nodes in each trip. The precedence constraints are satisfied by imposing constrains (2.55). The position value is bounded by restriction (2.56). Constraints (2.54- 2.56) enforce the continuous variable $Po_i, \forall i \in N$ takes the integer values between 0 and nb_r^l. We empirically observed that considering Po_i as continuous variable instead of integer one can reduce the computational time of the subproblem. Finally, constrains (2.57), (2.58) define the continuous and binary variables, respectively.

2.4.3 Optimality Cuts

The cuts play a vital role in the convergence of the LBBD algorithm. Unfortunately, there is no systematic procedure to derive such cuts as in classical Benders Decomposition. Therefore, tailored cuts must be developed according to the studied problem. There is no need to add any feasibility cut during the search since the subproblems are feasible for any master problem.

Let $Z^l_{SP(r)}$ and σ_r denote, respectively, the optimal cost of the sequencing subproblem associated with trip r and iteration l, and the cost of trip r. A simple variant of optimality cut is

$$\sigma_r \geq Z^l_{SP(r)} - \sum_{i|x^l_{ir}=1} (1 - x_{ir}), \tag{2.59}$$

i.e., the MP gives the same solutions as before, $(1 - x_{ir})$ for all i becomes zero. It means a better result cannot be achieved as the objective function σ_r being limited by $Z^l_{SP(r)}$. So, the master problem is enforced to change the nodes allocations till a new set, which has not been repeated before, is found.

Several researchers have used (2.59) and its equivalent version in different optimization problems, like [176], [69] and [21]. Unfortunately, the above-mentioned optimality cut (2.59) does not perform well in practice since the cut only affects a small number of solutions in the MP. Recently, Faganello Fachini and Armentano [62] have proposed an extended form of cut (2.59) in dealing with the heterogeneous fixed fleet vehicle routing problem with time windows. Their developed cut can be written as

$$\sigma_r \geq Z^l_{SP(r)} - \sum_{i|x^l_{ir}=1} (1 - x_{ir})(2 \max_{j \neq i|x^l_{jr}=1} \{c_{ij}\}). \tag{2.60}$$

The cut indicates that if the current solution of the master problem does not include a node of a previously obtained trip, it reduces by twice the maximum travel cost over all pairs of nodes in this previously obtained route.

In this research, we try to strengthen cut (2.60) to affect more solutions instead of excluding few ones. We propose the following optimality cuts for the master problem

$$\begin{aligned} \sigma_r \geq Z^l_{SP(r)} - \sum_{i|x^l_{ir}=1} (1 - x_{ir})(max_1\{c_{ij}\} + max_2\{c_{ij}\}) + \\ \sum_{i|x^l_{ir}=0} x_{ir}(min_1\{c_{ij}\} + min_2\{c_{ij}\}), \end{aligned} \tag{2.61}$$

where $max_1\{c_{ij}\}$ and $max_2\{c_{ij}\}$ are the two highest, and $min_1\{c_{ij}\}$ and $min_2\{c_{ij}\}$ are the two lowest values of the traveling cost linked to node i. In comparison

with cut (2.60), a stronger lower bound for σ_r can be obtained in (2.61). This is due to considering not only the nodes are excluded from trip r (by subtracting the two associated highest cost) but also the ones are newly included to that trip (by adding the two associated lowest cost). In addition, considering the two highest and the two lowest cost might drive tighter lower bound (2.61), since

$$\underset{j\neq i|x^l_{jr}=1}{max_1}\{c_{ij}\} + \underset{j\neq i|x^l_{jr}=1}{max_2}\{c_{ij}\} \leq 2 \max_{j\neq i|x^l_{jr}=1}\{c_{ij}\}, \tag{2.62}$$

and

$$\underset{j\neq i|x_{jr}=1}{min_1}\{c_{ij}\} + \underset{j\neq i|x_{jr}=1}{min_2}\{c_{ij}\} \geq 2 \min_{j\neq i|x_{jr}=1}\{c_{ij}\}. \tag{2.63}$$

To prove the cut validity, we somehow follow the same procedure proposed by [62].

The Benders optimality cut communicates to the MP for two purposes: (1) eliminating the current sub-optimal MP solution (Theorem 1), (2) not removing the globally feasible solutions (Theorem 2).

Theorem 1. *The optimality cut* (2.61) *remove the current sub-optimal solution.*

Proof. Let $\sigma^l_r, r \in \{1, ..., p\}$ be the travelling cost of trip r by the solution of master problem in iteration l. Solving the subproblems associated with all trips, three possible situations can be happened:

- $\sum_{r=1}^{p} \sigma^l_r > \sum_{r=1}^{p} Z^l_{SP(r)}$, which means that the LBBD terminates and the value of upper bound is optimal for the original problem;
- $\sum_{r=1}^{p} \sigma^l_r = \sum_{r=1}^{p} Z^l_{SP(r)}$, which indicates that the LBBD terminates and the solution of master problem is optimal for the original problem;
- $\sum_{r=1}^{p} \sigma^l_r < \sum_{r=1}^{p} Z^l_{SP(r)}$, i.e., the current solution of master problem is not globally feasible (sub-optimal solution). In this case , the optimality cuts derived from subproblems are added to the master problem. Lets suppose that the same solution (nodes allocation to trips) of master problem is obtained in the subsequent iteration, then for each trip r, $\sum_{i|x^l_{ir}=1}(1 - x_{ir}) = 0$ and $\sum_{i|x^l_{ir}=0} x_{ir} = 0$. From (2.61), the optimality cut for each trip r is

$$\sigma_r \geq Z^l_{SP(r)}. \tag{2.64}$$

Then, we obtain

$$\sum_{r=1}^{p} \sigma_r \geq \sum_{r=1}^{p} Z^l_{SP(r)} > \sum_{r=1}^{p} \sigma^l_r, \tag{2.65}$$

which means that the current solution obtained in iteration l is removed from the master problem space.

□

Theorem 2. *The optimality cut* (2.61) *does not exclude any globally feasible solutions in future iterations.*

Proof. We assume that there exists a globally feasible trip not satisfying the optimality cut and then showing a contradiction. Let Z^* denote the optimal cost of trip r associated with the cluster of customers H^*, which was assigned to that trip through a globally feasible solution for the original problem. Let $\bar{Z}$ be the optimal cost of the same trip associated with cluster $\bar{H}$, which was found in iteration l. We define three sets of nodes:

- $\phi_1 = \{H^* \setminus \bar{H}\}$: nodes in H^* not in $\bar{H}$;
- $\phi_2 = \{H^* \cap \bar{H}\}$: nodes in both H^* and $\bar{H}$;
- $\phi_3 = \{\bar{H} \setminus H^*\}$: nodes in $\bar{H}$ not in H^*.

An optimality cut (2.61) is generated for trip r associated with cluster $\bar{H}$ in iteration l. Let's assume that trip r obtained from the cluster H^* violate the optimality cut. Then, we have

$$Z^* < \bar{Z} - \sum_{i\in\bar{H}}(1-x_{ir})(\underset{j\neq i\in\bar{H}}{max_1}\{c_{ij}\} + \underset{j\neq i\in\bar{H}}{max_2}\{c_{ij}\}) + \sum_{i\notin\bar{H}} x_{ir}(\underset{j\neq i\in H^*}{min_1}\{c_{ij}\} + \underset{j\neq i\in H^*}{min_2}\{c_{ij}\}). \tag{2.66}$$

For the nodes in set ϕ_2, $\sum_{i\in\phi_2}(1-x_{ir}) = 0$ in the globally feasible Route(H^*). Since Route (H^*) dose not include the nodes in set ϕ_3, therefore, $x_{ir} = 0$ for all $i \in \phi_3$, and $\sum_{i\in\phi_3}(1-x_{ir}) = |\phi_3|$. Moreover, set ϕ_1 contains the nodes not included in $\bar{H}$. So, (2.66) can be presented as

$$Z^* < \bar{Z} - \sum_{i\in\phi_3}(\underset{j\neq i\in\bar{H}}{max_1}\{c_{ij}\} + \underset{j\neq i\in\bar{H}}{max_2}\{c_{ij}\}) + \sum_{i\in\phi_1}(\underset{j\neq i\in H^*}{min_1}\{c_{ij}\} + \underset{j\neq i\in H^*}{min_2}\{c_{ij}\}). \tag{2.67}$$

Note that if $\phi_2 = \emptyset$, then $\phi_3 = \bar{H}$ and $\phi_1 = H^*$. We can define an upper bound for $\bar{Z}$ as

$$\bar{Z} \leq \sum_{i\in\phi_3}(\underset{j\neq i\in\phi_3}{max_1}\{c_{ij}\} + \underset{j\neq i\in\phi_3}{max_2}\{c_{ij}\}), \tag{2.68}$$

including the two highest costs of node $i \in \phi_3$ linked to other nodes in that set, and a lower bound for Z^* as

$$Z^* \geq \sum_{i\in\phi_1}(\underset{j\neq i\in\phi_1}{min_1}\{c_{ij}\} + \underset{j\neq i\in\phi_1}{min_2}\{c_{ij}\}), \tag{2.69}$$

involving the two lowest cost of node $i \in \phi_1$ linked to other nodes in that set. So, equation (2.67) can be written as

$$Z^* < \bar{Z} - \sum_{i \in \phi_3} (max_1\{c_{ij}\}_{j \neq i \in \phi_3} + max_2\{c_{ij}\}_{j \neq i \in \phi_3}) + \sum_{i \in \phi_1} (min_1\{c_{ij}\}_{j \neq i \in \phi_1} + min_2\{c_{ij}\}_{j \neq i \in \phi_1}). \quad (2.70)$$

From (2.68), the value of $\bar{Z} - \sum_{i \in \phi_3}(max_1\{c_{ij}\}_{j \neq i \in \phi_3} + max_2\{c_{ij}\}_{j \neq i \in \phi_3})$ is equal to or less than zero. Therefore, according to (2.70), Z^* is less than a negative value plus $\sum_{i \in \phi_1}(min_1\{c_{ij}\}_{j \neq i \in \phi_1} + min_2\{c_{ij}\}_{j \neq i \in \phi_1})$ which leads to a contradiction with the lower bound of Z^* defined in (2.69).

Otherwise ($\phi_2 \neq \emptyset$), the contradiction (2.67) is more involved as we subsequently represent.

Given the globally feasible trip derived from a cluster of nodes, it is possible to reach a reduced or an extended route by removing or adding the nodes to that cluster. Let's consider the trip associated with cluster H^* with cost Z^*. By removing the nodes in ϕ_1 from H^*, Trip(ϕ_2) and the respective cost Z_{ϕ_2} is obtained. Removing the nodes $i \in \phi_1$ from Trip(H^*) results in a cost smaller than or equal to Z^*, since travel costs satisfy the triangular inequality, therefore,

$$Z_{\phi_2} = Z_{H^* \setminus \phi_1} \leq Z^* - \sum_{i \in \phi_1} (min_1\{c_{ij}\}_{j \neq i \in H^*} + min_2\{c_{ij}\}_{j \neq i \in H^*}). \quad (2.71)$$

The above equation indicates that removing nodes in ϕ_1, consequently, subtracting the two lowest cost of node i linked to node $j \in H^*$ from the value of Z^* defines an upper bound for the cost Z_{ϕ_2}.

Accordingly, (2.71), using the contradiction (2.67), becomes:

$$\begin{aligned} Z_{\phi_2} &\leq Z^* - \sum_{i \in \phi_1} (min_1\{c_{ij}\}_{j \neq i \in H^*} + min_2\{c_{ij}\}_{j \neq i \in H^*}) \leq Z^* \\ &\leq \bar{Z} - \sum_{i \in \phi_3} (max_1\{c_{ij}\}_{j \neq i \in \bar{H}} + max_2\{c_{ij}\}_{j \neq i \in \bar{H}}) \\ &+ \sum_{i \in \phi_1} (min_1\{c_{ij}\}_{j \neq i \in H^*} + min_2\{c_{ij}\}_{j \neq i \in H^*}). \end{aligned} \quad (2.72)$$

Now let us extend Trip(ϕ_2) to Trip($\phi_2 \cup \phi_3$) with cost $Z_{\phi_2 \cup \phi_3}$ by inserting the nodes belongs to ϕ_3 into Trip(ϕ_2). Consider Trip(ϕ_2) represented by the sequence of nodes $(0, ..., g, o, ..., z, 0)$. Let's assume that the node $i \in \phi_3$ is inserted into Trip(ϕ_2) such that $max_1\{c_{ij}\}_{j \neq i \in \phi_2} + max_2\{c_{ij}\}_{j \neq i \in \phi_2}$ is the sum of the two highest cost of $c_{i0}, ..., c_{ig}, c_{io}, ..., c_{iz}$. The insertion cost of node $i \in \phi_3$ between nodes g and o is given by $c_{gi} + c_{io} - c_{go}$. Moreover, observe that $c_{gi} - c_{go} \leq c_{oi}$, since the triangular inequality holds. Therefore, the insertion cost of node $i \in \phi_3$ between nodes g and o yields

$$c_{gi} + c_{io} - c_{go} \leq c_{gi} + c_{io} \leq \underset{j \neq i \in \phi_2}{max_1}\{c_{ij}\} + \underset{j \neq i \in \phi_2}{max_2}\{c_{ij}\}\,. \tag{2.73}$$

Proceeding this way for all nodes in ϕ_3, we obtain Trip$(\phi_2 \cup \phi_3)$ such that

$$Z_{(\phi_2 \cup \phi_3)} \leq Z^* + \sum_{i \in \phi_3} (\underset{j \neq i \in \phi_2}{max_1}\{c_{ij}\} + \underset{j \neq i \in \phi_2}{max_2}\{c_{ij}\}) - \sum_{i \in \phi_1} (\underset{j \neq i \in H^*}{min_1}\{c_{ij}\} + \underset{j \neq i \in H^*}{min_2}\{c_{ij}\}). \tag{2.74}$$

Since $\phi_2 \subset (\phi_2 \cup \phi_3) = \bar{H}$, for all $i \in \phi_3$, one has

$$\underset{j \neq i \in \phi_2}{max_1}\{c_{ij}\} + \underset{j \neq i \in \phi_2}{max_2}\{c_{ij}\}) \leq \underset{j \neq i \in \bar{H}}{max_1}\{c_{ij}\} + \underset{j \neq i \in \bar{H}}{max_2}\{c_{ij}\}). \tag{2.75}$$

Hence, (2.74) can be written as

$$Z_{(\phi_2 \cup \phi_3)} \leq Z^* - \sum_{i \in \phi_1} (\underset{j \neq i \in H^*}{min_1}\{c_{ij}\} + \underset{j \neq i \in H^*}{min_2}\{c_{ij}\}) + \sum_{i \in \phi_3} (\underset{j \neq i \in \bar{H}}{max_1}\{c_{ij}\} + \underset{j \neq i \in \bar{H}}{max_2}\{c_{ij}\}). \tag{2.76}$$

Regarding (2.76), one gets

$$Z^* \geq Z_{(\phi_2 \cup \phi_3)} + \sum_{i \in \phi_1} (\underset{j \neq i \in H^*}{min_1}\{c_{ij}\} + \underset{j \neq i \in H^*}{min_2}\{c_{ij}\}) - \sum_{i \in \phi_3} (\underset{j \neq i \in \bar{H}}{max_1}\{c_{ij}\} + \underset{j \neq i \in \bar{H}}{max_2}\{c_{ij}\}). \tag{2.77}$$

As Route$(\bar{H})$ is optimal for the cluster $\bar{H} = \phi_2 \cup \phi_3$ with cost $\bar{Z}$, then

$$Z_{(\phi_2 \cup \phi_3)} \geq \bar{Z}. \tag{2.78}$$

From (2.77) and (2.78) we obtain

$$Z^* \geq \bar{Z} + \sum_{i \in \phi_1} (\underset{j \neq i \in H^*}{min_1}\{c_{ij}\} + \underset{j \neq i \in H^*}{min_2}\{c_{ij}\}) - \sum_{i \in \phi_3} (\underset{j \neq i \in \bar{H}}{max_1}\{c_{ij}\} + \underset{j \neq i \in \bar{H}}{max_2}\{c_{ij}\}). \tag{2.79}$$

Equation (2.79) contradicts (2.72) and, consequently, the optimality cut (2.61) does not remove globally feasible solutions. □

2.5 Computational experiments

This section evaluates the performance of the proposed LBBD algorithm against the two-index MIP model, the best formulation among the three proposed ones as shown in section 2.3.4. We test three different versions of the LBBD algorithm.

The first one denoted by LBBD1 incorporates the optimality cut (2.60) proposed recently by Faganello Fachini and Armentano [62]. In LBBD2, our developed cut, represented as (2.61), is applied. The two algorithms LBBD1 and LBBD2 are the exact approaches in which both the MP and the SP are optimally solved (optimality gap of zero). LBBD3 is a heuristic version of LBBD2 in which the SP is optimally solved, yet the MP is solved for at most 5 seconds or 5% optimality gap (whichever comes first). This change to the master problem results in it no longer being a true lower bound. Such a stopping criterion was first proposed in [211] under which they speeded up the MP by allowing the solver to stop once a solution is found within some predetermined gap from the best lower bound obtained. However, they indicated that the quality of the solution found is within the chosen optimality gap. Moreover, Barzanji et al. [21] has recently improved their proposed LBBD by defining a gap of 5% for solving the SP in dealing with the integrated process planning and scheduling problem. As suggested in [211] and [21] and, also according to preliminary experiments, a gap of 5% is chosen in this research, which results in a good trade-off between computational time and quality.

The model and the algorithms are coded in GAMS 24.1 on corei5 pc with 2.50 GHz CPU and 4 GB of RAM, and the time limit is set to a maximum of 5400 seconds. In section 2.5.1, we describe the instances generated as a testbed for our assessment. Our computational experiment results are described and commented on in section 2.5.2.

2.5.1 Instance generation

Since we could not find a standard benchmark for the proposed problem, our testbed is generated by extending the same method described in [13] to deal with the CVRP. To build our instances, we assume that there is a single vehicle with the possibility of multiple trips. Then, we take the proposed number of vehicles as the maximum number of trips in our problem. Twenty-seven instances have nodes ranging from 32 to 80 randomly generated from points with real coordinates using a uniform distribution in the range [0,100]. Rounded Euclidean distances are taken as travel costs between each pair of nodes. The vehicle capacity Q is set to 100 for all instances. Demand $d_i, \forall i \in N \setminus \{0\}$ is picked from an uniform distribution $U(1,30)$, however $n/10$ of those demands are multiplied by 3.

For each instance, the Precedence Matrix (PM) is generated using three different values of scale $\tau = \{0,0.4,0.8\}$ to evaluate the PCs' impact on the complexity of instances. The procedure is the same described in section 2.3.4. As a result, the total number of instances is $27 \times 3 = 81$, as each instance is solved with three proposed scales of precedence constraints $\tau \in \{0,0.4,0.8\}$. All the instances are solved using the Two-index MIP model and the three algorithms LBBD1, LBBD2, and LBBD3.

2.5.2 Comparison of the Two-index MIP model and the three LBBD algorithms

In this section, we summarize and discuss our experimental results to evaluate the performance of the proposed algorithms compared to the Two-index MIP model on the sets of instances described in Section 2.5.1.

Tables 2.3-2.5 represents the results of experiments associated to the three proposed scales of PCs. In all the tables, column 1 displays the number of nodes, while column 2 indicates the proposed maximum number of trips for each instance. The CPU time (in seconds) is shown in columns 3, 5, 7, and 9, respectively, for the Two-index model, LBBD1, LBBD2, and LBBD3. A dash in these columns indicates the proposed time limit under which the model was unable to find optimal solutions. For each instance, the optimality gap is calculated with the following formula.

$$optimality\ gap = \frac{Upper\ bound - Lower\ bound}{Upper\ bound} \times 100. \tag{2.80}$$

Values in columns 4, 6, 8, and 10 represents the optimality gap (denoted by $\Delta(\%)$), respectively for the Two-index model, LBBD1, LBBD2, and LBBD3.

The first thing that should be noticed is that the results of the three tables highlight way better performance of the LBBD3 comparing the other approaches as it can heuristically solve all the instances in less computational time. Comparing the two algorithms LBBD1 and LBBD2 indicates the higher efficiency of the LBBD2 capable of solving almost all instances in less time and even some instances which can not be optimally dealt with the LBBD1. This is because of the cut (2.61) used in LBBD2, which can provide a tighter lower bound for the master problem than optimality cut (2.60) applied in the LBBD1. Not surprisingly, the Two-index model can only solve smaller instances with more CPU time than the LBBD algorithms for all the proposed PC scales.

Moreover, an increasing trend in the problem complexity can be observed as the size of instances grows in all three tables. More precisely, raising the number of nodes and the maximum number of trips may lead to more CPU time and the optimality gap, except for a few cases where the size of instances is close together. As expected, the maximum trip number parameter can affect the problem complexity more than the number of nodes in most cases for all the proposed solution approaches.

Considering different values of the PC scale, it can be observed that the complexity of instances is not necessarily affected by increasing or decreasing the PC scale value. In some instances, a larger value of PC scale leads to low computational complexity, consequently less CPU time, and in some other ones, the opposite happens. This behavior is the same for all the proposed approaches. For instance, let's consider the instance with 48 nodes and at most 7 trips solved using algorithm

Table 2.3: Computational results of the LBBD1, LBBD2, LBBD3 and the Two-index model considering $\tau = 0$.

Instances		Two-index model		LBBD1		LBBD2		LBBD3	
n	p	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)
32	5	1804	0	835	0	746	0	470	0
33	5	2117	0	794	0	762	0	496	0
33	6	-	13	863	0	859	0	509	0
34	5	-	11	807	0	693	0	431	0
36	5	3943	0	783	0	754	0	457	0
37	5	-	10	915	0	833	0	524	0
37	6	-	11	986	0	875	0	648	0
38	5	-	12	887	0	835	0	502	0
39	5	4820	0	936	0	917	0	530	0
39	6	-	12	973	0	940	0	597	0
44	6	-	14	1035	0	987	0	661	0
45	6	-	13	1090	0	1006	0	735	0
45	7	-	14	1271	0	998	0	672	0
46	7	-	16	1183	0	1055	0	704	0
48	7	-	22	1256	0	1107	0	682	0
53	7	-	24	1319	0	1143	0	704	0
54	7	-	23	1338	0	1117	0	756	0
55	9	-	28	1952	0	1736	0	962	0
60	9	-	31	1844	0	1771	0	1035	0
61	9	-	29	1996	0	1550	0	964	0
62	8	-	27	1638	0	1300	0	711	0
63	9	-	34	-	12	2644	0	805	0
63	10	-	38	-	8	2930	0	946	0
64	9	-	35	1950	0	1733	0	1061	0
65	9	-	36	-	10	1852	0	1143	0
69	9	-	39	1826	0	1677	0	972	0
80	10	-	37	-	14	-	9	2605	0

LBBD2. As shown, the CPU time is equal to 1107 when no PC is considered, while the time decreases once $\tau = 0.4$ and increases when $\tau = 0.8$. It seems, in this case, the instance complexity reduces considering precedence relations in comparison with the non-PC problem. As mentioned in section 2.3.4, not the number of PCs but their structure on the graph may affect the problem's complexity differently. This topic has also been addressed in [161]. According to their research, many problems can be NP-hard when considering general precedence constraints, while they become polynomially solvable for particular structures of precedence constraints.

2.6 Summary

In this chapter, the multi-trip single-vehicle routing problem is studied. The nodes associated with customers/target locations are related to each other through AND-type precedence constraints. Our interest originates from problems, such as

Table 2.4: Computational results of the LBBD1, LBBD2, LBBD3 and the Two-index model considering $\tau = 0.4$.

Instances		Two-index model		LBBD1		LBBD2		LBBD3	
n	p	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)
32	5	2289	0	972	0	815	0	397	0
33	5	1837	0	858	0	739	0	466	0
33	6	3035	0	793	0	645	0	503	0
34	5	-	14	927	0	732	0	628	0
36	5	4273	0	740	0	712	0	413	0
37	5	-	9	856	0	784	0	475	0
37	6	-	10	891	0	839	0	536	0
38	5	3792	0	863	0	807	0	570	0
39	5	5008	0	957	0	840	0	637	0
39	6	-	14	1016	0	917	0	716	0
44	6	-	16	993	0	846	0	741	0
45	6	-	13	1072	0	968	0	780	0
45	7	-	17	1335	0	1152	0	833	0
46	7	-	20	1494	0	1305	0	1025	0
48	7	-	18	1039	0	876	0	668	0
53	7	-	22	1266	0	1115	0	916	0
54	7	-	25	1392	0	1248	0	1037	0
55	9	-	29	1773	0	1566	0	1243	0
60	9	-	33	1827	0	1681	0	1469	0
61	9	-	30	1875	0	1736	0	1370	0
62	8	-	28	1745	0	1624	0	1288	0
63	9	-	31	-	15	-	9	2726	0
63	10	-	35	-	6	2072	0	1655	0
64	9	-	29	1713	0	1405	0	1168	0
65	9	-	26	-	16	-	7	2339	0
69	9	-	35	1645	0	1490	0	1173	0
80	10	-	39	-	17	-	11	2902	0

package delivery or picker routing problems. Some nodes have priorities to be visited after a set of other ones within and among the routes due to their emergency, importance, or physical features. Despite the AND-type PCs applications in real-life routing problems, non of the literature on logistics, even picker routing problems, focuses on these relations.

First, we develop and experimentally compare three mathematical formulations to address the problem. The computational results validate the significant superiority of the developed Two-index model in terms of computational time and problem size compared to the two other ones. Then, the problem is handled by proposing a solution approach based on the logic-based benders decomposition algorithm. The developed approach decomposes the original problem into an assignment master problem and sequencing subproblems. Moreover, a new optimality cut is provided, and its validity is proven. The performance of the optimality cut is experimentally investigated by comparing that with a recently proposed cut in the Literature.

Table 2.5: Computational results of the LBBD1, LBBD2, LBBD3 and the Two-index model considering $\tau = 0.8$.

Instances		Two-index model		LBBD1		LBBD2		LBBD3	
n	p	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)	CPU time (s)	Δ(%)
32	5	1998	0	950	0	923	0	551	0
33	5	2218	0	981	0	576	0	352	0
33	6	3249	0	883	0	806	0	714	0
34	5	-	16	956	0	851	0	693	0
36	5	4540	0	722	0	647	0	478	0
37	5	-	12	826	0	713	0	639	0
37	6	-	8	875	0	822	0	540	0
38	5	-	14	836	0	783	0	633	0
39	5	4712	0	972	0	852	0	689	0
39	6	-	17	990	0	964	0	616	0
44	6	-	15	1072	0	907	0	762	0
45	6	-	16	1039	0	932	0	750	0
45	7	-	19	1378	0	1105	0	961	0
46	7	-	20	1422	0	1213	0	1037	0
48	7	-	23	1156	0	1044	0	972	0
53	7	-	26	1437	0	1267	0	958	0
54	7	-	23	1335	0	1105	0	966	0
55	9	-	27	1822	0	1579	0	1203	0
60	9	-	38	1905	0	1601	0	1355	0
61	9	-	29	2016	0	1871	0	1290	0
62	8	-	28	1590	0	1358	0	1083	0
63	9	-	33	-	11	-	8	2295	0
63	10	-	30	3057	0	2472	0	1832	0
64	9	-	34	1582	0	1338	0	1172	0
65	9	-	30	-	15	-	8	2489	0
69	9	-	31	1937	0	1824	0	1362	0
80	10	-	36	-	13	-	9	2630	0

Additionally, we presented a relaxed version of LBBD by defining a limit for optimality gap and CPU time in deriving master-problem solutions. In such a way, the algorithm's efficiency improves as it allows the algorithm to find a feasible solution to the original problem in less CPU time and even larger instances. The performance of proposed LBBD algorithms is evaluated and compared together through extensive computational experiments. The results show that the two exact LBBDs can solve most instances, while the relaxed version of LBBD can heuristically solve all the generated instances in a shorter computational time.

Chapter 3

Vehicle routing problem with AND/OR precedence constraints and time windows

3.1 Introduction

The Vehicle Routing Problem (VRP) is a well-known problem in the Operations Research field. A given set of customers is served by a fleet of vehicles minimizing routing costs and respecting capacity constraints on vehicles. To better represent real-world problems, additional features need to be considered leading to different variants of the VRP. A popular extension of the VRP considers time window constraints, assuming that serving a given customer must occur in a limited range. Time windows are either hard when it is prohibited to deliver outside the time interval or soft, allowing deliveries outside the boundaries against a penalty cost.

In many real industrial or social service environments where a set of tasks or services are performed, partial orders known as Precedence Constraints (PCs) are defined to represent the dependency of tasks on each other. In general cases, precedence constraints are represented by an arbitrary directed acyclic graph in which each arc corresponds to a precedence constraint between a pair of nodes. In practice, there are cases in which alternative definitions for PCs, such as AND-type, OR-type, SOFT-type and S-type are defined. The 'AND' precedence constraint means that a node can be reached only after all of its predecessors. The 'OR' precedence constraint means that a node can be reached when at least one of its predecessors has been met before. The problem containing both 'AND' and 'OR' types is said to have AND/OR precedence constraints.

In many real-life routing problems, serving customers may include delivery or picking up items that need to be performed in a particular order due to some reasons. They could be physical restrictions such as fragility, stackability, shape, size, and weight or preferred loading or unloading sequences to avoid extra effort on sorting the collected items at the end of the retrieving process. These restrictions may impose constraints on the visiting sequence represented by the AND/OR predecessors associated with each customer.

In this chapter, we introduce a new variant of the VRP referred to as the vehicle routing problem subject to the AND/OR precedence constraints and time windows to minimize the total traveling and service time. Our interest in this problem originates from the picker routing problem, where the items are stored and retrieved manually with the order-picking sequences represented as AND/OR precedence constraints. In such a problem, to deal with various storage and replenishment policies that describe the availability of items like food or perishable products, time window limitations are defined for retrieving the items.

As mentioned in previous chapter, no research in logistics area refers to nodes visited according to AND/OR precedence constraints to the best of our knowledge. The Dial-a-Ride and the Pickup and Delivery problems are the most related routing problems that includes the most straightforward form of PCs. They are represented as a single pairwise relation between pickup and delivery (drop-off) points (we refer to them as conventional PCs). For instance, in the Dial-A-Ride problem, for any backhaul node j, there is a particular inhaul node i where $(i < j)$ must be met within any route. In such problems, any node has at most one predecessor that must be visited before it.

We formulate the problem as a Mixed Integer Linear Programming (MILP) model to optimally solve small-sized instances. A meta-heuristic algorithm based on Iterated Local Search (ILS) and Simulated Annealing (SA) approaches is developed and tuned using the Taguchi method [153] to effectively deal with problems in a reasonable time.

The contribution of this study is threefold: (i) the Vehicle Routing Problem with AND/OR Precedence Constraints and Time Windows is described; (ii) to formulate the problem, a MILP model is developed to solve small-scale instances optimally; (iii) we design and implement a meta-heuristic algorithm capable of obtaining reasonable solutions in terms of both quality and computational time.

This chapter is organized as follows. Section 3.2 is dedicated to the literature review of the problem. The proposed problem is described and formulated in Sections 3.3 and 3.4, respectively. In Section 3.5, we provide the meta-heuristic algorithm to address the problem. The computational results of the MILP model and the proposed algorithm are represented in Section 3.6. Finally, the chapter ends with a summary and interesting future research suggestions in Section 3.7.

3.2 Literature

Dantzig performed the first research on the VRP and Ramser [49]. Since then, thousands of papers on different variations of this problem have appeared in the logistics field literature. One of the extensions of this problem, encountered in many real-life applications, is defined as the Vehicle Routing Problem with Time Windows (VRPTW). Several customers are served within predefined time windows. Solomon and Desorios initially proposed this problem [192]. The excellent surveys by Toth and Vigo [210] and Kumar and Panneerselvam [105] have detailed the literature of VRPTW solution approaches. Also, Dixit et al. [56] have reviewed some of the recent advancements in solving VRPTW using various meta-heuristic techniques. They are Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony algorithm (ABC), etc. Fan and Feng [68] applied the hill-climbing algorithm to improve the genetic algorithm to address VRPTW. They could effectively increase the speed and globally optimal solution quality of the algorithm. Ye et al. [226] addressed the time-dependent vehicle routing problem with time windows by developing a multi-type ant system algorithm hybridized with the ant colony system and the max-min ant system. The nearest neighbor selection mechanism, an insertion local search procedure, and a local optimization procedure are applied to improve the efficiency of the insertion procedure. The particle tabu search algorithm designed by Schneider et al. [183] can significantly improve the computational efficiency of the VRPTW problem and give a complete Pareto foreword. Yang et al. [224] proposed the chaotic particle swarm optimization algorithm, which improves the speed, robustness, and speed of the solution of the VRPTW problem.

In many real operational research problems where a set of tasks or services are performed, it is tough to suppose that the tasks are independent of each other. In principle, the relative order between a couple of tasks is represented as a pairwise relation named Precedence Constraint (PC). These relations make the problems have a more comprehensive range of applications in various fields. The PCs arise whenever one activity or series of activities must be performed before beginning another activity or set of activities. Many examples can be mentioned, such as the assembly industry in which activities are carried out on products in different stations, logistics, construction projects, production scheduling, and maintenance support.

In the context of machine scheduling, Goldwasser and Motwani [83] derive inapproximability results for a specific single-machine scheduling problem with AND/OR precedence constraints. Gillies and Liu [80] addressed single and parallel machine scheduling problems to meet deadlines considering different structures of AND/OR precedence constraints. They proved NP-completeness of finding feasible schedules in many polynomially solvable settings with only AND-type precedence constraints. Moreover, they give priority-driven heuristic algorithms to minimize

the completion time on a multiprocessor. Mohring and Skutella [138] provided some algorithms for the more general and complex model of AND/OR precedence constraints. They showed that feasibility and questions related to generalized transitivity could be solved using essentially the same linear-time algorithm. Moreover, they discussed a natural generalization of AND/OR precedence constraints and prove that the same problems become NP-complete in this setting. Lee et al. [108] focused on flexible job-shop scheduling problems with AND/OR precedence constraints in the operations. They provided a MILP model, which can be used to compute optimal solutions for small-sized instances. They also developed a heuristic algorithm that results in a good solution for the problem regardless of its size. Moreover, a schedule builder who always gives a feasible solution and genetic and tabu search algorithms based on the proposed schedule builder were presented. Van Den Akker et al. [216] developed a solution framework that provides feasible schedules to minimize some objective function of the minimax type on a set of identical parallel machines subject to release dates, deadlines, AND/OR precedence constraints. They determined a high-quality lower bound by applying column generation to the LP relaxation.

In the context of routing problems, Moon et al. [140] addressed the traveling salesman problem with precedence constraints (TSPPC). The pairwise PCs form an order under which the nodes are visited. A genetic algorithm that involves a topological sort and a new crossover operation is proposed to solve the model. Savelsbergh and Sol [181] presented the TSPPC model to solve the Dial-A-Ride problem where a vehicle should transport several passengers. Each passenger should be transported from a given location to a given destination. Mingozzi et al. [135] dealt with the TSP with time windows and precedence constraints using a dynamic programming approach. Fagerholt and Christiansen [67] considered a TSPPC with a time window to solve the bulk ship scheduling problem. The model is solved as the shortest path problem on a graph. Renaud et al. [165] proposed a heuristic model to solve the pickup and delivery TSP formulated as the TSPPC. Bredstrom and Ronnqvist [30] developed a mathematical model for the combined vehicle routing and scheduling problem with time windows. The sets of pairwise synchronization and precedence constraints are considered between customer visits, independently of the vehicles. Also, they described some real-world problems to emphasize the importance of the mentioned constraints, such as homecare staff scheduling, airline scheduling, and forest operations. Bockenhauer et al. [29] studied a variant of TSP in which a given subset of nodes are visited in a prescribed order in the computed Hamiltonian cycle. They presented a polynomial-time algorithm to solve the problem. Haddadene et al. [162] modeled a home health care structure as a variant of vehicle routing problem with time windows and timing constraints. Some patients ask for more than one visit simultaneously or in given priority order. A MILP model, a greedy heuristic, two local search strategies, and three metaheuristics are proposed to solve the problem. Recently, the task assignment problem for a team

of heterogeneous vehicles has been investigated in which packages are delivered to a set of dispersed customers subject to precedence constraints. Using graph theory, a lower bound on the optimal time is constructed. Integrating with a topological sorting technique, several heuristic algorithms are developed to solve it [17].

The order-picking problem is one of the main applications of PCs in the context of routing problems. However, little works in order-picking problems have focused on PCs. Zulj et al. [239] considered the PCs in a warehouse of a German manufacturer of household products, where heavy items are not allowed to be stored on top of delicate items to prevent damage to the delicate items. To avoid the sorting effort at the end of the order-picking process, they propose a picker-routing strategy respecting the precedence constraints. An exact algorithm based on dynamic programming is used to evaluate the strategy and compared with the simple s-shape routing strategy. Dekker et al. [51] investigated combinations of storage assignment strategies and routing heuristics for a real case arising in a warehouse of a wholesaler of tools and garden equipment. A guideline has to be considered indicating that fragile products have to be picked last. Matusiak et al. [134] presented a simulated annealing method to address the joint order batching and precedence-constrained picker-routing problem in a warehouse with multiple depots. The shortest path through the warehouse is determined using the exact algorithm developed by Hart et al. (1968). Chabot et al. [39] introduced the order-picking routing problem underweight, fragility, and category constraints. They propose capacity-index and two-index vehicle-flow formulations as well as four heuristics to solve the problem. Furthermore, a branch-and-cut algorithm is applied to solve the two mathematical models.

As noticed in the previous chapter, the proposed precedence relations in most picker routing problems (mentioned above) have been represented as a pre-specified sequence of nodes. However, in our proposed problem, AND/OR PCs are defined under which the nodes assignments to vehicles and the sequence of nodes in each route are determined in the model.

The background study shows no available research in the literature of VRPs and even picker routing problems cover AND/OR precedence constraints. Thus, our research contributes to the literature by describing, modeling and providing a solution approach for the capacitated vehicle routing problem with AND/OR PCs and time windows.

3.3 Problem description

The proposed problem is defined as a capacitated vehicle routing problem. A set of nodes are visited using a fleet of homogeneous vehicles available at the depot in time zero, with capacity Q. Each vehicle can make one single trip during the planning time horizon. The problem can be represented on a directed graph $G = (N, A)$, where $N = \{0,1, ..., n, \acute{0}\}$ is a set of geographically located nodes including

the depot (node 0) and a dummy depot (node 0́), and $A = \{(i,j)|i,j \in N\}$ is a set of arcs. Each arc $(i,j) \in A$ is defined by a traveling time t_{ij}. For each node $i \in N \setminus \{0, 0́\}$, demand q_i, service time s_i, and a (hard) time window $[e_i, l_i]$ are given where e_i is the earliest possible arrival time and l_i is the latest possible one. Arriving at node i before e_i leads to a waiting time at this node. On the other side, late arrival at the node (after l_i) is not allowed. A time horizon T is given and establishes the working day. It can be viewed as a time window $[e_0, l_0] = [0, T]$ associated with the depot, which means the routes cannot start before e_0 and must be back to the depot up to time l_0.

Moreover, the nodes are related together by defining AND/OR precedence constraints. These relations must be met among the nodes visited by each vehicle. Given node i, two sets denoted by AND_i and OR_i are defined, including the AND-type and OR-type predecessors of node i, respectively. All predecessors $j \in AND_i$, visited by the same vehicle as node i, must be served in any positions before node i on the trip. Regarding OR-type precedence constraints, at least one of the predecessors $j \in OR_i$ needs to be served before node i by the same vehicle.

As an example, let's consider the feasible solution depicted in Figure 3.1. It can be seen that three vehicles are used to visit all the nodes. According to the defined PCs, the set of AND-type predecessors of node r includes $\{g, c, j\}$. As depicted, nodes g and c are visited before node r using vehicle 1, while node j is not, as it is not assigned to the same vehicle as node r. Concerning OR-type predecessors of node a, node k is visited before node a, while the corresponding precedence constraints $h < a$ and $t < a$ are not met. This is due to the definition of OR-type PCs, which denotes that given a node, at least one of its OR-type predecessors needs to be met before.

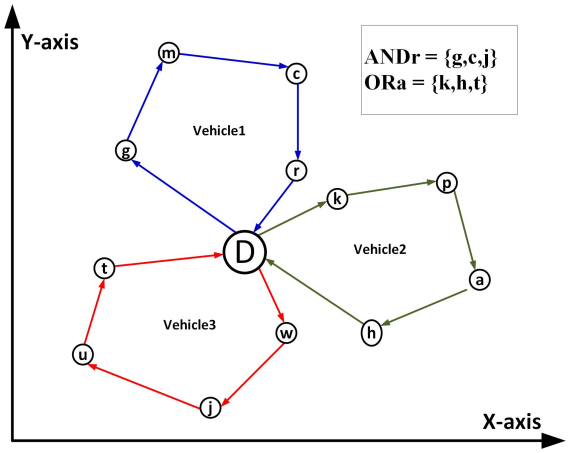


Figure 3.1: Illustration of an example

The proposed problem calls for determining an optimal trip of each vehicle to minimize the total traveling and service time to serve all the customers through

network edges.

The assumptions of this problem are as follows:

- each vehicle starts and ends the trip at the depot;
- each node is visited exactly once within its time window;
- the vehicles must be back to the depot up to time T;
- maximum number of used vehicles is $|K|$;
- vehicle capacity Q cannot be exceeded in each trip;
- the arrival time of each node must meet the time window limitations;
- AND/OR PCs are met between the nodes within each route.

In the following section, the proposed notations and the MILP model are presented to address the problem.

3.4 Mathematical model

Let us introduce the following notation:

- $N = \{0,1,2,...,n,\acute{0}\}$: set of nodes, (each vehicle starts the route from depot (node 0) and ends to the dummy depot (node $\acute{0}$));
- t_{ij}: travelling time from node i to node j;
- T: time horizon;
- Q: vehicle capacity;
- q_i: demand of node i;
- s_i: service time of node i;
- $K = \{1,2,...,f\}$: set of available homogenous vehicles;
- AND_i: set of AND-type predecessors of node i;
- OR_i: set of OR-type predecessors of node i;
- $[e_i, l_i]$: time window associated with the arrival time of node i, (e.i., arriving earlier than e_i introduces a waiting time at node i; arriving after l_i leads to infeasibility);
- M: an arbitrary large constant.

The variables and the proposed MILP model are presented as follows:

- y_{ijk}: binary variable takes value 1 if node i is visited before node j (not necessarily immediately) by vehicle k, 0 otherwise;
- z_{ik}: binary variable takes value 1 if node i is visited by vehicle k, 0 otherwise;
- u_k: binary variable takes value 1 if vehicle k is used;
- a_{ik}: continuous variable indicating the arrival time of node i visited by vehicle k;
- c_k: continuous variable indicating the completion time of the route performed by vehicle k.

Then, the MILP model becomes:

$$Z = \sum_{k=1}^{f} c_k \tag{3.1}$$

subject to

$$\sum_{k=1}^{f} z_{ik} = 1 \qquad \forall i \in N \setminus \{0, \acute{0}\}, \tag{3.2}$$

$$\sum_{i=1}^{n} q_i.z_{ik} \leq Q.u_k \qquad \forall k \in K, \tag{3.3}$$

$$z_{ik} = u_k \qquad \forall i \in \{0, \acute{0}\}, k \in K, \tag{3.4}$$

$$u_k \leq u_{\acute{k}} \qquad \forall k, \acute{k} \in K_{k>\acute{k}}, \tag{3.5}$$

$$a_{0k} = 0 \qquad \forall k \in K, \tag{3.6}$$

$$a_{ik} \leq M.z_{ik} \qquad \forall i \in N \setminus \{0, \acute{0}\}, k \in K, \tag{3.7}$$

$$c_k \geq a_{\acute{0}k} \qquad \forall k \in K, \tag{3.8}$$

$$a_{ik} \geq e_i.z_{ik} \qquad \forall i \in N \setminus \{0\}, k \in K, \tag{3.9}$$

$$a_{ik} \leq l_i.z_{ik} \qquad \forall i \in N \setminus \{0\}, k \in K, \tag{3.10}$$

$$a_{jk} + M(1 - y_{ijk}) \geq a_{ik} + t_{ij} + s_i - M(1 - z_{ik}) - M(1 - z_{jk}) \qquad \forall i \neq j \in N, k \in K, \tag{3.11}$$

$$1 + M(z_{ik} + z_{jk} - 2) \leq y_{ijk} + y_{jik} \qquad \forall i \neq j \in N, k \in K, \tag{3.12}$$

$$z_{ik} + z_{jk} \geq 2(y_{ijk} + y_{jik}) \qquad \forall i \neq j \in N, k \in K, \tag{3.13}$$

$$y_{ijk} - 1 \leq M(2 - z_{ik} - z_{jk}) \qquad \forall i \in AND_j, k \in K, \tag{3.14}$$

$$1 - y_{ijk} \leq M(2 - z_{ik} - z_{jk}) \qquad \forall i \in AND_j, k \in K, \tag{3.15}$$

$$\sum_{i \in OR_j} y_{ijk} \geq 1 + M(z_{jk} - 1) \qquad \forall j \in N, k \in K, \tag{3.16}$$

$$c_k, a_{ik} \geq 0 \qquad \forall i \in N, k \in K, \tag{3.17}$$

$$z_{ik}, y_{ijk}, u_k \in \{0,1\} \qquad \forall i, j \in N, k \in K. \tag{3.18}$$

The objective function in equation (3.1) minimizes the total completion time. Equations (3.2) guarantee that each node (except depot and dummy depot) is visited by exactly one vehicle. The vehicle capacity is ensured for each trip by equations (3.3). Equations (3.4), which link variables z and u, indicate that depot and dummy depot are both assigned to all the used vehicles. Using constraints (3.5), the sequence of used vehicles is determined so that vehicle k cannot be used unless vehicle $k-1$ has been started the route before. Equations (3.6) indicate that the depot's arrival time on each vehicle trip is equal to zero. Constraints (3.7) relate the two assignment and arrival time variables z and a in a way that if node i is not assigned to vehicle k, the corresponding arrival time of that node on that vehicle is zero. Constraints (3.8) ensure that each vehicle trip's completion time is larger than the arrival time of the dummy depot on that vehicle. Constraints (3.9) and (3.10) ensure that the nodes arrival time satisfy the time windows restrictions. As represented by constraints (3.11), if nodes i is visited before node j on a route performed by the same vehicle k, the arrival time of node j is larger than the sum of arrival time at node i, service time at this node and the travel time t_{ij}. Constraints (3.12) and (3.13) ensure that if node i and j are both assigned to the same vehicle k then node i is visited either before or after node j not necessarily immediately. Otherwise, the corresponding variables y_{ijk} and y_{jik} are equal to zero. Constraints (3.14-3.15) ensure respecting the AND-type precedence constraints among the nodes visited by each vehicle such that if node i is an AND-type predecessor of node j and both nodes are assigned to the same vehicle k, node i is visited before node j which indicates $y_{ijk} = 1$. The OR-type precedence constraints are satisfied using constraints (3.16) which imply that, given node j visited by vehicle k, at least one of its OR-type predecessors must be visited before node j by the same vehicle. Finally, constrains (3.17) and (3.18) define the continuous and binary variables, respectively.

3.5 Solution approach

In this section, our developed algorithm to address the proposed problem is introduced. Because of the high complexity of optimization problems, often exact algorithms are capable only for the smaller instances and spent a lot of computational time (see [112]). In contrast, meta-heuristics can find near-optimal solutions

for the instances with realistic sizes, generally with less computation time. Therefore, we concentrate on designing an effective and efficient meta-heuristic algorithm instead of exact methods.

In this research, the proposed approach is a hybridization of the Iterated Local Search (ILS) and Simulated Annealing (SA), which complements the advantages of both ILS and SA in a single optimization framework. Recently, ILS and SA have been hybridized to cope with the search space of complex optimization problems. Experimentally, it was found that the performance of the hybrid algorithm is better than that of SA and ILS algorithms when implemented individually (see, e.g., [131], [163], and [88]). In the following, we start by describing the general schemes of the proposed algorithm. Then, the different components of this approach are represented in detail.

3.5.1 General scheme

The general scheme of the proposed hybrid approach is represented in algorithm 1. This algorithm starts with an initial solution, denoted by SOL, generated from a constructive algorithm described in section 3.5.2. This starting point undergoes the main loop (lines 4-22) repeated until a maximum number of iterations given by Max_{iter} is reached. The loop contains four distinct parts: Perturbation Procedure (PP), Local Search (LS), check *Stack* set, and updating the best solution SOL^*.

More specifically, the current solution is first perturbed at each iteration associated with the algorithm's destruction phase. This process starts by recognizing some target routes and modifying the solution by transferring their nodes to the not-target ones. Then, the remaining target vehicles are removed from the solution, and the corresponding nodes are gathered in a set referred to as the *Stack*. More details on PP are included in Section 3.5.3.

After the destruction phase, the resulting partial solution undergoes the local search to reconstruct the solution to obtain a feasible one finally. This process is performed by randomly exploring different neighborhoods and reinserting *stack* nodes to the active vehicles. In this phase, the move acceptance criterion of Simulated Annealing is incorporated with LS, which provides another way of avoiding the local optima to enhance the performance. The details on the LS are provided in Section 3.5.4.

At the end of the local search loop, the current solution's feasibility is investigated by checking the *stack* set that contains the nodes removed from the solution during the PP and have not been reinserted during the iterations of LS. In case of an empty *stack* set, the current solution would be feasible and can be saved as the best one if the current solution cost $F(SOL)$ is lower than the cost of the best solution $F(SOL^*)$ already found in the algorithm. Otherwise, in the non-empty *stack* set, a new vehicle may need to be added to the current partial solution. This process is implemented, taking into account parameter *vehicle number*, which reports the

total number of already used (active) vehicles. This parameter is updated during the algorithm by adding or eliminating each vehicle. Constructing the new vehicle route is carried out in a way that the nodes in the *stack* are randomly selected, one by one, and checked to be assigned to the new vehicle till the route gets full (which means considering all feasibility aspects, no additional *stack* node can be inserted to that trip).

Algorithm 1 General procedure of the proposed hybrid algorithm

```
1: Generating an initial solution SOL
2: SOL* ← SOL
3: iter ← 1
4: while iter < Max_iter do
5:     Perturbation Procedure
6:     Local Search
7:     if stack is not empty then                    ▷ Check Stack Set
8:         if vehicle number < |K| then
9:             add a new vehicle
10:            vehicle number ++
11:            Assign stack nodes to the new vehicle.
12:            GO TO Local Search.
13:        else
14:            Keep the stack
15:            Go to PP
16:        end if
17:    end if
18:    if F(SOL) < F(SOL*) then                      ▷ Update Best Solution
19:        SOL* ← SOL
20:    end if
21:    iter ← iter + 1
22: end while
```

The algorithm then starts the local search process again to search the neighborhoods and give the remaining *stack* nodes to be inserted into the current partial solution. In case of non-empty *stack* set, if parameter *vehicle number* is larger than the maximum number of vehicles (K), the algorithm preserves *stack*. It goes back to the Perturbation Procedure to destruct the current partial solution and rebuilt it through the LS process. The following subsections are devoted to the different parts of the proposed algorithm.

3.5.2 Building initial solution

In this section, the procedure for generating an initial solution (SOL) is introduced. This approach constructs a solution sequentially by building the routes associated with the vehicles one after another. This process is performed by initializing an individual vehicle that travels a route during the time horizon $[0, T]$. The nodes from a sorted list of candidates are selected to be assigned to that vehicle one by one. They are located in positions one after another to form a route up to time T. Whenever inserting none of the unvisited nodes to the current position leads to a feasible partial solution, the depot is used to end the current trip. A new vehicle is initialized.

The partial solution is becoming complete over the stages as the number of unvisited nodes is reducing. Finally, an initial solution is obtained, which contains all the nodes that have been allocated to several vehicles (routes) that might be larger than the maximum number of available vehicles. The nodes' assignments to the vehicles are performed, considering that the selection does not violate the time horizon, vehicle capacity, time windows, and the proposed precedence constraints. However, the initial solution's feasibility in terms of the maximum number of active vehicles is achieved during the next stages of the algorithm, as described in the next sections.

In this context, we introduce parameter $AllPRE_i$, $\forall i \in N \setminus \{0, \acute{0}\}$ representing the total number of AND/OR predecessors of node i. Moreover, a *candidate set* is constructed made of all the unvisited nodes sorted in increasing order of parameter $AllPRE$. By sorting this set, the predecessors are more likely assigned before the successors. During the nodes insertion process, the value of parameters $AllPRE$, the size and order of *candidate set* are updating. When a new vehicle is added, the *candidate set* is reconstructed using the unvisited vertices and their corresponding $AllPRE$. Also, this set is repeatedly updated as a node insertion is performed. Every individual node from the beginning of *candidate set* is checked whether its assignment to the current vehicle leads to a feasible solution or not. Whenever a node is found whose insertion to the current position is feasible, the movement is performed.

The feasibility of each node insertion is checked in three consecutive stages. In the first stage, the partial solution's feasibility in terms of the vehicle capacity and the time horizon is investigated. In contrast, in the second and third stages, the precedence constraints and time windows are taken into account, respectively. Two parameters are defined for each vehicle, which specify the cumulative demand of already assigned nodes to that vehicle (denoted by CD) and the trip's completion time (denoted by CT).

Algorithm 2 Constructing an initial solution

1: **for all** $i \in N \setminus \{0\}$ **do**
2: $AllPRE_i \leftarrow |AND_i| + |OR_i|$
3: $VISITED(i) \leftarrow 0$
4: **end for**
5: candidate set $\leftarrow N \setminus \{0\}$
6: *vehicle number* $\leftarrow 1$
7: *current position* $\leftarrow 1$
8: **while** candidate set $\neq \emptyset$ **do**
9: Sort candidate set in increasing order of $AllPRE_i$
10: *insertion process* $\leftarrow FALSE$
11: **for** $i = 1$ to |candidate set| **do**
12: Choose node i ▷ Check feasibility of inserting node i
13: feasibility feedback = TRUE
14: stage 1:
15: **if** ($CD_{current\ position}$ > vehicle capacity) and ($CT_{current\ position} > T$) **then**
16: feasibility feedback = FALSE
17: GO to line 11 and choose the next node
18: **end if**
19: stage 2:
20: **if** AND/OR PCs corresponding to node i are not met **then**
21: feasibility feedback = FALSE
22: GO to line 11 and choose the next node
23: **end if**
24: stage 3:
25: **if** ($e_i < A_i$) or ($A_i > l_i$) **then**
26: feasibility feedback = FALSE
27: GO to line 11 and choose the next node
28: **end if**
29: **if** feasibility feedback = TRUE **then**
30: Insert node i to current position
31: *insertion process* $\leftarrow TRUE$
32: current position ++
33: $VISITED(i) \leftarrow 1$
34: Remove node i from candidate set
35: **for all** $j \in$ candidate set **do** ▷ Update candidate set
36: **if** $PC(i < j) = AND/OR$ *type PC* **then**
37: $AllPRE_j - -$
38: **end if**

Algorithm 2 Constructing an initial solution (continued)

```
39:                 if PC(j < i) = AND type PC then
40:                     Remove j from candidate set
41:                 end if
42:             end for
43:             Break and GO to line 8
44:         end if
45:     end for
46:     if insertion process = FALSE then
47:         Add a new vehicle
48:         vehicle number++
49:         current position ← 1
50:         for all i ∈ N \ {0} do                    ▷ reconstruct candidate set
51:             if VISITED(i) = FALSE then
52:                 Add i in candidate set
53:             end if
54:         end for
55:         Go to line 8
56:     end if
57: end while
```

For each new node insertion, the two parameters CD and CT are checked whether this operation leads to a partially feasible solution in terms of the vehicle capacity and time horizon. In case of a node insertion's infeasibility, the depot is placed at the end of the corresponding route, which means the vehicle ends the trip by going back to the depot.

Other feasibility aspects include precedence constraints and time windows. For each node j in the *candidate set*, the feasibility of corresponding precedence constraints is investigated in terms of the AND/OR PCs by considering the two following conditions:

- Inserting node j, an AND-type predecessor of any of the already assigned nodes on the route, leads to an infeasible solution.
- Inserting node j when at least one of its OR-type predecessor has not visited yet on the route leads to an infeasible solution.

Finally, the insertion of node j in the *candidate set* is checked by considering the time windows limitations so that node j can be inserted to the current position if its arrival time meets its associated time windows.

As a node insertion is performed, the candidate set needs to be updated. Given node j insertion to vehicle k, all the AND-type predecessors of node j, which have

not been visited yet, will be removed from the *candidate set*. This is due to the definition of AND-type PCs, which indicates that a vehicle cannot visit a successor before its predecessors. Also, all successors of node j, which have not been assigned to any vehicles yet, are updated in the candidate set by subtracting constant value one from their corresponding $AllPRE$. Then, the candidate set is sorted again, which gives the nodes, whose predecessors have been allocated already, the chance to be assigned sooner. This procedure stops when all the nodes are assigned to the current vehicle, or no more nodes can be inserted (due to feasibility conditions). In the second case, a new vehicle is added to the current partial solution. The above explanations on the generation of the initial feasible solution are represented as pseudo-code in algorithm 2.

3.5.3 Perturbation procedure

After constructing an initial solution, the algorithm starts a loop where the current solution is first perturbed with the aim of escaping from local optima. In the perturbation phase, the vehicles whose trips are not appropriate for the proportion of total traveling and service time and the number of assigned nodes are recognized and referred to as target vehicles. This process is performed using parameter $I_k = \frac{Traveling\ and\ service\ time_k}{Number\ of\ nodes_k}$ representing the ratio of the total traveling and service time to the number of assigned nodes associated to vehicle k. The vehicles whose index I_k are higher than and equal to $I_{threshold}$ computed as equation (3.19) are considered as target ones and may need to be modified.

$$I_{threshold} = \frac{Average\ traveling\ and\ service\ time}{Average\ number\ of\ assigned\ nodes} \tag{3.19}$$

In equation (3.19), the average values of total traveling and service time and several nodes are taken over all active vehicles.

After recognizing the target routes, the perturbation procedure is performed in two successive steps: pre-improvement and route removal. During the first step, every individual node in target vehicles is randomly chosen and checked to transfer to one of the not-target vehicles. Whenever a not-target vehicle is found to remove the node from its current position and insert it to the new position is feasible, it is performed. The pre-improvement step aims at emptying and finally eliminating the target trips as their nodes are transferred to the not-target ones. This process continues till transferring all the target nodes are investigated, and no improvement in terms of reducing the total number of nodes in the target trips can be achieved. Any time a movement is accepted according to the feasibility test described in section 3.5.5, the entire solution, including the target and non-target vehicles and their assigned nodes, is updated.

Algorithm 3 Perturbation Procedure of the proposed hybrid algorithm

```
Perturbation Procedure
for k = 1 to vehicle number do
    Compute I_k
    if I_k > I_threshold then
        Vehicle k is a target vehicle
    end if
end for
for all target vehicles do
    for all target nodes do
        Transferring target node to a not-target route
    end for
end for
Sorting the vehicles in decreasing order of I_k
Eliminating the first |vehicle number − |K|| + 1 vehicles
Constructing stack set using the nodes of the removed vehicles
```

After modifying the solution in the pre-improvement step, the current solution might contain several target vehicles. In the second phase (routes removal), the algorithm destructs the solution by eliminating the first $|vehicle\ number - |K|| + 1$ vehicles from the set of all vehicles sorted in decreasing order of I_k. The corresponding nodes of the removed vehicles are also gathered in the *stack* set. After PP, the algorithm undergoes the local search iterations, described in subsection 3.5.4, where the current partial solution is improved as long as the *stack* nodes are reinserted to the active vehicles or the newly added ones. The above explanations on PP are represented as pseudo-code in algorithm 3.

3.5.4 Local search

In this section, the proposed local search approach attempts to construct a feasible solution by repeatedly searching several specified neighborhoods and reinserting the *stack* nodes to the current partial solution simultaneously. This process stops when either the *stack* set is empty or the maximum number of tries made without success in improving the current solution (given by Max_{notImp}) is reached. At each iteration t of the LS, logical parameter $Improve$ checks whether or not any improvements in terms of the objective function or reinserting nodes from the *stack* set to the current partial solution is achieved. If no improvement is obtained, the parameter $iter_{notImp}$ is updated.

Most neighborhoods used in the vehicle routing problems are based on one or usually more nodes exchanges or relocations inside or across the vehicles. In this context, we apply five neighborhoods widely used in the vehicle routing problems as follows (see [157]):

- **Transferring within a vehicle:** where a node is transferred from its current position to another position in the same vehicle.
- **Transferring across the vehicles:** where a node is transferred from its current position to another vehicle.
- **Exchange within a vehicle:** where two nodes that belong to the same vehicle are exchanged.
- **Exchange across the vehicles:** where two nodes that belong to different vehicles are exchanged.
- **Insert a vehicle:** select a node from one vehicle and create a route associated with a new vehicle with it if parameter *vehicle number* is less than K.

During the LS procedure, the proposed five neighborhoods of the current solution are explored iteratively in a random sequence provided by a list. For example, the following sequence [3,1,5,2,4] implies that neighborhood 3 is the first and neighborhood 4 is the final one to be explored at each iteration of LS.

Preliminary tests have indicated that looking for the best feasible move needs much time with no real overall improvement of the solution cost than a first improvement strategy. It should be noted that only those moves cannot lead to an infeasible solution concerning all the feasibility constraints.

To empower the LS in searching different space regions, Simulated Annealing (SA) is implemented which allows non-improving moves and avoid being trapped in a local minima. In the SA approach, a variable threshold value named *SArate* is calculated as:

$$SArate = \exp\left(\frac{F(\acute{s}) - F(s)}{Temp}\right), \quad F(\acute{s}) \geq F(s) \tag{3.20}$$

where $F(s)$ and $F(\acute{s})$ are the objective function values of the current and the new solutions, respectively. The initial temperature $Temp$ is exponentially decreased by a fraction α denoted by the cooling rate, where $Temp_{t+1} = \alpha \times Temp_t$. In general, the SA begins with a high-temperature value, and it gradually decreases during the search. The SA performance depends on the two factors: initial temperature value $Temp$ and the cooling rate α calibrated in Section 3.6.2.

For each feasible move, the origin's value and destination vehicles before and after replacement are computed. The move is accepted if it results in the lower value in terms of objective function, i.e. $(F(\acute{s}) \leq F(s))$. Otherwise, the move does not improve the current solution, and it is only accepted if a random value in the range of (0,1) is less than the *SArate*.

Algorithm 4 Local Search of the proposed hybrid algorithm

```
1: Local Search
2: t ← 1
3: iter_notImp ← 1
4: while (stack is not empty) and (iter_notImp < Max_notImp) do
5:     Improve=FALSE
6:     Generate a sequence of neighborhoods randomly
7:     for n = 1 to 5 do
8:         Explore the neighborhood n^th
9:         if F(current solution) > F(new solution) or random number < SArate
   then
10:            Move is accepted
11:            Improve=TRUE
12:        else
13:            Move is not accepted.
14:        end if
15:    end for
16:    for all stack nodes do
17:        Choose a node randomly
18:        for all vehicles do
19:            Choose a vehicle randomly
20:            for all positions in the route do
21:                if Inserting the node in the position is feasible then
22:                    Insert the node
23:                    Improve=TRUE
24:                    Break and go to line 19
25:                end if
26:            end for
27:        end for
28:    end for
29:    if Improve=FALSE then
30:        iter_notImp ← iter_notImp + 1
31:    end if
32:    t ← t + 1
33:    Temp ← α × Temp
34: end while
```

After navigating different search space regions, adding the *stack* nodes to the current partial solution is performed at each iteration of the local search cycle. The *stack* nodes are randomly chosen, one by one, and checked to be inserted into the current partial solution. Given a *stack* node, a vehicle is randomly selected to be allocated that node. This vehicle is chosen from all active vehicles that have not

been checked before for that particular node. Chosen the vehicle, all the possible positions are assessed to be assigned the node one after another. Whenever a vehicle and a position inside it are found under which the node insertion is feasible, the movement from the *stack* to that position is implemented. This procedure aims at emptying the *stack* set to construct a feasible solution finally. The above explanations on LS are represented as pseudo-code in algorithm 4.

3.5.5 Feasibility test

Since in many heuristic algorithms, millions of movements are evaluated during the search process, their feasibility must be checked as efficiently as possible. In our proposed problem, due to various side constraints such as AND/OR precedence constraints, time windows, and vehicle capacity, checking the feasibility of moves leads to prohibitive computation time. So, a critical factor of a heuristic algorithm performance is assessing the feasibility of a solution quickly. In this section, we will describe how to handle the side constraints for the considered neighborhoods efficiently.

As introduced in section 3.5.4, five different moves are proposed as nodes transferring (forward or backward) or exchanging within or across the vehicles and adding new vehicles. Each move may lead to one or more operations like node removal (deletion), node insertion, forward and/or backward transfer on the corresponding vehicles. Whenever a move's feasibility is checked, a screening procedure is performed on the partial solution to return either true or false feedback. It should be noticed that when nodes transferring or exchanges are carried out within or across the routes, the feasibility conditions are only checked for the partial solution, which includes the corresponding origin and destination trips.

The feasibility test screening procedure includes three consecutive stages corresponding to checking the vehicle capacity restriction, AND/OR precedence relationships, and the time windows associated with the nodes and the depot. During the feasibility test execution, whenever an infeasibility is detected, false feedback is returned, and the procedure avoids checking other feasibility aspects in the remaining stages of the test. The proposed order has shown good performance in our preliminary experiments since the test starts from the feasibility aspects with less computational complexity to the most.

In the first stage, a partial solution's feasibility is verified concerning vehicle capacity. To do so, given a move that includes a node insertion or replacement to a position on the specific vehicle's trip, the cumulative demand of the already assigned nodes and the new one on that particular trip cannot exceed the capacity restriction.

The screening procedure checks a partial solution's feasibility in the second stage according to the precedence constraints. The cases must be considered to check the feasibility of the proposed moves concerning the AND/OR precedence constraints

for each proposed neighbourhoods are represented as:

- **Transferring forward within a route:** Node i placed in position p is transferred to a forward position $\acute{p}$ on the route.

  case 1: node i should not be an AND-type predecessor of any node between location p and $\acute{p}$;

  case 2: node i should not be only OR-type predecessor of any node between location p and $\acute{p}$.

- **Transferring backward within a route:** Node i placed in position p is transferred to a backward position $\acute{p}$ on the route.

  case 1: nodes between location p and $\acute{p}$ should not be AND-type predecessor of node i;

  case 2: nodes between location p and $\acute{p}$ should not be only OR-type predecessor of node i.

- **Transferring across routes:** Node i placed in position p on route k is transferred to position $\acute{p}$ on route $\acute{k}$.

  case 1: node i should not be only OR-type predecessor of any nodes placed after position p on route k ;

  case 2: node i should not be AND-type predecessor of any node placed before position $\acute{p}$ on route $\acute{k}$;

  case 3: node i should not be AND-type successor of any node placed after position $\acute{p}$ on route $\acute{k}$.

- **Exchange within a route:** Node i placed in position p is exchanged with node j located in position $p > p$ on the same route.

  case 1: node i should not be the AND-type predecessor of node j.

  case 2: node i should not be AND-type predecessor of any node placed between position p and $\acute{p}$ on the route;

  case 3: node i should not be only OR-type predecessor of any node placed between position p and $\acute{p}$ on the route;

  case 4: nodes between location p and $\acute{p}$ should not be AND-type predecessor of node j;

  case 5: nodes between location p and $\acute{p}$ should not be only OR-type predecessor of node j.

- **Exchange across routes:** Node i placed in position p on route k is exchanged with node j located in position $\acute{p}$ on route $\acute{k}$.

case 1: node i should not be AND-type predecessor of any node placed before position $\acute{p}$ on route $\acute{k}$;

case 2: node i should not be AND-type successor of any node placed after position $\acute{p}$ on route $\acute{k}$;

case 3: node i should not be only OR-type predecessor of any node placed after position p on route k;

case 4: node j should not be AND-type predecessor of any node placed before position p on route k;

case 5: node j should not be AND-type successor of any node placed after position p on route k;

case 6: node j should not be only OR-type predecessor of any node placed after position $\acute{p}$ on route $\acute{k}$.

- **Insert a vehicle:** Node i placed in position p on route k is removed and create a new route $\acute{k}$.

  case 1: node i should not be only OR-type predecessor of any node placed after location p on route k;

  case 2: node i should not have any OR-type predecessor.

In the third stage, the partial solution's feasibility is checked concerning the time windows limitations. This procedure is based on the principle of push backward and forward proposed by Kindervater and Savelsberg [102] for the Vehicle Routing Problems with Time Windows (VRPTW) where the moves are direction preserving.

Consider a path $(u, u+1, ..., v)$ with associated arrival times of the nodes. Lets suppose that the arrival time of the first node in the path is decreased. This defines a push backward as

$$B_u = A_u - A_u^{new}, \tag{3.21}$$

Where A_u and A_u^{new} define the current and new arrival time at vertex u. The push backward at the next vertex on the path is calculated as

$$B_{u+1} = \min\{B_u, A_{u+1} - e_{u+1}\}. \tag{3.22}$$

As long as $B_k > 0$, all nodes on the path remain feasible, and their associated arrival times need to be adjusted sequentially for $k = u, ..., v$.

Algorithm 5 Time windows feasibility test of forward transferring within a route

Description: node i placed in position p is transferred to a forward position $\acute{p}$ on the route. The direct successor of node i is denoted by $\sigma(i)$.
step 1: compute the values of push backward for the nodes placed between positions $p+1$ and $\acute{p}-1$ and update their associated arrival times;
step 2: compute arrival time of node i at new position $\acute{p}$;
step 3: compute arrival time of node $\sigma(i)$ and determine the type of push for the successors of node i;
step 4: compute the arrival times of the successors of node i placed after position $\acute{p}+1$;
step 5: check if the new arrival times meet the time windows limitations;
step 6: check if the depot time window is met.

Algorithm 6 Time windows feasibility test of backward transferring within a route

Description: node i placed in position p is transferred to a backward position $\acute{p}$ on the route. The direct successor of node i is denoted by $\sigma(i)$.
step 1: compute arrival time of node i at new position $\acute{p}$;
step 2: compute the values of push forwards for the nodes placed between positions $\acute{p}+1$ and $p-1$ and update their associated arrival times;
step 3: compute arrival time of node $\sigma(i)$ and see the push is forward or backward. Then, update arrival times for the successors of node i;
step 4: check if the new arrival times meet the time windows limitations;
step 5: check if the depot time window is met.

Algorithm 7 Time Windows Feasibility test of transferring across routes

Description: node i placed in position p on route k is transferred to position $\acute{p}$ on route $\acute{k}$.
step 1: update the arrival times of the nodes placed after position $p+1$ on route k using the backward push values;
step 2: compute arrival time of node i in position $\acute{p}$ on route $\acute{k}$;
step 3: update the arrival times of the nodes placed after position $\acute{p}+1$ on route $\acute{k}$ using the push forward values;
step 4: check if the new arrival times meet the time windows limitations;
step 5: check if the depot time window is met.

Algorithm 8 Time windows feasibility test of exchange within a route

Description: node i placed in position p is exchanged with node j located in position $p > p$ on the same route. The direct successor of node i and j on route k are denoted by $\sigma(i)$ and $\sigma(j)$, respectively.
step 1: compute arrival time of node j in new position p;
step 2: compute arrival time of node $\sigma(i)$ and determine the type of push;
step 3: compute the arrival times of the nodes placed between positions $p + 2$ and $\acute{p} - 1$ using the push values;
step 4: compute arrival time of node i in new position $\acute{p}$;
step 5: compute arrival time of node $\sigma(j)$ and determine the type of push;
step 6: compute the arrival times of the nodes placed after positions $\acute{p} + 2$ using the push values;
step 7: check if the new arrival times meet the time windows limitations;
step 8: check if the depot time window is met.

Algorithm 9 Time windows feasibility test of exchange across routes

Description: node i placed in position p on route k is exchanged with node j located in position $\acute{p}$ on route $\acute{k}$. The direct successor of node i and j on route k are denoted by $\sigma(i)$ and $\sigma(j)$, respectively.
step 1: compute arrival time of node j in position p on route k;
step 2: compute arrival time of node $\sigma(i)$ on vehicle k and determine the type of push for the next nodes on that route;
step 3: compute the arrival times of the nodes placed after positions $p + 1$ using the push values;
step 4: compute arrival time of node i in position $\acute{p}$ on route $\acute{k}$;
step 5: compute arrival time of node $\sigma(j)$ on the vehicle $\acute{k}$ and determine the type of push for the next nodes on that vehicle;
step 6: compute the arrival times of the nodes placed after positions $\acute{p} + 1$ using the push values;
step 7: check if the new arrival times meet the time windows limitations;
step 8: check if the depot time window is met.

Algorithm 10 Time windows feasibility test of vehicle insertion

Description: node i placed in position p on route k is removed and create a new route $\acute{k}$.
step 1: update the arrival times of the nodes placed after position $p+1$ on route k using the backward push values;
step 2: compute arrival time of node i in the first position on the route performed by vehicle $\acute{k}$;
step 3: check if the new arrival times meet the time windows limitations;
step 4: check if the depot time window is met.

Similarly, when the arrival time of the first node on path $(u, u+1, ..., v)$ is postponed, a push forward is defined

$$F_u = A_u^{new} - A_u. \tag{3.23}$$

The push forward at the next node on the path is calculated by

$$F_{u+1} = \min\{F_u - W_{u+1}, 0\}. \tag{3.24}$$

where W_{u+1} represents the waiting time at node $u+1$. The nodes on the path have to be checked sequentially, in such a way that if $A_k + F_k > l_k, u \leq k \leq v$, the path is no longer feasible. In case that $F_k = 0$, the path from node k to node v remains unchanged.

In this context, checking the feasibility of a partial solution concerning the time windows is implemented using the nodes push forward and backward for each proposed move. As mentioned before, the proposed moves may include various operations like node insertion, removal, and/or exchange with other nodes simultaneously. For example, exchanging nodes across the vehicles contains both operations of removal and insertion for two nodes on two different vehicles. So, the type of push (forward or backward) resulting from a move may not be known first. Given a movement of node i on the first position of path $(u, ..., v)$, it suffices only the arrival time of the direct successor of node i is computed and compared with the old value. If $A_u^{new} < A_u$, the push is backward and dented by B_u, while in case of $A_u^{new} > A_u$ the push is forward represented by F_u. Then, the next nodes push on the path is sequentially computed according to (3.22) or (3.24) associated with the backward and forward push, respectively. Finally, the nodes' updated arrival times on the path are checked whether the associated time windows are met.

Algorithms 5-10 provide the procedures implemented to check the feasibility of the proposed neighbourhoods in terms of the time windows associated with the related nodes and the depot (time horizon $[0, T]$).

3.6 Computational results

In this section, we present the results of the computational experiments carried out to evaluate the MILP model and the proposed hybrid algorithm on a set of instances. GAMS solve the MILP model. The algorithm is implemented in $C++$ on an Intel(R) Core(TM)Processor $i5-6200U$ (CPU2.30GHz) with 16 GB RAM.

In section 3.6.1, we describe the instances generated as a testbed for our assessment. In section 3.6.2, the value of parameters involved in the algorithm is determined using the Taguchi tuning procedure. Our computational experiment results are described and commented on in Section 3.6.3.

3.6.1 Design of experiments

In this section, we describe how we generate test instances for the proposed problem. Due to the novelty of the problem, no instances are available in the literature. So, we modify the well-known Solomon's benchmark instances (see [191]) used by most papers on vehicle routing problems with time windows. These instances are divided into six classes obtained by combinations of two criteria. The first criterion concerns the spatial position of nodes, which includes three different options: randomly generated by Uniform distribution (denoted by R), clustered (denoted by C), and semi-clustered (denoted by RC). The second criterion is the tightness of the planning horizon, which contains two types: a short time horizon (type 1) and a long time horizon (type 2). All possible combination are therefore: "R1", "C1", "RC1", "R2", "C2" and "RC2", making a total of 56 benchmark instances. Instances are encoded as follows: C201-50 corresponds to the first instances of the class "C2", where only the first 50 customers are considered. In this work, instances with a tight planning time horizon (type 1) are discarded since the short horizon does not define a significant number of AND/OR PCs for each node. Results are thus reported for "R2" (11 instances), "C2" (8 instances), and "RC2" (8 instances) for a total of 27 instances.

We classify the test problems into two categories in our experiments, referred to the small and large-sized instances. Both instance sets are solved using the MILP model and the proposed algorithm. The small-sized instances take the first 10, 20, and 30 nodes, while the large-sized ones take the first 40 and 50 nodes from each original instance. The nodes' locations, demands, time windows, service times, and the time horizon are set as in the original Solomon's instances. Similar to the previous literature, the travel time is the same as the Euclidean distance between two node locations. The vehicle capacity and the maximum number of available vehicles are set to specific values, all empirically determined as listed in Table 3.1.

An upper triangular matrix without the diagonal called Precedence Matrix (PM) is developed to represent the precedence constraints. Each element of PM denotes whether or not a precedence relation exists between the two corresponding nodes. If

Table 3.1: Parameters setting

Number of Nodes	Vehicle Capacity	Maximum Vehicle
10	100	3
20	200	4
30	200	4
40	300	5
50	300	5

node s have AND-type predecessors $\{i, j, k\}$ and OR-type predecessors $\{m, n\}$, then $PM_{is} = AND$, $PM_{js} = AND$, $PM_{ks} = AND$, $PM_{ms} = OR$, and $PM_{ns} = OR$. If there is no PC between the two nodes, the corresponding element of the matrix is zero. The representation of precedence constraints as an upper triangular matrix leads to these relations' feasibility. It is never possible to have nodes with a smaller number than one of its predecessors.

In our proposed problem, to construct a precedence matrix, time window limitations need to be taken into account in a way that if node i is a predecessor (no matter the type of PC) of node j, the late time of node j needs to be larger than that of node i. Otherwise, the PC ($i < j$) cannot be feasible in time windows. To define a feasible precedence matrix corresponding to the time windows constraints, the set of customer nodes needs to be sorted to increase the latest arrival time. The n^{th} row of the matrix represents all the node's possible successors associated with that row. The m^{th} column includes all the possible predecessors of the node corresponding to that column. For each column of the matrix, starting from the second column (we do not consider any predecessors for the first node) to the final one, the associated node's predecessors are generated.

To do so, we somehow adopt the scheme proposed by Derriesel and Monch [55] who addressed the parallel machines with sequence-dependent setup times, precedence constraints, and ready times. The precedence relations are inserted using the factor $\tau = \{0.4,0.8\}$ to evaluate PCs' impact by considering two different sizes. Given a column, if a chosen random number from $U[0,1]$ is higher than τ, we do not consider any predecessors for the node associated with that column. Otherwise, the number of predecessors is chosen according to $U[0, n-1^{th}]$, where n^{th} is the number of that column. Then, the predecessors are randomly selected from the set of already generated nodes $\{1^{th}, ..., n-1^{th}\}$. To determine the AND/OR types of precedence constraint between the randomly selected node in $\{1^{th}, ..., n-1^{th}\}$ and the one associated with the column, we use a random number from $U[0,1]$. The corresponding PC is an AND-type one if the chosen random number is less than 0.5. Otherwise, the PC is an OR-type relation.

As a result, the total number of instances is 270 which is the combinations of the type of instance $\{R2, C2, RC2\} = 27$, number of nodes $\{10,20,30,40,50\}$, and the rate of precedence constraints $\{0.4,0.8\}$. The proposed meta-heuristic algorithm

is run five times over 270 instances, and each run is stopped after 5 minutes of computation time. The upper limit of the CPU time for solving the MILP models is set to 14400 seconds.

3.6.2 Tuning

Because the choice of parameters has a remarkable influence on the meta-heuristic algorithms' efficiency, the Taguchi method for designing experiments is utilized to adjust the parameters. The motivation to apply the Taguchi method in this research is that it has been recognized as an effective approach that can simultaneously consider several factors and quickly distinguish the factors with principal impacts on final solutions by performing minimal possible experiments. For more information about the Taguchi method, the interested readers can refer to [153].

The proposed algorithm relies on four parameters, namely the maximum number of iterations (Max_{iter}), the maximum number of tries in LS loop are made without success to improve the current solution ($notImpMax$), the initial temperature ($Temp$) and the cooling rate (α) that need to be tuned. The set of representative tuning instances consists of the first two instances in any combinations of the type $\{R2, C2, RC2\}$, size $N = \{10,20,30\}$, and the PC scale $\tau = \{0.4,0.8\}$ for a total number of 36 instances. Based on some initial screening tests, three levels are selected for each parameter, as shown in Table 3.2.

Table 3.2: Parameters (factors) and their levels

Parameters	**Level**		
	1	**2**	**3**
Max_{iter}	500	900	1200
$notImpMax$	50	70	90
$Temp$	80	100	120
α	0.7	0.9	0.95

Since the number of parameters and their associated levels are equal to 4 and 3, respectively, we use the orthogonal array $L_9(3^4)$. Subsequently, the number of trials and the combination of parameter levels in each trial can be specified. To enable the comparison between the objective functions of the problems, the Relative Error (RE) value is calculated for each instance

$$RE(\%) = \frac{F_{sol} - F_{opt}}{F_{opt}} \times 100, \tag{3.25}$$

where F_{sol} is the objective function value found by the algorithm and F_{opt} corresponds to the optimal value for a given instance. Since MRE, i.e. the mean of REs, is to be minimized and therefore it is of "the smaller the better" category, S/N ratios are obtained using following equation

$$S/N = -10\log(\frac{1}{n}\sum_{i=1}^{n} F_i^2), \tag{3.26}$$

where n is the number of instances, and F_i is the objective function of instance i obtained by the algorithm.

Table 3.3 shows the mean value of the S/N ratios and means in every level of the parameters. In this table, the parameters are ranked according to $\Delta = \max() - \min()$. It can be seen that the effect of $Temp$ and Max_{iter} according to both signals to noise ratios and the means are the largest and the smallest, respectively, in comparison with the other factors. Main effects plots for S/N ratios and means are depicted in Figure 3.2 and 3.3, respectively. According to the signal to noise ratios, the level with maximum S/N value should be selected, and then we will have $Max_{iter} = 1200$, $notImpMax = 70$, $Temp = 100$ and $\alpha = 0.95$; whereas considering the means, the level with a minimum value of mean should be chosen and therefore, we have the same results in terms of the three parameters $notImpMax$, $Temp$ and α, with only except for Max_{iter} which should be equal to 900. Finally, we conclude that to maintain the algorithm's robustness, the adjustment corresponding to signal-to-noise ratios should be selected.

Table 3.3: Response table for the algorithm

Level	S/N				**Mean**			
	Max_{iter}	$notImpMax$	$Temp$	α	Max_{iter}	$notImpMax$	$Temp$	α
1	-6.471	-6.369	-6.827	-6.396	0.845	1.085	0.882	0.860
2	-5.972	-4.970	-4.739	-5.473	0.636	0.710	0.359	0.992
3	-5.383	-5.264	-6.285	-5.072	0.974	0.958	0.625	0.526
Δ	1.088	1.399	2.088	1.324	0.338	0.375	0.523	0.466
Rank	4	2	1	3	4	3	1	2

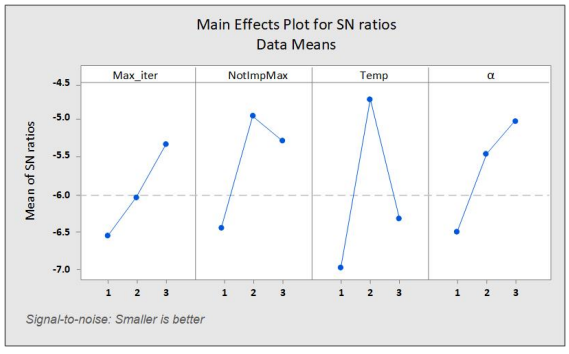


Figure 3.2: Main effects Plots for S/N ratios

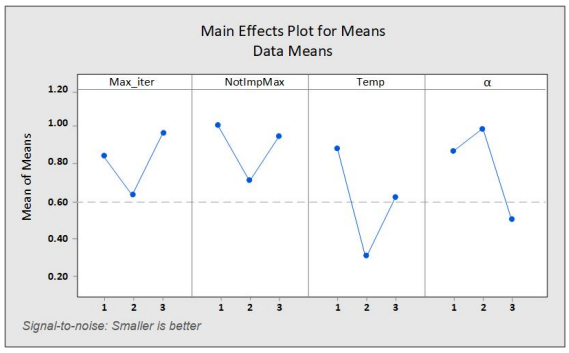


Figure 3.3: Main effects Plots for Means

3.6.3 Evaluation of the MILP model and the hybrid algorithm

This section summarizes and discusses our experiments' results to evaluate the proposed MILP model and the algorithm performances on the different set of instances generated and described in Section 3.6.1.

Both approaches (the MILP model and the algorithm) can optimally solve all the small-sized instances. The developed algorithm's performance in terms of the computational time compared to the exact model in dealing with the first category is reported in Table 3.4. The table shows the average time (in seconds) needed to optimally solve instances for each combination of type $\{R2, RC2, C2\}$, size $N = \{10,20,30\}$, and the PC scale $\tau = \{0.4,0.8\}$ for both solution approaches.

The first thing that should be noticed is that the proposed algorithm can find optimal solutions in less average time for all combinations except for the smallest size ($N = 10$). It can be seen that the difference in time for such instances is not much as the average time over different types of size $N = 10$ and PC factor $\tau = 0.4$ associated with the MILP model and the algorithm are equal to 16.76 and 21.01, respectively.

Considering different sizes of instances, as expected, the two approaches spend more computational time to obtain optimal solutions as the number of nodes increases for all cases.

Comparing different types indicates that instances of types $R2$ and $C2$ need, respectively, the least and the most average running time to reach an optimal solution using the exact model. Instead, the developed algorithm spends average time with a small discrepancy for such instances. It means that different types of instances do not significantly affect the algorithm's computational time for instances of each size and PC factor.

Other trends can be noticed by looking at the PC scale. It seems that the larger PC scale ($\tau = 0.8$) results in less complexity of the MILP model as it spends a lower average running time. On the contrary, as expected, the proposed algorithm needs

Table 3.4: Computational average time of the MILP model and the algorithm on small-sized instances.

Instance	**MILP**		**Hybrid Algorithm**	
Type-N	$\tau = 0.4$	$\tau = 0.8$	$\tau = 0.4$	$\tau = 0.8$
R2-10	22.51	16.77	18.92	24.65
RC2-10	17.47	15.92	17.38	19.59
C2-10	57.26	17.59	26.73	29.13
Average:	25.75	16.76	21.01	24.46
R2-20	218.39	170.62	38.72	41.68
RC2-20	378.62	326.35	27.94	44.80
C2-20	583.55	467.37	36.37	52.52
Average:	393.52	321.45	34.34	46.33
R2-30	692.07	583.55	84.99	127.60
RC2-30	1073.51	828.30	76.12	253.49
C2-30	3802.64	2480.73	92.86	180.24
Average:	1856.07	1297.53	84.66	187.11

more time to deal with instances with higher PC factors than those of the lower one.

We now want to assess the quality of solutions obtained by the second category's two approaches, including large-scale instances. The evaluation is performed by comparing their solutions before exceeding the CPU time limit of 300 and 14400 seconds associated with the algorithm and the MILP model. The performance, in terms of percentage gap, is evaluated through the calculation of the Relative Percentage Error (RPE) as follows

$$RPE = \frac{Alg_{sol} - M_{sol}}{M_{sol}} \times 100, \tag{3.27}$$

where M_{sol} and Alg_{sol} represent the feasible solution found by the MILP model and the solution obtained by the algorithm, respectively, for a given instance. It should be noticed that an optimal solution of the large-sized instances cannot be found by solving the exact model. So, the reported solution obtained by the MILP model is an upper bound with the determined optimality gap for each instance.

Tables 3.5 and 3.6 report the feasible solution of the MILP model and its optimality gap (denoted by OptGap), the solution found by the algorithm, as well as the factor RPE, to evaluate the performance of the algorithm concerning the feasible solution of the MILP model within the proposed time limit for instances with $N = 40$ and $N = 50$, respectively.

The first thing that should be noticed is that the algorithm's solution values are way less than those found by solving the MILP model for all instances of any

Table 3.5: Performance results of the MILP model and the algorithm on the instances with $N = 40$.

Type-N	Instance	$\tau = 0.4$				$\tau = 0.8$			
		MILP		Algorithm	RPE	MILP		Algorithm	RPE
		OptGap	Sol	Sol		OptGap	Sol	Sol	
R2-40	1	52.14	1382	951	31.19	48.02	1739	1226	29.50
	2	31.05	1449	893	38.37	34.66	1977	993	49.77
	3	18.12	1650	909	44.91	22.48	2049	1308	36.16
	4	5.89	1734	786	54.67	9.25	1971	961	51.24
	5	64.04	1250	991	20.72	37.61	1550	1370	11.61
	6	56.37	1484	852	42.59	40.07	1786	1148	35.72
	7	26.92	1451	1008	30.53	19.22	1879	1473	21.61
	8	29.4	1359	661	51.36	17.39	1538	862	43.95
	9	47.66	1355	946	30.18	29.58	2150	1244	42.14
	10	33.37	1340	750	44.03	31.7	1624	959	40.95
	11	69.03	1240	837	32.50	46.44	1733	1232	28.91
	Average:	39.45	1426.73	871.27	38.28	30.58	1817.82	1161.45	35.60
RC2-40	1	50.62	1419	1103	22.27	40.32	1792	1052	41.29
	2	36.48	1372	936	31.78	34.25	1885	1163	38.30
	3	54.17	1590	1108	30.31	42.66	1942	1470	24.30
	4	28.39	1648	950	42.35	32.72	1766	986	44.17
	5	41.7	1520	825	45.72	37.49	1827	992	45.70
	6	22.91	1428	736	48.46	24.18	1907	1373	28.00
	7	40.57	1639	914	44.23	12.66	1945	1328	31.72
	8	50.02	1376	738	46.37	47.83	1739	1014	41.69
	Average:	40.61	1499.00	913.75	38.94	34.01	1850.38	1172.25	36.90
C2-40	1	66.38	1588	942	40.68	51.47	1837	1059	42.35
	2	42.5	1226	880	28.22	36.8	1914	953	50.21
	3	43.99	1740	1156	33.56	41.94	1955	1274	34.83
	4	31.74	1972	971	50.76	29.17	2006	1368	31.80
	5	43.16	1662	956	42.48	44.62	1950	1076	44.82
	6	50.03	1759	1003	42.98	48.77	2173	1279	41.14
	7	25.48	1801	1149	36.20	26.83	1988	1193	39.99
	8	62.7	1839	944	48.67	52.9	2107	1109	47.37
	Average:	45.75	1698.38	1000.13	40.44	41.56	1991.25	1163.88	41.56

combinations of types, sizes, and PC factors. This result shows the promising performance of the developed algorithm in dealing with such instances where the exact model cannot give optimal or even near-optimal solutions.

Considering the different types of instances, it can be seen that both OptGap and RPE associated with the instances of types $R2$ and $C2$ are the smallest and the largest value for all the cases of the two considered sizes ($N = 40$ and $N = 50$). It means that type $R2$ instances contain the least complexity as their optimality gap is the lower values than the two other types, while type $C2$ instances are more complex since their corresponding feasible solutions have larger OptGap values. Similar behavior can be seen for the RPE, which means that, for the instances of type $R2$, the percentage gap between the feasible solution of the MILP model and the one obtained by the algorithm is lower than those of the instances of the two other types.

Considering the two different PC factors ($\tau = 0.4$ and $\tau = 0.8$), a fluctuating

Table 3.6: Performance results of the MILP model and the algorithm on the instances with $N = 50$.

Type-N	Instance	$\tau = 0.4$				$\tau = 0.8$			
		MILP		**Algorithm**	**RPE**	**MILP**		**Algorithm**	**RPE**
		OptGap	**Sol**	**Sol**		**OptGap**	**Sol**	**Sol**	
R2-50	1	39.62	1473	884	39.99	18.48	1672	962	42.46
	2	48.5	1362	1038	23.79	51.72	1850	1425	22.97
	3	27.31	1846	1003	45.67	22.37	1996	1346	32.57
	4	61.17	1930	869	54.97	54.8	2173	990	54.44
	5	42.84	1393	920	33.96	38.51	1630	1380	15.34
	6	50.73	1560	1027	34.17	46.04	1782	1233	30.81
	7	32.91	1528	940	38.48	30.18	1866	1302	30.23
	8	57.62	1402	841	40.01	67.82	1751	939	46.37
	9	40.88	1438	883	38.60	39.94	1849	1427	22.82
	10	36.67	1567	852	45.63	41.03	1730	883	48.96
	11	25.17	1346	733	45.54	23.74	1694	1193	29.57
	Average:	42.13	1531.36	908.18	40.07	39.51	1817.55	1189.09	34.23
RC2-50	1	37.18	1583	1063	32.85	28.66	1846	1342	27.30
	2	42.93	1649	1139	30.93	38.5	1973	1274	35.43
	3	32.66	1662	826	50.30	30.82	1860	940	49.46
	4	39.03	1879	955	49.18	47.2	2035	1075	47.17
	5	46.82	1703	809	52.50	42.71	2082	969	53.46
	6	50.24	1534	783	48.96	40.9	1755	895	49.00
	7	56.9	1950	1094	43.90	52.52	2120	1266	40.28
	8	52.42	1526	1138	25.43	48.35	1796	1357	24.44
	Average:	44.77	1685.75	975.88	41.75	41.21	1933.38	1139.75	40.82
C2-50	1	68.27	1837	1149	37.45	55.27	2263	1296	42.73
	2	44.62	1659	937	43.52	40.12	1970	1071	45.63
	3	44.09	1930	1174	39.17	43.74	2159	1289	40.30
	4	33.15	1984	1005	49.34	31.29	2324	1162	50.00
	5	44.73	1760	959	45.51	42.89	1973	1293	34.47
	6	66.49	1985	1340	32.49	57.18	2288	1371	40.08
	7	30.75	1874	967	48.40	25.36	2230	1028	53.90
	8	66.94	1936	1050	45.76	52.15	2166	1139	47.41
	Average:	49.88	1870.63	1072.63	42.71	43.50	2171.63	1206.13	44.32

behavior can be seen for both OptGap and RPE values so that they can be increased or decreased by considering the smaller or larger PC factor for each instance. The problem complexity is not entirely dependent on the amount of imposed PCs, but the generated PCs' structure may significantly affect. However, the average value of OptGap over the instances of each type and size follows an increasing trend when the smaller PC factor is applied. The same behavior can be seen for RPE's average value over the instances except for instances of type $C2$ for both sizes.

Finally, comparing the results for the two instance sizes ($N = 40$ and $N = 50$) shows that, as expected, the average value of solutions obtained by the model and the algorithm and the OptGap increases as the size of instances grows. The average RPE values over the instances also follow the same described behavior with little discontinues associated with the instance of type $R2$ and PC factor $\tau = 0.8$.

3.7 Summary

In this chapter, we have studied for the first time, to the best of our knowledge, a generalization of the vehicle routing problems with time windows. The AND/OR precedence constraints are defined among the customers visited by each vehicle. This generalization comes after considering the partial orders of servicing the customers due to some physical restrictions or having the preferred loading or unloading sequence and avoiding extra effort to sort the collected items at the end of the retrieving process. To address the problem, we have formulated it as a MILP model capable of solving only small-sized instances by spending a lot of CPU time. We have also developed a meta-heuristic algorithm as the hybridization of Iterated Local Search and Simulated Annealing approaches. The computational result of the developed algorithm highlights this approach's promising performance in CPU time and its quality. This proves that the integration between SA and ILS can balance exploration and exploitation and thus achieve reasonable optimization results.

Chapter 4

Background on the Deterministic Approximation approach

4.1 Introduction

In this chapter, the deterministic approximation approach, as well as some background information, are represented. The approach is based on looking at the optimization problem as a so-called Random Utility Models (we refer to the static case) or its multi-stage version. Random Utility Models (RUMs) are typical of operations management's applications such as supply chain optimization, logistics, and transportation, in which decisions (or a part of them) must be taken with limited knowledge of the alternatives and their attributes (see, e.g., [26], [47], [127], [155],[204]).

When facing the static RUMs where the decision-maker is asked to choose an alternative among a static set of choices, it is well-known that the choice probability is modeled as a Multinomial Logit (MNL) model under the assumption that the random term utilities are independent and identically distributed (i.i.d.) and the common distribution is a *Gumbel* function (see [124], [24], [25], [57]).

Some contributions have shown that the assumption of a Gumbel distribution for the random term utilities is too restrictive when the number of alternatives becomes large and that an MNL model can be still derived under the milder assumption that the common distribution of such i.i.d. random utilities has an asymptotically exponential behavior in its right or left tail ([200], [201]). The effectiveness of such an asymptotic approximation has been proved in several applications in the context of routing, loading, packing, and other logistics operations ([200], [154], [203], [155]).

In several decision processes, the decision-maker is asked to solve several RUMs

consecutively over multiple discrete stages aiming to maximize (minimizing) the expected value of the total utility (cost) originating from the overall decision process. It means the decision-maker needs to select, at each stage, an alternative among a finite set of choices. Each alternative is associated with a certain level of utility (cost) depending on both stochastic variables with unknown probability distributions and the utilities associated with the selected alternatives in the subsequent stages. In such a way, the decisions are nested each other throughout the multi-stage decision network. So, the decision process cannot be decomposed into distinct stages, and, in turn, the approximation approach can not be applied straightforwardly.

For this reason, Tadei et al. [202] recently generalized the approximation framework to provide results for the multi-stage case too. In particular, under appropriate assumptions, the probability distribution of the best alternative can still be asymptotically approximated by a Gumbel distribution. In turn, the total utility of the process can be analytically derived. Moreover, the choice probability can be modeled as a Nested Multinomial Logit model.

This chapter is organized as follows. In Section 4.2, a review of discrete choice models, random utility models, and the multinomial logit models are represented. After a brief discussion of general assumptions, in Section 4.3, the deterministic approximation approach for the static RUM is provided. At the same time, its applications in the literature are reviewed in Section 4.4. Then, the deterministic approximation approach for the maximization problem to deal with the multi-stage RUMs is discussed in Section 4.5. Moreover, the approach for the minimization problem is presented in 4.6. Finally, a summary is provided in Section 4.7.

4.2 Preliminaries

4.2.1 Discrete Choice Models

Discrete Choice Models (DCM) are used to model the choices made between well-defined alternatives. Typically a decision-maker might choose between a finite set of alternate products or services. The set of all alternatives available is called the choice set. If there are just two alternatives, it is called a binary choice. A discrete choice model with more than two alternatives is called a multinomial discrete choice model.

The framework of a discrete choice model is distinguished as

- decision-maker – defining the decision making entity and its characteristics;
- alternatives – determining the choices available to the decision-maker;
- attributes – measuring either utility or cost of alternatives to the decision-maker;

- decision rule – describing the process used by the decision-maker to choose an alternative.

In the following, the above entities are better described.

Decision-maker: In discrete choice models (also referred to as disaggregate models), the decision-maker is assumed to be an individual. The individual entity depends on the particular application. For instance, we may consider a group of persons as the decision-maker. In doing so, we may ignore all internal interactions within the group and consider only the group's decisions as a whole.

Alternatives: Analyzing decision-making requires knowledge of what has been chosen and what has not been chosen. Therefore, assumptions must be made about alternatives that an individual considers during a decision-making process. The set of available alternatives is called the choice set. A discrete choice set contains a finite number of alternatives that can be explicitly listed. The set of travel modes is a typical example of a discrete choice set. Identifying the list of alternatives is a complex process usually referred to as a choice set generation. The most widely used method for choice set generation uses deterministic criteria of alternative availability. The universal choice set contains all potential alternatives in the application's context. The choice set is the subset of the universal choice set considered by, or available to, a particular individual. In addition to availability, the decision-makers' awareness of the alternative could also affect the choice set. The behavioral aspects of awareness introduce uncertainty in modeling the choice set generation process and motivate probabilistic choice set generation models that predict the probability of each feasible choice set within the universal set.

Attributes: Each alternative in the choice set is characterized by attributes such as utility, cost, etc. Note that some attributes may be generic to all alternatives, and some may be alternative-specific. An attribute is not necessarily a directly measurable quantity. It can be any function of available data. Alternative definitions of attributes as functions of available data must usually be tested to identify the most appropriate.

Decision Rule: The decision rule is the process used by the decision-maker to assess the attributes of the alternatives in the choice set and make a choice. Most models are based on utility theory, which assumes that a value, called utility, captures the decision-maker's preference for a choice. The decision-maker selects the alternative in the choice set with the highest utility. The complexity of human behavior suggests that the decision rule should include a probabilistic dimension. Some models assume that the decision rule is intrinsically probabilistic, and even complete knowledge of the problem would not overcome the uncertainty. Others

consider the individuals' decision rules as deterministic and motivate the uncertainty from the limited capability of the analyst to observe and capture all the dimensions of the choice process due to its complexity. Specific families of models can be derived depending on the assumptions about the source of uncertainty. Random utility models, described in the following subsection, are based on deterministic decision rules, where random variables represent utilities.

4.2.2 Random Utility Models

Random Utility Models (RUMs) are behavioral models [130]. A decision-maker chooses among mutually exclusive alternatives, each associated with a certain level of utility. The basic assumptions of these settings are that the choice set is discrete, and the decision-maker follows a rational *utility-maximizing* behavior. Also, the utility associated with each choice alternative can be decomposed into a deterministic part and a random term oscillation. Since the latter term is not known as a priori, it is treated as a stochastic variable. In turn, this means that the possible realizations (observations) of such a random variable hypothetically increase the number of alternatives to choose from, i.e., the decision-maker would select the best alternative if he knew the actual realizations of the utilities. It is important to notice that we do not assume that the decision-maker can decide for the best alternative having complete knowledge of the realizations in advance. Instead, we want to focus on modeling the extreme behavior of such unknowns, i.e., the one with the maximum utility (minimum cost), which theoretically corresponds to the decision maker's wish. In other words, it is assumed that the decision-maker has an optimistic vision of the future. Not knowing what will happen in the future, he assumes that he will have to pay as little as possible (or gain benefit as much as possible).

Manski [129] identifies four different sources of uncertainty: unobserved alternative attributes, unobserved individual characteristics, measurement errors, and proxy or instrumental variables. The utility is modeled as a random variable to reflect the uncertainty. More specifically, the utility that decision-maker n associates with alternative i in the choice set C_n is given by

$$U_{in} = V_{in} + \theta_{in}, \tag{4.1}$$

where V_{in} is the deterministic part of the utility, and θ_{in} in is the random term oscillation capturing the uncertainty. Among the alternatives in the choice set, the one with the highest utility is selected. Therefore, the probability that alternative i is chosen by decision-maker n from the choice set C_n is

$$P(i \mid C_n) = P[U_{in} \geq U_{jn}, \forall j \in C_n] = P[U_{in} = \max_{j \in C_n} U_{jn}]. \tag{4.2}$$

The deterministic term V_{in} is a function of the attributes of the alternative i and the characteristics of decision-maker n. That is

$$V_{in} = h(Z_{in}, S_n), \tag{4.3}$$

where Z_{in} is the vector of alternative i attributes as perceived by decision-maker n, and S_n is the vector of individual n characteristics. Function h is defined as any appropriate vector of attributes from both Z_{in} and S_n. The choice of function h is very general, and several forms may be tested to identify the best representation in a specific application.

Many potential models are derived for the random parts of the utility functions. The Logit family models are based on a probability distribution function of the maximum or minimum of a series of random variables introduced by Gumbel distribution. The models within the Logit family has been much more popular because of its tractability, but it imposes restrictions on the covariance structure. The derivation of other models in the Logit family is aimed at relaxing restrictions while maintaining tractability.

4.2.3 Multinomial Logit Models

Among the discrete choice models, the multinomial logit model is the most widespread and used in many different fields. This disaggregated model seeks to study the decision of choice or the perception of the value of an event among a set of mutually exclusive alternatives.

The Logit model was first introduced in the context of binary choices where the logistic distribution is used. Its extension to more than two choices is referred to as the Multinomial Logit Model. This model is derived from the assumption that the random terms of the utility are independent and identically random variables distributed as a Gumbel, i.e.,

$$F(\theta) = \exp[-e^{-\mu(\theta-\eta)}], \mu > 0, \tag{4.4}$$

$$f(\theta) = \mu e^{-\mu(\theta-\eta)} \exp[-e^{-\mu(\theta-\eta)}], \tag{4.5}$$

where η is a location parameter and μ is a strictly positive scale parameter. The mean of this distribution is $\eta + \gamma/\mu$ where $\gamma = 0.5772$ is the Euler constant. The variance of the distribution is $\pi^2/6\mu^2$.

The probability that decision-maker n chooses alternative i within the choice set C_n is given by

$$P(i \mid C_n) = \frac{e^{\mu V_{in}}}{\sum_{j \in C_n} e^{\mu V_{jn}}}. \tag{4.6}$$

An important property of the Multinomial Logit Model is Independence from Irrelevant Alternatives (IIA). This property can be stated as follows: The ratio

of the probabilities of any two alternatives is independent of the choice set. That is, for any choice sets C_1 and C_2 such that $C_1 \subseteq C_n$ and $C_2 \subseteq C_n$, and for any alternatives i and j in both C_1 and C_2, we have

$$\frac{P(i \mid C_1)}{P(j \mid C_1)} = \frac{P(i \mid C_2)}{P(j \mid C_2)}. \tag{4.7}$$

An equivalent definition of the IIA property is: The ratio of the choice probabilities of any two alternatives is unaffected by the systematic utilities of any other alternatives.

4.3 Deterministic approximation for static Random Utilities Models

The theory of extreme values is particularly appropriate for Random Utilities Models, as it deals with the asymptotic behavior of maxima and minima over sequences of variables [75]. When the decision-maker has a static set of alternatives to choose from, it is well-known that the choice probability can be modeled as a Multinomial Logit (MNL) under the assumption that the random term of utilities are independent and identically distributed as a *Gumbel* distribution (see [124], [24], [25], [57]).

Other contributions ([114], [113], [204]) have shown that the assumption of a Gumbel distribution for the random utilities is too restrictive when the number of alternatives becomes large. An MNL model can still be derived under the milder assumption that the common distribution of the i.i.d. random utilities has an asymptotically exponential behavior in its right or left tail. Such a model can be seen as an asymptotic deterministic approximation of the static decision-making process. It can be theoretically derived only when the number of alternatives and, therefore, possible realizations of the unknowns tend to infinity.

Recently, Tadei et al. [204] have shown that the Gumbel distribution assumption for the i.i.d. random term utilities is unjustified to derive the MNL model for choice probability. They reformulated the random utility choice theory in terms of the asymptotic extreme values theory [75]. This theory deals with maxima (or minima) properties of sequences of random variables with many terms. In particular, they showed that under a milder assumption on a random term of utilities and considering many alternatives, the Gumbel distribution assumption is not necessary anymore to derive the MNL model. In the following, we present all the results for the maximization case, but the same results can be adapted to the minimization case.

It is assumed that $F(x)$ is asymptotically exponential in its right tail, i.e., there is a constant $\beta > 0$ such that

$$\exists \beta > 0 \mid \lim_{y \to +\infty} \frac{1 - F(x+y)}{1 - F(y)} = e^{-\beta x}. \tag{4.8}$$

This property is widely used in the extreme value theory and defines the so-called domain of attraction of the double exponential distribution.

Let us consider a set J of $N = |J|$ alternatives. We assume that J is partitioned into n nonempty disjoint subsets $J_j, j = 1, ..., n$, called clusters, of $N_j = |J_j|$ alternatives. Many choice processes face the partition into clusters of alternatives. For instance, when a household is looking for a dwelling, it will select the district where to live (this is the cluster) and, then, inside that district, it will choose the actual dwelling among all the alternatives. Let $\tilde{u}_{ij}^z$ be the utility for decision maker i for choosing alternative $z \in J_j$. As already stated, in a random utility model we assume that $\tilde{u}_{ij}^z$ is the sum of a deterministic variable v_{ij} and a random variable $\tilde{\theta}_{iz}$, i.e.,

$$\tilde{u}_{ij}^z = v_{ij} + \tilde{\theta}_{iz}. \tag{4.9}$$

The deterministic variable v_{ij} of the utility includes variables representing attributes of the cluster and the decision context. The random variable $\tilde{u}_{ij}^z$ represents aspects of utility that the researcher does not observe, e.g., idiosyncrasies of decision-maker i. The decision-maker i has an optimistic vision, and not knowing the observations, he assumes that the one with the maximum utility is chosen among all alternatives.

Let us define the distribution of the maximum utility for decision-maker i among all alternatives z in all clusters j as

$$\tilde{u}_i = \max_{j=1,..,n; z \in J_j} \tilde{u}_{ij}^z = \max_{j=1,..,n} (v_{ij} + \max_{z \in J_j} \tilde{\theta}_{iz}) = \max_{j=1,..,n} (v_{ij} + \tilde{\theta}_i^j). \tag{4.10}$$

Moreover, let

$$G_i(x) = Pr(\tilde{u}_i < x), \tag{4.11}$$

be the distribution of $\tilde{u}_i$ and

$$P_{ij}(x) = Pr(\tilde{\theta}_i^j < x), \tag{4.12}$$

be the distribution of $\tilde{\theta}_i^j$.

By the i.i.d. assumption of the random variables, the distribution $P_{ij}(x)$ becomes

$$P_{ij}(x) = \prod_{z \in J_j} Pr(\tilde{\theta}_{iz} < x) = [F(x)]^{N_j}. \tag{4.13}$$

Now, because of (4.10) and (4.13), $G_i(x)$ becomes

$$G_i(x) = Pr(\tilde{u}_i < x) = Pr(\max_{j=1,..,n}(v_{ij} + \tilde{\theta}_i^j) < x)$$
$$= \prod_{j=1,..,n} Pr(v_{ij} + \tilde{\theta}_i^j < x) = \prod_{j=1,..,n} Pr(\tilde{\theta}_i^j < x - v_{ij})$$
$$= \prod_{j=1,..,n} P_{ij}(x - v_{ij}) = \prod_{j=1,..,n} [F(x - v_{ij})]^{N_j}. \tag{4.14}$$

Following [154] and [155], It is shown that under assumption (4.8) the distribution $G_i(x)$ tends towards a Gumbel function as the total number of alternatives N becomes large. So, under these results, the MNL model for the choice probability can be still derived. First, It is considered that we can fix the origin for the utility scale arbitrarily, i.e., the choice probabilities are unaffected by a shift in the utility scale, and any additive constant to the utilities can be ignored. Let's choose this constant as the root a_N of the equation

$$1 - F(a_N|N) = 1/N, \tag{4.15}$$

where we remind N is the total number of alternatives. By replacing $\tilde{u}_i$ with $\tilde{u}_i - a_N$ in (4.14) one has

$$G_i(x|N) = \prod_{j=1,..,n} [F(x - v_{ij}) + a_N|N]^{N_j}. \tag{4.16}$$

Let us consider the ratio

$$\alpha_j = N_j/N, \tag{4.17}$$

and assume that this ratio remains constant for each j while the values of $N = 1,2,...$ vary, as needed later to compute the asymptotic behavior while N increases.

Because of (4.17), Eq. (4.16) can be written as

$$G_i(x|N) = \prod_{j=1,..,n} [F(x - v_{ij}) + a_N|N]^{\alpha_j N}. \tag{4.18}$$

Let us assume that N is large enough to use $\lim_{N \longrightarrow +\infty} G_i(x|N)$ as an approximation of $G_i(x)$. Then, the following theorem holds.

Theorem 1. *Under condition (4.8), the probability distribution $G_i(x)$ becomes the following Gumbel distribution*

$$G_i(x) = \lim_{N \longrightarrow +\infty} G_i(x|N) = \exp(-A_i e^{-\beta x}), \tag{4.19}$$

where

$$A_i = \sum_{j=1,..,n} \alpha_j e^{\beta v_{ij}}, \tag{4.20}$$

is the accessibility in the sense of Hansen [89] to the overall set of alternatives.

Theorem 2. *The choice probability p_{ij} for decision maker i to choose cluster j is given by*

$$p_{ij} = \frac{N_j e^{\beta v_{ij}}}{\sum_{k=1}^{n} N_k e^{\beta v_{ik}}}. \tag{4.21}$$

Note that the choice probability in (4.21) still represents an MNL model.

Even if condition in (4.8) yet represents a mild assumption on the shape of the distribution of the stochastic variables, Fadda et al. [66] have recently proved that Theorem 1 still holds when (4.8) is relaxed to the following condition

$$\exists \beta > 0 \mid \lim_{|N| \to +\infty} B(x + a_N)^{|N|} = exp(-e^{-\beta x}), \tag{4.22}$$

where $B(x)$ is the probability distribution of $\tilde{x}$, i.e.,

$$B(x) = Pr\{\tilde{x} \leq x\}, \tag{4.23}$$

and a_N is chosen equal to the root of the equation

$$1 - B(x) = \frac{1}{|N|}. \tag{4.24}$$

Assumption (4.22) is equivalent to ask that the unknown distribution of the stochastic maximum utility belongs to the domain of attraction of a Gumbel distribution. This new result further enlarges the applicability of the approximation approach, which theoretically holds for any distribution of the form $1-e^{-p(x)}$ where $p(x)$ is a polynomial function, such as the Normal, the Gumbel, the Weibull, the Logistic, the Laplace, the Lognormal, and many others.

It is easy to see that the new condition is milder, since assumption in (4.8) implies assumption in (4.22), while the converse is not true (e.g., the Normal distribution does not satisfy (4.8)).

4.4 Related works on the deterministic approximation for static Random Utilities Model

The accuracy of the approximation has been experimentally shown in several application domains.

As the first application, Tadei et al. [204] addressed the location of p facilities, which minimize the expected total cost when the cost for using a facility is a stochastic variable with an unknown probability distribution. They stated that in several papers dealing with uncertainty in the p-median problem, assumptions on the type of the cost probability distribution are given (either the probability distribution is known or a finite number of possible states, each occurring with

nonzero probability, is assumed). Unfortunately, in many real-life situations, the exact shape of this distribution is unknown. In this paper, a set of customers V, a set of potential facility locations U, partitioned into n nonempty disjoint subsets called clusters $U_1, .., U_n$ is assumed. The problem consists of finding p, with $p \leq n$, facility locations, no more than one per cluster. The stochastic costs r_{ij} of customer $i \in V$ pays for using facility $j \in U_k, k = 1, ..., n$ are given by the sum of a deterministic cost c_k associated to each cluster plus a random term θ_{ij}, with an unknown probability distribution, which represents the cost heterogeneity inside each cluster as

$$r_{ij}(\theta) = c_k + \theta_{ij}, \forall i \in V, j \in U_k, k = 1, ..., n. \tag{4.25}$$

In their proposed work, it is proved that under a quite mild and reasonable assumption on the shape of the unknown random term θ_{ij} probability distribution (see, eq.(4.8)) and the number of the potential facility locations, the probability distribution of the minimum cost becomes a Gumbel (or double exponential) one. Moreover, using such a distribution, a multinomial Logit function for the allocation variables of the p-median problem is derived.

As another application of the asymptotic approximation, Tadei et al. [200] proposed a problem that aims to find a transshipment facilities location that maximizes the total net utility when the handling utilities at the facilities are stochastic variables. In this work, the two main levels of a transshipment network, i.e., the network design (upper level), which leads to a network flow formulation with origins, transshipment facilities and destinations as nodes of the network, and the transshipment facilities management (lower level), where the management variables considered as stochastic handling utilities at the facilities are integrated. Also, the total net utility is given by the expected total shipping utility minus the total fixed cost of the located facilities. Shipping utilities are given by a deterministic utility s_{ij}^k for shipping freight from origin i to destinations j via transshipment facilities k plus a stochastic handling utility at the facilities $\tilde{u}^{kl}$ under scenario l, whose probability distribution is unknown. They showed that using some results of the extreme values theory, the probability distribution of the maximum stochastic utility is derived, and the expected value of the optimum of the stochastic model is found. An efficient heuristics for solving real-life instances was also given. Moreover, the i.i.d. assumption for $\tilde{u}^{kl}$, which is necessary for deriving the asymptotic approximation, was justified as follows. The stochastic utility of a handling operating scenario at any facility k is extremely difficult to be measured in practice. Its probability distribution is generally unknown, and it would be rather arbitrary to assume a particular shape for it. Of course, the mildest hypothesis which can be made for the shape of such unknown probability distribution is that it does not vary within different scenarios and facilities, which corresponds to the "identically distributed" assumption. Moreover, the alternative handling operating scenario inside a facility do not obviously depend on the scenario of the remaining facilities, and inside the same facility, they slightly interact with each other in practice, allows the stochastic

utilities $\tilde{u}^{kl}$ to be considered as independent variables too.

The stochastic Generalized Bin Packing Problem was dealt in Perboli et al. [154]. The problem consists of finding a subset of items to be loaded into a subset of bins, which maximizes the expected total net profit, given by the difference between the expected total profit of the loaded items and the total cost bins while satisfying the volume and bin availability constraints. In this work, the item profits are random variables to consider the profit oscillations due to the handling operations for bin loading. They also assumed that such profit oscillations randomly depend on the handling scenarios adopted for bin loading. The probability distribution of these random variables is assumed to be unknown. They showed that using some results of the asymptotic theory of extreme values, the probability distribution of the maximum random profit of any item becomes a Gumbel (or double exponential) probability distribution, and the total expected profit of the loaded items can be easily calculated. By using this result, the deterministic approximation is derived.

The Multi-Handler Knapsack Problem under Uncertainty was proposed in Perboli et al. [155], a new stochastic variant of the knapsack problem. In this problem, given a set of items, characterized by volume and random profit, and a set of potential handlers, a subset of items is found, which maximizes the expected total profit. The profit is given by the sum of a deterministic profit and a stochastic profit oscillation, with an unknown probability distribution, due to the random handling costs of the handlers. A specific application of this problem can be found in the automotive sector. The delivery of cars from manufacturers to dealers is not managed by the manufacturers themselves but is delegated to specialized companies. These companies manage both the finishing operations on the cars and the logistics operations linked to delivery to the dealers. To have a more flexible structure, the fleet of auto-carriers used to deliver the cars is only partially owned by each company. At the same time, a substantial part of the deliveries is sub-contracted to micro-companies with highly variable random costs. Moreover, the auto-carriers have different capacities due to the presence of specific technical features. From the point of view of the cars that must be delivered, the net profit for the company is affected by different factors, including delays in the finishing operations, additional costs due to violations of the negotiated deadlines, or additional transportation costs. This paper introduced a formulation of the stochastic problem where a deterministic approximation is derived. In particular, under a mild hypothesis on the unknown probability distribution, the deterministic approximation becomes a knapsack problem where the total expected profit of the loaded items is proportional to the logarithm of the total accessibility of those items to the set of handlers. Moreover, at optimality, the percentage of an item handled by any handler is given by a Multinomial Logit model.

The multi-path Traveling Salesman Problem was introduced by Tadei et al. [203] where given a set of nodes, several paths connect each pair of nodes, and each path shows a stochastic travel cost with an unknown probability distribution.

The problem aims at finding an expected minimum Hamiltonian tour connecting all nodes. This is computed as the sum of the expected travel costs of the paths interconnecting the pairs of nodes, where, for each pair of nodes, only one path can be selected among the several ones. The travel cost is a generalized cost that includes both fuel consumption and driving time. Additionally, each travel cost is composed of a deterministic term plus a random term, representing the travel cost oscillation due to traffic congestion, driving style, etc. The several combinations between the powertrain and the travel cost oscillation generate different paths between two given nodes. The model chooses the path between two nodes according to an efficiency-based decision, i.e., the path with the minimum expected travel cost is chosen. Moreover, the probability distributions of the travel costs are assumed to be unknown. In this work, the deterministic approximation becomes a TSP problem where the minimum expected total travel cost is equivalent to the maximum of the logarithm of the total accessibility of the Hamiltonian tours to the path set. They also evaluate the quality of the deterministic approximation by comparing it with the Perfect Information results obtained by a Monte Carlo method. The comparison shows good accuracy of the deterministic approximation, reducing the computational times of two orders of magnitude. Besides, computational results show how the derived model can be solved with difficulty within the timing restrictions of the application with reasonable accuracy.

Recently, Fadda et al. [66] showed that the independence assumption on the path travel costs could be relaxed, and a deterministic approximation of the stochastic multi-path traveling salesman problem by assuming just asymptotically independent travel costs is derived. They also show that this deterministic approximation has strong operational implications because it deals with realistic traffic models. Computational tests on extensive sets of random and realistic instances show very good efficiency and accuracy of the deterministic approximation.

4.5 Multi-stage Random Utilities Models for maximization problem

Several stochastic decision-making problems can be interpreted as multi-stage Random Utilities Models. A decision-maker selects, at each stage, an alternative among a finite set of choices. Depending on the application, the objective may be expressed in terms of utility maximization or cost minimization. In the current section, we mainly focus on the former case. The multi-stage stochastic decision-making process structure can be generally represented in Figure 4.1.

More precisely, let us introduce the following notation

- $t = 1, \ldots, T$: stage;
- N_t: set of choice alternatives at stage t;

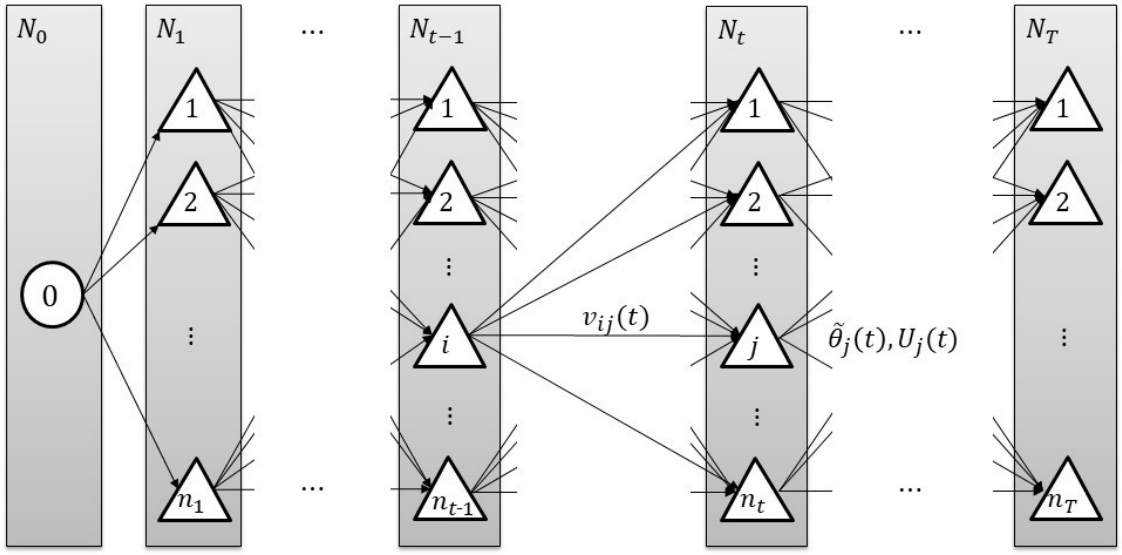


Figure 4.1: Multi-stage stochastic decision process.

- $N_0 = \{0\}$: initial start of the decision process, containing a singleton alternative 0;
- $L_j(t)$: set of realizations for alternative j at stage t;
- $l_j(t) = |L_j(t)|$: number of realizations for alternative j at stage t;
- $L = \cup_{t=1}^{T} \cup_{j \in N_t} L_j(t)$: total set of realizations of the decision process;
- $l = |L| = \sum_{t=1}^{T} \sum_{j \in N_t} l_j(t)$: total number of realizations of the decision process;
- $v_{ij}(t)$: deterministic utility of alternative j at stage t when the decision process starts from alternative i at stage $t-1$;
- $\tilde{\theta}_j^l(t)$: random term utility oscillation of the alternative j at stage t under realization $l \in L_j(t)$.

As in Tadei et al. [202], it is assumed that random oscillations $\tilde{\theta}_j^l(t)$ are independent and identically distributed (i.i.d.) stochastic variables for all j, l, and t, with a common *unknown* probability distribution

$$F(x) = Pr\{\tilde{\theta}_j^l(t) \leq x\}, \quad j \in N_t, l \in L_j(t), t = 1, \ldots, T, \tag{4.26}$$

and $\tilde{\theta}_j(t)$ is defined as the unknown maximum value of the random utility oscillations $\tilde{\theta}_j^l(t)$ among all the realizations $l \in L_j(t)$, i.e.,

$$\tilde{\theta}_j(t) = max_{l \in L_j(t)} \tilde{\theta}_j^l(t), \quad j \in N_t, t = 1, \ldots, T. \tag{4.27}$$

Since probability distribution $F(x)$ is unknown, $\tilde{\theta}_j(t)$ is still a random variable with the unknown probability distribution as follows

$$B_j(x,t) = \Pr\left\{\tilde{\theta}_j(t) \leq x\right\}, \quad j \in N_t, t = 1, ..., T. \tag{4.28}$$

Since $\tilde{\theta}_j(t) \leq x \Longleftrightarrow \tilde{\theta}_j^l(t) \leq x,\ l \in L_j(t)$ and $\tilde{\theta}_j^l(t)$ are independent, using (4.26), (4.28) becomes

$$B_j(x,t) = \prod_{l \in L_j(t)} \Pr\left\{\tilde{\theta}_j^l(t) \leq x\right\} = \prod_{l \in L_j(t)} F(x) = [F(x)]^{l_j(t)}, \tag{4.29}$$

where $l_j(t)$ is the total number of scenarios for alternative j at stage t.

Let $\tilde{v}_{ij}(t+1)$ is defined as the random utility of alternative j at stage $t+1$ when alternative i has been chosen at previous stage $t = 0, ..., T-1$. It is assumed that the decision process is efficiency-based so that, for any alternative $j \in N_{t+1}, t = 0, ..., T-1$, among the different realizations $l \in L_j(t+1)$ the one which maximizes the random choice utility will be considered. In other words, the decision-maker has an optimistic vision of the future. Not knowing the realizations and what will happen in the future, he assumes that he will benefit as much as possible. The random utility $\tilde{v}_{ij}(t)$ is composed of three terms, including the deterministic utility component $v_{ij}(t+1)$ of choosing choice j at stage $t+1$ after alternative i, random term $\tilde{\theta}_j^l(t+1)$ assumed i.i.d. with unknown probability distribution, and the expected utility alternative j at stage $t+1$ denoted as $U_j(t+1)$. Because of (4.27), the random utility $\tilde{v}_{ij}(t)$ becomes

$$\begin{aligned} \tilde{v}_{ij}(t+1) &= v_{ij}(t+1) + max_{l \in L_j(t+1)} \tilde{\theta}_j^l(t+1) + U_j(t+1) \\ &= v_{ij}(t+1) + \tilde{\theta}_j(t+1) + U_j(t+1), \\ &\quad i \in N_t, j \in N_{t+1}, t = 0, ..., T-1. \end{aligned} \tag{4.30}$$

with $U_i(T) = 0, i \in N_T$, where $U_i(t)$ is defined as, i.e.,

$$U_i(t) = \mathbb{E}_{\tilde{\theta}}\left[\max_{j \in N_{t+1}} \tilde{v}_{ij}(t+1)\right], \quad i \in N_t, t = 0, ..., T-1. \tag{4.31}$$

Equation (4.31) is based on the Bellman equation and indicates that the utility of choice j at stage $t+1$ is evaluated not only by its instantaneous utility $v_{ij}(t+1) + \tilde{\theta}_j(t+1)$ associated with the action made at current state but also by the expected utility $U_j(t+1)$ of the future selected alternatives. It means the decision-maker chooses an alternative (make an action) given the current state in a process having the Markov property. In such a way, the decisions become nested over stages. Also, it should be noted that a deterministic environment of decision

making is assumed, such that a given state and action will result in a new state deterministically in the next stage.

Now, by defining

$$\tilde{v}_i(t) = \max_{j \in N_{t+1}} \tilde{v}_{ij}(t+1), \quad i \in N_t, t = 0, ..., T-1, \tag{4.32}$$

equation (4.31) becomes

$$U_i(t) = \mathbb{E}_{\tilde{\theta}} \left[\tilde{v}_i(t) \right], \quad i \in N_t, t = 0, ..., T-1, \tag{4.33}$$

and the maximum utility U of the whole multi-stage stochastic decision process is

$$U = U_0(0) = \mathbb{E}_{\tilde{\theta}} \left[\tilde{v}_0(0) \right]. \tag{4.34}$$

However, the calculation of $U_0(0)$ requires the calculation of $\mathbb{E}_{\tilde{\theta}} \left[\tilde{v}_0(0) \right]$, which in turn requires to know the probability distribution of $\tilde{v}_0(0)$, or, because of the nested structure of the utilities, of $\{\tilde{v}_i(t), i \in N_t, t = 0, ..., T-1\}$. Let us call the probability distribution of $\tilde{v}_i(t)$ as

$$G_i(x, t) = Pr\{\tilde{v}_i(t) \leq x\}, \quad i \in N_t, t = 0, ..., T-1. \tag{4.35}$$

that is still unknown, since $\tilde{\theta}_j^l(t)$ have an unknown probability distribution. Nevertheless, the asymptotic approximation of $G_i(x, t)$, i.e. an approximation valid when the total number l of scenarios of the decision process becomes very large, will be derived in the next session.

4.5.1 Deterministic approximation approach

From Tadei et al. [202], it is assumed that $F(x)$, the probability distribution of $\tilde{\theta}_j^l(t)$, is asymptotically exponential in its right tail, i.e.,

$$\exists \beta > 0 \ \text{ such that } \ \lim_{y \to +\infty} \frac{1 - F(x+y)}{1 - F(y)} = e^{-\beta x}. \tag{4.36}$$

Following [200], [203], and [155], by using some results of the asymptotic extreme value theory [75], it can be shown that under assumption (4.36) the distribution $G_i(x, t)$ asymptotically converges to a Gumbel function as the total number of scenarios l becomes large. This is a very mild condition, as we observe that many probability distributions show such behavior, among them the Gamma, Gumbel, Laplace, and Logistic distributions.

First note that, because of (4.28), (4.29), (4.31), and (4.32), equation (4.35) becomes

$$\begin{aligned} G_i(x,t) &= Pr\{\tilde{v}_i(t) \le x\} = Pr\{\max_{j\in N_{t+1}} \tilde{v}_{ij}(t+1) \le x\} \\ &= Pr\{\max_{j\in N_{t+1}} [v_{ij}(t+1) + \tilde{\theta}_j(t+1) + U_j(t+1)] \le x\} \\ &= \prod_{j\in N_{t+1}} Pr\{v_{ij}(t+1) + \tilde{\theta}_j(t+1) + U_j(t+1) \le x\} \\ &= \prod_{j\in N_{t+1}} Pr\{\tilde{\theta}_j(t+1) \le x - v_{ij}(t+1) - U_j(t+1)\} \\ &= \prod_{j\in N_{t+1}} B_j(x - v_{ij}(t+1) - U_j(t+1), t+1) \\ &= \prod_{j\in N_{t+1}} [F(x - v_{ij}(t+1) - U_j(t+1)]^{l_j(t+1)}, \\ &\quad i \in N_t, t = 0, ..., T-1. \end{aligned} \tag{4.37}$$

Moreover, note that it is possible to fix the origin for the utility scale arbitrarily, i.e., the choice probabilities are unaffected by a shift in the utility scale, and any additive constant to the utilities can be ignored. Lets this constant be as the root a_l of the equation

$$1 - F(a_l|l) = 1/l, \tag{4.38}$$

where l is the total number of scenarios of the decision process.

By replacing $\tilde{v}_i(t)$ with $\tilde{v}_i(t) - a_l$ in (4.37) one has

$$G_i(x,t|l) = \prod_{j\in N_{t+1}} [F(x - v_{ij}(t+1) - U_j(t+1) + a_l)|l]^{l_j(t+1)}, \tag{4.39}$$

where $G_i(x,t|l)$ is used to underline the dependency of $G_i(x,t)$ from l.

Let us consider the ratio

$$\alpha_j(t) = l_j(t)/l, \quad j \in N_t, t = 1, ..., T, \tag{4.40}$$

and assume that this ratio remains constant for each pair (j,t) while the values of $l = 1,2, ...$ do increase.

Then, equation (4.39) can be written as

$$G_i(x,t|l) = \prod_{j\in N_{t+1}} [F(x - v_{ij}(t+1) - U_j(t+1) + a_l)|l]^{\alpha_j(t+1)l}. \tag{4.41}$$

Now, let us assume that l is large enough to use $\lim_{l\to+\infty} G_i(x,t|l)$ as an approximation of $G_i(x,t)$. Then, the following theorem holds.

Theorem 3. *Under condition (4.36), the probability distribution $G_i(x,t)$ becomes the following Gumbel function*

$$G_i(x,t) = \lim_{l\to+\infty} G_i(x,t|l) = \exp\left(-A_i(t)e^{-\beta x}\right), \quad i \in N_t, t = 0, ..., T-1, \tag{4.42}$$

where

$$A_i(t) = \sum_{j \in N_{t+1}} \alpha_j(t+1) e^{\beta[v_{ij}(t+1)+U_j(t+1)]}, \quad i \in N_t, t = 0, ..., T-1, \tag{4.43}$$

is the accessibility in the sense of Hansen [89] to the overall set of alternatives at stage(t+1).

Proof. By (4.39) and (4.40) one has

$$\begin{aligned} G_i(x,t) &= \lim_{l \to +\infty} G_i(x,t|l) = \\ &\lim_{l \to +\infty} \prod_{j \in N_{t+1}} [F(x - v_{ij}(t+1) - U_j(t+1) + a_l|l)]^{\alpha_j(t+1)l} = \\ &\prod_{j \in N_{t+1}} \lim_{l \to +\infty} [F(x - v_{ij}(t+1) - U_j(t+1) + a_l|l)]^{\alpha_j(t+1)l}. \end{aligned} \tag{4.44}$$

As $\lim_{l \to +\infty} 1/l = 0$, from (4.38) we have

$$\lim_{l \to +\infty} 1 - F(a_l|l) = 0, \tag{4.45}$$

then $\lim_{l \to +\infty} F(a_l|l) = 1$, i.e. $\lim_{l \to +\infty} a_l|l = +\infty$.

From (4.36), where $a_l|l$ plays the role of y, one obtains

$$\lim_{l \to +\infty} \frac{1 - F(x - v_{ij}(t+1) - U_j(t+1) + a_l|l)}{1 - F(a_l|l)} = e^{-\beta(x - v_{ij}(t+1) - U_j(t+1))}. \tag{4.46}$$

By (4.46) and (4.38) one gets

$$\begin{aligned} &\lim_{l \to +\infty} F(x - v_{ij}(t+1) - U_j(t+1) + a_l|l) = \\ &\lim_{l \to +\infty} \left(1 - [1 - F(a_l|l)] e^{-\beta(x - v_{ij}(t+1) - U_j(t+1))}\right) = \\ &\lim_{l \to +\infty} \left(1 - \frac{e^{-\beta(x - v_{ij}(t+1) - U_j(t+1))}}{l}\right). \end{aligned} \tag{4.47}$$

By substituting, after multiplying numerator and denominator by $\alpha_j(t+1)$, (4.48) into (4.45) one has

$$G_i(x,t) = \prod_{j \in N_{t+1}} \lim_{l \to +\infty} \left[1 - \frac{\alpha_j(t+1) e^{-\beta(x - v_{ij}(t+1) - U_j(t+1))}}{\alpha_j(t+1) l}\right]^{\alpha_j(t+1)l}, \tag{4.48}$$

and, by reminding that $\lim_{l \to +\infty} (1 + \frac{x}{l})^l = e^x$ and by using (4.43), (4.48) becomes

$$G_i(x,t) = \prod_{j \in N_{t+1}} \exp\left(-\alpha_j(t+1) e^{-\beta(x - v_{ij}(t+1) - U_j(t+1))}\right) = \exp\left(-A_i(t) e^{-\beta x}\right). \tag{4.49}$$

The probability distribution derived in (4.49) is a Gumbel distribution. □

Having now an explicit form for $G_i(x,t)$, we can calculate $\mathbb{E}_{\tilde{\theta}}[\tilde{v}_i(t)]$ in (4.33) as follows

$$U_i(t) = \mathbb{E}_{\tilde{\theta}}[\tilde{v}_i(t)] = \int_{-\infty}^{+\infty} x dG_i(x,t) = \tag{4.50}$$

$$\int_{-\infty}^{+\infty} x \exp\left(-A_i(t)e^{-\beta x}\right) A_i(t)e^{-\beta x}\beta dx,\ i \in N_t, t = 0, ..., T-1.$$

By substituting $z = A_i(t)e^{-\beta x}$, one gets

$$\begin{aligned} U_i(t) &= -1/\beta \int_0^{+\infty} \ln(z/A_i(t))e^{-z}dz = \\ &= -1/\beta \int_0^{+\infty} e^{-z}\ln z dz + 1/\beta \ln A_i(t) \int_0^{+\infty} e^{-z}dz = \\ &= \gamma/\beta + 1/\beta \ln A_i(t) = \\ &= 1/\beta(\ln A_i(t) + \gamma), \end{aligned} \tag{4.51}$$

where $\gamma = -\int_0^{+\infty} e^{-z}\ln z\, dz \simeq 0.5772$ is the Euler constant.

Because of (4.51), the optimal utility U of the whole multi-stage stochastic decision process in (4.34) becomes

$$U = 1/\beta \ln A_0(0) + \gamma/\beta. \tag{4.52}$$

4.5.2 A Nested Multinomial Logit model for the choice probability

As mentioned in the previous section, where the decision-maker has only a static set of alternatives to choose from (i.e., when there is only one stage of the decision making process), it is well-known that the choice probability reduces to a Multinomial Logit (MNL) model under the assumption that the random term utilities are independent and identically distributed (i.i.d.) and the common distribution has an asymptotically negative exponential behavior in its tail as well as having a large number of scenarios. Tadei et al. [202] has recently shown that the choice probability $p_{ij}(t+1)$ for decision-maker i at stage t to select alternative j at stage $t+1$ can be determined as follows. The decision-maker will choose alternative j at stage $t+1$ if and only if alternative j will have the largest utility among all the alternatives at that stage, i.e.,

$$\begin{aligned} v_{ij}(t+1) + \tilde{\theta}_j(t+1) + U_j(t+1) \geq v_{ik}(t+1) + \tilde{\theta}_k(t+1) + U_k(t+1), \\ i \in N_t,\ j,k \in N_{t+1},\ k \neq j, t = 0, ..., T-1. \end{aligned} \tag{4.53}$$

Then,

$$\begin{aligned} p_{ij}(t+1) = Pr\{v_{ij}(t+1) + \tilde{\theta}_j(t+1) + U_j(t+1) \\ \geq v_{ik}(t+1) + \tilde{\theta}_k(t+1) + U_k(t+1)\}, \\ \forall i \in N_t,\ j \neq k \in N_{t+1},\ t = 0, ..., T-1, \end{aligned} \tag{4.54}$$

and

$$Pr\{v_{ij}(t+1)+\tilde{\theta}_j(t+1)+U_j(t+1)\geq \max_{k\in N_{t+1};\, k\neq j} v_{ik}(t+1)+\tilde{\theta}_k(t+1)+U_k(t+1)\}, \quad \forall i\in N_t, j\in N_{t+1}, t=0,...,T-1. \tag{4.55}$$

By using (4.28) one gets

$$\begin{aligned} & Pr\{v_{ij}(t+1)+\tilde{\theta}_j(t+1)+U_j(t+1)\leq x\} \\ = \;& Pr\{\tilde{\theta}_j(t+1)\leq x-v_{ij}(t+1)-U_j(t+1)\} \\ = \;& B_j[x-v_{ij}(t+1)-U_j(t+1),t+1]), \end{aligned} \tag{4.56}$$

and, since $\{\tilde{\theta}_k(t), k\in N_{t+1}, t=0,...,T-1\}$ are independent,

$$\begin{aligned} & Pr\{\max_{k\in N_{t+1};\, k\neq j} v_{ik}(t+1)+\tilde{\theta}_k(t+1)+U_k(t+1)\leq x, i\in N_t, t=0,...,T-1\} \\ = \;& \prod_{k\in N_{t+1};\, k\neq j} B_k[x-v_{ik}(t+1)-U_k(t+1),t+1]. \end{aligned} \tag{4.57}$$

Now, from the Total Probability Theorem, equation (4.56) becomes

$$p_{ij}(t+1)=\int_{-\infty}^{+\infty}\left[\prod_{k\in N_{t+1};\, k\neq j} B_k[x-v_{ik}(t+1)-U_k(t+1),t+1]\right] dB_j[x-v_{ij}(t+1)-U_j(t+1),t+1], \quad i\in N_t, j\in N_{t+1}, t=0,...,T-1, \tag{4.58}$$

and the following theorem holds.

Theorem 4. *The choice probability $p_{ij}(t+1)$ for decision maker i at stage t to select alternative j at stage $t+1$ is given by*

$$p_{ij}(t+1)=\frac{l_j(t+1)e^{\beta[v_{ij}(t+1)+U_j(t+1)]}}{\sum_{k\in N_{t+1}} l_k(t+1)e^{\beta[v_{ik}(t+1)+U_k(t+1)]}}, \quad i\in N_t, j\in N_{t+1}, t=0,...,T-1, \tag{4.59}$$

which is a Nested Multinomial Logit (NMNL) model.

Proof. By using (4.29) and (4.40), from (4.58) one obtains

$$p_{ij}(t+1)=\int_{-\infty}^{+\infty}\prod_{k\in N_{t+1};\, k\neq j}\{F[x-v_{ik}(t+1)-U_k(t+1)]\}^{\alpha_k(t+1)l} d\{F[x-v_{ij}(t+1)-U_j(t+1)]\}^{\alpha_j(t+1)l}. \tag{4.60}$$

As per Theorem 3, by comparing (4.45) and (4.49), one can show that when $l\longrightarrow+\infty$

$$\{F[x-v_{ij}(t+1)-U_j(t+1)]\}^{\alpha_j(t+1)l}\longrightarrow \exp\left[-\alpha_j(t+1)e^{-\beta(x-v_{ij}(t+1)-U_j(t+1)-a_l)}\right]. \tag{4.61}$$

Finally, by setting $\gamma = e^{\beta a_l}$ and $z = e^{-\beta x}$, by (4.43) and (4.61), equation (4.60) becomes

$$
\begin{aligned}
& p_{ij}(t+1) = \\
& = \int_{-\infty}^{+\infty} \prod_{k \in N_{t+1};\, k \neq j} e^{\left[-\alpha_k (t+1) e^{-\beta[x - v_{ik}(t+1) - U_k(t+1) - a_l]}\right]} de^{\left[-\alpha_j (t+1) e^{-\beta[x - v_{ij}(t+1) - U_j(t+1) - a_l]}\right]} = \\
& = \int_{-\infty}^{+\infty} \prod_{k \in N_{t+1};\, k \neq j} e^{\left[-\gamma \alpha_k (t+1) e^{-\beta[x - v_{ik}(t+1) - U_k(t+1)]}\right]} de^{\left[-\gamma \alpha_j (t+1) e^{-\beta[x - v_{ij}(t+1) - U_j(t+1)]}\right]} = \\
& = \gamma \alpha_j (t+1) e^{\beta[v_{ij}(t+1) + U_j(t+1)]} \int_{-\infty}^{+\infty} \beta e^{-\beta x} exp[-\gamma A_i(t) e^{-\beta x}] dx = \\
& = \gamma \alpha_j (t+1) e^{\beta[v_{ij}(t+1) + U_j(t+1)]} \int_{0}^{+\infty} e^{-\gamma A_i(t) z} dz = \\
& = \frac{\alpha_j (t+1) e^{\beta[v_{ij}(t+1) + U_j(t+1)]}}{A_i(t)} = \\
& = \frac{l_j (t+1) e^{\beta[v_{ij}(t+1) + U_j(t+1)]}}{\sum_{k \in N_{t+1}} l_k (t+1) e^{\beta[v_{ik}(t+1) + U_k(t+1)]}}, \qquad i \in N_t, j \in N_{t+1}, t = 0, \ldots, T-1.
\end{aligned}
$$

□

4.6 Deterministic approximation for a minimization problem

The deterministic approximation approach computes the total expected value of the shortest path on the multi-stage decision-making process for the minimization problem. The procedure is similar to what was mentioned in the previous sections. However, all the utilities, namely, the deterministic ones v_{ij}^t, the random oscillations $\tilde{\theta}_j(t)$, and the expected utility U_j^t must be interpreted as costs. This means that U becomes the expected total cost of the decision process. Also, "max" must be substituted by "min" in all the related equations.

In the case of the minimization problem, the theoretical results can be obtained by assuming that $F(x)$ represents the survival function of the probability distribution of $\tilde{\theta}_j^l(k)$, i.e.,

$$
F(x) = Pr\{\tilde{\theta}_j^l(t) > x\}, \forall j \in N_t, l \in L_j(t), t = 0, ..., T, \tag{4.62}
$$

and it has an asymptotic exponential behavior in its left tail, i.e.,

$$
\exists \beta > 0 \text{such that} \lim_{y \to -\infty} \frac{1 - F(x+y)}{1 - F(y)} = e^{\beta x}. \tag{4.63}
$$

Considering the above assumptions, the final equations in Theorem 3, but in the minimization approach, are

$$G_i(x,t) = \lim_{l\to+\infty} G_i(x,t|l) = \exp\left(-A_i(t)e^{\beta x}\right), \quad i \in N_t, t = 0,...,T-1, \tag{4.64}$$

where the accessibility measure $A_i(t)$ and expected cost of alternative i at stage t are, respectively, computed as

$$A_i(t) = \sum_{j\in N_{t+1}} \alpha_j(t+1)e^{-\beta[v_{ij}(t+1)+U_j(t+1)]}, i \in N_t, t = 0,...,T-1, \tag{4.65}$$

and

$$U_i(t) = -1/\beta(\ln A_i(t) + \gamma), i \in N_t, t = 0,...,T-1. \tag{4.66}$$

So, the minimum total cost can be approximated by

$$U = -1/\beta \ln A_0(0) - \gamma/\beta. \tag{4.67}$$

Moreover, the Nested Multinomial Logit model for modeling the choice probability in the cost minimization problem is

$$p_{ij}(t+1) = \frac{l_j(t+1)e^{-\beta[v_{ij}(t+1)+U_j(t+1)]}}{\sum_{k\in N_{t+1}} l_k(t+1)e^{-\beta[v_{ik}(t+1)+U_k(t+1)]}}, \quad i \in N_t, j \in N_{t+1}, t = 0,...,T-1. \tag{4.68}$$

4.7 Summary

In this chapter, some background information and the deterministic approximation approach for both static and multi-stage Random Utilities Models proposed by Tadei et al. [202], are represented. For the static case, the related literature that contains the papers have applied the deterministic approximation approach in dealing with various applications. Moreover, the DA approach for multi-stage RUMs is derived with the main focus on the maximization problem. For the minimization problems, the assumptions and the final theoretical results are presented. In addition, the choice probability formulated as the Nested Multinomial Logit model is provided for both maximization and minimization multi-stage problems.

Chapter 5

Optimal paths in multi-stage stochastic decision networks

5.1 Introduction

Finding an optimal path is one of the most fundamental problems on networks with broad applications in various fields like computer science, robotics, operations research, and transportation planning. This problem can be interpreted as a multi-stage decision-making process where decisions are taken step by step to achieve an optimal sequence of choices over stages eventually. Most of the operations management areas such as logistics, routing, scheduling, project management, and finance face concrete settings that lead to finding an optimal sequence (path) of decisions over a multi-stage structure. Depending on the applications, the concept of *stage* may represent different discretization of the decision-making process and not necessarily a discretization of the time horizon.

In these problems, the choice utility at each stage is affected by the subsequent stages' utilities associated with the selected choices. In this sense, the decision process cannot be decomposed into distinct stages. So, the process finally leads to an optimal path made by the choices selected stage by stage in a sequential fashion. The idea of using a sequential decision-making model to describe a path has been around for quite some time (see, e.g., [4], [18], and [72]).

Depending on the application, the objective function can be expressed in terms of maximization of utilities (profits) or minimization of costs, which in turn may lead to a search for the most profitable or the less-costly path on a network, respectively. The most profitable path can be seen as the longest path when the arcs are associated with the utilities, while the less-costly path corresponds to the shortest path. In this chapter, we mainly provide a detailed approach to the maximization problem.

When all parameters of the problem are deterministically known a priori, finding

optimal paths is an easy problem to solve in general. However, in most real-life applications, parameters are highly affected by uncertainty, and there might be circumstances in which parameters are being changed dynamically over the entire decision horizon. It is easy to understand that, in those cases, ignoring the parameter variability and uncertainty may lead to inferior or, even worse, simply wrong decisions. It is also well-known that explicitly addressing uncertainty in an optimization problem generally increases the decision-making process's complexity and poses significant computational challenges. Therefore, it always makes sense to see whether it is possible to incorporate stochasticity in an approximated way, converting the stochastic model into a deterministic one and, if so, how accurate this approximation is.

This chapter defines the optimal path problem as a multi-stage stochastic decision process where the choice utilities are varying over stages and are also assumed to be stochastic variables with an unknown probability distribution.

This research provides the first application of the multi-stage dynamic stochastic decision process approach proposed by Tadei et al. [202], which is somehow consistent with a dynamic programming problem to determine the total utility of the optimal path. In this approach, the total utility is assumed to be stage additive, and each choice utility is the sum of three components. The first one represents a deterministic stage-dependent term of utility. The second part is a stochastic oscillation with unknown probability distribution. The third one is the expected utility of the selected choices of future stages (value function). This definition of utility is obtained by looking at the Bellman equation. In such a way, the decision-maker chooses a choice (make an action) given the current state in a stochastic process with the Markov property. So, the decisions become nested over the entire stages. It should be noted that a deterministic environment of decision making is assumed, such that a given state and action will result in a new state deterministically in the next stage.

For example, in routing problems, the deterministic utility component (or cost in this case) can be associated with a travel time between each pair of nodes, which changes over periods (stages). This value can be deterministically changed over various time stages due to different levels of traffic congestion. Also, we must consider stochastic utility oscillations of the travel time due to several factors like driving style, moving targets, or mobile obstacles in different periods. Finally, the selection of the next node on the network is affected by the expected travel time from that node on, making the random travel time at each stage be affected by the future alternatives.

This chapter mainly focuses on (i) providing the first concrete application of the deterministic asymptotic approximation proposed by Tadei et al. [202]. This approximation is used to determine the total utility of an optimal longest path over a network with a multi-stage structure. Its accuracy is tested versus benchmarks obtained by optimally solving the expected value problem over a great number of

different instances; (ii) deriving path solutions heuristically using a Nested Multinomial Logit model for the choice probability and investigates its quality; (iii) giving a way to calibrate a parameter inside the deterministic approximation approach, which is critical for its accuracy.

This chapter is organized as follows. In Section 5.2, a literature review of relevant problems that can be approached by the proposed method is listed, while Section 5.3 presents the mathematical model to find an optimal path in multi-stage stochastic decision networks. In Section 5.4, we derive a deterministic asymptotic approximation for the problem along the line of the approach proposed in [202]. In Section 5.5, we propose a procedure for the calibration of a parameter that is critical for the accuracy of the approach and we compare the results coming from the approximation with those of the expected value problem over several experiments. Finally, a brief summary is given in Section 5.6.

5.2 Literature review

Most of the operational management problems under uncertainty involve sequential decision processes. A decision-making action is made stage by stage, taking into account the state of the process and stage-dependent uncertain parameters. This section includes different problems in the related literature (clustered into application fields) that have been encapsulated in a multi-stage stochastic decision structure. Often, but not necessarily always, stages represent different periods that discretize the decision process.

It should be highlighted that some common optimization approaches in Operations Research, like Stochastic or Dynamic Programming, generally provide a conceptual framework based on multi-stage decision-making processes. Hence, all these problems have the potential to be addressed by our proposed deterministic asymptotic approximation approach since the inner problem they solve can be seen as finding an optimal path of choices in a multi-stage stochastic decision process.

5.2.1 Routing problems

Various classes of routing problems such as the Vehicle Routing Problem (VRP), Traveling Salesman Problem (TSP), and Travelling Purchaser Problem (TPP) have been considered under the assumption that necessary information is being dynamically changed at different stages of the horizon ([208], [159], [128]). Since time-dependency arises naturally in various routing applications due to traffic congestion, weather conditions, moving targets, or mobile obstacles, a huge body of research has been conducted in this area. Interested readers may refer to the survey by [79] for a comprehensive review of the works on different time-dependent routing problems and to [142] for a very good review on integrated transportation-inventory models.

A common feature of these problems is finding an optimal path or cycle, considering the variability of parameters like time, speed, and cost, which asks for a decision process on a multi-stage network. For instance, [12] address a multi-period VRP in which customers with due dates exceeding the planning period may be postponed by paying a cost. The objective of the problem is to find vehicle routes for each day (period) such that the overall cost of the distribution, including transportation costs, inventory costs, and penalty costs for postponed service, is minimized. [220] consider the dynamic multi-period VRP, which deals with the distribution of orders from a depot to a set of customers over a multi-period time horizon. Customer orders and their feasible service periods are dynamically revealed over time. The goal is to minimize the total travel cost and customer waiting by balancing the daily workload over the planning horizon. Other studies on dynamic multi-period routing problems can be found in [11], [115] and [5].

5.2.2 Network design

Decisions in network design problems (associated with strategic, tactical, and operational levels) concern with complex inter-relationships between suppliers, plants, distribution centers, customer zones, location, capacity, inventory, and financial decisions. In the past decades, a huge body of research has been conducted in various classes of logistics network design. Particular attention has been devoted to stochasticity and uncertainty, which lead to considering multi-period stochastic frameworks ([147], [152]). [178] and [53] develop capacitated multi-stage multi-product supply-chain models for reverse logistics operations. [233] propose a multi-stage stochastic model to deal with the design and planning problem of multi-period multi-product closed loop supply-chains with both uncertain supply and demand. Other multi-echelon similar models in closed-loop supply-chains can be found in [190] and [149]. A comprehensive review of studies in supply-chain network design under uncertainty is provided by [85].

5.2.3 Scheduling

Scheduling problems generally focus on allocating resources to different jobs to find an optimal sequence of jobs with minimum cost. In practice, parameters such as task processing times ([76]), availability of resources ([90]), as well as demand ([148]) and prices ([34], [141]) can all be subject to considerable changes over the horizon of scheduling decision. As an example, the uncertainty of job processing times may stem from different possible sources such as learning effect ([41], [109]), deterioration functions ([94], [217]), and resource allocation ([219], [118]). Considering variations in these parameters, scheduling problems can be converted into a multi-stage decision process where the decision on allocation is taken at each distinct stage.

5.2.4 Financial planning

Financial optimization, involving asset allocation and risk management, is one of the most attractive areas in decision-making under uncertainty. The problem determines how an investor should allocate funds among possible investment choices taking into account the optimal trade-off between return and risk. A comprehensive review of the approaches developed to address the problem is provided by [103]. Investors deal with uncertainty in different financial parameters such as return, risk, and turnover rates in the real world. Moreover, to construct a more realistic model, it is often necessary to investigate a multi-period optimization model (as in [20] and [82]). Therefore our decision process structure can be considered.

5.2.5 Project management

As already mentioned, project management problems often deal with the search of optimal paths, and, in real applications, parameters are affected by uncertainty. They may depend on the progress of the project itself. In these problems, a set of tasks with a given duration must be performed to complete a project as soon as possible (i.e., to minimize its make-span). Tasks can be represented as nodes of a network that is clustered into *ranks*, according to the precedence constraints between tasks [101]. The well-known Critical Path Method can be applied to this layered graph to obtain the most critical sequence of decisions that affect the project's completion time. It is easy to see that, in this setting, ranks are the stages of the problem, tasks are the different alternatives for each stage, and a critical path finding resorts to find the longest path linking decisions throughout the stages.

5.3 Longest path problem formulation

This section provides a formal description of the decision process at hand and its mathematical formulation as an optimal path problem under uncertainty. As mentioned before, the formulation proposed here is suitable for both the maximization approach (longest/most-profitable path) and the minimization approach (shortest path). Here, we will consider the maximization approach in detail.

5.3.1 Problem setting and notation

To achieve a more realistic representation of optimal path problems, we consider a multi-stage stochastic decision network. An optimal path is created by sequentially selecting nodes throughout the stages. Since utilities associated with each node are affected by uncertainty, we can interpret the decision process as a multi-stage Random Utility Model. At each stage, the decision-maker faces a set of alternatives to choose from in the next stage. According to their similarities, these

alternatives are grouped into clusters, which are represented by the *nodes*. Each node is characterized by a deterministic utility of choosing the next stage nodes (the deterministic arc weight) and the expected utility on the rest of the decision process (future stages). Moreover, inside each node, several alternatives are associated with a random utility oscillation that depends on their own dispersion in the cluster and the incomplete knowledge of the decision-maker.

Given the above interpretation, the deterministic approximation approach proposed by [202] described in the previous chapter can be applied to this problem. Please note that, in that work, the authors called "mutually exclusive alternatives" the nodes of our problem, while our different alternatives inside each node were called "realizations".

More precisely, let us introduce the following notation

- $k = 0, \ldots, K$: stage;
- $N_k = 1, \ldots, n_k$: set of nodes at stage $k = 0, \ldots, K$. Note that the set N_0 at the initial stage of the decision process contains only a singleton node 0, which can be seen as the decision maker of the process, and therefore it does not contains any alternative;
- $N = \bigcup_{k=1}^{K} N_k$: the entire set of nodes in the network;
- $L_j(k)$: set of alternatives inside node $j \in N_k$, $k = 1, \ldots, K$;
- $l_j(k) = |L_j(k)|$: number of alternatives inside node $j \in N_k$, $k = 1, \ldots, K$;
- $L(k) = \bigcup_{j \in N_k} L_j(k)$: total set of alternatives at stage k, $k = 1, \ldots, K$;
- $l(k) = |L(k)|$: number of alternatives at stage k, $k = 1, \ldots, K$;
- $L = \bigcup_{k=1}^{K} \bigcup_{j \in N_k} L_j(k)$: total set of alternatives of the decision process
- $w_{ij}(k)$: deterministic utility of node $j \in N_k$ when it is reached from node $i \in N_{k-1}$, $k = 1, \ldots, K$;
- $\tilde{\theta}_j^l(k)$: random oscillation over the deterministic utility associated with alternative $l \in L_j(k)$ inside node $j \in N_k$, $k = 1, \ldots, K$;
- $W_j(k)$: expected utility of node $j \in N_k$, $k = 0, \ldots, K-1$.

Then, the general structure of the considered multi-stage stochastic decision process can be represented as a network shown in Figure 5.1, where nodes are layered in stages and arcs connecting them have weights corresponding to the deterministic utilities $w_{ij}(k)$. Note that node j at stage k is zoomed-in to show the existence of several alternatives associated with utilities within a certain radius of stochasticity. Moreover, since each alternative l has a utility affected not only by its dispersion in

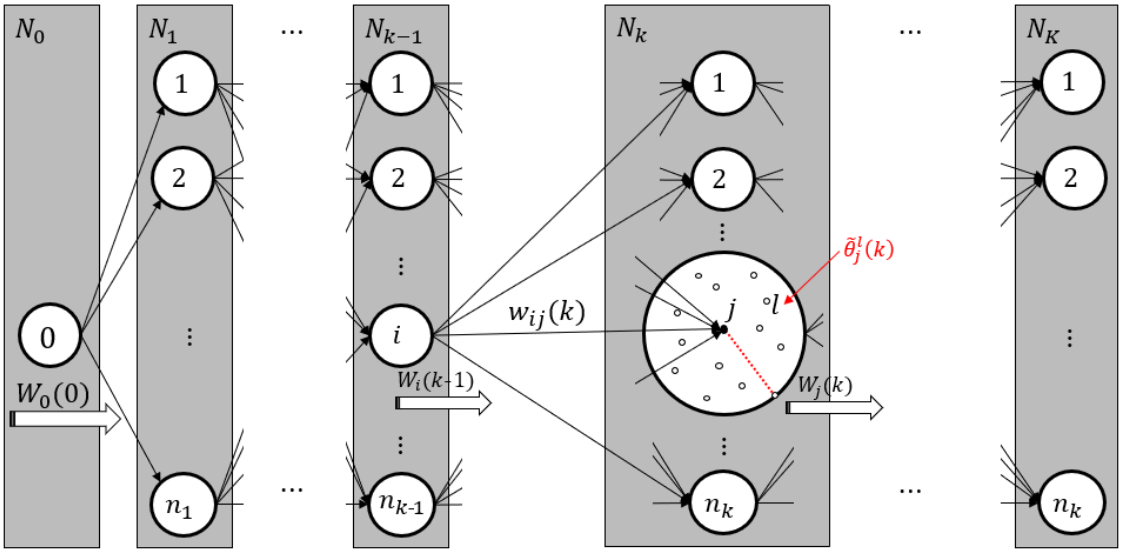


Figure 5.1: Illustration of a multi-stage stochastic decision process.

the node j but also by the incomplete knowledge of the decision-maker, a further oscillation $\tilde{\theta}_j^l(k)$ can be considered for each alternative l inside node j at stage k. Finally, expected utilities of the nodes in future stages $W_j(k)$ are represented as large white arrows at node j.

5.3.2 Toward a mathematical formulation

Let $\tilde{w}_{ij}(k)$ be the random utility of node $j \in N_k$ when it is reached from node $i \in N_{k-1}, k = 1, \ldots, K$. Since we assume an efficiency-based process, the decision-maker would choose, among the different alternatives $l \in L_j(k)$, the one which maximizes the random choice utility. In other words, it is assumed that the decision maker has a totally optimistic vision of the future. Not knowing the alternatives (random utility oscillations), he assumes that he will gain benefit as much as possible, i.e.,

$$\tilde{w}_{ij}(k) = \max_{l \in L_j(k)} (w_{ij}(k) + \tilde{\theta}_j^l(k) + W_j(k)) = w_{ij}(k) + \max_{l \in L_j(k)} \tilde{\theta}_j^l(k) + W_j(k), \quad (5.1)$$
$$i \in N_{k-1}, j \in N_k, k = 1, \ldots, K.$$

Now, by defining the maximum oscillation over the alternatives inside node j at stage k

$$\tilde{\theta}_j(k) = \max_{l \in L_j(k)} \tilde{\theta}_j^l(k), \quad j \in N_k, k = 1, \ldots, K, \quad (5.2)$$

one obtains

$$\tilde{w}_{ij}(k) = w_{ij}(k) + \tilde{\theta}_j(k) + W_j(k), \quad i \in N_{k-1}, j \in N_k, k = 1, \ldots, K, \quad (5.3)$$

where the expected utility of future nodes given stage k

$$W_i(k) = \begin{cases} \mathbf{E}_{\tilde{\theta}}\left[\max_{j\in N_{k+1}} \tilde{w}_{ij}(k+1)\right], & i \in N_k, k = 0, ..., K-1 \\ 0, & k = K \end{cases}. \tag{5.4}$$

Now by defining

$$\tilde{w}_i(k) = \max_{j\in N_{k+1}} (\tilde{w}_{ij}(k+1)), \quad i \in N_k, k = 0, \ldots, K-1, \tag{5.5}$$

equation (5.4) becomes

$$W_i(k) = \begin{cases} \mathbf{E}_{\tilde{\theta}}[\tilde{w}_i(k)], & i \in N_k, k = 0, ..., K-1 \\ 0, & k = K \end{cases}. \tag{5.6}$$

The recursive formula (5.3) is interpreted as the Bellman equation. It shows that the random utility of node $j \in N_k$, when it is reached from $i \in N_{k-1}$, is composed of the instantaneous utility, which is the sum of the deterministic utility $w_{ij}(k)$ and the random utility oscillation $\tilde{\theta}_j(k)$, and an expected utility of the future selected nodes $W_j(k)$. In such a way, the random utilities $\tilde{w}_{ij}(k)$ become nested over stages. In the terminology of Dynamic Programming, node $i \in N_{k-1}$ is a state, and selecting node $j \in N_k$ is a potential action state, i already chosen at the previous stage. At each stage, a node is chosen given the current state in a stochastic process with the Markov property. In our setting the value function is defined by $W(k)$, which is computed in (5.6) as a Bellman equation ([23]). It should be noted that, differently from other paradigms for modeling multi-stage optimization problems under uncertainty (e.g., Stochastic Programming or Stochastic Dynamic Programming), in our case, we assume that the uncertainty of random oscillations is not revealed over the process. Such a recursive formula allows determining the total utility of an optimal path without identifying the path (i.e., the sequence of decisions) itself.

Because of the nested structure of the decision process, the maximum expected total utility of the whole multi-stage stochastic decision process would be

$$W = W_0(0) = \mathbf{E}_{\tilde{\theta}}[\max_{j\in N_1} \tilde{w}_{0j}(1)] = \mathbf{E}_{\tilde{\theta}}[\tilde{w}_0(0)]. \tag{5.7}$$

Therefore, the optimal longest path problem in a multi-stage stochastic decision network can be formulated as follows

$$W = \mathbf{E}_{\tilde{\theta}}[\max_{j\in N_1} \tilde{w}_{0j}(1)x_{0j}^1], \tag{5.8}$$

subject to

$$\sum_{i\in N_{k-1}} x_{ih}^k - \sum_{j\in N_{k+1}} x_{hj}^{k+1} = 0 \qquad h \in N_k, k = 1, \ldots, K-1, \tag{5.9}$$

$$\sum_{i \in N_{k-1}} \sum_{j \in N_k} x_{ij}^k = 1 \qquad k = 1, \ldots, K, \tag{5.10}$$

$$x_{ij}^k \in \{0,1\} \qquad i \in N_{k-1}, j \in N_k, k = 1, \ldots, K, \tag{5.11}$$

where x_{ij}^k are Boolean variables taking value 1 if node j is selected at stage k after node $i \in N_{k-1}$, $k = 1, \ldots, K$, and 0 otherwise. The objective function (5.8) expresses the expected value of the maximum total utility of the whole multi-stage stochastic decision process. Please note that all the other variables x_{ij}^k are embedded into the variables x_{0j}^1 of the objective function. Constraint (5.9) ensure that the computed result is indeed a path between a source and a designated destination passing through stages. Constraint (5.10) indicates that only one decision is taken at each stage (choosing the next node). Finally, binary conditions on the variables are stated in (5.11).

5.4 Deterministic approximation

According to the model formulation above, in Section 5.4.1 we will derive in detail a deterministic approximation for the longest/most-profitable path value (maximization approach). In Section 5.4.2, we provide a way to derive a feasible solution for the optimization perspective. In Section 5.4.3, we briefly discuss the large applicability of the approximation approach.

5.4.1 Expected value of the longest path problem in maximization approach

Since we want to provide an approximation of the maximum total utility, let us first consider just the objective function (5.8). In that case, i.e., without any path constraint involved, the problem has the following trivial solution

$$x_{0j}^1 = \begin{cases} 1, & \text{if } \tilde{w}_{0j}(0) = \max_{q \in N_1} \tilde{w}_{0q}(1) \\ 0, & \text{otherwise} \end{cases} \qquad j \in N_1. \tag{5.12}$$

Now, by defining

$$\tilde{w}_0(0) = \max_{q \in N_1} \tilde{w}_{0q}(1), \tag{5.13}$$

and because of (5.12), the objective function (5.8) becomes

$$W = \mathbf{E}_{\{\tilde{\theta}\}}[\tilde{w}_0(0)]. \tag{5.14}$$

Note that the value of W in (5.14) cannot be calculated analytically since we do not have precise information on the distribution of the random oscillations $\theta_j^l(k)$. Let us call the probability distribution of $\theta_j^l(k)$ as

$$F(x) = Pr\{\tilde{\theta}_j^l(k) \leq x\}, \quad j \in N_k, l \in L_j(k), k = 1, \ldots, K, \tag{5.15}$$

and the cumulative right distribution function of $\tilde{w}_i(k)$ as

$$G_i(x,k) = Pr\{\tilde{w}_i(k) \leq x\}, \quad i \in N_k, k = 0, ..., K-1. \tag{5.16}$$

Then, we can use the following theorem

Theorem 5. *It is assumed that*

- *the random oscillations $\theta_j^l(k)$ are independent and identically distributed (i.i.d) random variables;*
- *$F(x)$ has an asymptotic negative exponential behavior in its right tail, i.e.,*

$$\exists \beta > 0 \mid \lim_{y \to +\infty} \frac{1 - F(x+y)}{1 - F(y)} = e^{-\beta x}. \tag{5.17}$$

Then,

$$G_i(x,k) = \lim_{|L| \to +\infty} G_i(x,k \mid |L|) = \exp\left(-A_i(k)e^{-\beta x}\right), \quad i \in N_k, k = 0, ..., K-1, \tag{5.18}$$

where $\beta > 0$ is a parameter to be calibrated,

$$A_i(k) = \sum_{j \in N_{k+1}} \alpha_j(k+1) e^{\beta[w_{ij}(k+1) + W_j(k+1)]}, \quad i \in N_k, k = 0, ..., K-1, \tag{5.19}$$

is the accessibility in the sense of [89] to the overall set of nodes at stage (k+1), and $\alpha_{ij}(k)$ is the ratio

$$\alpha_j(k) = l_j(k)/l(k), \quad j \in N_k, k = 1, ..., K, \tag{5.20}$$

which remains constant for each pair (j,k) while the number of alternatives do increase [202].

Theorem 5 states that the probability distribution $G_i(x,k)$ can be asymptotically approximated by a double-exponential (Gumbel) distribution. So, when the total number of alternatives of the decision process becomes very large, the expected value of the maximum total utility (5.14) can be approximated by

$$W = 1/\beta(\ln A_0(0) + \gamma), \tag{5.21}$$

where $\gamma \simeq 0.5772$ is the Euler constant, and

$$A_0(0) = \sum_{j \in N_1} \alpha_j(1) e^{\beta[w_{ij}(1) + W_j(1)]}, \tag{5.22}$$

is the *accessibility* to the set of nodes at stage 1. Note that the parameter β is interpreted as the dispersion of preferences among the different nodes at each stage under different realization of random oscillations. More precisely, this parameter aims at capturing the diversity of all the available nodes and their alternatives in each stage of decision making process.

5.4.2 Finding a feasible solution

It is important to highlight that the expected utility calculated in (5.21) represents just an approximation of the expected value of the maximum total utility. As already stated, the approximation is obtained by following the recursive formula in (5.3) that somehow also embeds the creation of a sequence of nodes, one selected at each stage. However, given the nature of the approximation, it is clear that the above sequence of nodes could not satisfy the path constraints (5.9)–(5.11).

Therefore, creating a feasible solution for the problem (besides the approximation of its objective function) is still an open issue, which can be addressed in the following way. Let us first recall a theorem, which holds under the same conditions discussed for Theorem 5 in Section 5.4.1.

Theorem 6. *The probability $p_{ij}(k)$ for choosing node j at stage k after node i has been selected at stage $k-1$ is given by*

$$p_{ij}(k) = \frac{l_j(k) e^{\beta[v_{ij}(k)+W_j(k)]}}{\sum_{q \in N_k} l_q(k) e^{\beta[v_{iq}(k)+W_q(k)]}}, \quad i \in N_{k-1}, j \in N_k, k = 1, ..., K, \tag{5.23}$$

which represents a Nested Multinomial Logit model [202].

Theorem 6 defines a model for the continuous probabilities of the choices and does not address the construction of a feasible path made by one single node per stage. However, let us consider a graph in a multi-stage structure, where to each arc (i, j) at stage k a weight equal to the probability $p_{ij}(k)$ (as defined in (5.23)) is assigned. By finding the longest path on that graph, an approximated optimal path solution over stages can be then determined. The rationale behind this choice is to look for the most probable sequence of nodes that satisfies constraints (5.9)-(5.11). We will also discuss the quality of this approach through various experiments in Section 5.5.

5.4.3 Applicability of the approximation

In this section, we want to highlight the quite large applicability of the proposed approximation. Even if condition in (5.17) yet represents a mild assumption on the shape of the distribution of the stochastic variables, [66] have recently proved that Theorem 5 still holds when (5.17) is relaxed to the following condition

$$\exists \beta > 0 \mid \lim_{|L(k)| \to +\infty} F(x + a_{|L(k)|})^{|L(k)|} = exp^{-e^{-\beta x}}, \tag{5.24}$$

where $F(x)$ is the probability distribution of $\tilde{\theta}_j(k)$, i.e.,

$$F(x, k) = Pr\{\tilde{\theta}_j(k) \leq x\}, \quad j \in N_k, k = 1, \ldots, K, \tag{5.25}$$

and $a_{|L(k)|}$ is chosen equal to the root of the equation

$$1 - F(x) = \frac{1}{|L(k)|}. \tag{5.26}$$

Assumption (5.24) is equivalent to ask the unknown distribution of the random oscillations $\tilde{\theta}$ to belong to the *domain of attraction* of a Gumbel distribution. The Gumbel domain of attraction describes distributions with light tails, i.e., probability distributions whose tails decrease exponentially. This assumption enlarges the set of distributions for which the deterministic approximation approach can be deployed in practice. It can be shown that such assumption is satisfied by widely used distributions such as the Normal, the Gumbel, the exponential, the Weibull, the Logistic, the Laplace, the Lognormal, and any cumulative distribution in the form $1 - e^{-p(x)}$, where $p(x)$ is a positive polynomial function. Thus, this is a very mild assumption and does not significantly restrict the deployment of the deterministic approximation approach. It is important to notice that some empirical results presented in [203], [66], and [173] have demonstrated the effectiveness of the DA framework even when the distribution of random oscillations does not satisfy the assumption (5.24) like the Uniform distribution.

5.5 Computational results

In this section, we present the computational experiments' results to evaluate the effectiveness of the deterministic approximation approach to finding the solution and value of optimal path in stochastic multi-stage networks. The evaluation is performed by considering the expected value problem, in which their expected values replace the random variables, and comparing its optimum value with that of the proposed approach. The deterministic approximation has been coded using MATLAB version R2016b, while the expected value problem has been implemented in GAMS 24.5.6. All the computational tests have been performed on an Intel(R) Core(TM) Processor $i5$-6200U (CPU 2.30 GHz) with 16 GB RAM.

In Section 5.5.1, we describe the instances set generated as a testbed for our assessment. The calibration of β parameter is presented in Section 5.5.2, while the results of our computational experiment are described and commented in Section 6.5.4.

5.5.1 Instance sets

To evaluate the performance of the proposed approximation approach, we randomly generated networks with $|N_k| = \{5, 10, 20, 50, 100\}$ nodes per stage $k = 1, \ldots, K$. Without loss of generality, we assume $N_k = N, k = 1, \ldots, K$, which indicates the nodes associated with each N_k represent the entire set of nodes for the

decision process at hand. This means that we can consider the same set of nodes at each decision stage and, to make sure to take into account all possible paths in the network, the number of stages K will be equal to the number of nodes (which is the same at each stage). In all the experiments, 100 alternatives for each node $j \in N_k, k = 1, \ldots, K$, have been considered, i.e., $|L_j(k)| = 100$.

The utility observations associated to the selection of alternative l inside node j coming from node i, $i \in N_{k-1}, j \in N_k, l \in L_j(k), k = 1, \ldots, K$, are generated using the Uniform, Normal, and Gumbel distributions in the range $[1, \delta]$, with $\delta = \{50, 100, 150\}$. The parameter δ simply allows us to control the behavior of the approximation against different magnitude of utilities. The deterministic utility $w_{ij}(k)$ is calculated as a mean value of the utility observations over the alternatives. For each one of the possible 45 combinations of some nodes per stage $|N_k|$, δ value, and the three distribution types (Uniform, Normal, and Gumbel), we generated ten random instances, which results in 450 instances in total. Note that, in generating observations using the Gumbel and Normal distributions, the *location* parameter $\mu = \delta/2$ is used. For the Gumbel distribution, a proportional scale parameter $\sigma = 0.5\mu$ is adopted. As suggested in [127], the scale factor has been chosen experimentally so that 98% of the probability lies in the considered truncated domain $[1, \delta]$ for each possible instance. Similarly, for the Normal distribution, the standard deviation $\sigma = \delta/6$ is set to give a 99% confidence interval.

Besides being very common in many practical applications, the three above distributions have been chosen to represent quite extreme cases of possible unknown distributions of observations to test throughout our experiments. Theoretically, the Gumbel distribution represents the best case to be approximated, whereas the Uniform distribution does not even satisfy the assumptions needed to derive the approximation. The Normal distribution is somehow between the two extremes, satisfying assumption (5.24) but not satisfying (5.17). A final remark is necessary to justify the Uniform distribution in our tests, for which the theoretical framework does not hold. In fact, in practical applications, it is often the case that a set of observed scenarios are available. Still, it is impossible to derive precise or even partial knowledge in terms of their probability distribution. Therefore, the experiments with the Uniform distribution can give us insights on whether the approach results accurate and robust even against unknown distributed scenarios, which may not satisfy our approximation assumptions.

5.5.2 Calibration of parameter β

The effectiveness of deterministic approximation is mainly dependent on an appropriate value of the positive parameter β. This parameter describes the dispersion of preferences among the different nodes and different realization of random oscillations at each stage of the decision-making process. Taking into account the concept of the expected utility of node j at stage k, the parameter β is computed

in a way that $W_j(k)$ should be equal to the maximum utility that can be achieved by choosing the next node and then an alternative inside it at stage $k+1$. Since depending on the current node and stage, the maximum possible utility in the next stage is different. The specific parameter β_{ik} should be calculated for any node i at any stage k, which allows us to model the uncertainty better. i.e.,

$$W_i(k) \quad = \quad 1/\beta_{ik}(lnA_i(k)+\gamma) = w_i^{max}(k), \quad i \in N_k, k = 0, \ldots, K-1. \tag{5.27}$$

Here, $w_i^{max}(k)$ represents, given node i at stage k, the maximum utility that can be achieved by making decision at stage $k+1$, i.e.,

$$w_i^{max}(k) = \max_{j \in N_{k+1}} (w_{ij}(k+1)+\bar{\theta}_j(k+1)+W_j(k+1)), \ i \in N_k, k = 0, \ldots, K-1, \tag{5.28}$$

where $\bar{\theta}_j(k+1)$ is the average value of random oscillation utilities over the alternatives inside node j, i.e.,

$$\bar{\theta}_j(k+1) = \sum_{l=1}^{L_j(k+1)} \frac{\theta_j^l(k+1)}{|L_j(k+1)|}, \quad j \in N_{k+1}, k = 0, \ldots, K-1. \tag{5.29}$$

In our case, since we take the deterministic term of utility $w_{ij}(k)$ as the average value of observations over alternatives, it is possible to collapse stages into a single one (containing all the possible alternatives) and to find just one β parameter value which works for the entire network (in line with Theorem 5). More precisely, let us consider only one decision making stage which contains all nodes of the network and assign to each of them a new utility weight $\bar{w}_j, j \in N$, computed as a mean value of deterministic utility $w_{ij}(k)$, i.e.,

$$\bar{w}_j = \sum_{k=1}^{K} \sum_{i \in N_{k-1}} \frac{w_{ij}(k)}{K \cdot |N|}, \quad j \in N. \tag{5.30}$$

Hence, the unique value of parameter β is calculated such that the expected utility of the decision making process (i.e., choosing one of these nodes) equals the maximum utility that can be achieved, i.e.,

$$1/\beta(ln(A_0(0)+\gamma)) = w^{max}. \tag{5.31}$$

The left hand side of Eq. (5.31) represents the expected utility W, $A_0(0)$ is the accessibility to all the nodes with new weights, and w^{max} is the maximum utility that can be achieved by choosing a node in a one-stage decision making process and is computed as

$$w^{max} = \max_{j \in N}(\bar{w}_j + \bar{\theta}_j), \tag{5.32}$$

in which $\bar{\theta}_j$ is the average value of random oscillation utilities over alternatives inside node j in all stages, i.e.,

$$\bar{\theta}_j = \sum_{k=1}^{K} \sum_{l=1}^{L_j(k)} \frac{\theta_j^l(k)}{K \cdot |L_j(k)|}, \quad j \in N. \tag{5.33}$$

Please note that this calibration approach is somewhat general and can be used with any available dataset. The experiments reported in Section 6.5.4 over randomly generated instances will show good confidence in this calibration method.

5.5.3 Results of the computational experiments

In this section, we summarize and discuss the results obtained from our experiments on all the instance sets generated as discussed in Section 5.5.1 and using the β value calibrated as presented in Section 5.5.2. The results contain two main distinct parts. The first one aims to assess the accuracy of the deterministic approximation of the maximum total utility W. In contrast, the second part evaluates the quality of the optimal path solutions derived by calculating an optimal path of a probability-weighted network as described in Section 5.4.2.

For maximum total utility and path solutions, the results of the approximation approach are compared with the Expected Value Problem (EVP) considered a benchmark. In the expected value problem, each arc's weight is considered the expected value over observations. Since the experiments are performed for the maximization approach, the benchmark path solution is derived by finding the longest/most profitable path in the network. The performance, in terms of percentage gap, is evaluated through the calculation of the Relative Percentage Error (RPE) as follows

$$RPE := \frac{W - opt_{EVP}}{opt_{EVP}} \times 100,$$

where W and opt_{EVP} represent the optimum of the deterministic approximation (see Equation (5.21)) and the optimum of the expected value problem, respectively.

Tables 5.1, 5.2, and 5.3 present the RPE associated with the instances generated by using the Uniform, Normal, and Gumbel distributions, respectively. Each entry of these tables reports statistics of the RPE over ten randomly generated instances, given a specific combination of values of $|N_k|$ and δ. In particular, the tables show the average, the best, the worst RPE, and its standard deviation (columns RPE_{avg}, RPE_{best}, RPE_{worst}, and RPE_{σ}, respectively).

The first thing that should be noticed is that, as expected, the average RPE increase as the size of the network grows for the three distributions with the worst case of 1.30, 1.40, and 2.45 in the 100-node network (average over δ values) for the Gumbel, Normal and Uniform distributions, respectively. Other trends can be noticed by looking at the average RPE for various values of δ. It seems that smaller intervals give better approximation results for the three distributions and across all the sizes. It means the approximation approach behaves better with smaller dispersion and magnitude of the realizations. The best RPEs also follow the same described behavior with little discontinuities, while the worst ones have more fluctuations for all the combinations of different network sizes and distributions. Considering the standard deviation RPEs, it can be seen that the values are very

Table 5.1: RPE of the maximum total utility between the deterministic approximation and expected value problem for the Uniform distribution.

Instance		$RPE(\%)$			
$\|N_k\|$	δ	RPE_{avg}	RPE_{best}	RPE_{worst}	RPE_{σ}
5	50	0.97	0.02	2.44	0.75
5	100	1.25	0.18	2.25	0.67
5	150	1.30	0.14	2.30	0.79
	Average:	1.17	0.12	2.33	0.74
10	50	1.33	0.21	2.18	0.68
10	100	1.38	0.14	2.15	0.62
10	150	1.40	0.35	2.69	0.68
	Average:	1.37	0.24	2.34	0.66
20	50	1.37	0.54	2.20	0.49
20	100	1.41	0.74	2.07	0.46
20	150	1.47	0.86	2.23	0.40
	Average:	1.41	0.71	2.17	0.45
50	50	1.99	1.44	2.32	0.26
50	100	2.06	1.56	2.71	0.37
50	150	2.16	1.54	2.56	0.30
	Average:	2.07	1.51	2.53	0.31
100	50	2.41	2.09	2.74	0.19
100	100	2.43	2.19	2.77	0.19
100	150	2.52	2.22	2.80	0.21
	Average:	2.45	2.17	2.77	0.20
Global Average:		**1.69**	**0.95**	**2.42**	**0.47**

similar together, i.e., less than 0.5% in the global average, which denotes good stability in terms of variance.

Comparing the results for the three distributions highlights better performance, as expected, of the Normal and Gumbel distribution concerning the Uniform distribution for all considered sizes and dispersions in terms of the average, best, worst, and standard deviation RPE.

We now want to assess the quality of the path solutions obtained using the Nested Multinomial Logit (NML) model explained in Section 5.4.2. We again use as a performance indicator the RPE, modified as follows

$$RPE := \frac{opt_{NML} - opt_{EVP}}{opt_{EVP}} \times 100,$$

where opt_{NML} represents the value of the feasible solution obtained through the

Table 5.2: RPE of the maximum total utility between the deterministic approximation and expected value problem for the Normal distribution.

Instance		$RPE(\%)$			
$\|N_k\|$	δ	RPE_{avg}	RPE_{best}	RPE_{worst}	RPE_{σ}
5	50	0.57	0.11	1.36	0.41
5	100	0.69	0.05	1.78	0.53
5	150	0.79	0.25	2.00	0.61
	Average:	0.69	0.14	1.71	0.52
10	50	0.54	0.02	1.31	0.50
10	100	0.60	0.14	1.01	0.25
10	150	0.68	0.34	1.03	0.24
	Average:	0.61	0.16	1.12	0.33
20	50	0.65	0.14	1.14	0.29
20	100	0.92	0.20	1.53	0.40
20	150	0.99	0.42	1.70	0.37
	Average:	0.85	0.25	1.45	0.35
50	50	1.01	0.84	1.29	0.14
50	100	1.29	0.83	1.78	0.30
50	150	1.34	0.88	1.72	0.25
	Average:	1.21	0.85	1.60	0.23
100	50	1.38	1.02	1.59	0.21
100	100	1.39	1.14	1.72	0.22
100	150	1.44	1.15	1.81	0.25
	Average:	1.40	1.10	1.70	0.22
Global Average:		**0.95**	**0.50**	**1.51**	**0.33**

application of the NML model. Table 5.4 reports RPE averages and standard deviations for the three distributions.

The results show promising performance of the Nested Multinomial Logit model for deriving optimal path solutions in terms of average and standard deviation RPE, indicating a good performance and overall robustness of the approach. As it can be seen, the quality of the path solution is not dependent on the network size and the distribution.

Finally, despite the quality obtained by the deterministic approximation approach in terms of accuracy, we also want to point out its efficiency. The computational times for deriving the maximum total utility (t_{DA}) and the path solution (t_{path}) are reported in Table 5.5. Since we have noticed that the computational times were somehow independent of the distribution used and the range of randomly generated utility weights, we report average computational times for the

Table 5.3: RPE of the maximum total utility between the deterministic approximation and expected value problem for the Gumbel distribution.

Instance		$RPE(\%)$			
$\|N_k\|$	δ	RPE_{avg}	RPE_{best}	RPE_{worst}	RPE_{σ}
5	50	0.43	0.08	1.06	0.36
5	100	0.48	0.13	1.30	0.35
5	150	0.70	0.03	1.55	0.55
	Average:	0.53	0.08	1.30	0.42
10	50	0.62	0.01	1.45	0.44
10	100	0.76	0.09	1.63	0.42
10	150	0.79	0.33	1.68	0.43
	Average:	0.72	0.14	1.59	0.43
20	50	0.68	0.18	1.33	0.39
20	100	0.79	0.40	1.24	0.26
20	150	0.86	0.31	1.27	0.31
	Average:	0.77	0.29	1.28	0.32
50	50	0.98	0.06	1.85	0.66
50	100	1.01	0.67	1.35	0.19
50	150	1.13	0.88	1.40	0.19
	Average:	1.04	0.54	1.53	0.35
100	50	1.22	0.95	1.46	0.15
100	100	1.33	1.05	1.60	0.17
100	150	1.36	1.20	1.58	0.13
	Average:	1.30	1.07	1.55	0.15
Global Average:		**0.87**	**0.42**	**1.45**	**0.33**

various sizes of the network. First, note that finding the optimal path solutions needs just slightly more time than the deterministic approximation calculation for larger-size instances. Moreover, not surprisingly, the time increases as the size of the network increases. The CPU time is in particular affected by the curse of dimensionality due to the recursive formula in (5.3). However, considering the difficulty of solving the underlying problem under uncertainty for a suitable number of scenarios, the approximation approach shows reasonably small CPU times. Few seconds are needed for networks with up to 20 stages and 20 nodes per stage, while about 10 minutes are needed for networks with up to 100 stages and 100 nodes per stage. This gives the possibility to embed the approximation into the most sophisticated optimization algorithms for multi-stage problems under uncertainty.

Table 5.4: RPE of the optimal path between the approximation approach and expected value problem for the Uniform, Normal and Gumbel distribution.

Instance		Uniform distribution		Normal distribution		Gumbel distribution	
$\|N_k\|$	δ	RPE_{avg}	RPE_{σ}	RPE_{avg}	RPE_{σ}	RPE_{avg}	RPE_{σ}
5	50	0.13	0.09	0.05	0.08	0.06	0.10
5	100	0.05	0.09	0.06	0.12	0.05	0.03
5	150	0.08	0.06	0.06	0.11	0.07	0.06
	Average:	0.09	0.08	0.06	0.10	0.06	0.06
10	50	0.12	0.04	0.05	0.12	0.06	0.06
10	100	0.07	0.07	0.03	0.04	0.05	0.10
10	150	0.11	0.12	0.04	0.03	0.04	0.05
	Average:	0.10	0.08	0.04	0.06	0.05	0.07
20	50	0.04	0.02	0.05	0.07	0.06	0.10
20	100	0.08	0.07	0.03	0.04	0.04	0.04
20	150	0.05	0.06	0.04	0.05	0.03	0.02
	Average:	0.05	0.05	0.04	0.05	0.05	0.06
50	50	0.05	0.05	0.04	0.07	0.05	0.07
50	100	0.04	0.03	0.05	0.05	0.03	0.04
50	150	0.04	0.05	0.05	0.06	0.04	0.05
	Average:	0.04	0.04	0.05	0.06	0.04	0.05
100	50	0.06	0.04	0.04	0.04	0.04	0.04
100	100	0.04	0.03	0.03	0.03	0.05	0.04
100	150	0.05	0.03	0.04	0.03	0.03	0.03
	Average:	0.05	0.03	0.03	0.03	0.04	0.04
Global Average:		**0.06**	**0.05**	**0.04**	**0.06**	**0.04**	**0.05**

Table 5.5: Computational time in seconds for calculating the deterministic approximation and path solutions.

N_k	$t_{DA}(s)$	$t_{path}(s)$
5	0.36	0.39
10	1.68	1.71
20	8.20	8.25
50	120	132
100	900	997

5.6 Summary

This chapter has used a quite efficient and accurate approach to estimate the value and structure of optimal paths in a multi-stage stochastic decision network. In this network, decisions are made under uncertainty, and the oscillations of the stochastic parameters follow an unknown probability distribution. The optimal path is seen as a sequential decision making over stages, where the uncertain utility of nodes at each stage is affected by both previous and next decisions. Using some results from [202], we have determined a deterministic approximation for the longest path value. Moreover, a feasible solution is obtained by heuristically using a Nested Multinomial Logit model, which gives the probability to choose each node optimally. On many random generated networks, numerical tests have shown accurate estimations concerning analogous results obtainable from solving the expected value problem. The performance of our deterministic approximation seems particularly good as the size of networks increase, making the proposed approach a valuable tool to support decision-making in stochastic multi-stage networks for large and complex applications.

Chapter 6

Stochastic single machine scheduling problem as a multi-stage dynamic random decision process

6.1 Introduction

Single machine scheduling is a decision-making process that plays a critical role in all manufacturing and service systems. This problem has been extensively investigated for a long time because of its practical importance in developing scheduling theory in more complex job shops and integrated processes. Here, the machine can be used for different jobs, but processing them depends on the configuration used. In general, switching from one configuration to another one implies a so-called setup time.

In machine scheduling problems, the setup times are considered either sequence-independent or sequence-dependent. In the former case, the setup times are negligible or assumed to be a part of job processing times, while in the latter case, the setup times depend on the job currently being scheduled and the last scheduled job. Sequence-dependent setup time between two different activities is encountered in many industries such as the printing industry, paper industry, automotive industry, chemical processing, and plastic manufacturing industry. Dudek et al. [59] reported that 70% of industrial activities include sequence-dependent setup times.

Another realistic aspect to consider is that the efficiency of workers or machines increases depending on the time spent by the jobs or repetition of activities in many manufacturing and service industries. Therefore, the actual processing time of a job could be shorter if scheduled at the end of a queue. This phenomenon, known as

learning effect, has been observed in various practical situations in several branches of industry and for a variety of activities [227],[77].

Depending on the production system, learning effects can be based on position or on the sum of processing times. In the former case, they depend only on the number of jobs being processed, while in the latter case, they depend on the sum of the processing times of the already processed jobs. In this chapter, the single machine scheduling with position-dependent learning effects is considered.

Finally, it is important to notice that the manufacturing and service systems operate under uncertainty in many realistic situations. The uncertain environment stems from various random events, such as machine breakdown, job cancellation, rush orders, and inaccurate expected jobs information. The majority of the literature on stochastic single machine scheduling mainly considers uncertain job processing time. However, in some real-world situations, the setup times may also be uncertain due to some random factors like crew skills, tools and setup crews, or unexpected breakdown of fixtures and tools. Despite it is a common case in several industries, we found only a few studies that consider stochastic sequence-dependent setup times [6].

In this work, we focus on a scheduling problem on a single machine with configurations that can perform all the jobs in various processing times. Each job has a deterministic processing time affected by the job-dependent learning effect and the selected machine configuration. The deterministic setup time of switching the machine from a configuration mode to another is sequence-dependent. A random variable associated with the job attributes, including processing time and setup time between the machine configurations, is defined. The problem objective is to determine the sequence of jobs and choose the configuration to process each job such that the makespan is minimized. This measure (makespan) is proposed, one of the most popular and critical objective functions for single machine scheduling problems. Our proposed problem belongs to NP-hard class since the deterministic single machine scheduling with sequence-dependent setup time is NP-hard [19].

This chapter addresses the proposed problem with three models, including two-stage and multi-stage stochastic programming and deterministic approximation approach. Stochastic Programming (SP) is one of the main existing paradigms to deal with uncertain data. It assumes that the random input data follow probability distributions and pursues optimality in the average sense, adopting a risk-neutral perspective. However, it is difficult to measure the probabilistic distribution of input data in practice. Even if an estimate of such a distribution is available, many scenarios are necessarily needed to approximate it accurately. Unfortunately, as the number of scenarios increases, the problem becomes more complex since the number of decision variables and constraints grows. To deal with these drawbacks, we approximate the problem using the multi-stage dynamic random decision process proposed recently by Tadei et al. [202], where the knowledge of the probabilistic distribution of uncertain data is not needed. According to this approach, the problem

is seen as a multi-stage decision process in which jobs and configurations are chosen step by step to achieve an optimal sequence eventually. A Gumbel distribution can asymptotically approximate the probability distribution of the best alternative, and, in turn, the makespan of the process can be analytically derived. Using this approach, the problem is first reformulated as a non-linear integer programming model. Then, it is converted into a simpler integer one to be solved.

In this chapter, we mainly focus on: (i) a single machine scheduling involving simultaneously stochastic sequence-dependent setup times and stochastic position-dependent processing times affected by learning effect is considered for the first time; (ii) two-stage and multi-stage stochastic programming formulation are derived from modeling the problem formally; (iii) approximated makespan and optimal solutions are found by adapting a deterministic approximation approach from the literature. This provides a powerful decision support tool that overcomes the computational burden of solving fat stochastic programs that depend on the number of scenarios considered.

This chapter is organized as follows. In Section 6.2, a literature review of the problem is given. Section 6.3 describes the problem and formulates it using the Two-stage and Multi-stage stochastic models. In Section 6.4, we propose a solution approach based on a Deterministic Approximation recently introduced in the literature. We first derive the approximation and, based on that, we then formulate the problem as a non-linear model. By defining a new accessibility measure, the model is converted into a shortest path problem. In Section 6.5, the computational results of the proposed models are provided. Finally, conclusions are given in Section 6.6.

6.2 Literature review

The first research on job scheduling problems was performed in the mid-1950s. Since then, thousands of papers on different scheduling problems have appeared in the literature. In the manufacturing industries, the machine environment is generally considered as the resource of scheduling problems. The production system sometimes includes a machine bottleneck which affects, in some cases, all the jobs. Since the management of this bottleneck is crucial, the single machine scheduling problem has been gaining importance for a long time. Here, we explored the scheduling literature within the single machine scheduling problems. The excellent surveys by [228], [160], [1], and the work proposed by [111] have detailed the literature on the theory and applications about this problem in the past several decades.

The majority of papers assumed sequence-dependent setup times, which occur in many different manufacturing environments. Angel Bello et al. [9] addressed the single machine scheduling with sequence-dependent setup times and maintenance with the aim of makespan minimizations. They developed a MIP model, its linear relaxation model, and an efficient heuristic approach to solve the larger instances.

Kaplanoglu [100] also addressed this problem but for the case of dynamic job arrivals. He developed a collaborative multi-stage optimization approach. Bahalke et al. [16] proposed a tabu search and genetic algorithm to deal with the single machine scheduling with sequence-dependent setup times and deteriorating jobs. Stecco et al. [196] deals with the single machine scheduling where the setup time is not only sequence-dependent but also time-dependent. They developed a branch-and-cut algorithm that solves the instances up to 50 jobs. Ying and Bin-Mokhtar [231] addressed this problem with the secondary objective of minimizing the total setup time where jobs dynamically arrive. They proposed a heuristic algorithm based on the dynamic scheduling system. There are numerous studies on single machine scheduling with sequence-dependent setup times and various performance measures, such as minimizing total flow time, tardiness, lateness, waiting time. We refer the reader to Allahverdi ([6], [7]) for the comprehensive survey of the models, applications, and algorithms.

As Allahverdi [6] emphasizes, the literature on stochastic sequence-dependent setup times is scarce. However, in some real-world situations, setup times may be uncertain due to random factors such as crew skills, tools and setup crews, and unexpected breakdown of fixtures and tools. In the literature of single machine scheduling, we found only a few papers that consider sequence-dependent setup times as random variables. Lu et al. [121] addressed a robust single machine scheduling problem with uncertain job processing times and sequence-dependent family setup times. They formulated the problem as a robust constrained shortest path problem and solved it by a simulated annealing-based heuristic. The objective was to minimize the absolute deviation of total flow time from the optimal solution under the worst-case scenario. Also, the interval data were used to generate uncertain parameters of sequence-dependent setup times. Ertem and O-T-Sarac [60] focuses on the single machine scheduling with stochastic sequence-dependent setup times to minimize the total expected tardiness. They proposed two-stage stochastic programming and the sample average approximation (SAA) method to model and solve the problem. The genetic algorithm is used to solve larger-size problems. Soroush [194] deals with position and sequence-dependent setup times in the single machine scheduling under uncertain job attributes, including processing time and setup times.

Many researchers have been focused on various scheduling problems with learning effects on processing times. Biskup [28] demonstrated that the makespan minimization on single machine scheduling with position-based learning could be optimally solved in polynomial time by using the shortest processing time (SPT) rule. Since then, many researchers have focused on scheduling with a position-based learning model and various performance measures. The most well-known ones include those of Mosheiov [143], Lee et al. [110], Zhao et al. [238], and Kuo and Yang ([106], [146]). A comprehensive review on different kinds of learning effect is proposed by Azzouz et al. [15]. Some extensions of the basic position-based

learning model have been presented, including the consideration of job-dependent position-based learning effects ([229], [42]), autonomous position-based and induced learning effects ([237], [96]), position-based learning and deteriorating effects ([206], [43], [197]), and both position-based and sum-of-the-processing-time-based learning effects ([225], [41]).

In the above studies, the processing time is assumed to be a deterministic value. However, real-world manufacturing and service systems usually work in the uncertain environment due to various random interruptions. The ignorance of uncertainty makes the schedules not executed as they are proposed and causes a gap between scheduling theory and practice. Most works in the literature of stochastic single machine scheduling mainly study uncertain processing time. Various objective functions have been considered, such as minimizing expected total tardiness, earliness and tardiness penalty costs, expected number of tardy jobs, expected total weighted number of early and tardy jobs, the expected value of the sum of a quadratic cost function of idle time and the weighted sum of a quadratic function of job lateness, mean completion time and earliness and tardiness costs, worst-case conditional value at risk of the job sequence total flow time, total weighted completion time or the total weighted tardiness [60]. Hu et al. [95] used uncertainty theory to study the single machine scheduling problem with deadlines and stochastic processing times with known uncertainty distributions. The aim is to derive a deterministic integer programming model using the operational law for inverse uncertainty distributions to maximize the expected total weight of batches of jobs. Pereira [156] addressed single machine scheduling under a weighted completion time performance metric in which the processing times are uncertain but can only take values from closed intervals. The objective is to minimize the maximum absolute regret for any possible realization of the processing times. An exact branch-and-bound method to solve the problem has been developed. Seo et al. [184] studied single machine scheduling to minimize the expected number of tardy jobs. The jobs have normally distributed processing times and a common deterministic due date. They proposed a non-linear integer programming model and some relaxations to solve it approximately.

Limited work exists to address problems with both learning effect and uncertainty on processing time in the single machine context. Li et al. [115] addressed the single machine scheduling problem with random nominal processing time and/or random job-based learning rate to minimize the expected total flow time and expected makespan. It has shown that the shortest expected processing time (SEPT) rule is optimal for minimizing the expected total flow time or makespan in the position-based learning model with only job processing time being random. The job-based learning model has proved that minimizing the expected total flow time or makespan is equivalent to solving a random assignment problem with uncertain assignment costs. Zhang et al. [236] studied the single machine scheduling problem with both learning effect and uncertain processing time. They proved that

the SEPT rule is optimal for minimizing the expected makespan and maximum lateness. Also, they studied the case with stochastic machine breakdowns.

Concerning the need to consider uncertain data, researchers have applied various methodologies to achieve optimal solutions, such as Robust Optimization and Stochastic Programming [125]. In Robust Optimization (RO), uncertain data are represented using continuous intervals, and the aim is to optimize the performance measure in the worst-case scenario. Lu et al. [121] studied the robust single machine scheduling with uncertain processing time and sequence-dependent family setup time represented by interval data. The objective is to minimize the absolute deviation of total flow time from the optimal solution of the worst-case scenario. They formulated the problem as a robust constrained shortest path problem and solved it by simulated annealing algorithm, which embeds a generalized label correcting method. Daniels and Kouvelis [48] addressed the single machine scheduling with the uncertain processing time and objective of total flow time. They used a branch-and-bound algorithm and two surrogate relation heuristics to find robust schedules. Yang and Yu [223] studied the same problem, but with a discrete finite set of processing time scenarios rather than interval data. They developed an exact dynamic programming algorithm and two heuristics to obtain robust schedules. Stochastic Programming is another approach to tackle machine scheduling. Job attributes (e.g., processing time, release time, setup time, due dates) follow given probability distributions and reach optimality in the average sense. Some studies indicate that the SP models of single machine scheduling are NP-hard under certain distributional assumptions of job processing time. For example, Soroush [193] addressed static stochastic single machine scheduling problem in which jobs have random processing times with arbitrary distributions, known due dates with certainty, and fixed individual penalties imposed on both early and tardy jobs. He showed that the problem is NP-hard and developed certain conditions under which the problem is solvable exactly. In the literature, optimizing various performance measures, such as flow time [3], [122]), maximum lateness [32], the number of late jobs ([215], [184], [212]), weighted number of early and tardy jobs [193], and the total tardiness ([60], [170]) have been addressed.

It is well-known that explicitly addressing uncertainty in an optimization problem generally poses significant computational challenges. Therefore, another line of research is to see whether it is possible to incorporate uncertainty in an approximated way and convert the stochastic model into a deterministic one. Tadei et al. [202] have recently proposed a deterministic approximation approach to deal with uncertainty in a multi-stage decision-making process, where the knowledge of the probability distribution of input data is not required. According to this approach, under appropriate and mild assumptions, the probability distribution of the best alternative can still be asymptotically approximated by a Gumbel distribution, and, in turn, the makespan can be analytically derived. This method was experimentally proved under a concrete application in Chapter 5 where we experimentally

showed how the approximation approach performs in finding the optimal sequence of choices in a multi-stage stochastic structure by comparing with the expected value problem. The accuracy of this approximation in the case of single-stage has been experimentally shown in several application domains (see, e.g., [154], [155], [204]).

6.3 Problem definition and mathematical formulation

The proposed problem aims to sequence a set of jobs that require processing on a single machine to minimize the makespan. The machine can handle one job at a time and works under a set of operating modes (from now on called *configurations*) which affect the job processing times. No pre-emption is allowed, i.e., a job must be completed without interruptions once it is started to be processed. Sequence-dependent setup times are assumed when the machine switches from a configuration to another one. Finally, considering the learning effect, the job processing times also depend on the job positions in the sequence. Let us consider the following notation

- $I = \{1,2,...,n\}$: set of jobs;
- $R = \{1,2,...,n\}$: set of positions in the job sequence;
- $F = \{0,1,2,...,m\}$: set of possible machine configurations. Note that 0 indicates an initial dummy configuration of the machine;
- $\alpha < 0$: learning effect rate;
- P_i^k: nominal processing time of job i under configuration k;
- $P_{ir}^k := P_i^k \cdot r^\alpha$: deterministic processing time of job i processed at position r under configuration k;
- S_r^{jk}: deterministic sequence-dependent setup time of the machine to switch from configuration j to configuration k at position r.

A stochastic fluctuation associated to the machine setup time and job processing time is considered to achieve a more realistic representation of manufacturing and service systems. The fluctuation related to process job i at position r under configuration k after switching from configuration j is represented by a random variable $\tilde{\theta}_{ir}^{jk}$, defined over a given probability space. As commonly done in the literature (see, e.g., [27]), we approximate the distribution of the random variable through a sufficiently large set of realizations. Therefore, let us consider the following additional notation:

- L_{ir}^{jk}: set of realizations of time oscillations related to process job i at position r under configuration k after switching from configuration j;
- π^l: probability of each realization $l \in L_{ir}^{jk}$;
- $\tilde{\theta}_{ir}^{jkl}$: random oscillation of processing job i at position r and configuration k, when the machine is switched from configuration j, under realization $l \in L_{ir}^{jk}$;
- L_r: set of total realizations of the random oscillations at position r.

In the following, we provide two different mathematical formulations for the problem described, using the well-known two-stage and multi-stage SP paradigms.

6.3.1 Two-stage Stochastic Programming formulation

The proposed problem can be formulated as a mixed-integer linear model using a two-stage SP model. The first-stage variables have to be decided before the actual realization of the uncertain parameters becomes available. Once the random events occur, the value of the second-stage (or *recourse*) variables can be decided.

There are two different types of operational decisions: assigning jobs to positions and choosing a configuration machine to process each job. Two variables sets corresponding to decisions before and after revealing information must be defined in the two-stage SP model. The first-stage decisions, common to all realizations, represent the jobs assignments to positions. The second-stage decisions, specific to each realization and dependent on the first-stage decisions, represent the choice of configuration to process each job. Obviously, in the two-stage model, the term *stage* corresponds to the period before and after revealing information. Therefore, the first stage is completed when all the jobs are assigned to positions. Then the second stage starts as soon as the uncertainties are revealed.

We consider the following variables

- y_{ir}: boolean variable equal to 1 if job i is assigned to position r, 0 otherwise;
- x_{ir}^{jkl}: boolean variable equal to 1 if job i is assigned to position r and processed under configuration k after switching from configuration j and realization l, 0 otherwise.

Then, a two-stage SP model for the problem is as follows

$$\min_{x} \sum_{l \in L_{ir}^{jk}} \pi^l \sum_{i=1}^{n} \sum_{r=1}^{n} \sum_{j=0}^{m} \sum_{k=1}^{m} (S_r^{jk} + P_{ir}^{k} + \tilde{\theta}_{ir}^{jkl})\ x_{ir}^{jkl} \tag{6.1}$$

subject to

$$\sum_{r=1}^{n} y_{ir} = 1 \qquad \forall i \in I, \tag{6.2}$$

$$\sum_{i=1}^{n} y_{ir} = 1 \qquad \forall r \in R, \tag{6.3}$$

$$\sum_{j=0}^{m}\sum_{k=1}^{m} x_{ir}^{jkl} = y_{ir} \qquad \forall i \in I, r \in R, l \in L_{ir}^{jk}, \tag{6.4}$$

$$\sum_{i=1}^{n}\sum_{j=0}^{m} x_{ir}^{jkl} = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,r+1}^{kjl} \qquad \forall k \in F \setminus \{0\}, r \in R \setminus \{n\}, l \in L_{ir}^{jk}, \tag{6.5}$$

$$\sum_{k=1}^{m} x_{i1}^{0kl} = y_{i1} \qquad \forall i \in I, l \in L_{i1}^{0k}, \tag{6.6}$$

$$y_{ir} \in \{0,1\} \qquad \forall i \in I, r \in R, \tag{6.7}$$

$$x_{ir}^{jkl} \in \{0,1\} \qquad \forall i \in I, r \in R, j, k \in F, l \in L_{ir}^{jk}. \tag{6.8}$$

The objective function (6.1) expresses the minimum expected makespan. Constraints (6.2) and (6.3) ensure that each job must be selected to be processed in exactly one position, and in each position, exactly one job is assigned. Constraint (6.4) states that the machine is switched from a configuration to another one to process each job (it should be noted that the two consecutive configurations are not necessarily different from each other). These constraints are also linking the two types of decisions. Constraints (6.5) form the sequence (flow) of configurations to process all jobs over the positions. It establishes that it is possible to switch from a certain configuration k to another if and only if the machine is already under configuration k. Without this constraint, the sequence of configurations switch over positions would not be continuous. Constraint (6.6) indicate that the machine is switched from the initial configuration 0 to one of the configurations to process the job in the first position. Finally, (6.7) and (6.8) are binary constraints on the variables.

It is important to notice that the above two-stage vision of the problem could be too simplistic since it assumes to collapse the implementation of the optimal second-stage decisions (those relating to the assignment of configurations to jobs) when all the uncertainty is revealed. In practice, for this kind of strongly-layered operational problems, a multi-stage SP approach could be more suitable, which is described in the next subsection.

6.3.2 Multi-stage Stochastic Programming formulation

The proposed scheduling problem can also be modeled using a Multi-Stage Stochastic Programming formulation. The decisions are taken stage by stage, along with the realizations of some random variables. Using the multi-stage stochastic programming paradigm, the uncertainty of random oscillations is dealt with a *scenario tree* as a branching structure representing the evolution of realizations over

stages. In this context, we define a symmetric and balanced scenario tree. The job positions in our proposed problem correspond to stages in the stochastic programming approach in which the decisions on choosing jobs and configurations are taken. In a scenario tree, a path of realizations from the root node to a leaf node represents a scenario ω, occurring with probability π^{ω}. We call Ω the entire set of scenarios, i.e., the set of the path realizations up to any leaf of the scenario tree. Two scenarios $\omega, \acute{\omega}$ are called *indistinguishable* at stage r if they share a common history of realizations until that stage. At the same time, they are represented by distinct paths. Also, each node o at stage r in the tree can be associated with a scenario group, represented as Ω_r^o, such that two scenarios that belong to the same group have the same realizations up to that stage. Moreover, the set of all the nodes at stage r is depicted as Φ_r.

Let us consider the following variables

- $x_{ir}^{jk\omega}$: boolean variable equal to 1 if job i is assigned to position r and processed under configuration k after switching from configuration j under scenario ω, 0 otherwise.

Then, a multi-stage SP model for the problem is as follows

$$\min_{x} \sum_{\omega \in \Omega} \pi^{\omega} \sum_{i=1}^{n} \sum_{r=1}^{n} \sum_{j=0}^{m} \sum_{k=1}^{m} (S_r^{jk} + P_{ir}^{k} + \tilde{\theta}_{ir}^{jk\omega})\ x_{ir}^{jk\omega} \tag{6.9}$$

subject to

$$\sum_{r=1}^{n} \sum_{j=0}^{m} \sum_{k=1}^{m} x_{ir}^{jk\omega} = 1 \qquad \forall i \in I, \omega \in \Omega, \tag{6.10}$$

$$\sum_{i=1}^{n} \sum_{j=0}^{m} \sum_{k=1}^{m} x_{ir}^{jk\omega} = 1 \qquad \forall r \in I, \omega \in \Omega, \tag{6.11}$$

$$\sum_{i=1}^{n} \sum_{j=0}^{m} x_{ir}^{jk\omega} = \sum_{i=1}^{n} \sum_{t=1}^{m} x_{i,r+1}^{kt\omega} \qquad \forall k \in F \setminus \{0\}, r \in R \setminus \{n\}, \omega \in \Omega, \tag{6.12}$$

$$x_{ir}^{jk\omega} = x_{ir}^{jk\acute{\omega}} \qquad \forall i \in I, j, k \in F, r \in R \setminus \{n\}, \omega, \acute{\omega} \in \Omega_r^o, o \in \Phi_r, \tag{6.13}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x_{i1}^{0k\omega} = 1 \qquad \forall \omega \in \Omega, \tag{6.14}$$

$$x_{ir}^{jk\omega} \in \{0,1\} \qquad \forall i \in I, r \in R, j, k \in F, \omega \in \Omega. \tag{6.15}$$

The objective function (6.9) expresses the minimum expected makespan. Constraint (6.10) ensure that each job must be selected to be processed in exactly one position under a switched configuration. In contrast, constraint (6.11) indicates that exactly one job and switched configurations are assigned to each position. Constraints (6.12) imposes the continuity of the sequence of configurations switch, as

explained for constraint (6.5). Constraints (6.13) indicate explicit non-anticipative conditions, which ensure that, for any pair of scenarios with the same history of realizations up to the stage (position) r, the decisions must be the same. These constraints imply that the decisions taken at any stage do not depend on future realizations of uncertainty. Still, they are affected by the previous realizations of uncertainty and the knowledge of previous decisions. Constraint (6.14) indicate that the machine is switched from the initial configuration 0 to one of the configurations to process the job assigned to the first position. Finally, (6.15) are binary conditions on the variables.

To sum up, in this section, we proposed a couple of formulations according to the two-stage and multi-stage stochastic programming paradigms. However, those models are highly computationally demanding. So, in the following section, relying on the multi-stage dynamic random decision process proposed by Tadei et al. (2019), we derive a deterministic approximation of the problem to allow the resolution of large-scale instances.

6.4 Deterministic Approximation (DA)-based solution approach

In the proposed problem, two types of operational decisions are taken: assigning jobs to positions and choosing configurations to process each job. This problem can be interpreted as a multi-stage stochastic decision-making process in which jobs and machine configurations are chosen stage by stage to create a sequence of jobs and configurations. Recently, Tadei et al. [202] proposed a Deterministic Approximation (DA) approach which computes the total expected cost of the optimal sequence of alternatives in a multi-stage network representing a random decision-making process. It should be noted that, differently from other paradigms for modeling multi-stage optimization problems under uncertainty (e.g., Stochastic Programming or Stochastic Dynamic Programming), the knowledge of the probability distribution of the random variables is not needed to derive the DA approach. The method also implies a static vision of the entire decision-making process. The uncertainty of the random oscillations is not revealed over time, and where decisions collapse to a single-stage approximation.

6.4.1 DA for multi-stage dynamic random decision processes

The DA approach can be applied in this case since we can interpret the proposed problem as a multi-stage Random Utility Model (RUM) as explained in [173]. In this problem, the positions where the jobs are processed form a multi-stage framework. At each stage (i.e., in each position), the decision-maker faces a set of choices,

including the combinations of jobs and machine configurations to choose from for the next stage. Each combination of job i and configuration k switched from configuration j at position r is characterized by a deterministic configurations setup S_r^{jk} and processing time P_{ir}^k, as well as an expected completion time of the remaining choices (combinations of jobs and machine configurations) in future positions. Moreover, for each choice, several realizations are associated with a random time oscillation that depends on its own dispersion around the deterministic times of a future decision and the incomplete knowledge of the decision-maker.

More formally, let $\tilde{t}_{ir}^{jk}$ be the random completion time of job i processed under configuration k at position r when the machine is switched from configuration j.

To derive a deterministic approximation of the stochastic problem, the framework from Tadei et al. [202] assumes that the decision-maker has an optimistic vision of the future. Not knowing what will happen in the future, he assumes that he will have to pay as little as possible. Then, from this assumption, it is derived an estimate of the expected future cost, which depends on a parameter whose appropriate calibration allows to mitigate the risk associated with the initial optimistic attitude. Therefore, in our context, it is optimistically assumed that we have to incur in a random choice cost $\tilde{t}_{ir}^{jk}$ represented by

$$\tilde{t}_{ir}^{jk} = \min_{l \in L_{ir}^{jk}} [S_r^{jk} + P_{ir}^k + \tilde{\theta}_{ir}^{jkl} + T_{ir}^k], \tag{6.16}$$

where T_{ir}^k indicates the expected completion time of the future schedules (job h and configuration $\acute{j}$) when they are reached from job i and configuration k at position r, i.e.,

$$T_{ir}^k = \begin{cases} \mathbb{E}_{\tilde{\theta}} \left[\min_{\acute{j} \in F, h \in I} \tilde{t}_{h,r+1}^{k\acute{j}} \right], & i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\} \\ 0, & r = n \end{cases}. \tag{6.17}$$

T_{ir}^k is unknown because it depends on future realizations of the random oscillations and future scheduling decisions. From Tadei et al. [202], some approximation results for the expected values of T_{ir}^k can be derived under the following assumptions:

1. the random oscillations $\tilde{\theta}$ are independent and identically distributed (i.i.d.) according to an unknown probability distribution;

2. the survival function $F(x)$ of the probability distribution of $\tilde{\theta}$ has an asymptotic exponential behavior in its left tail, i.e.,

$$\exists \beta > 0 \text{ such that } F(x) = \lim_{y \to -\infty} \frac{1 - F(x+y)}{1 - F(y)} = e^{\beta x}. \tag{6.18}$$

Then, two main results can be derived

- T_{ir}^{k} can be approximated by

$$T_{ir}^{k} = -\frac{1}{\beta}(\ln A_{ir}^{k} + \gamma) \qquad \forall i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\}, \tag{6.19}$$

where $\gamma \simeq 0.5772$ is the Euler constant and A_{ir}^{k} is the *accessibility* in the sense of [89] to the overall set of choices at position $r+1$ when it is reached from job i and configuration k at position r, i.e.,

$$A_{ir}^{k} = \sum_{h=1}^{n}\sum_{j=1}^{m} \alpha_{h,r+1}^{kj} \exp^{-\beta\left(S_{r+1}^{kj}+P_{h,r+1}^{j}+T_{h,r+1}^{j}\right)} \tag{6.20}$$
$$\forall i \in I, j, k \in F \setminus \{0\}, r \in R \setminus \{n\},$$

where $\alpha_{ir}^{kj} = \frac{|L_{ir}^{kj}|}{|L_r|}$ indicates the proportion of the number of realizations of job i and configuration k at position r with respect to the total number of all available choices realizations at position r;

- the expected makespan of all jobs on the single machine is represented by T_0 determined as:

$$T_0 = -\frac{1}{\beta}(\ln A_0 + \gamma), \tag{6.21}$$

where A_0 is the accessibility to all the choices and realizations in the first stage of decision making process.

Note that the parameter β can be interpreted as the dispersion of preferences among the different choices (each combination of job i and configuration k) at position (stage) r under realization l of random oscillations. More precisely, this parameter aims to capture the diversity of all the available choices and their random oscillations in each decision-making process stage. Also, this parameter allows to model more effectively and exhaustively the effect of uncertainty on the final decision while minimizing any risks associated with the optimistic nature of the approach.

However, in practice, it is not straightforward to calibrate this parameter reliably. So, despite a single β appears in the above equation, as highlighted in [173], the specific beta values β_{ir}^{k} for different triples of indexes (i, r, k) can be used associated with each job processing time. The configurations switch at any stage of the decision process. It allows us to model the uncertainty better since, depending on the current schedule and stage, the possible minimum expected completion time in the next stage is different. More details will be provided on β_{ir}^{k} calibration in section 6.5.2.

According to [173], the job positions on the machine are equivalent to the various stages of the decision-making process in which each stage includes a set of nodes corresponding to choices (combinations of jobs and configurations) in our problem. Also, the realizations which we consider for each choice correspond to the various alternatives inside each node.

6.4.2 DA-based Mixed Integer Non-Linear Programming formulation

Both the sequence of jobs and configurations and the expected makespan can be determined by using a non-linear model that embeds the Deterministic Approximation results.

Let us consider the following decision variables

- y_{ir}^k: boolean variable equal to 1 if job i is processed at position r under configuration k, 0 otherwise;
- A_{ir}^k: accessibility of job i and configuration k at position r to the set of available choices at position $r+1$;
- A_0: accessibility to the set of available choices at the first position;
- T_{ir}^k: expected total completion time of future schedules when they are reached from job i and configuration k at position r;
- T_0: expected makespan.

Then, a Mixed Integer Non-Linear Programming (MINLP) model for the problem is as follows

$$\min T_0 \tag{6.22}$$

subject to

$$\sum_{r=1}^{n}\sum_{k=1}^{m} y_{ir}^k = 1 \qquad \forall i \in I, \tag{6.23}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} y_{ir}^k = 1 \qquad \forall r \in R, \tag{6.24}$$

$$T_{ir}^k = -\frac{1}{\beta_{ir}^k}(\ln A_{ir}^k + \gamma) \qquad \forall i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\}, \tag{6.25}$$

$$A_{ir}^k = \sum_{h=1}^{n}\sum_{j=1}^{m}\left(\alpha_{h,r+1}^{kj}\exp^{-\beta_{ir}^k(S_{r+1}^{kj}+P_{h,r+1}^j+T_{h,r+1}^j)}\right)(1-\sum_{\acute{r}=1}^{r}\sum_{\acute{j}=1}^{m} y_{h\acute{r}}^{\acute{j}}) \tag{6.26}$$
$$\forall i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\},$$

$$T_{in}^k = 0 \qquad \forall i \in I, k \in F \setminus \{0\}, \tag{6.27}$$

$$A_0 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\alpha_{i1}^{0j}\exp^{-\beta_0(S_1^{0j}+P_{i1}^j+T_{i1}^j)}\right), \tag{6.28}$$

$$T_0 = -\frac{1}{\beta_0}(\ln A_0 + \gamma), \tag{6.29}$$

$$y_{ir}^{k} \in \{0,1\} \qquad \forall i \in I, r \in R, k \in F \setminus \{0\}, \tag{6.30}$$

$$T_{ir}^{k}, T_0, A_{ir}^{k}, A_0 \geq 0 \qquad \forall i \in I, k \in F \setminus \{0\}, r \in R. \tag{6.31}$$

The objective function (6.22) expresses the minimum expected makespan. Constraints (6.23) and (6.24) ensure that each job must be selected to be processed under one configuration in exactly one position each position exactly one job and one configuration are assigned, respectively. Using constraints (6.25) and (6.26), the expected completion time and the accessibility of job i and configuration k at position r to the set of available choices at position $r+1$ are computed, respectively. It should be noted that, in computing A_{ir}^{k}, the part $(1 - \sum_{\acute{r}=1}^{r} \sum_{\acute{j}=1}^{m} y_{h\acute{r}}^{\acute{j}})$ indicates that the set of available choices at position r contains the jobs that have not yet been processed at any positions before r. According to constraint (6.27), the expected completion time of all choices at the last position is equal to zero. The accessibility to all choices at the first position and the expected makespan are calculated in (6.28) and (6.29). Clearly, in calculating A_0, the set of available choices include all the combinations of jobs and configurations since no job has been assigned before. Finally, the binary and non-negative variables are indicated in (6.30) and (6.31).

6.4.3 Linearizing DA-based model

The mixed-integer non-linear formulation presented in the previous section is computationally challenging. This complexity is mainly due to the non-linearity of the model and the nested structure of some of its constraints. For instance, in constraints (6.25) and (6.26), the expected completion time T_{ir}^{k} is computed using the accessibility A_{ir}^{k} that in turn is depending on the expected completion time T_{ir+1}^{k} of the chosen schedule (job and configuration) in the next position. In the following, to make possible an efficient resolution of realistic instances of the problem, a linear model inspired by the non-linear formulation in (6.22)–(6.31) is developed.

As yet mentioned, the constraint (6.26) expresses the accessibility in the sense of Hansen. In the context of this work, such accessibility measures the attractiveness of a given schedule at the position r by computing a weighted sum of the exponential cost associated with the choices that would still be available at the position $r+1$ is the case such schedule is chosen. Unfortunately, the deployment of Hansen's accessibility leads to the deterministic approximation model in (6.22)–(6.31) that is computationally demanding. For this reason, by maintaining the model structure derived from the DA approach framework, we consider a different measure of accessibility that allows us to obtain a new formulation of the problem that can be effectively linearized. In particular, rather than taking a convex sum of the exponential cost of the alternative available at the next position as suggested by Hansen' measure of accessibility, the new measure assesses the attractiveness of a

given schedule at the position r by looking at the exponential cost of the best alternative (lowest processing plus switching time) among those still available at the position $r+1$ if such schedule is chosen. Therefore, the new accessibility measure A_{ir}^{k} which determines the attractiveness of choosing job i and configuration k at position r is computed as

$$A_{ir}^{k} = \sum_{h=1}^{n}\sum_{j=1}^{m}\left(\exp^{-\beta_{ir}^{k}(S_{r+1}^{kj}+P_{h,r+1}^{j}+T_{h,r+1}^{j})}\right) y_{h,r+1}^{j} \qquad (6.32)$$
$$\forall i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\},$$

Let us suppose, job h under configuration j is the most profitable choice at position $r+1$ when it is chosen after job i and configuration k at stage r, which means $y_{h,r+1}^{j} = 1$. Hence, the accessibility A_{ir}^{k} in equation (6.32) is represented as

$$A_{ir}^{k} = \exp^{-\beta_{ir}^{k}(S_{r+1}^{kj}+P_{h,r+1}^{j}+T_{h,r+1}^{j})} \qquad \forall i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\}, \qquad (6.33)$$

So, the expected completion time T_{ir}^{k} can be reformulated as

$$T_{ir}^{k} = -\frac{1}{\beta_{ir}^{k}}(\ln A_{ir}^{k} + \gamma) = S_{r+1}^{kj} + P_{h,r+1}^{j} + T_{h,r+1}^{j} - \frac{\gamma}{\beta_{ir}^{k}} \qquad (6.34)$$
$$\forall i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\}.$$

Now, by considering the new accessibility measure, a simpler non-linear model can be derived as follows

$$\min T_0 \qquad (6.35)$$

subject to

$$T_{ir}^{k} = \sum_{h=1}^{n}\sum_{j=1}^{m}\left(S_{r+1}^{kj} + P_{h,r+1}^{j} + T_{h,r+1}^{j} - \frac{\gamma}{\beta_{ir}^{k}}\right) y_{h,r+1}^{j} \qquad (6.36)$$
$$\forall i \in I, k \in F \setminus \{0\}, r \in R \setminus \{n\},$$

(6.37)

$$T_0 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(S_1^{0j} + P_{i1}^{j} + T_{i1}^{j} - \frac{\gamma}{\beta_0}\right) y_{i1}^{j}, \qquad (6.38)$$

$$T_{ir}^{k}, T_0 \geq 0 \qquad \forall i \in I, k \in F \setminus \{0\}, r \in R. \qquad (6.39)$$

and constraints (6.23), (6.24), and (6.30).

The objective function (6.35) expresses the minimum expected makespan. Using constraints (6.36), the expected completion time of job i and configuration k at position r are computed. The expected makespan is calculated in (6.38).

By comparing equations (6.17) and (6.34), we can notice that $-\frac{\gamma}{\beta_{ir}^{k}}$ is equivalent to the expected minimum random oscillation of job $\acute{i}$ processed at position $r+1$ under configuration j over the realizations, i.e.,

$$-\frac{\gamma}{\beta_{ir}^{k}} \equiv \mathbb{E}_{\tilde{\theta}}\left[\min_{l \in L_{h,r+1}^{kj}} \tilde{\theta}_{h,r+1}^{kjl}\right], \qquad \forall i \in I, r \in R, k \in F \setminus \{0\}. \tag{6.40}$$

where the switch from configuration k to configuration j is performed at position $r+1$ to process job h. Having β_{ir}^{k} value, the value of $\mathbb{E}_{\tilde{\theta}}\left[\min_{l \in L_{h,r+1}^{kj}} \tilde{\theta}_{h,r+1}^{kjl}\right]$ can be approximated using (6.40).

Given the above reasoning, we can convert the non-linear formulation into a deterministic linear model. In particular, the optimal sequence of jobs and configurations as well as the makespan are computed by finding a shortest path on the multi-stage network (as done in [173]), through the following model where the expected minimum random oscillation is represented as $\bar{\theta}$

$$\min_{x}\left[\sum_{i=1}^{n}\sum_{r=1}^{n}\sum_{j=0}^{m}\sum_{k=1}^{m}(S_{r}^{jk}+P_{ir}^{k}+\bar{\theta}_{ir}^{jk})x_{ir}^{jk}\right] \tag{6.41}$$

subject to

$$\sum_{r=1}^{n}\sum_{j=0}^{m}\sum_{k=1}^{m} x_{ir}^{jk} = 1 \qquad \forall i \in I, \tag{6.42}$$

$$\sum_{i=1}^{n}\sum_{j=0}^{m}\sum_{k=1}^{m} x_{ir}^{jk} = 1 \qquad \forall r \in I, \tag{6.43}$$

$$\sum_{i=1}^{n}\sum_{j=0}^{m} x_{ir}^{jk} = \sum_{i=1}^{n}\sum_{t=1}^{m} x_{i,r+1}^{kt} \qquad \forall k \in F, r \in R \setminus \{n\}, \tag{6.44}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{i1}^{0k} = 1, \tag{6.45}$$

$$x_{ir}^{jk} \in \{0,1\} \qquad \forall i \in I, r \in R, j,k \in F. \tag{6.46}$$

The objective function (6.41) expresses the minimum makespan. Constraints (6.42) ensure that each job must be selected to be processed at exactly one position under a switched configuration. In contrast, constraints (6.43) indicate that exactly one job and switched configurations are assigned to each position. Constraints (6.44) ensures the continuity of configurations switch over the positions. Constraints (6.45) indicate that the machine is switched from the initial configuration 0 to one of the configurations to process the job assigned to the first position. Finally, (6.46) are binary conditions on the variables.

6.5 Computational results

In this section, we present the results of the computational experiments to evaluate the effectiveness of the DA approach compared to the two-stage and multi-stage recourse models to address the proposed problem. In section 6.5.1 we describe the instances set generated as a testbed for our assessment. The calibration of parameter β is presented in Section 6.5.2, while the value of stochastic solutions including both two-stage and multi-stage recourse models are discussed in Sections 6.5.3 and 6.5.3, respectively. The results of our computational experiment are described and commented on in Section 6.5.4.

Because of the computational complexity of the non-linear DA-based model presented in Section 6.4.2, the derived equivalent shortest path model described in Section 6.4.3 is applied to solve the instances. All the models are implemented in GAMS[1] on an Intel(R) Core(TM) i5-6200U (CPU2.30GHz) computer with 16GB RAM.

6.5.1 Instance generation

To evaluate the performance of the proposed approximation approach, we randomly generated instances classified into small and large-scale groups. The small-scale group involves instances with 3 and 5 jobs processed on a machine with two configurations, while the larger ones include 10, 20, 30, and 40 jobs scheduled on a machine with 3, 4, 4, and 5 configurations. The equivalent shortest path model is compared to the two-stage and multi-stage stochastic models in dealing with large and small-scale instances.

Our test generating procedure is somehow similar to the work proposed by Ertem and O. T. Sarac [60]. The nominal processing times are generated from the Uniform distribution in the range [1,100]. In contrast, the sequence-dependent setup times between machine configurations are produced using the Uniform distribution in the range [50,150]. The learning effect is assumed to be $\alpha = \log_2(0.8)$. The random oscillations are generated according to the Uniform, Normal, and Gumbel distributions in two ranges $[-0.5d,0.5d]$ and $[-0.9d,0.9d]$, where d is the sum of job processing time and the machine setup time associated with each combination of job, configuration switch, and position. Besides being common in many practical applications, the three above distributions have been chosen to represent quite extreme cases of possible unknown distributions of observations. Also, the two smaller and larger ranges allow us to control the behavior of the proposed approaches against the different magnitude of random oscillations. For each problem, 10 random instances are generated, which result in 360 instances in total.

[1] https://www.gams.com/

Note that, in generating realizations using the Gumbel and Normal distributions, the location parameter $\mu = 0$ is used for small and large ranges. For the Gumbel distribution, the proportional scale parameter $\sigma = 0.125d$ and $\sigma = 0.220d$ are adopted for the smaller and larger ranges, respectively. As suggested by Manerba et al. [127], the scale factor has been chosen experimentally so that 96% of the probability lies in the considered truncated domains $[-0.5d,0.5d]$ and $[-0.9d,0.9d]$ for each possible instance. Similarly, for the Normal distribution, the standard deviation $\sigma = 0.166d$ and $\sigma = 0.3d$ is set to give a 99% confidence interval for the smaller and larger ranges, respectively.

6.5.2 Calibration of parameter β

The effectiveness of the deterministic approximation model is highly dependent on the appropriate value of the positive parameter β. As shown by [173], the parameter β represents the dispersion of preferences among all choices and their realizations at each stage of the decision-making process. They proposed a method to calibrate this parameter per choice and stage, considering that each choice's expected utility (cost) at each stage should be equal to the maximum utility (minimum cost) that can be achieved by choosing the next stage. Moreover, as described by [202], β is a parameter of a Nested Multinomial Logit model under which the choice probability of each alternative is determined in each stage of the decision-making process. The analytical results derived from the choice probability model highlight that when β tends to zero, the choices probabilities are equal, which means that all the choices are equivalent. On the other hand, when the magnitude of uncertainty increases, the effect of random oscillations on the deterministic cost of choices becomes large. In this case, all the choices tend to be equivalent since it is difficult to decide which choice is the best. The above explanations suggest that parameter β should be inversely proportional to the range of random oscillations of the choices in the next position. Based on the above discussion, we decided to calibrate this parameter as

$$\beta_{ir}^{k} = \frac{\gamma}{\delta\, STD_{h\in I, j\in F\setminus\{0\}, l\in L_{h,r+1}^{kj}}(\tilde{\theta}_{h,r+1}^{kjl})} \qquad i \in I, r \in R, k \in F \setminus \{0\}, \tag{6.47}$$

where the nominator represents the Euler constant, while the denominator indicates a proportion $\delta < 1$ of the standard deviation of the realizations associated with all the jobs and the configurations (switched form k) in the next position. We define $\delta = \frac{\sigma}{2}$, which indicates that this parameter is depending on the standard deviation of the estimated probability distribution derived from the realizations. The values of this parameter used in this paper are summarized in Table 6.1. As it can be seen, the δ value associated with the smaller range is less than that of the larger interval for the three distributions. Also, the δ value in the Uniform

distribution is larger for both small and large ranges than the two other ones, which implies the larger dispersion of this distribution. The experiments reported in Section 6.5.4 over randomly generated instances will show the accuracy of this calibration method.

Table 6.1: Value of parameter δ for problem sets.

Distribution	**Small range**	**Large range**
Uniform	0.25	0.45
Normal	0.09	0.20
Gumbel	0.08	0.18

6.5.3 Value of the stochastic solutions

Stochastic programs, in general, have the reputation of being computationally difficult to solve. Before solving a stochastic model, the user must investigate and solve several deterministic models, each corresponding to one particular scenario or to a group of them to see if the scenarios, stages, and other aspects of the model are well defined, and the solution of the stochastic model (Recourse Problem) is justified. In the following two subsections, the value of stochastic solutions is determined for both two-stage and multi-stage recourse models.

Value of two-stage stochastic solutions

The Value of Stochastic Solution (VSS) is a concept to compare the recourse problem (RP) and the expected values approach (EEV) to see whether or not the approximation of the stochastic program by the program with expected values instead of random variables is a good one. This measure in the two-stage recourse model is calculated as

$$VSS = EEV - RP, \tag{6.48}$$

where EEV is obtained as follows: (1) solve the related average value problem (EV); (2) fix the first stage solution for each scenario (in the recourse problem) at the optimal one obtained for the first stage of the EV problem; (3) solve the resulting problem for each scenario; (4) calculate the expectation over the set of scenarios of the value at the objective function of these modified recourse problems.

To compute the VSS, the number of observations considered in the model plays a key role. There is a trade-off between the optimality of the results and the computational complexity caused by increasing the realizations number. Here, to assess the required number of observations to solve the recourse problem in larger-sized instances, we implement a set of experiments and vary the number of realizations from 30 to 50 with the same probability. For all the instances in

the larger-sized group, 50 realizations show that the upper and lower bounds are converging well enough. But, due to the computational complexity in the instances with 30 and 40 jobs (with 4 and 5 configurations), we fix the number of realizations to 30 for these instances. It should be noted that this number still shows well-enough results in a reasonable time. Table 6.2 reports the average value of the percentage VSS concerning the RP value for the large-scale instances computed as

$$VSS = \frac{EEV - RP}{RP} \times 100. \tag{6.49}$$

It can be observed, as expected, the VSS of instances associated with the larger range of random variables is higher than those obtained in the smaller one. Also, this parameter is affected by the type of probability distributions. It seems, the Uniform distribution results in larger average VSS in comparison with the two other distributions.

Table 6.2: Average value of the percentage VSS with respect to the RP value for large-scale instances.

Instance			**Uniform**	**Normal**	**Gumbel**
$\|I\|$	$\|F\|$	Range	VSS_{avg}	VSS_{avg}	VSS_{avg}
10	3	small	6.68	1.47	1.39
10	3	large	42.64	8.08	7.86
20	4	small	8.95	1.48	1.44
20	4	large	52.43	10.87	10.09
30	4	small	9.36	1.43	1.07
30	4	large	57.71	10.95	10.20
40	5	small	10.34	1.50	1.13
40	5	large	60.22	11.05	10.61

Value of multi-stage stochastic solutions

In this work, the value of the multi-stage stochastic solution is calculated using the generalization of parameter VSS proposed by Escudero et al. [61]. They defined the value of stochastic solution at stage r, denoted by VSS_r, as

$$VSS_r = EEV_r - mRP \qquad \forall r \in R, \tag{6.50}$$

where

- EEV_r for $r = 2, ..., n$, is the optimal value of the recourse problem in which the decision variables until stage $r-1$ are fixed at the optimal values obtained in the solution of the average value model, i.e.,

$$EEV_r = \begin{cases} (6.9) - (6.15) \\ s.t. \\ x_{i1}^{jk\omega} = \bar{x}_{i1}^{jk} \quad \forall \omega \in \Omega \\ \dots \\ x_{ir-1}^{jk\omega} = \bar{x}_{ir-1}^{jk} \quad \forall \omega \in \Omega \end{cases} \tag{6.51}$$

in which $\bar{x}_{ir}^{jk}$ denotes the optimal solution of the EV problem;

- mRP is the value of the multi-stage SP model (6.9)-(6.15).

It has been proved ([61]) that the following relations for any multi-stage stochastic program hold

$$0 \leq VSS_r \leq VSS_{r+1} \qquad \forall r \in R. \tag{6.52}$$

This sequence of non-negative values represents the cost of ignoring uncertainty until stage r in multi-stage models' decision-making.

Because of the branching structure in the scenario tree, even with a small number of realizations and stages, the tree becomes extremely large and difficult to manage and solve. Therefore, only small-scale instances have been solved using the multi-stage recourse model. Here, we fix the number of realizations to 7 and 3 associated with the instances with 3 and 5 jobs, respectively, which leads to the trees with $7^3 = 343$ and $3^5 = 243$ scenarios. These numbers of realizations are the values that can be handled in a reasonable computational time. Tables 6.3 and 6.4 report the average values of percentage VSS_r with respect to the mRP value for instances with 3 and 5 jobs, respectively, computed as

$$VSS_r = \frac{EEV_r - mRP}{mRP} \times 100. \tag{6.53}$$

As it can be observed, the VSS average values increase over stages. The average VSS per stage associated with the small range is less than those of the large interval for the three considered distributions. Moreover, comparing the results for the three distributions shows that the uniform distribution has a higher value than average VSS for both small and large ranges concerning the two other distributions.

6.5.4 Results for the DA-based solution approach

This section summarizes and discusses the results obtained from our experiments on the instance sets generated as discussed in Section 6.5.1. The results contain two distinct parts associated with the small and large-scale instances. The first part aims to assess the accuracy of the shortest path model derived from the deterministic approximation approach compared to the two-stage stochastic model for large-scale problems. In contrast, in the second part, the multi-stage stochastic

Table 6.3: Average values of percentage VSS_r with respect to the mRP value for instances with $|I| = 3$ jobs and $|F| = 2$ configurations.

Distributions	**Range**	VSS_1	VSS_2	VSS_3
Uniform	small	7.51	12.22	13.76
Uniform	large	11.90	39.43	43.39
Normal	small	1.34	2.24	2.70
Normal	large	5.26	8.62	11.71
Gumbel	small	0	1.49	4.19
Gumbel	large	4.46	8.15	10.18

Table 6.4: Average values of percentage VSS_r with respect to the mRP value for instances with $|I| = 5$ jobs and $|F| = 2$ configurations.

Distributions	**Range**	VSS_1	VSS_2	VSS_3	VSS_4	VSS_5
Uniform	small	4.77	7.56	10.14	11.21	11.26
Uniform	large	11.88	34.95	43.93	53.34	63.85
Normal	small	1.37	2.32	2.39	3.34	3.34
Normal	large	4.06	10.16	11.64	14.16	14.20
Gumbel	small	0.95	2.54	3.67	3.67	3.69
Gumbel	large	6.50	14.43	17.79	20.90	21.94

programming formulation is used in dealing with small-scale instances. The performance, in terms of percentage gap, is evaluated through the calculation of the Relative Percentage Error (RPE) as follows

$$RPE = \frac{T_0 - opt_{SP}}{opt_{SP}} \times 100, \tag{6.54}$$

where T_0 and opt_{SP} represent the optimum makespan of the shortest path model (see Eq. (6.41)) and the optimum of either two-stage or multi-stage stochastic model, respectively.

Tables 6.5–6.7 represent the RPE associated with the large-scale instances generated using the Uniform, Normal, and Gumbel distributions, respectively. Each entry of these tables reports statistics of the RPE over 10 randomly generated instances, given a specific combination of the size of the instance and the range of random variables. In particular, the tables show the average, the best, the worst RPE, and its standard deviation (columns RPE_{avg}, RPE_{best}, RPE_{worst}, and RPE_{σ}, respectively).

The first thing that should be noticed is that, as expected, the average RPE associated with the larger range of observations is bigger than those of the smaller

range across all instances sizes and the three distributions. It means the approximation model behaves better with smaller dispersion and magnitude of the realizations. The best, worst, and standard deviation RPE also follows the same described behavior with few discontinuities.

Comparing the results of the three distributions highlights better performance, as expected, of the Normal and Gumbel distributions concerning the Uniform distribution in terms of the global average, best, worst, and standard deviation RPE.

Taking into account different sizes of instances, it can be seen that the average RPEs become larger as the scale of instances increases for the three distributions. It means the quality of the approximation approach is also affected by the scale of instances. The best, worst, and standard deviation RPE also follow a similar behavior with little discontinuities.

Table 6.5: RPE of the makespan between the deterministic approximation and two-stage stochastic model for the Uniform distribution.

Instance			$RPE(\%)$			
$\|I\|$	$\|F\|$	Range	RPE_{avg}	RPE_{best}	RPE_{worst}	RPE_{σ}
10	3	small	0.70	0.05	2.00	0.69
10	3	large	1.19	0.27	3.90	1.09
		Average:	0.94	0.16	2.95	0.89
20	4	small	1.48	0.50	2.88	0.67
20	4	large	2.04	0.78	3.13	0.77
		Average:	1.76	0.64	3.00	0.72
30	4	small	1.58	0.26	3.55	1.23
30	4	large	2.11	0.23	5.35	1.78
		Average:	1.84	0.24	4.45	1.50
40	5	small	1.68	0.30	3.57	0.99
40	5	large	2.51	0.41	6.11	1.60
		Average:	2.09	0.35	4.84	1.29
Global Average:			**1.64**	**0.34**	**3.81**	**1.10**

We now want to evaluate the quality of the shortest path model concerning the multi-stage model in dealing with the small-scale instances. Table 6.8 reports the average, best, worst, and standard deviation RPE of the small-scale instance for the three considered distributions. Although we use a small number of realizations in dealing with these small-sized instances, the results show promising performance of the shortest path model.

Similar to the results obtained from the two-stage model in dealing with the

Table 6.6: RPE of the makespan between the deterministic approximation and two-stage stochastic model for the Normal distribution.

Instance			$RPE(\%)$			
$\|I\|$	$\|F\|$	Range	RPE_{avg}	RPE_{best}	RPE_{worst}	RPE_{σ}
10	3	small	0.42	0.04	0.88	0.27
10	3	large	0.79	0.06	2.25	0.75
		Average:	0.60	0.05	1.56	0.51
20	4	small	0.53	0.13	1.14	0.37
20	4	large	1.64	0.82	2.11	0.52
		Average:	1.08	0.47	1.62	0.44
30	4	small	0.79	0.40	1.04	0.23
30	4	large	1.67	0.34	2.81	1.06
		Average:	1.23	0.37	1.92	0.64
40	5	small	0.90	0.36	1.52	0.41
40	5	large	1.93	0.18	3.97	1.09
		Average:	1.41	0.27	2.74	0.75
Global Average:			**1.08**	**0.29**	**1.96**	**0.58**

large-scale instances, the average RPE associated with the small range of realizations is less than those of the larger support for the considered distributions. As it can be seen, the average RPE values of the Uniform distribution are more than those of the Normal and Gumbel distributions. It means, as expected, these two distributions perform better in comparison with the Uniform distribution. The best, worst, and standard deviation results do not show an exactly similar trend in some cases. These fluctuations are due to the very small number of realizations applied in the multi-stage stochastic model. As mentioned before, by computing the appropriate value of the parameter β using the considered observations, the performance of the deterministic approximation model is maintained. Here, due to the complexity of the multi-stage stochastic model, a few realizations are applied, which may not be enough to derive the appropriate value of β.

Considering the various sizes shows that the average RPE grows as the scale of instances increases for the three distributions. This behavior is also similar to what we noticed in the large-sized instances solved by the two-stage model.

Finally, despite the quality obtained by the deterministic approximation approach in terms of accuracy, we also want to point out its efficiency. The computational times for obtaining the minimum makespan of the shortest path model (t_{SPM}) and the stochastic problem (t_{SP}) are reported in Table 6.9. As we pointed out before, the small-scale instances are dealt with in the multi-stage recourse model with

Table 6.7: RPE of the makespan between the deterministic approximation and two-stage stochastic model for the Gumbel distribution.

Instance			$RPE(\%)$			
$\|I\|$	$\|F\|$	Range	RPE_{avg}	RPE_{best}	RPE_{worst}	RPE_{σ}
10	3	small	0.38	0.07	0.97	0.33
10	3	large	0.64	0.04	1.51	0.43
		Average:	0.51	0.05	1.24	0.38
20	4	small	0.42	0.12	1.18	0.42
20	4	large	1.26	0.13	2.17	0.79
		Average:	0.84	0.12	1.67	0.60
30	4	small	0.75	0.02	1.39	0.52
30	4	large	1.42	0.06	2.84	0.88
		Average:	1.08	0.04	2.11	0.70
40	5	small	0.89	0.20	1.49	0.41
40	5	large	1.75	0.32	3.53	0.91
		Average:	1.32	0.26	2.51	0.66
Global Average:			**0.93**	**0.11**	**1.88**	**0.58**

a few realizations, while the larger ones are solved using the two-stage problem with a suitable number of observations. Since we have noticed that the computational times were not affected by the distribution and the range of observations, we report average computational times for the various sizes of the instances. First, note that the CPU time increases as the size of the problem increases for both shortest path formulation and stochastic models. Comparing these two approaches, it is clear that the shortest path model derived from the deterministic approximation approach can solve all instances in far less time than the stochastic models. Few seconds are needed for even the largest size (40 jobs and 5 configurations), while it takes 42700 seconds to deal with this scale using the stochastic model.

6.6 Summary

This chapter has addressed the stochastic single machine scheduling problem where learning effects on processing time, sequence-dependent setup times, and machine configuration selection are considered simultaneously. Also, random variables are assumed to represent uncertainty associated with job processing time and machine setup times. The problem aims to find the sequence of jobs and choose a configuration to process each job, which minimizes the makespan under uncertainty. First, we formulate the proposed problem as two-stage and multi-stage

Table 6.8: RPE of the makespan between the deterministic approximation and multi-stage stochastic model in dealing with small-scale instances for the Uniform, Normal, and Gumbel distributions.

Instance				$RPE(\%)$			
Distribution	$\|I\|$	$\|F\|$	Range	RPE_{avg}	RPE_{best}	RPE_{worst}	RPE_{σ}
Uniform	3	2	small	2.12	0.49	4.76	1.25
	3	2	large	3.61	0.06	6.67	1.79
			Average:	2.86	0.27	5.71	1.52
Uniform	5	2	small	2.20	0.26	4.85	1.66
	5	2	large	4.13	0.28	6.76	2.47
			Average:	3.16	0.27	5.80	2.06
Normal	3	2	small	1.75	0.26	3.72	1.17
	3	2	large	2.69	0.38	5.04	1.34
			Average:	2.22	0.32	4.38	1.25
Normal	5	2	small	1.83	0.09	3.86	1.21
	5	2	large	2.81	0.30	5.63	2.03
			Average:	2.32	0.19	4.74	1.62
Gumbel	3	2	small	1.60	0.27	3.33	1.03
	3	2	large	2.67	0.16	4.59	1.39
			Average:	2.13	0.21	3.96	1.21
Gumbel	5	2	small	1.78	0.11	4.21	1.34
	5	2	large	2.85	0.42	4.94	1.81
			Average:	2.31	0.26	4.57	1.57
	Global		**Average:**	**2.50**	**0.25**	**4.86**	**1.53**

Table 6.9: Computational time in seconds for calculating the minimum makespan using the deterministic approximation and stochastic models.

$\|I\|$	$\|F\|$	t_{SPM}	t_{SP}
3	2	1	345
5	2	1	378
10	3	2	46
20	4	5	630
30	4	16	11130
40	5	35	42700

stochastic models, which are computationally demanding. In addition, a deterministic approximation formulation is developed using the multi-stage dynamic random

decision process, which has been proposed recently by [202], where the knowledge of the probability distribution of the random variables is not needed. Using this approach, the problem is formulated as a mixed-integer non-linear programming model. Then, by defining a new measure of accessibility, the model is converted to a shortest path problem on a multi-stage network solvable in a few seconds, even for large-sized instances. Finally, the extensive computational experiments showed particularly good performance of the deterministic approximation model concerning the stochastic models in terms of accuracy of solutions and computational time.

Chapter 7

Online single-machine scheduling problem

7.1 Introduction

Scheduling of operations is one of the planning functions in the manufacturing companies (see [31]). Due to disruptions or arrival of new information, the incumbent schedule can become suboptimal or even infeasible thus motivating the need for rescheduling. Online scheduling is a generalization of ongoing rescheduling process in which decisions are taken and revised in real-time during the course of production process (see [87]). This is in contrast to static case where all the specifications and requirements are fully and deterministically known in advance, before any execution begins.

In this research, we will consider *online scheduling*, mainly fostered by our experience on an industrial project (Plastic and Rubber 4.0[1]) in which frequent occurrences of unexpected events call for more dynamic and flexible scheduling (see [116]).

In particular, we focus on online scheduling of a set $\mathcal{J}$ of jobs on a single machine, where preemption is allowed. The jobs are released over time, and as soon as a new job arrives, it is added to the end of a waiting queue. For each job $j \in \mathcal{J}$, let d_j be its due date and c_j its completion time. A job is early if its completion time is shorter than its due date. On the contrary, a job is tardy if its completion time is larger than its due date. When the completion time is equal to the due date, the job is on time. The goal of the problem is to arrange the queue's jobs to minimize two different objective functions: total tardiness ($\Gamma 1$) and the total earliness and tardiness ($\Gamma 2$) of the jobs. The two objectives are calculated as

[1]Plastic&Rubber 4.0. Piattaforma Tecnologica per la Fabbrica Intelligente (Technological Platform for Smart Factory), URL: https://www.openplast.it/en/homepage-en/

- $\Gamma 1 = \sum_{j \in \mathcal{J}} T_j$,
- $\Gamma 2 = \sum_{j \in \mathcal{J}} (E_j + T_j)$,

where T_j and E_j represent the tardiness and the earliness, respectively, and are computed as $T_j := \max\{0, c_j - d_j\}$ and $E_j := \max\{0, d_j - c_j\}$. The two objective functions are the most widely used ones in scheduling, focusing on meeting jobs due dates. In particular, the minimization of the second objective characterizes the Just-In-Time principle in production.

The motivation of the research comes from plastic and rubber manufacturing, transforming raw material into a final product goes through one or two machines. On the other hand, even those manufacturing require multiple-machine scheduling problems. Each machine represents a chain's primary block. Thus improper usage of a machine can slow down the whole production process.

The *dispatching rules* are the easiest approach to deal with scheduling in a dynamic context. These rules prioritize jobs waiting for being processed and then select the job with a greedy evaluation whenever a machine gets free. While most dispatching rules schedule on a local view basis, other smarter approaches can provide better results in the long run. For instance, Reinforcement Learning (RL) is a continuing and goal-directed learning paradigm, and it represents a promising approach to deal with online scheduling. The potential of RL on online scheduling has been revealed in several works (see, e.g., [74], [185], [235]). However, while most works compare a single RL algorithm with commonly-used dispatching rules, they do not compare different RL algorithms.

In this research, we investigate the applicability of four RL algorithms (namely *Q-learning*, *Sarsa*, *Watkins's Q(λ)*, and *Sarsa(λ)*) on online single-machine scheduling in comparison with a random assignment (*Random*) which simply selects a job randomly and the most popular dispatching rules, namely the *earliest due date* (*EDD*) rule. Furthermore, the algorithms are tested under different operating conditions (e.g., the frequency of job arrivals).

Therefore, we contribute the literature on two different aspects: getting insights on the compared methods and giving practitioners suggestions on selecting the best method against the specific situation.

Finally, we also propose some preliminary results obtained by the use of *Deep Q Network* (*DQN*), which utilizes the power of neural networks to approximate the value function (see [137] for a review about DQN).

The current chapter is structured as follows. Section 7.2 is dedicated to a general overview of RL techniques, while Section 7.3 introduces and reviews some previous works using RL approaches on scheduling problems. Section 7.4 describes the algorithmic framework for the online single-machine problem. Section 7.5 defines the simulation procedure, and the simulation results from three different types of

experiments (Section 7.6). Finally, in Section 7.7, the chapter concludes with a summary of the findings and some future lines.

7.2 Reinforcement Learning

RL is a subfield of Machine Learning which involves learning the optimal behavior in an environment to obtain maximum reward. This optimal behavior is learned through interactions with the environment and observations of how it responds. It comes from three main research branches: the first relates to learning by trial-and-error, the second relates to optimal control problems, and the last links to temporal-difference methods (see [198]). The three approaches converged together in the late eighties to produce the modern RL.

In particular, RL approaches are concerned with how a goal-directed decision-maker called *agent* to interact with a set of *states* called *environment* by means of a set of possible *actions*. A *reward* is given to the agent in each specific state. In this research, we consider a discrete time system, i.e. defined over a finite set $\mathcal{T}$ of time steps with its cardinality being called *time horizon*. As shown in Figure 7.1, at each time step $t \in \mathcal{T}$, an agent in state S_t takes action A_t, then, the environment reacts by changing into state S_{t+1} and by rewarding the agent of R_{t+1}. The interaction starts from an initial state, and it continues until the end of the time horizon. Such a sequence of actions is named an *episode*. In the following, $\mathcal{E}$ will represent the set of episodes.

Each *state* of the system is associated with a *value function* that estimates the expected future reward achievable from that state. Each state-action pair (S_t, A_t) is associated with a so-called *Q-function* $Q(S_t, A_t)$ that measures the future reward achievable by implementing action A_t in state S_t. The agent's goal is to find the best *policy*, which is a function mapping the set of states to the set of actions, maximizing the cumulative *reward*. If exact knowledge of the Q-function is available, the best policy for each state is defined by $\max_a Q(S_t, a)$.

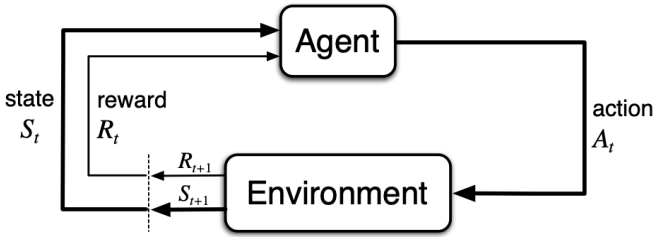


Figure 7.1: The agent-environment interaction in RL [198].

Almost all reinforcement learning algorithms are based on estimating value functions that estimate how good it is for the agent to be in a given state and discover the optimal policies. To do so, three main classes of RL techniques exist: Monte

Carlo (MC)-based, Dynamic Programming (DP)-based methods, and temporal-difference (TD)-based methods. Unlike DP-based methods, which require complete knowledge of all the possible transitions, MC-based techniques only require some experience and the possibility to sample from the environment randomly. TD-based methods are a sort of combination of MC-based and DP-based ones: they sample from the environment like in MC-based methods and perform updates based on current estimates like DP-based ones. Moreover, TD-based techniques are also appreciated for being flexible, easy to implement, and computationally fast. For these reasons, in this research, we will consider only RL algorithms belonging to the TD-based methods. Even if several TD-based RL algorithms have been introduced in the literature, the most used are *Sarsa* (an acronym for State-Action-Reward-State-Action), *Q-learning* and their variations, e.g. the *Watkins's Q(λ)* method and the *Sarsa(λ)* (see [218]).

7.3 Literature Review

Since the first research on scheduling problem was performed in the mid-1950s, many articles have been published in the literature, considering different problem variants and solution approaches.

The manufacturing industries sometimes include a machine bottleneck, which affects, in some cases, all the jobs. Studies on single machine scheduling problems have been gaining importance for a long time since this bottleneck's management is crucial. The excellent surveys by Pinedo [160], Adamu and Adewumi [1], and the work proposed by Leksakul and Techanitisawad [111] have detailed the literature on the theory and applications about this problem in the past several decades.

In the manufacturing environment, various objectives can be considered to use the resources and provide good customer service efficiently. Scheduling against due dates has received considerable attention to meet principles like Lean Management, Just-in-Time, Simultaneous Engineering, etc. For example, the Just-in-Time principle states that jobs are expected to be on time since both late and early processing may negatively influence the manufacturing costs. While late processing does not meet customer expectations, early processing increases inventory costs and causes possible wastes since some products have a limited lifetime. One of the pioneers addressing minimizing the sum of earliness and tardiness (also referred to as the sum of deviations from a common due date) was [99]. Ying [230] addressed a single-machine problem against common due dates concerning earliness and tardiness penalties. He proposed a recovering beam search algorithm to solve this problem. Behnamian et al. [22] considered the problem of parallel machine scheduling to minimize both makespan and total earliness and tardiness. Fernandez-Viagas et al. [70] studied the problem of scheduling jobs in a permutation flow shop to minimize the sum of total tardiness and earliness. They developed and compared four heuristics to deal with the problem. More recently, the two-machine permutation

flow shop scheduling problem to minimize total earliness and tardiness has been addressed by two branch-and-bound algorithms utilizing lower bounds and dominance conditions [182].

Total tardiness minimization is another common criterion in the scheduling literature where only the tardiness penalties are considered. Koulamas [104] surveyed theoretical developments, exact and approximation algorithms for the single-machine scheduling problem with the aim of total tardiness minimization. In [91], single machine scheduling with family setup and resource constraints to minimize total tardiness minimization was addressed. A mathematical formulation and a heuristic solution approach were presented. Recently, Silva et al. [187] studied the single machine scheduling problem that minimizes the total tardiness. They presented two algorithms to deal with the situation in which the processing time is uncertain.

As for the scheduling modes, research on online scheduling is one of the popular streams. Since this problem has been an active field for several decades, an in-depth analysis of the literature review is beyond the present paper's scope. Thus, in this section, we recall some of the most traditional approaches to online scheduling, and we review the main applications of RL to this problem.

Differently from tailored algorithms (heuristic and exact methods), which might require effort in implementation and calibration over a broad set of parameters, dispatching rules are widely adopted for online scheduling for their simplicity (see, e.g., [97]). For instance, the *earliest due date* (*EDD*) dispatching rule is one of the most commonly used ones in practical applications [199]. *EDD* schedules first the job with the earliest due date. Again, in [86], the authors propose a deterministic greedy algorithm known as *list scheduling* (*LS*), which assigns each job to the machine with the smallest load. For more details, we refer the reader to the work [151] that classified over one hundred dispatching rules. In [46], the authors designed a deterministic algorithm and a randomized one for online machine sequencing problems using Linear Programming techniques. At the same time, in [123], the authors proposed an algorithm to make jobs artificially available to the online scheduler by delaying the release time of jobs.

In online scheduling, a decision-maker is regularly scheduling jobs over time, attempting to reach the overall best performance. Therefore, it is reasonable that RL represents one of the possible techniques to exploit such a setting.

In [74], the authors interpreted job-shop scheduling problems as sequential decision processes. They try to improve the job dispatching decisions of the agent by employing an RL algorithm. Experimental results on numerous benchmark instances showed the competitiveness of the RL algorithm. More recently, in [235], the authors modeled the scheduling problem as a Markov Decision Process and solved it through a simulation-based value iteration and a simulation-based *Q-learning*.

Their results clearly showed that such RL algorithms could achieve better performance concerning several dispatching heuristics, disclosing RL application's potential in the field. In the context of an online single-machine environment, in [222], the authors compared the performance of *neural fitted Q-learning* techniques using combinations of different states, actions, and rewards. They proved that taking only the necessary inputs of states and actions is more efficient.

While all the discussed works revealed RL's competitiveness on scheduling problems, a further comparison of the performance among various RL algorithms is still missing in the scheduling literature. With the knowledge of the available studies showing RL's potential and the demand from the industrial application, we are motivated to compare different RL approaches' performance on online scheduling for getting more insights. In particular, we carry out experimental studies on four of the most commonly used model-free RL algorithms, namely *Q-learning*, *Sarsa*, *Watkins's Q(λ)*, and *Sarsa(λ)*. Our comparison methodology is inspired by [222], in which the best configuration for minimizing maximal lateness is pursued. In our work, instead, we propose two different objective functions to minimize: the total tardiness and the total earliness and tardiness. Moreover, another significant difference with their work lies in the way we evaluate the results. While they used the result from one run, our results come from 50 runs with different random seeds, and two different time step sizes are tested (the interaction between agent and environment is checked in each step). We further test a neural network-based RL technique showing that it is unnecessary to use such a combination when the state space is limited.

7.4 Reinforcement Learning Algorithms for Online Scheduling

In this section, we describe the algorithmic framework used to deal with our online single-machine scheduling problem. Our problem setting are defined as

- *state*: a state is associated with each possible length of the jobs in the waiting queue;
- *action*: if not all the jobs are finished, the action is either to select one new job from a specific position of the waiting queue and start processing it (we recall that preemption is allowed) or to continue processing the job which has been already assigned to the machine in the previous step;
- *reward*: since RL techniques aim at maximizing rewards while our problem seeks to minimize its objective function (either the total tardiness or the total earliness and tardiness), we set the reward of a state as the opposite value of the considered measure.

When the action implies selecting a job from a certain position in the waiting queue, it is important to decide the order in which jobs are stored inside the queue. Therefore, we implemented three possible ordering of jobs that provide very different scheduling effects

- jobs are unsorted (*UNSORT*), i.e., they have the same order as the arrivals;
- jobs are sorted by increasing value of due time (*DT*);
- all unfinished jobs are sorted by increasing the value of the sum of due time and processing time (*DT+PT*).

For instance, by using *DT*, if the action is to select a job in the second position of the queue, the job with the second earliest due time will be processed.

We have decided to implement four different RL algorithms, namely *Q-learning*, *Sarsa*, *Watkins's Q(λ)*, and *Sarsa(λ)*. They are described in the following. Here are the notation used

- s: state;
- a: action;
- $\mathcal{S}$: set of non-terminal states;
- $\mathcal{A}(s)$: set of actions possible in state s;
- S_t: state at time step t;
- A_t: action at time step t;
- R_t: reward at time step t.

Q-learning

Q-learning is a technique that learns the value of an optimal policy independently of the agent's action. It is largely adopted for its simplicity in the analysis of the algorithm and for the possibility of early convergence proofs by directly approximating the optimal action-value function (see [218] and [198]). The updating rule for the estimation of the Q-function is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]. \quad (7.1)$$

The $Q(S_t, A_t)$ function estimates the quality of state-action pair. At each time step t, the reward R_{t+1} from state S_t to S_{t+1} is calculated and $Q(S_t, A_t)$ is updated accordingly. The coefficient α is the learning rate ($0 \leq \alpha \leq 1$); it determines

the extent that new information overrides the old information. Furthermore, γ is the discount factor determining the importance of future reward and finally, $\max_a Q(S_{t+1}, a)$ is the estimation of best future value.

The values of the Q-function are stored in a look-up table called *Q-table*. Figure 7.2 displays an example of Q-table storing Q-function values for states from 0 to 10 (in row) and actions from selecting *Job 1* to *Job 5* (in column). By overlooking

Q Table		Actions				
		Select Job 1	Select Job 2	Select Job 3	Select Job 4	Select Job 5
States	0	0	0	0	0	0
	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
	5	-20	-15	-34	-14	-31
	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
	10	-15	-21	-22	-16	-23

Figure 7.2: An example of Q table.

the actual policy being followed in deciding the next action, *Q-learning* simplifies the analysis of the algorithm and enabled early convergence proofs.

Sarsa

Sarsa is a technique that updates the estimated Q-function by following the experience gained from executing some policies (see [188] and [198]). The updating rule for the estimation of the Q-function is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]. \quad (7.2)$$

The structure of formula (7.2) is similar to (7.1). The only difference is that (7.2) considers the actual action implemented in the next step A_{t+1}, instead of the generic best action $\max_a Q(S_{t+1}, a)$.

As for *Q-learning*, also in *Sarsa* the values of the Q-function are stored in a *Q table*. Despite the more expensive behaviour with respect to *Q-learning*, *Sarsa* may provide better online performances in some scenarios (as shown by the *Cliff Walking* example in [198]).

Watkins's Q(λ)

Watkins's Q(λ) is a well-known variant of *Q-learning*. The main difference with respect to classical *Q-learning* is the presence of a so-called *eligibility trace*, i.e. a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. The trace marks the memory parameters associated with

the event as eligible for undergoing learning changes. A trace is initialized when a state is visited or an action is taken, and then the trace gets decayed over time according to a decaying parameter λ (with $0 \leq \lambda \leq 1$). Let us call $e_t(s,a)$ the trace for a state-action pair (s,a). Let us also define an indicator parameter $\mathbb{1}_{xy}$ that takes value 1 if and only if x and y are the same, and 0 otherwise. Then, for any (s,a) pair (for all $s \in \mathcal{S}$, $a \in \mathcal{A}$), the updating rule for the estimation of the Q-function is

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha\delta_t e_t(s,a) \tag{7.3}$$

where

$$\delta_t = R_{t+1} + \gamma \max_{a'} Q_t(S_{t+1}, a') - Q_t(S_t, A_t) \tag{7.4}$$

and

$$e_t(s,a) = \gamma\lambda e_{t-1}(s,a) + \mathbb{1}_{sS_t}\mathbb{1}_{aA_t} \tag{7.5}$$

if $Q_{t-1}(S_t, A_t) = \max_a Q_{t-1}(S_t, a)$, and $\mathbb{1}_{sS_t}\mathbb{1}_{aA_t}$ otherwise.

As the reader can notice, by plugging equation (7.4) into equation (7.3), we obtain an equation similar to (7.1) but with the additional eligibility term that increments the value of δ_t if the state and action selected by the algorithm are one of the eligibility states. In the rest of the paper we use *Q(λ)* referring to *Watkins's Q(λ)*.

Sarsa(λ)

Similarly to *Q(λ)*, the *Sarsa(λ)* algorithm represents a combination between *Sarsa* and eligibility traces to obtain a more general method that may learn more efficiently. Here, for any (s,a) pair (for all $s \in \mathcal{S}$, $a \in \mathcal{A}$), the updating rule for the estimation of the Q-function is

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha\delta_t e_t(s,a) \tag{7.6}$$

where

$$\delta_t = R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t) \tag{7.7}$$

and

$$e_t(s,a) = \gamma\lambda e_{t-1}(s,a) + \mathbb{1}_{sS_t}\mathbb{1}_{aA_t} \tag{7.8}$$

Unlike equation (7.5), there is no other condition (set the eligibility traces to 0 whenever a non-greedy action is taken) added. A more in-depth discussion about the interpretation of the formulas is given in [198].

Algorithm 11 Online scheduling simulation through RL algorithms

Require: $|\mathcal{E}|$ number of episodes; $|\mathcal{T}|$ number of time-steps;
Initialize $Q(s,a) = 0, \forall\ s \in \mathcal{S}, a \in \mathcal{A}$;
for $\eta \leftarrow 1$ to $|\mathcal{E}|$ **do**
    Initialize S
    for $t \leftarrow 1$ to $|\mathcal{T}|$ **do**
        if new jobs arrive **then**
            Update waiting list L
        end if
        if L is not empty **then**
            Take A_t in S_t, observe R_t, S_{t+1}
            Calculate A_{t+1} and update Q_t
            $S_t \leftarrow S_{t+1}$, $A_t \leftarrow A_{t+1}$
        end if
    end for
end for

7.5 Simulation Setting

In this section, an online scheduling simulation procedure is described in Algorithm 11.

We first update Q tables through a training phase then use the Q tables to select actions in the test phase.

The arrival time of job j are distributed according to an exponential distribution, i.e., $X_j \sim exp(r)$ with the rate parameter valued $r = 0.1$. It is simulated in this way: at the first time step, a random number of jobs (from 1 to 6 jobs) and an interval time (following the exponential distribution) are generated. Once a job is generated (simulating the job's arrival), it will immediately be put into the waiting queue. Then at the next time step, if the interval time is passed, new jobs will be generated and put into the waiting queue; meanwhile, a new interval time will be created. Otherwise, nothing is created. Then the same procedure repeats till reaching a final state.

In an episode, we start a new schedule by initializing state S and terminates when either reaching the maximum steps or no jobs to process. To simulate real-time scheduling, for each episode, we check the arrivals of new jobs and update the waiting queue if there are, then we choose the action A, and calculate the reward R and the next state S' accordingly. The Q-functions are updated according to the exact RL algorithms used. The same procedure is carried out in both training and test phases except that in the test, the Q-table is not initialized with empty values but obtained from the training phase.

It is worth noting that all the algorithms considered are heuristics. They focus on

finding a good solution quickly by finding a balance between the solution space's intensified and diversified explorations. Nevertheless, the direct implantation of the algorithms above does not ensure enough diversification. For this reason, it is common to use a ϵ-greedy method. Thus, with probability ϵ, exploration is chosen, which means the action is chosen uniformly at random between the available ones. Instead, with probability $1 - \epsilon$, exploitation is chosen by taking the actions with the highest values greedily. After knowing how to balance exploration and exploitation, we need to define a learning method for finding out policies leading to higher cumulative rewards.

For the settings regarding RL algorithms

- In the policy, $\epsilon = 0.1$ enabling highly greedy actions while keeping some randomness in job selections;
- In the value function, $\alpha = 0.6$, i.e., there is a bit higher tendency to explore more possibilities while a bit lower in keeping exploiting old information, whereas $\gamma = 1.0$, which means it strives for a long-term high reward;
- In the eligibility traces, λ is 0.95, a high decaying value leads to a longer-lasting trace.

Let us show how the total tardiness value evolves, for an example in which *Q-learning* is used to schedule the jobs. In Figure 7.3, the graph on the bottom shows that the reward increases and reaches the maximum and holds steady after 80 episodes. Accordingly, the objective value (the total tardiness) decreases with more noticeable fluctuations and drops more slowly after 80 episodes. While the reward keeps stable, total tardiness continues dropping to around 40000. To summarize, using total tardiness as a goal is useful, but it is still challenging to represent the trend of this objective value adequately.

7.6 Experimental Results

In this section, we propose three different experimental results. Section 7.6.1 compares the performance among random assignment (*Random*), *EDD*, and the four RL approaches implemented for both minimizing the total tardiness and the total earliness and tardiness. Section 7.6.2 investigates the possible impact of different operating conditions (i.e., frequency of jobs arrivals) on the RL approaches. Finally, Section 7.6.3 compares *Q(λ)* and *DQN*.

The algorithms have been implemented in Python 3.6. To avoid possible ambiguities, we locate the related code in a public repository[2]. All the experiments are

[2] https://github.com/Yuanyuan517/RL_OnlineScheduling.git

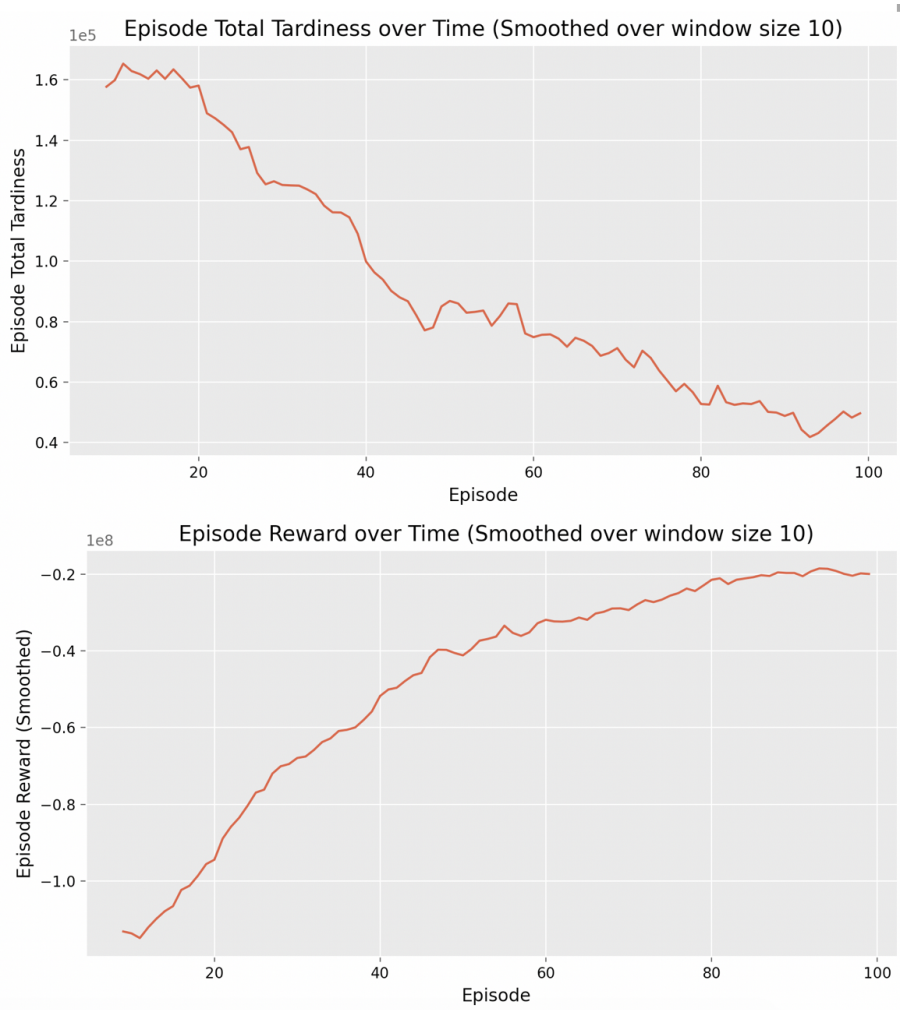


Figure 7.3: The changes to reward and the objective value (total tardiness) of 100 episodes.

carried out on an *Intel Core i5* CPU@2.3GHz machine equipped with 8GB RAM and running *MacOS* v10.15.4 operating system.

7.6.1 RL algorithms vs *Random* and *EDD*

To check if considering different time horizons leads to different results, we consider two experiments in which the time horizon $\mathcal{T}$ is set to 2500 and 5000, respectively. For each of the settings, we ran 50 tests with different random seeds. For each algorithm Θ, we call $\Gamma_{\zeta\Theta}$ the objective value achieved in simulation ζ . Furthermore, we define $\rho_{\zeta\Theta}$ to be the percentage gap between the objective value

achieved by the best algorithm and algorithm Θ during run ζ, i.e.,

$$\rho_{\zeta\Theta} = \frac{\Gamma_{\zeta\Theta}}{\min_{\zeta\Theta} \Gamma_{\zeta\Theta}}. \tag{7.9}$$

To compare the different algorithms, we consider the average value of $\rho_{\zeta\Theta}$ concerning all the runs.

The simulation results with the algorithms (under different job orders, time horizons) for the total tardiness and the total earliness and tardiness minimization are displayed in Table 7.1 and 7.2, respectively. Note that avg($\rho_{\zeta\Theta}$) and std($\rho_{\zeta\Theta}$) represent respectively the mean value and standard deviations of $\rho_{\zeta\Theta}$.

Table 7.1: Simulations of the algorithms with different settings and considering the total tardiness minimization.

Algorithm	**Jobs order**	$\lvert\mathcal{T}\rvert$=2500		$\lvert\mathcal{T}\rvert$=5000	
		avg($\rho_{\zeta\Theta}$)	std($\rho_{\zeta\Theta}$)	avg($\rho_{\zeta\Theta}$)	std($\rho_{\zeta\Theta}$)
Random	-	2.59	0.50	3.06	0.69
EDD	-	7.67	1.76	9.19	1.47
Q-learning	*UNSORT*	2.15	0.43	2.04	0.35
Q-learning	*DT*	1.45	0.28	1.29	0.20
Q-learning	*DT+PT*	1.44	0.30	1.25	0.18
Sarsa	*UNSORT*	2.55	0.53	2.47	0.39
Sarsa	*DT*	1.65	0.40	1.76	0.36
Sarsa	*DT+PT*	1.66	0.47	1.68	0.33
Sarsa(λ)	*UNSORT*	4.42	0.93	5.04	0.93
Sarsa(λ)	*DT*	7.04	1.35	7.73	1.34
Sarsa(λ)	*DT+PT*	3.08	1.03	7.70	1.33
Q(λ)	*UNSORT*	2.04	0.42	2.01	0.40
Q(λ)	*DT*	**1.11**	**0.18**	1.13	0.17
Q(λ)	*DT+PT*	1.19	0.26	**1.09**	**0.14**

The best value among all the combinations of algorithms and jobs order policies for each time horizon is highlighted in bold font.

While in [222] the authors show that *EDD* gets a better result than RL in minimizing the maximum tardiness, as shown in Table 7.1, all the implemented RL algorithms outperform *EDD* in minimizing the total tardiness. This result is exciting and probably depends on whether the learning paradigm is more tailored to optimize min-sum problems than min-max ones. Also, it can be seen that the size of running time steps influences the result on job order but does not affect the algorithm. For the case with 2500 steps, the configuration *Q(λ)* plus *DT* gets the best result, instead for 5000 steps, the configuration *Q(λ)* plus *DT+PT* outperforms the others.

Besides, we find with the sorting choice *DT+PT* that all algorithms get smaller

Table 7.2: Simulations of the algorithms with different settings and considering the total tardiness and earliness minimization.

Algorithm	**Jobs order**	$\|\mathcal{T}\|$=2500		$\|\mathcal{T}\|$=5000	
		avg($\rho_{\zeta\Theta}$)	std($\rho_{\zeta\Theta}$)	avg($\rho_{\zeta\Theta}$)	std($\rho_{\zeta\Theta}$)
Random	-	5.85	9.95	19.34	42.96
EDD	-	4.17	**1.86**	**6.24**	**2.99**
Q-learning	*UNSORT*	5.33	9.33	12.62	30.34
Q-learning	*DT*	3.95	6.71	10.20	23.10
Q-learning	*DT+PT*	3.72	6.20	9.91	22.20
Sarsa	*UNSORT*	5.72	9.62	17.45	39.43
Sarsa	*DT*	4.43	7.87	16.34	37.85
Sarsa	*DT+PT*	4.46	8.13	13.77	33.89
Sarsa(λ)	*UNSORT*	10.77	17.78	36.59	75.93
Sarsa(λ)	*DT*	13.29	23.84	49.71	111.14
Sarsa(λ)	*DT+PT*	6.04	9.87	55.77	116.36
Q(λ)	*UNSORT*	4.68	8.25	15.45	34.97
Q(λ)	*DT*	3.89	6.66	10.21	23.98
Q(λ)	*DT+PT*	**3.29**	5.62	9.23	21.36

average values except for the configuration *Q(λ)* with 2500 steps. Comparatively, a randomly sorting job leads to a much worse result.

Instead, as reported in Table 7.2, *EDD* outperforms the other algorithms in minimizing the total earliness and tardiness, in terms of both the mean and the standard deviation for the larger time horizon. Moreover, it achieves the smallest standard deviation for both time horizons. However, the configuration using *Q(λ)* and *DT+PT* gets the smallest mean for the case with 2500 time steps. Taking into account the three job's ordering, it can be noticed that all the algorithms in combination with *UNSORT* have the worst results in terms of both the mean and the standard deviation, except for the algorithm *Sarsa(λ)* (which instead performs very poorly with the sorting choice *DT*).

Finally, it can be noticed that the mean and the standard deviation obtained by the algorithms in minimizing the total earliness and tardiness are larger than those achieved in Table 7.1. Unlike the total tardiness minimization's objective, the total earliness and tardiness may not be well addressed by the proposed RL algorithms. Considering the measure of jobs, earliness can negatively affect the effectiveness of RL algorithms. A possible reason can be found in the test environment settings. In the experiments, the due date is calculated by first taking a random value, namely the processing time of the job, from an exponential distribution $X \sim Exp(\iota)$ where

$$\iota = \frac{1}{7 \times \max_{j \in \mathcal{J}}\{processingTimeJob_j\}},$$

and adding that value to the current simulation time. Reminding that the tardiness of a job j is defined as $T_j := \max\{0, c_j - d_j\}$ where $c_j = startTime_j + processingTimeJob_j$, then the more the jobs accumulated as time running, the bigger the difference between the start time and due date for a job. Hence, more delays will occur, which might cause the simulation results in favor of tardiness calculation.

7.6.2 *Q(λ)* performance against different job arrival rates

We carried out another test against different frequencies of job arrivals (controlled by the rate parameter r) by considering the two best RL algorithm combinations resulted from the previous tests, i.e., *Q(λ)* plus *DT* and *Q(λ)* plus *DT+PT*. To understand whether the value of r affects the performance, we experimented with 2 more values, i.e. $r = \{0.05, 0.2\}$ in addition to the previous one $r = 0.1$. Tables 7.3 and 7.4 show the results of this test in the case of minimization of total tardiness and total earliness and tardiness, respectively. Note that the results have been normalized by following Eq. (7.9) with 50 tests and $|\mathcal{T}| = 2500$ for each test.

Table 7.3: Experiments on the rate parameter with best settings from *Q(λ)* concerning the total tardiness minimization.

Jobs order	r	avg($\rho_{\zeta\Theta}$)	std($\rho_{\zeta\Theta}$)
DT	0.05	**1.14**	**0.18**
DT+PT	0.05	1.17	0.55
DT	0.10	**1.10**	**0.17**
DT+PT	0.10	1.17	0.26
DT	0.20	1.17	0.28
DT+PT	0.20	**1.12**	**0.24**

Table 7.4: Experiments on the rate parameter with best settings from *Q(λ)* concerning the total earliness and tardiness minimization.

Jobs order	r	avg($\rho_{\zeta\Theta}$)	std($\rho_{\zeta\Theta}$)
DT	0.05	**1.74**	**0.83**
DT+PT	0.05	1.94	0.90
DT	0.10	3.89	6.66
DT+PT	0.10	**3.29**	**5.62**
DT	0.20	5.97	6.66
DT+PT	0.20	**5.75**	**6.37**

As shown in the Table 7.3, with small values of r (e.g., 0.05, 0.10), i.e., when

jobs arrive much less frequently than the last one, the version with jobs ordered by DT performs better. When jobs arrive much more frequently, the version sorted by DT+PT wins. Hence, a careful selection of algorithms and settings according to the operating conditions matters.

Table 7.4 shows results on comparing the total earliness and tardiness with the same settings as the ones of Table 7.3. However, even with a different objective, the results for $r = 0.05$ and $r = 0.20$ are similar: the version using DT (for the former) and $DT + PT$ (for the latter) perform better. The difference lies on $r = 0.10$, which gets better performance with $DT + PT$ instead of DT in Table 7.3. Thus, a combination of factors (settings, operating conditions, and objective) is clearly necessary to be considered when selecting the RL algorithm.

7.6.3 Comparison between *Q(λ)* and *DQN*

Finally, in this section, we compare a four-layer DQN and $Q(\lambda)$ plus DT+PT, which is the best performing RL algorithm. Figure 7.4 shows such a comparison in the total tardiness minimization, while Figure 7.5 is dedicated to the case minimizing the total earliness and tardiness. We run 50 tests and $|\mathcal{T}| = 5000$ in each test. The horizontal axis represents the total tardiness and the vertical axis shows the probability the objective value falls in. Note that the brown area indicates the overlapping between $Q(\lambda)$ and DQN.

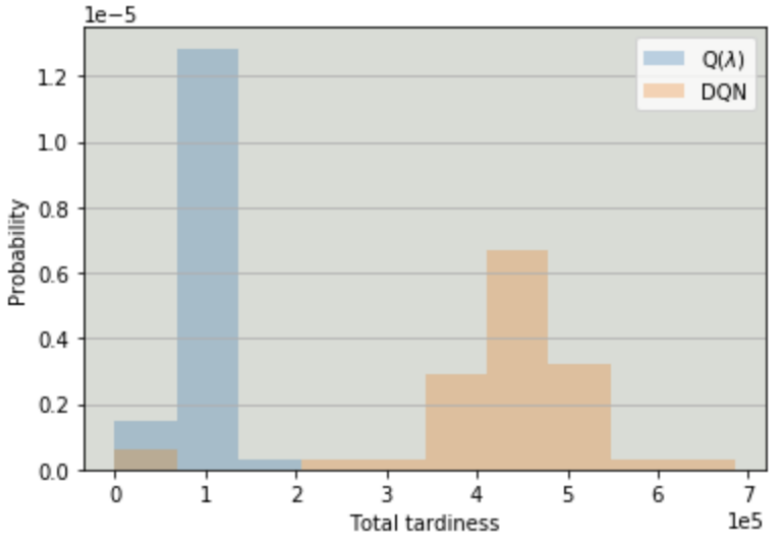


Figure 7.4: Comparison between $Q(\lambda)$ and DQN on the total tardiness of 50 runs with different seeds representing different schedules.

From Figure 7.4, we can see $Q(\lambda)$ has much higher probability with smaller objective value, which indicates $Q(\lambda)$ outperforms DQN. Taking into account the

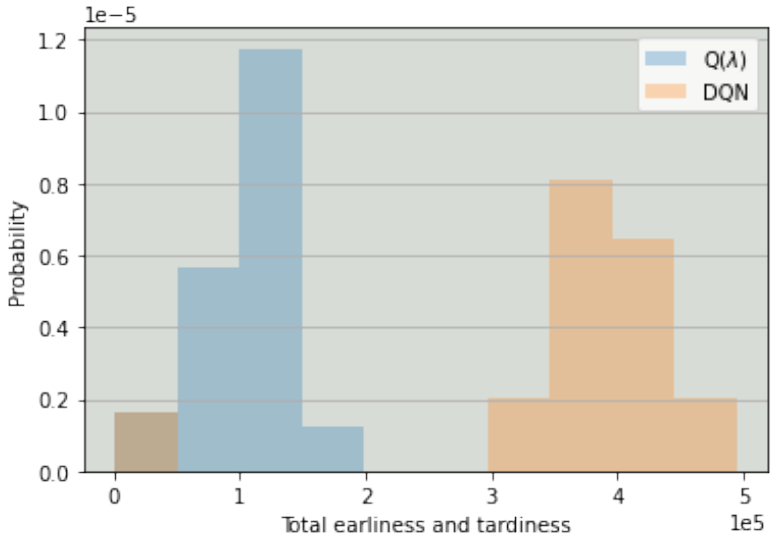


Figure 7.5: Comparison between *Q(λ)* and *DQN* on the total earliness and tardiness of 50 runs with different seeds representing different schedules.

time spent in training *DQN* is almost 10 times of *Q(λ)*, *Q(λ)* is a better option, especially for guaranteeing a flexible and adaptive scheduling in realtime.

The results in Figure 7.5 are very similar to the previous ones. Compared to *Q(λ)*, *DQN* has a much higher probability with a bigger objective, which stands for its poor performance.

7.7 Summary

In this paper, we compared four RL methods, namely *Q-learning*, *Sarsa*, *Watkins's Q(λ)*, and *Sarsa(λ)*, with *EDD* and random assignment on an online single-machine scheduling problem with two different objectives, as the total tardiness and the total earliness and tardiness minimization. The experiments show that

- better scheduling performance in minimizing the total tardiness is achieved by the RL method *Watkins's Q(λ)*, especially when the action concerns the selection of jobs sorted by due date for the smaller time horizon ($|\mathcal{T}| = 2500$) and the selection of jobs sorted by due date and processing time for bigger time horizon ($|\mathcal{T}| = 5000$);
- considering the measure of earliness may negatively affect the performance of RL algorithms. In minimizing the total earliness and tardiness, *Watkins's Q(λ)* with the sorting choice *DT+PT* performs better for the small-time horizon in terms of mean values. In contrast, *EDD* can get better results for the large-time horizon;

- when considering different frequencies of jobs arrival, the combination of $Q(\lambda)$ and job orders have different performances in various operating conditions with different objectives;
- slight differences in algorithms and objectives can profoundly change the results.

Besides, with limited input, using DQN is too costly for extended running time and energy spent adjusting parameters to guarantee a good result. In addition to the numerical results explicitly presented in the paper, according to our previous experience, RL algorithms also do not perform well on a single job-related objective (e.g., maximum tardiness [222]). These indicate careful analysis should be done from different viewpoints (running time, operating conditions, average results from multiple experiments) for making a wiser selection of algorithms.

Chapter 8

Conclusions and future research

In the following, for each problem addressed in the thesis (two deterministic, two stochastic, and one dynamic COPs involving scheduling decisions), we summarize the main conclusions and sketch some possible future lines of research.

8.1 Multi-trip single vehicle routing problem with AND-type precedence constraints

The first proposed COP under the deterministic environment is the multi-trip single-vehicle routing problem. The nodes associated with customers/target locations are related to each other through AND-type precedence constraints. Our motivation comes from realistic settings, such as package delivery or picker routing problems. Some nodes have priorities to be visited after a set of other ones within and among the routes. For instance, in picker routing problems where a picker walks or drives through the warehouse to collect the requested items, AND-type PCs need to be considered due to physical features like fragility restrictions, stackability, shape, size, and weight. For example, to prevent damage to light items, pickers cannot put heavy items on top of light items. Moreover, preferred loading or unloading sequences (to avoid extra effort on sorting and packing the collected items at the end of the retrieving process) can be represented as PCs that specify which items should be collected before other ones.

Despite the AND-type PC applications in real-life routing problems, none of the studies in the literature of logistics, even picker routing problems, focuses on these relations. Closely related research to the VRP with PCs is the Dial-A-Ride or pickup and delivery problem where the pair-wise relations (known as conventional PC) are inherently represented between pickup and delivery (drop-off) points within

a route, i.e., for any backhaul node j, there is a particular inhaul node i where the PC ($i < j$) must be met within a route.

In this thesis, we develop and experimentally compare three mathematical formulations to address the proposed problem. The computational results validate the significant superiority of the developed two-index model in terms of both computational time and problem size in comparison with the two other ones on an extended set of small instances proposed by Martinez-Salazar et al. [133].

Then, the problem is handled by developing a solution approach based on the logic-based Benders decomposition (LBBD) algorithm. The proposed approach decomposes the original problem into an assignment master problem. The nodes are allocated to some required trips and independent sequencing subproblems with the particular structure of the traveling salesman problem considering AND-type PCs.

Moreover, a new optimality cut is provided to obtain faster convergence, and its validity is proven. The performance of the optimality cut is experimentally investigated by comparing that with a recently proposed cut in the literature. Additionally, we present a relaxed version of LBBD by defining a limit for optimality gap and CPU time in deriving master-problem solutions. In such a way, the algorithm's efficiency improves. It allows the algorithm to find a feasible solution to the original problem in less CPU time and even larger instances. The performance of proposed LBBD algorithms is evaluated and compared together through extensive computational experiments. The results show that the two exact LBBDs can solve most instances, while the relaxed version of LBBD can heuristically solve all the generated instances in a shorter computational time.

Since this research is the first attempt to propose AND-type PCs in the routing area, various future topics can be explored, including different vehicle routing problems and even real-life applications. Designing and developing other exact solution approaches, well-known heuristic, meta-heuristic algorithms, and enhancements in both parts of the master and subproblem of the proposed LBBD algorithm are highly recommended for future research studies.

8.2 Vehicle routing problem with AND/OR precedence constraints and time windows

The second proposed COP under the deterministic environment is an extension of the vehicle routing problems with time windows in which AND/OR precedence constraints are defined among the customers' visits. This generalization comes after considering the customers' partial orders due to their priorities or physical restrictions.

Let us consider a delivery problem in which a customer's order includes various items collected from different locations. In such a situation, the customer must be

visited in a route only after (not necessarily immediately) visiting the locations of requested goods. A similar situation can also be seen in order picker routing problems where a picker walks or drives through the warehouse to collect the requested items and put them in a roll container considering fragility restrictions, stackability, shape, size, and weight. For example, to prevent damage to light items, pickers are not allowed to put heavy items on top of light items. Such physical features and preferred loading or unloading sequences (to avoid extra effort to sort and pack the collected items at the end of the retrieving process) can be represented as AND/OR PCs.

Despite the applicability of such constraints, no available research in the literature of VRPs and even the picker routing problems consider AND/OR PCs. As mentioned before, the precedence relations have been represented as a pre-specified sequence of nodes in most picker routing problems.

To address the proposed problem, we formulate it as a MILP model capable of solving only small-sized instances by spending a lot of CPU time. We have also developed a meta-heuristic algorithm as the hybridization of Iterated Local Search (ILS) and Simulated Annealing (SA) approaches which complement the advantages of both ILS and SA in a single optimization framework. The computational experiments highlight the promising performance of the developed algorithm in terms of CPU time and solution quality. This proves that the integration between SA and ILS can balance exploration and exploitation and achieve reasonable optimization results.

Future works could be devoted to different routing problems like stochastic dynamic models considering AND/OR precedence constraints. Considering the solution approaches, exact methods and other algorithms can be developed to be compared with our proposed meta-heuristic algorithm in dealing with larger instances.

8.3 Optimal paths in multi-stage stochastic decision networks

As a first application of the multi-stage Deterministic Approximation (DA) approach, we propose a problem that aims at finding the optimal path value in a multi-stage stochastic decision-making network. In this problem, decisions are made under uncertainty, and the random term oscillations of the stochastic parameters follow an unknown probability distribution. The optimal path is seen as a sequential decision-making process over stages, where the uncertain utility of nodes at each stage is affected by the subsequent decisions. In such a way, the decisions are nested, and the decision process cannot be decomposed into distinct stages.

We show how, under appropriate assumptions and using some results of the extreme value theory, the probability distribution of the best alternative can still

be asymptotically approximated by a Gumbel distribution and, in turn, the total utility of the optimal path can be analytically derived.

Moreover, using a Nested Multinomial Logit model, which gives the probability to choose each node at each stage optimally, a feasible solution is heuristically derived. The solution is obtained by finding the longest path (the most probable sequence) on the graph in which arcs are characterized with their choice probabilities.

Extensive numerical tests on a significant number of randomly generated instances have shown accurate estimations concerning analogous results obtainable from solving the expected value problem. The performance of the deterministic approximation seems particularly good as the size of networks increases, making the proposed approach a valuable tool to support decision-making in stochastic multi-stage networks for large and complex applications.

Future works could consider using the given deterministic approximation in different and more specific operational management problems involving multi-stage stochastic decision processes. Finally, from a more methodological point of view, one might embed this approach into a shifting-window framework that iteratively considers a restricted horizon to mitigate the approximation errors in finding optimal paths over stages.

8.4 Stochastic single machine scheduling problem as a multi-stage dynamic random decision process

As a more specific application of the deterministic approximation approach in a multi-stage stochastic decision-making process, a single machine scheduling problem is addressed. The problem is defined as the stochastic single machine job scheduling problem where the learning effect on processing time, sequence-dependent setup times, and machine configuration selection are considered simultaneously. Also, random variables are assumed to represent uncertainty for job processing times and machine setup times. The machine can handle one job at a time and works under configurations that affect the job processing times. The problem aims at finding the sequence of jobs and configurations that minimizes the makespan under uncertainty.

We formulated the problem as two-stage and multi-stage stochastic models to compare with the deterministic approximation model, where knowing the probability distribution of the random variables is not needed. In the two-stage SP model, two variable sets, corresponding to decisions before and after revealing information, are defined. The first-stage decisions, common to all realizations, represent the jobs assignments to positions. The second-stage decisions, specific to each realization and dependent on the first-stage decisions, represent the choice of configuration to

process each job.

The two-stage vision of the problem could be too simplistic since it assumes to collapse the implementation of the optimal second-stage decisions when all the uncertainty is revealed. In practice, a multi-stage SP approach could be more suitable for this kind of strongly-layered operational problem. The decisions are taken stage by stage, along with the realizations of some random variables.

However, both two-stage and multi-stage SP models are highly computationally demanding. So, we apply the deterministic approximation approach proposed in [202] to address the problem and make possible the resolution of large-scale instances. Using the approximation approach, the problem was first reformulated as a non-linear integer programming model. Then, by defining the new measure of accessibility, it was converted to the shortest path problem on a multi-stage network, which can be solved in a few seconds, even for large-sized instances.

In conclusion, the extensive computational experiments showed outstanding performance of the deterministic approximation model vs. the stochastic models in terms of accuracy of solutions and computational time.

Future works could consider using the Deterministic Approximation in different scheduling problems involving multi-stage random decision processes. From a methodological point of view, it could be interesting to develop and assess a moving-window DA-based framework that iteratively considers a restricted horizon to provide optimal decisions over stages.

8.5 Online Single-Machine Scheduling via Reinforcement Learning

In the context of dynamic COPs, the online single machine scheduling problem has been studied. To deal with the problem, we compared four of the most used RL methods, namely *Q-learning*, *Sarsa*, *Watkins's Q(λ)*, and *Sarsa(λ)*, with *EDD* and random assignment method. Two different objective functions are considered as total tardiness and total earliness and tardiness minimization.

The result of computational experiments indicates that the slight differences in algorithms, jobs order, objective functions, and time horizon can profoundly change the results. In particular, the RL method *Watkins's Q(λ)* show better scheduling performance in minimizing the total tardiness, especially when the action concerns the selection of jobs sorted by the due date for the smaller time horizon ($|\mathcal{T}| = 2500$) and the selection of jobs sorted by the due date and processing time for larger time horizon ($|\mathcal{T}| = 5000$).

Considering the measure of earliness may negatively affect the performance of RL algorithms. In minimizing the total earliness and tardiness, *Watkins's Q(λ)* with the sorting choice *DT+PT* performs better for the small-time horizon in terms of mean values. In contrast, *EDD* can get better results for the large-time horizon.

Moreover, the computational tests against different frequencies of job arrivals show that the combination of $Q(\lambda)$ and job orders have different performances in various operating conditions with different objectives. Besides, with limited input, using DQN is too costly for extended running time and energy spent adjusting parameters to guarantee a good result.

In addition to the numerical results explicitly presented in the paper, according to some previous experience, RL algorithms also do not perform well on a single job-related objective (e.g., maximum tardiness [222]). This indicates that, in the future, a careful analysis could be done from different viewpoints (running time, operating conditions, average results from multiple experiments) for making a wiser selection of algorithms.

In multiple machines scheduling, more transitions must be considered, which need more representational state information. Thus it will be impossible to store values of all state-action pairs in a Q-table. DQN may take a leading role then.

As indicated by the work [73], unpredictable changes may happen at different places in the state-action space, and more care should be taken to avoid instabilities of DQN. One technique that can achieve this goal is the so-called *kernel function* (see [38]), which builds a future research avenue. Another possibility is creating an algorithm selection framework, as explored in work by Rice [177]. In particular, by mapping from the problem characteristics to the appropriate algorithms considered in the framework, we can achieve an automatic selection of the best one to use.

Bibliography

[1] M.O. Adamu and A.O. Adewumi. "A survey of single machine scheduling to minimize weighted number of tardy jobs". In: *Journal of Industrial and Management Optimization* 10 (2014), pp. 219–241.

[2] A. Agnetis, F. Rossi, and S. Smriglio. "Some Results on Shop Scheduling with S-Precedence Constraints among Job Tasks". In: *Algorithms* 12 (2019), pp. 1–12.

[3] A.K. Agrawala et al. "A static optimization algorithm expected flow time on uniform processors". In: *IEEE Transaction on Computing* 33 (1984), pp. 351–357.

[4] T. Akamatsu. "Cyclic flows, markov process and stochastic traffic assignment". In: *Transportation research part B* 30.5 (1996), pp. 369–386.

[5] M. Albareda-Sambola, E. Fernandez, and G. Laporte. "The dynamic multiperiod vehicle routing problem with probabilistic information". In: *Computers & Operations Research* 48 (2014), pp. 31–39.

[6] A. Allahverdi. "The Third Comprehensive Survey on Scheduling Problems with Setup Times/Costs". In: *European Journal of Operational Research* 246 (2015), pp. 345–378.

[7] A. Allahverdi, J.N.D. Gupta, and T. Aldowaisan. "A review of scheduling research involving setup considerations". In: *Omega* 27 (1999), pp. 219–239.

[8] F. Angel-Bello, L. Martinez-Salazar, and A. Alvarez. "Minimizing waiting times in a route design problem with multiple use of a single vehicle". In: *Journal of Electronic Notes in Discrete Mathematics* 41 (2013), pp. 269–276.

[9] F. Angel-Bello et al. "A single machine scheduling problem with availability constraints and sequence-dependent setup costs". In: *Applied Mathematical Modeling* 35 (2011), pp. 2041–2050.

[10] E. Angelelli, R. Mansini, and M. Vindigni. "The Stochastic and Dynamic Traveling Purchaser Problem". In: *Transportation Science* 50 (Jan. 2016), pp. 642–658.

[11] E. Angelelli et al. "Short term strategies for a dynamic multi period routing problem". In: *Transportation Research Part C* 17 (2009), pp. 106–119.

[12] C. Archetti, O. Jabali, and M.G. Speranza. "Multi-period Vehicle Routing Problem with Due dates". In: *Computers & Operations Research* 61 (2015), pp. 122–134.

[13] P. Augerat et al. *Computational results with a branch and cut code for the capacitated vehicle routing problem.* Tech. Rep. 949-M, Universit e Joseph Fourier, Grenoble, France, 1995.

[14] N. Azi et al. "An exact algorithm for a single-vehicle routing problem with time windows and multiple routes". In: *European Journal of Operational Research* 178 (2007), 755—766.

[15] A. Azzouz, M. Ennigrou, and L. Ben Said. "Scheduling problems under learning effects: classification and cartography". In: *International Journal of Production Research* 56 (2018), pp. 1642–1661.

[16] U. Bahalke, A.M. Ulmeh, and K. Shahanaghi. "Meta-heuristics to solve single machine scheduling problem with sequence-dependent setup time and deteriorating jobs". In: *International Journal of Advanced Manufacturing Technology* 50 (2010), pp. 749–759.

[17] X. Bai, M. Cao, and S. S-Ge. "Efficient Routing for Precedence-Constrained Package Delivery for Heterogeneous Vehicles". In: *Journal of IEEE Transaction on Automation* (2019), pp. 1–13.

[18] J.B. Baillon and R. Cominetti. "Markovian traffic equilibrum". In: *Mathematical programming* 111 (2008), pp. 33–56.

[19] K. R. Baker and D. Trietsch. *Principles of Sequencing and Scheduling.* Ed. by New Jersey John Wiley. 2009.

[20] L. Baringo and A.J. Conejo. "Risk-Constrained Multi-Stage Wind Power Investment". In: *IEEE transaction on power systems* 28 (2013), pp. 401–411.

[21] R. Barzanji, B. Naderi, and M.A. Begen. "Decomposition algorithms for the integrated process planning and scheduling problem". In: *Omega* 93 (2020), pp. 1–13.

[22] J. Behnamiana, S.M.T. Fatemi Ghomi, and M. Zandieh. "A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic." In: *Expert systems with applications* 36 (2009), pp. 11057–11069.

[23] R. Bellman. *Dynamic programming.* Princeton University Press, 1957.

[24] M. Ben-Akiva and S. R. Lerman. "Disaggregate Travel and Mobility Choice Models and Measures of Accessibility". In: *Behavioral Travel Modeling.* Ed. by D. Hensher and P. Stopher. Croom Helm, London, 1979.

[25] M.E. Ben-Akiva and S.R. Lerman. *Discrete choice analysis: theory and application to travel demand*. Vol. 9. MIT press, 1985.

[26] P. Beraldi et al. "A stochastic programming approach for the traveling purchaser problem". In: *IMA Journal of Management Mathematics* 28.1 (2017), pp. 41–63.

[27] J.R. Birge and F. Louveaux. *Introduction to stochastic programming*. Ed. by 2nd edition. Springer Publishing Company, 2011.

[28] D. Biskup. "Single-machine scheduling with learning considerations". In: *European Journal of Operational Research* 115 (1999), pp. 173–178.

[29] H.J. Bockenhauer, T. Momke, and M. Steinova. "Improved approximations for TSP with simple precedence constraints". In: *Journal of Discrete Algorithms* 21 (2013), pp. 32–40.

[30] D. Bredstrom and M. Ronnqvist. "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints". In: *European Journal of Opertional Research* 191 (2008), pp. 19–31.

[31] P. Brucker. *Scheduling Algorithms*. 5th. Springer Publishing Company, Incorporated, 2010.

[32] X. Cai, L. Wang, and X. Zhou. "Single-machine scheduling to stochastically minimize maximum lateness". In: *Journal of Scheduling* 10 (2007), pp. 293–301.

[33] M. Casazza, A. Ceselli, and R. Wolfler-Calvo. "A branch and price approach for the Split Pickup and Split Delivery VRP". In: *Electronic Notes in Discrete Mathematics* 69 (2018), pp. 189–196.

[34] P.M. Castro, I. Harjunkoski, and I.E. Grossmann. "Rolling-Horizon Algorithm for Scheduling under Time-Dependent Utility Pricing and Availability". In: *Computer Aided Chemical Engineering* 28 (2010), pp. 1171–1176.

[35] P. Castrogiovanni et al. "Smartphone Data Classification Technique for Detecting the Usage of Public or Private Transportation Modes". In: *IEEE Access* 8 (2020), pp. 58377–58391.

[36] D. Cattaruzz, N. Absi, and D. Feillet. "Vehicle routing problems with multiple trips". In: *Journal of Operations Research* 14 (2016), pp. 223–259.

[37] D. Cattaruzz, N. Absi, and D. Feillet. "Vehicle routing problems with multiple trips". In: *Journal of Annals of Operations Research* 271 (2018), pp. 127–159.

[38] V. Cerone, E. Fadda, and D. Regruto. "A robust optimization approach to kernel-based nonparametric error-in-variables identification in the presence of bounded noise". In: *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 831–838.

[39] T. Chabot et al. “Order picking problems under weight, fragility and category constraints”. In: *International Journal of Production Research* 55 (2017), pp. 6361–6379.

[40] C.B. Cheng and K.P. Wang. “Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm”. In: *Expert Systems with Applications* 36 (2009), pp. 7758–7763.

[41] T.C.E. Cheng, W.H. Kuo, and D.L. Yang. “Scheduling with a position-weighted learning effect based on sum-of-logarithm-processing-times and job position”. In: *Information Science* 221 (2013), pp. 490–500.

[42] T.C.E. Cheng, C.C. Wu, and W.C. Lee. “Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects”. In: *Information Sciences* 178 (2008), pp. 2476–2487.

[43] T.C.E. Cheng, W.H. Wu, and S.R. Cheng. “Two-agent scheduling with position-based deteriorating jobs and learning effects”. In: *Applied Mathematics and Computations* 217 (2011), pp. 8804–8824.

[44] A.A. Cire, E. Coban, and J.N. Hooker. “Logic-based Benders decomposition for planning and scheduling: A computational analysis”. In: *The Knowledge Engineering Review* 31 (2016), pp. 440–451.

[45] J.F. Cordeau and G. Laporte. “The dial-a-ride problem: Models and algorithms”. In: *Journal of Annaul operational Research* 153 (2007), pp. 29–46.

[46] J.R. Correa and M.R. Wagner. “LP-based online scheduling: from single to parallel machines”. In: *Mathematical Programming* 119 (2009), pp. 109–136.

[47] T.G. Crainic et al. “Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic”. In: *European Journal of Operational Research* 253.2 (2016), pp. 404–417.

[48] R.L. Daniels and P. Kouvelis. “Robust scheduling to hedge agains processing time uncertainity in single-stage production”. In: *Management Science* 41 (1995), pp. 363–376.

[49] G.B. Dantzig and J.H. Ramser. “The Truck Dispatching Problem”. In: *Management science* 6 (1959), pp. 1–140.

[50] R.S. De-Camargo, G. de Miranda, and A. Lokketangen. “A new formulation and an exact approach for the many-to-many hub location-routing problem”. In: *Applied Mathematical Modelling* 37 (2013), pp. 7465–7480.

[51] R. Dekker et al. “Improving order-picking response time at ankor’s warehouse”. In: *Interfaces* 34 (2004), pp. 303–313.

[52] E. Demeulemeester, M. Vanhoucke, and W. Herroelen. “A random network generator for the activity-on-the-node networks”. In: *Journal of Scheduling* 6 (2003), pp. 17–38.

[53] E. Demirel, N. Demirel, and H. Gokcen. "A mixed integer linear programming model to optimize reverse logistics activities of end-of-life vehicles in Turkey". In: *Journal of Cleaner Production* 112 (2016), pp. 2101–2113.

[54] Y. Deng et al. "Multi-type ant system algorithm for the time dependent vehicle routing problem with time windows". In: *Journal of Systems Engineering and Electronics* 29 (2018), pp. 625–638.

[55] R. Derriesel and L. Monch. "Variable neighbourhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times". In: *Computers and Industrial Engineering* 61 (2011), pp. 336–345.

[56] A. Dixit, A. Mishra, and A. Shukla. "Vehicle Routing Problem with Time Windows Using Meta-Heuristic Algorithms: A Survey". In: *Harmony Search and Nature Inspired Optimization Algorithms* 741 (2018), pp. 539–546.

[57] T. Domencich and D. McFadden. *Urban travel dynamics: a behavioral analysis*. North Holland, Amsterdam, 1975.

[58] Y. Dong et al. "Solution methods for vehicle-based inventory routing problems". In: *Computers and Chemical Engineering* 101 (2017), pp. 259–278.

[59] R. Dudek, M. Smith, and S. Panwalkar. "Use of a case study in sequencing/scheduling research". In: *Omega* 2 (1974), pp. 253–261.

[60] F. Ertem and F. Ozcelik Tugba Sarac. "Single machine scheduling problem with stochastic sequence-dependent setup times". In: *International Journal of Production Research* (2019), pp. 1–17.

[61] L.F. Escudero et al. "The value of the stochastic solution in multistage problems". In: *Sociedad de Estadistica e Investigacion Operativa* 15 (2007), pp. 48–64.

[62] R. Faganello Fachini and V.A. Armentano. "Logic-based Benders decomposition for the heterogeneous fixed fleet vehicle routing problem with time windows". In: *Computers and Industrial Engineering* 148 (2020), pp. 1–18.

[63] E. Fadda, G. Perboli, and G. Squillero. "Adaptive Batteries Exploiting On-Line Steady-State Evolution Strategy". In: *Applications of Evolutionary Computation. EvoApplications 2017. Lecture Notes in Computer Science*. Ed. by Giovanni Squillero and Kevin Sim. Vol. 10199. Springer, Cham., 2017, pp. 329–341.

[64] E. Fadda, P. Plebani, and M. Vitali. "Optimizing Monitorability of Multi-cloud Applications". In: *Advanced Information Systems Engineering. CAiSE 2016. Lecture Notes in Computer Science*. Ed. by S. Nurcan et al. Vol. 9694. Springer, Cham, June 2016, pp. 411–426.

[65] E. Fadda et al. "KPIs for Optimal Location of charging stations for Electric Vehicles: the Biella case-study". In: *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems*. Ed. by Maria Ganzha, Leszek Maciaszek, and Marcin Paprzycki. Vol. 18. Annals of Computer Science and Information Systems. IEEE, 2019, pp. 123–126.

[66] E. Fadda et al. "The stochastic multi-path traveling salesman problem with dependent random travel costs". In: *Transportation Science* 54 (2020), pp. 1372–1387.

[67] K. Fagerholt and M. Christiansen. "A travelling salesman problem with allocation time window and precedence constraints an application to ship scheduling". In: *Journal of International Transactions in Operational Research* 7 (2000), pp. 231–244.

[68] W.B. Fan and W. Feng. "Optimization of Vehicle Routing Problem with Time Window Cigarette Logistics Based on Hybrid Genetic Algorithm". In: *Journal of Modern Electronic Technology* 11 (2018), pp. 119–123.

[69] M. Fazel-Zarandi and C. Beck. "Using Logic-Based Benders Decomposition to Solve the Capacity- and Distance-Constrained Plant Location Problem," in: *INFORMS Journal on Computing* 24 (2012), pp. 387–398.

[70] V. Fernandez-Viagas, M. Dios, and J. M-Framinan. "Ecient constructive and composite heuristics for the Permutation Flowshop to minimise total earliness and tardiness." In: *Computers and operations research* 75 (2016), pp. 38–48.

[71] R. Fisher and V.A. Jaikumar. "A generalized assignment heuristic for vehicle routing". In: *Networks* 11 (1981), pp. 109–124.

[72] M. Fosgerau. "A link based network route choice model with unrestricted choice set". In: *Transportation Research part B* 56 (2013), pp. 70–80.

[73] V. François-Lavet, R. Fonteneau, and D. Ernst. "How to discount deep reinforcement learning: Towards new dynamic strategies". In: *arXiv preprint arXiv:1512.02011* (2015).

[74] T. Gabel and M. Riedmiller. "Adaptive reactive job-shop scheduling with reinforcement learning agents". In: *International Journal of Information Technology and Intelligent Computing* 24 (2008), pp. 14–18.

[75] J. Galambos. *The asymptotic theory of extreme order statistics*. John Wiley, New York, 1978.

[76] S. Gawiejnowicz. *Time-Dependent Scheduling*. Springer-Verlag, Berlin, 2008.

[77] S.A. Gawiejnowicz. "A note on scheduling on a single processor with speed dependent on a number of executed jobs". In: *Information Processing Letters* 57 (1996), pp. 297–300.

[78] R. Gedik et al. "Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals". In: *European Journal of Operational Research* 251 (2016), pp. 640–650.

[79] M. Gendreau, G. Ghiani, and E. Guerriero. "Time-dependent routing problems". In: *Computers & Operations Research* 64 (2015), pp. 189–197.

[80] D.W. Gillies and W.S. Liu. "Scheduling tasks with AND/OR precedence constraints". In: *SIAM Journal on Computing* 24 (1995), pp. 797–810.

[81] R. Giusti et al. "Sustainable and de-stressed international supply-chains through the SYNCHRO-NET approach". In: *Sustainability* 11 (2019), pp. 1–26.

[82] B. Glensk and R. Madlener. "Multi-period portfolio optimization of power generation assets". In: *Journal of operation research and decisions* 23 (2013), pp. 21–38.

[83] M.H. Goldwasser and R. Motwani. "Complexity measures for assembly sequences". In: *International Journal of Computational Geometry and Applications* 9 (1999), pp. 371–418.

[84] V.S. Gordon et al. "Single machine scheduling models with deterioration and learning: Handling precedence constraints via priority generation". In: *Scheduling* 11 (2008), pp. 357–370.

[85] K. Govindan, M. Fattahi, and E. Keyvanshokooh. "Supply chain network design under uncertainty: A comprehensive review and future research directions". In: *European Journal of Operational Research* 263 (2017), pp. 108–141.

[86] R.L. Graham. "Bounds for certain multiprocessing anomalies". In: *Bell System Technical Journal* 45 (1966), pp. 1563–1581.

[87] S.C. Graves. "A review of production scheduling". In: *Operations Research* 29 (1981), pp. 646–675.

[88] A.I. Hammouri et al. "ISA: a hybridization between iterated local search and simulated annealing for multiple-runway aircraft landing problem". In: *Neural Computing and Applications* 32 (2020), pp. 11745–11765.

[89] W. Hansen. "How Accessibility Shapes Land Use". In: *Journal of the American Institute of Planners* 25 (1959), pp. 73–76.

[90] S. Hartmann and D. Briskorn. "A survey of variants and extensions of the resource-constrained project scheduling problem". In: *European Journal of Operational Research* 207 (2010), pp. 1–14.

[91] O. Herr and A. Goel. "Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints". In: *European Journal of Operational Research* 248 (2016), pp. 123–135.

[92] T. Hintsch and S. Irnich. "Large multiple neighborhood search for the clustered vehicle-routing problem". In: *European Journal of Operational Research* 270 (2018), pp. 118–131.

[93] S. Ho et al. "A survey of dial-a-ride problems: Literature review and recent developments". In: *Transportation Research Part B* 111 (2018), pp. 395–421.

[94] C.J. Hsu et al. "Unrelated parallel-machine scheduling problems with aging effects and deteriorating maintenance activities". In: *Information Sciences* 253 (2013), pp. 163–169.

[95] K. Hu et al. "A new model for single machine scheduling with uncertain processing time". In: *Journal of Intelligent Manufacturing* 28 (2015), pp. 717–725.

[96] J.Z. Huo, L. Ning, and L. Sun. "Group Scheduling with General Autonomous and Induced Learning Effect". In: *Mathematical Problems in Engineering* 2018 (2018), pp. 1–5.

[97] A.K. Kaban, Z. Othman, and D. Rohmah. "Comparison of dispatching rules in job-shop scheduling problem using simulation: a case study". In: *International Journal of Simulation Modelling* 11 (2012), pp. 129–140.

[98] B. Kallehauge et al. *Vehicle Routing Problem with Time Windows*. Ed. by MA Springer Boston. Column Generation, 2005.

[99] J.J. Kanet. "Minimizing the average deviation of job completion times about a common due date." In: *Naval Research Logistics Quarterly* 28 (1981), pp. 643–651.

[100] V. Kaplanoglu. "Multi-agant based approach for single machine scheduling with sequence-dependent setup times and machine maintenance". In: *Applied Mathematical Modeling* 13 (2014), pp. 165–179.

[101] J.E. Kelley. "Critical-path planning and scheduling: Mathematical basis". In: *Operations Research* 9 (1961), pp. 296–320.

[102] G. A. P. Kindervater and M. W. P. Savelsbergh. "Local Search in Combinatorial Optimization". In: Princeton University Press, 1997. Chap. Vehicle routing: handling edge exchange, pp. 337–360. DOI: https://doi.org/10.1515/9780691187563-013.

[103] P.N. Kolm, R. Tutuncu, and F.J. Fabozzi. "60 Years of portfolio optimization: Practical challenges and current trends". In: *European Journal of Operational Research* 234 (2014), pp. 356–371.

[104] C. Koulamas. "The single-machine total tardiness scheduling problem: Review and extensions". In: *European Journal of Operational Research* 202 (2010), pp. 1–7.

[105] S.N. Kumar and R. Panneerselvam. "A survey on the vehicle routing problem and its variants". In: *Intelligent Information Management* 4 (2012), pp. 1–66.

[106] W.H. Kuo and D.L. Yang. "Single machine scheduling with past-sequence-dependent setup times and learning effects". In: *Information Processing Letters* 102 (2007), pp. 22–26.

[107] H. Lee. "Stochastic Single-Machine Scheduling With Learning Effect". In: *IEEE Transactions on Engineering Management* 64 (2016), pp. 94–102.

[108] S. Lee et al. "Flexible job-shop scheduling problems with AND/OR precedence constraints". In: *International Journal of Production Research* 50 (2012), pp. 1979–2001.

[109] W.C. Lee, C.C. Wu, and P.H. Hsu. "A single-machine learning effect scheduling problem with release times". In: *Omega* 38 (2010), pp. 3–11.

[110] W.C. Lee, C.C. Wu, and H.J. Sung. "A bi-criterion single-machine scheduling problem with learning considerations". In: *Acta Informatica* 40 (2004), pp. 303–315.

[111] K. Leksakul and A. Techanitisawad. "An application of the neural network energy function to machine sequencing". In: *Computational Management Science* 2 (2005), pp. 309–338.

[112] J.K. Lenstra and A.H.G. Rinooy Kan. "Complexity of vehicle routing and scheduling problems". In: *NETWORKS* 11 (1981), pp. 221–227.

[113] G. Leonardi. "Asymptotic approximations of the assignment model with stochastic heterogeneity in the matching utilities". In: *Environment and Planning A* 17 (1985), pp. 1303–1314.

[114] G. Leonardi. "The structure of random utility models in the light of the asymptotic theory of extremes". In: *Transportation planning models*. Ed. by M. Florian. Elsevier, 1984, pp. 107–133.

[115] H. Li. "Stochastic Single Machine Scheduling With Learning Effect". In: *IEEE Transactions on Engineering Management* 64 (2016), pp. 94–102.

[116] Y. Li et al. "Machine learning and optimization for production rescheduling in Industry 4.0". In: *The International Journal of Advanced Manufacturing Technology* 110 (2020), pp. 2445–2463.

[117] Y. Li et al. "Recent Advances in Computational Optimization, Results of the Workshop on Computational Optimization WCO (in press)". In: ed. by Springer Studies in Computational Intelligence / SN SL. Studies in Computational Intelligence, 2020. Chap. Online single-machine scheduling via reinforcement learning.

[118] L. Liu, J.J. Wang, and X.Y. Wang. "Single machine due-window assignment scheduling with resource-dependent processing times to minimize total resource consumption cost". In: *International Journal of Production Research* 54.4 (2016), pp. 1186–1195.

[119] M. Liu et al. "An optimal online algorithm for single machine scheduling to minimize total general completion time". In: *Journal of combinatorial optimization* 23.2 (2012), pp. 189–195.

[120] H.R. Lourenci, O.C. Martin, and T. Stutzle. "Iterated Local Search: Framework and Applications". In: *International Series in Operations Research & Management Science* 146 (2010), pp. 363–397.

[121] C.C. Lu, S.W. Lin, and K.C. Ying. "Robust scheduling on a single machine to minimize total flow time". In: *Computers and Operations Research* 39 (2012), pp. 1682–1691.

[122] C.C. Lu, K.C. Ying, and S.W. Lin. "Robust single machine scheduling for minimizing total flow time in the presence of uncertain processing times". In: *Computers and Industrial Engineering* 74 (2014), pp. 102–110.

[123] X. Lu, R.A. Sitters, and L. Stougie. "A class of on-line scheduling algorithms to minimize total completion time". In: *Operations Research Letters* 31.3 (2003), pp. 232–236.

[124] R.D. Luce. *Individual choice behavior: a theoretical analysis*. John Wiley, New York, 1959.

[125] F. Maggioni, F. A-Maggioni, and M. Bertocchi. "A scenario-based framework for supply planning under uncertainty: stochastic programming versus robust optimization approaches". In: *Computational Management Science* 14 (2017), pp. 5–44.

[126] D. Manerba and R. Mansini. "The Nurse Routing Problem with Workload Constraints and Incompatible Services". In: *IFAC-PapersOnLine* 49.12 (2016), pp. 1192–1197.

[127] D. Manerba, R. Mansini, and G. Perboli. "The Capacitated Supplier Selection problem with Total Quantity Discount policy and Activation Costs under uncertainty". In: *International Journal of Production Economics* 198 (2018), pp. 119–132. ISSN: 0925-5273.

[128] D. Manerba, R. Mansini, and J. Riera-Ledesma. "The Traveling Purchaser Problem and its Variants". In: *European Journal of Operational Research* 259.1 (2017), pp. 1–18.

[129] C. Manski. "The structure of random utility models". In: *Theory and Decision* 8 (1977), pp. 229–254.

[130] A. Marley. "Random utility models and their applications: recent developments". In: *Mathematical Social Sciences* 43 (2002), pp. 289 –302.

[131] O.C. Martin and S.W. Otto. "Combining simulated annealing with local search heuristics". In: *Annals of Operations Research* 63 (1996), pp. 57–75.

[132] K.P. Martinez et al. "An exact optimization approach for an integrated process configuration, lot-sizing, and scheduling problem". In: *Computers and Operations Research* 103 (2019), pp. 310–323.

[133] I. Martinez-Salazar et al. "A customer-centric routing problem with multiple trips of a single vehicle". In: *Journal of the Operational Research Society* 66 (2014), pp. 1312–1323.

[134] M. Matusiak, R. De-Koster, and J. Saarinen. "A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse". In: *European Journal of Operational Research* 236 (2014), pp. 968–977.

[135] A. Mingozzi, L. Bianco, and A. Ricciardelli. "Dynamic programming strategies for the TSP with time windows and precedence constraints". In: *Journal of Operations Research,* 45 (1997), pp. 365–377.

[136] P.L. Miranda et al. "A decomposition heuristic for a rich production routing problem". In: *Computers and Operations Research* 98 (2018), pp. 211–230.

[137] V. Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[138] R.H. Mohring, M. Skutella, and F. Stork. "Scheduling with AND/OR Precedence Constraints". In: *SIAM Journal on Computing* 33 (2004), pp. 393–415.

[139] Y. Molenbruch et al. "Typology and literature review for dial-a-ride problems". In: *Journal of Expert Systems with Applications* 259 (2017), pp. 295–325.

[140] C. Moon et al. "An efficient genetic algorithm for the traveling salesman problem with precedence constraints". In: *European Journal of Opertional Research* 140 (2002), pp. 606–617.

[141] J.Y. Moon, K. Shin, and J. Park. "Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency". In: *International Journal of Advanced Manufacturing Technology* 68.1-4 (2013), pp. 523–535.

[142] A. Mosca, N. Vidyarthi, and A. Satir. "Integrated transportation-inventory models: A review". In: *Operations Research Perspectives* 6 (2019), pp. 1–12. DOI: https://doi.org/10.1016/j.orp.2019.100101.

[143] G. Mosheiov. "Scheduling problems with a learning effect". In: *European Journal of Operational Research* 132 (2001), pp. 687–693.

[144] S.E. Moussavi, M. Mahdjoub, and O. Grunder. "A matheuristic approach to the integration of worker assignment and vehicle routing problems: Application to home healthcare scheduling". In: *Expert Systems with Applications* 125 (2019), pp. 317–332.

[145] J.M. Mulvey and R.J. Vanderbei. "Robust optimization of large-scale system". In: *Operations Research* 43 (1995), pp. 264–281.

[146] S. Mustu and T. Eren. "The single machine scheduling problem with sequence-dependent setup times and a learning effect on processing times". In: *Applied Soft Computing* 71 (2018), pp. 291–306.

[147] S. Nickel, F. Saldanha da Gama, and H.P. Ziegler. "A multi-stage stochastic supply network design problem with financial decisions and risk management". In: *Omega* 40 (2012), pp. 511–524.

[148] H. Niu and X. Zhou. "Optimizing urban rail timetable under time-dependent demand and oversaturated conditions". In: *Transportation Research Part C* 36 (2013), pp. 212–230.

[149] E. Ozceylan and T. Paksoy. "Interactive fuzzy programming approaches to the strategic and tactical planning of a closed-loop supply chain under uncertainty". In: *International Journal of Production Research* 52.8 (2014), pp. 2363–2387.

[150] C. Ozguven, L. Ozbakir, and Y. Yavuz. "Mathematical models for job-shop scheduling problems with routing and process plan flexibility". In: *Applied Mathematical Modelling* 34 (2010), pp. 1539–1548.

[151] S.S. Panwalkar and W. Iskander. "A survey of scheduling rules". In: *Operations Research* 25.1 (1977), pp. 45–61. DOI: 10.1287/opre.25.1.45.

[152] S.H.R. Pasandideh, S.T. Akhavan-Niaki, and K. Asadi. "Bi-objective optimization of a multi-product multi-period three-echelon supply chain problem under uncertain environments: NSGA-II and NRGA". In: *Information Sciences* 292 (2015), pp. 57–74.

[153] G.S. Peace. *Taguchi Methods: A Hands-on Approach*. Addison Wesley Publishing Company, 1993.

[154] G. Perboli, R. Tadei, and M.M. Baldi. "The stochastic generalized bin packing problem". In: *Discrete Applied Mathematics* 160 (2012), pp. 1291–1297.

[155] G. Perboli, R. Tadei, and L. Gobbato. "The Multi-Handler Knapsack Problem under Uncertainty". In: *European Journal of Operational Research* 236.3 (2014), pp. 1000–1007.

[156] J. Pereira. "The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective". In: *Computers and Operations Research* 66 (2016), pp. 141–152.

[157] C.P. Petrica, F. Levente, and M. Andrei Horvat. "A Variable Neighborhood Search Approach for Solving the Generalized Vehicle Routing Problem". In: *International Conference on Hybrid Artificial Intelligence Systems* (2014), pp. 13–24.

[158] F. Pezzella, G. Morganti, and G. Ciaschetti. "A genetic algorithm for the flexible job-shop scheduling problem". In: *Computer and Operational Research* 35 (2008), pp. 3202–3212.

[159] V. Pillac et al. "A review of dynamic vehicle routing problems". In: *European Journal of Operational Research* 225.1 (2013), pp. 1–11. ISSN: 0377-2217.

[160] M.L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, New York, NY, USA, 2012.

[161] D. Prot and O. Bellenguez-Morineau. "How the structure of precedence constraints may change the complexity class of scheduling problems". In: *Journal of Scheduling* 21 (2018), pp. 3–16.

[162] S. R-A-Haddadene et al. "A GRASP × ILS for the vehicle routing problem with time windows, synchronization and precedence constraints". In: *Journal of Expert Systems with Applications* 66 (2016), pp. 274–294.

[163] K. Rajalakshmi, P. Kumar, and H.M. Bindu. "Hybridizing iterative local search algorithm for assigning cells to switch in cellular mobile network". In: *International Journal of Soft Computing* 5 (2010), pp. 7–12.

[164] "Recent Advances in Computational Optimization, Results of the Workshop on Computational Optimization WCO 2020 (in press)". In: Springer Studies in Computational Intelligence, 2020. Chap. Online single-machine scheduling via reinforcement learning.

[165] J. Renaud, F.F. Boctor, and J. Ouenniche. "A heuristic for the pickup and delivery traveling salesman problem". In: *Journal of Computers and Operations Research* 27 (2000), pp. 905–916.

[166] M. Riedmiller. "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method". In: *European Conference on Machine Learning*. Springer. 2005, pp. 317–328.

[167] A. Riise, C. Mannino, and L. Lamorgese. "Recursive logic-based Benders' decomposition for multi-mode outpatient scheduling". In: *European Journal of Operational Research* 255 (2016), pp. 719–728.

[168] J.C. Rivera, H.M. Afsar, and C. Prins. "Mathematical formulations and exact algorithm for the multitrip cumulative capacitated single-vehicle routing problem". In: *European Journal of Operational Research* 249 (2016), pp. 93–104.

[169] M. Roderick, J. MacGlashan, and S. Tellex. "Implementing the deep q-network". In: *arXiv preprint arXiv:1711.07478* (2017).

[170] D.P. Ronconi and W.B. Powell. "Minimizing total tardiness in a stochastic single machine scheduling problem using approximate dynamic programming". In: *Journal of Scheduling* 13 (2010), pp. 597–607.

[171] M. Roohnavazfar and S.H.R. Pasandideh. "Decomposition Algorithm for the Multi-Trip Single Vehicle Routing Problem with AND-type Precedence Constraints". In: *Operational Research* (2021).

[172] M. Roohnavazfar, S.H.R. Pasandideh, and R. Tadei. "A Hybrid Algorithm for the Vehicle Routing Problem with AND/OR Precedence Constraints and Time Windows". In: *arXiv:2106.01652v1* (2021).

[173] M. Roohnavazfar et al. "Optimal paths in multi-stage stochastic decision networks". In: *Operations Research Perspectives* 6 (2019), pp. 1–10.

[174] M. Roohnavazfar et al. "Stochastic single machine scheduling problem as a multi-stage dynamic random decision process". In: *Computational Management Science* (2021).

[175] S. Ropke and D. Pisinger. "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows". In: *Journal of Transportation Science* 40 (2006), pp. 455–472.

[176] V. Roshanaei et al. "Propagating logic-based Benders' decomposition approaches for distributed operating room scheduling". In: *European Journal of Opeartional Research* 257 (2017), pp. 439–455.

[177] J. R.Rice. "The Algorithm Selection Problem". In: *Advances in Computers* 15 (1976), pp. 65–118.

[178] T. John Sajan, R. Sridharan, and P.N. Ram Kumar. "Multi-period reverse logistics network design with emission cost". In: *The International Journal of Logistics Management* 28 (Feb. 2017), pp. 127–149.

[179] R. Salman. *Algorithms for the Precedence Constrained Generalized Travelling Salesperson Problem*. Master's thesis: Chalmers University of Technology. University of Gothenburg, 2016.

[180] S.C. Sarin, H.D. Sherali, and A. Bhootra. "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints". In: *Operations Research Letters* 33 (2005), pp. 62–70.

[181] M.W. Savelsbergh and M. Sol. "The general pickup and delivery problem". In: *Journal of Transportation Research* 29 (1995), pp. 17–29.

[182] J. Schaller and J. Valente. "Branch-and-bound algorithms for minimizing total earliness and tardiness in a two-machine permutation flow shop with unforced idle allowed". In: *Computers and Operations Research* 109 (2019), pp. 1–11.

[183] M. Schneider, F. Schwahn, and D. Vigo. "Designing granular solution methods for routing problems with time windows". In: *European Journal of Operational Research* 263 (2018), pp. 493–509.

[184] D.K. Seo, C.M. Seo, and W. Jang. "Single machine stochastic scheduling to minimize the expected number of tardy jobs using mathematical programming models". In: *Computers and Industrial Engineering* 48 (2005), pp. 153–161.

[185] H. Sharma and S. Jain. "Online Learning Algorithms for Dynamic Scheduling Problems". In: *Second International Conference on Emerging Applications of Information Technology*. 2011, pp. 31–34.

[186] V.K. Shetty, M. Sudit, and R. Nagi. "Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles". In: *Computers and Operations Research* 35 (2008), pp. 1813–1828.

[187] M. Silva, M. Poss, and N. Maculan. "Solution Algorithms for Minimizing the Total Tardiness with Budgeted Processing Time Uncertainty". In: *European Journal of Operational Research* 283 (2020), pp. 70–82.

[188] S. Singh et al. "Convergence results for single-step on-policy reinforcement-learning algorithms". In: *Machine learning* 38.3 (2000), pp. 287–308. DOI: 10.1023/A:1007678930559.

[189] R. Soares et al. "Multiple vehicle synchronisation in a full truck-load pickup and delivery problem: A case-study in the biomass supply chain," in: *European Journal of Operational Research* 277 (2019), pp. 174–194.

[190] H. Soleimani, M. Seyyed-Esfahani, and M. Akbarpour-Shirazi. "Designing and planning a multi-echelon multi-period multi-product closed-loop supply chain utilizing genetic algorithm". In: *International Journal of Advanced Manufacturing Technology* 68 (2013), pp. 917–931.

[191] M.M. Solomon. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Operations Research* 35 (1987), pp. 254–265.

[192] M.M. Solomon and J. Desorios. "Time Window Constrained Routing and Scheduling Problems". In: *Transportation Science* 22 (1987), pp. 1–13.

[193] H.M. Soroush. "Minimizing the weighted number of early and tardy jobs in a stochastic single machine scheduling problem". In: *European Journal of Operational Research* 181 (2007), pp. 266–287.

[194] H.M. Soroush. "Stochastic bicriteria single machine scheduling with sequence-dependent job attributes and job-dependent learning effects". In: *European Journal of Industrial Engineering* 8 (2014), pp. 421–456.

[195] F. Sourd. "Earlinee-tardiness scheduling with setup considerations". In: *Computers and Operations Research* 32 (2005), pp. 1849–1865.

[196] G. Stecco, J. Cordeau, and E. Moretti. "A branch and cut algorithm for the production scheduling problem with sequence-dependent and time-dependent setup time". In: *Computers and Operations Research* 35 (2008), pp. 2635–2655.

[197] L. Sun. "Single-machine scheduling problems with deteriorating jobs and learning effects". In: *Computers and Industrial Engineering* 57 (2009), pp. 843–846.

[198] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[199] H. Suwa and H. Sandoh. *Online scheduling in manufacturing: A cumulative delay approach*. Springer Science & Business Media, 2012.

[200] R. Tadei, G. Perboli, and M. M. Baldi. "The capacitated transshipment location problem with stochastic handling costs at the facilities". In: *International Transactions in Operational Research* 19 (2012), pp. 789–807.

[201] R. Tadei, G. Perboli, and D. Manerba. "A recent approach to derive the Multinomial Logit model for choice probability". In: *Special Volume of the AIRO Springer Series. ODS2018, Sept 10-13, 2018. Taormina, Italy* (2018).

[202] R. Tadei, G. Perboli, and D. Manerba. "The multi-stage dynamic stochastic decision process with unknown distribution of the random choice utilities." In: *Optimization letters* (2019), pp. 1–12. DOI: https://doi.org/10.1007/s11590-019-01412-1.

[203] R. Tadei, G. Perboli, and F. Perfetti. "The multi-path Traveling Salesman Problem with stochastic travel costs". In: *EURO J Transp Logist* 6 (2017), pp. 3–23. DOI: https://doi.org/10.1007/s13676-014-0056-2.

[204] R. Tadei, N. Ricciardi, and G. Perboli. "The stochastic p-median problem with unknown cost probability distribution". In: *Operations Research Letters* 37 (2009), pp. 135–141.

[205] K. Takadama and H. Fujita. "Toward guidelines for modeling learning agents in multiagent-based simulation: Implications from q-learning and sarsa agents". In: *International Workshop on Multi-Agent Systems and Agent-Based Simulation*. Springer. 2004, pp. 159–172. DOI: 10.1007/978-3-540-32243-6_13.

[206] M.D. Toksari and E. Guner. "Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration". In: *Computers and Operations Research* 36 (2009), pp. 2394–2417.

[207] L. Tong et al. "Customized bus service design for jointly optimizing passenger-to-vehicle assignment and vehicle routing". In: *Transportation Research Part C: Emerging Technologies* 85 (2017), pp. 451–475.

[208] A. Toriello, W.B. Haskell, and M. Poremba. "A Dynamic Traveling Salesman Problem with Stochastic Arc Costs". In: *Operations Research* 62.5 (2014), pp. 1107–1125.

[209] P. Toth and D. Vigo. *The Vehicle Routing Problem: Society for Industrial and Applied Mathematics*. 2002.

[210] P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods and applications*. MOS-Siam Series on Optimization, 2nd edn. SIAM, Philadelphia, 2014.

[211] T.T. Tran and J.C. Beck. "Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups". In: *In Proceedings of the 20th European Conference on Artificial Intelligence* (2012), pp. 774–779.

[212] M. Trietsch and K. R-Baker. "Minimizing the number of tardy jobs with stochastically-ordered processing times". In: *Journal of Scheduling* 11 (2008), pp. 59–69.

[213] M. Tzur and E. Drezner. "A lookahead partitioning heuristic for a new assignment and scheduling problem in a distribution system". In: *European Journal of Operational Research* 215 (201‘), pp. 325–336.

[214] O. Unsal and C. Oguz. "An exact algorithm for integrated planning of operations in dry bulk terminals". In: *Transportation Research Part E: Logistics and Transportation Review* 126 (2019), pp. 103–121.

[215] J.M. Van-Den-Akker and J.A. Hoogeveen. "Minimizing the number of late jobs in a stochastic setting using a chance constraint". In: *Journal of Scheduling* 11 (2008), pp. 59–69.

[216] J.M. Van-Den-Akker, J.A. Hoogeveen, and J.W. Van-Kempen. "Parallel machine scheduling through column generation: minimax objective functions, release dates, deadlines, and/or generalized precedence constraints". In: *Lecture Notes in Computer Science* 4168 (2005), pp. 648–659.

[217] J.B. Wang and L.Y. Sun. "Single-machine group scheduling with linearly decreasing time-dependent setup times and job processing times". In: *International Journal of Advanced Manufacturing Technology* 49 (2010), pp. 765–772.

[218] C.J. Watkins. *Learning from delayed rewards*. Thesis Submitted for Ph.D., King's College, Cambridge, 1989.

[219] C.M. Wei, J.B. Wang, and P. Ji. "Single-machine scheduling with time-and-resource-dependent processing times". In: *Applied Mathematical Modelling* 36 (2012), pp. 792–798.

[220] M. Wen et al. "The dynamic multi-period vehicle routing problem". In: *Computers & Operations Research* 37 (2010), pp. 1615–1623.

[221] S. Whiteson et al. "Protecting against evaluation overfitting in empirical reinforcement learning". In: *2011 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*. IEEE. 2011, pp. 120–127.

[222] S. Xie, T. Zhang, and O. Rose. "Online Single Machine Scheduling Based on Simulation and Reinforcement Learning". In: *Simulation in Produktion und Logistik*. Simulation in Produktion und Logistik. 2019.

[223] J. Yang and G. Yu. "The robust single machine scheduling problem". In: *Journal of Combinatorial Optimization* 6 (2002), pp. 17–33.

[224] Q. Yang, Q. Chen, and Z.Z. Li. "Chaotic Particle Swarm Optimization Algorithm for Vehicle Routing Problem with Time Windows". In: *Computer Technology and Development* 8 (2015), pp. 119–122.

[225] S.J. Yang and D.L. Yang. "Single-machine scheduling simultaneous with position-based and sum-of-processing-times-based learning considerations under group technology assumption". In: *Applied Mathematical Modeling* 35 (2011), pp. 2068–2074.

[226] D. Ye et al. "Multi-type ant system algorithm for the time dependent vehicle routing problem with time windows". In: *Systems engineering and electronics* 29 (2018), pp. 625–638.

[227] L.E. Yelle. "The learning curve: Historical review and comprehensive survey". In: *Decision Sciences* 10 (1979), pp. 302–328.

[228] B.P.C. Yen and G. Wan. "Single machine bicriteria scheduling:a survey". In: *International Journal of Industrial Engineering: Theory Applications and Practice* 10 (2003), pp. 222–231.

[229] Y. Yin, D. Xu, and J. Wang. "Single-machine scheduling with a general sum-of-actual-processing-times-based and job-position-based learning effect". In: *Applied Mathematical Modelling* 34 (2010), pp. 3623–3630.

[230] K.C. Ying. "Minimizing earliness–tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm". In: *Computers and Industrial Engineering* 55 (2008), pp. 494–502.

[231] K.C. Ying and M. Bin Mokhtar. "Heuristic model for dynamic single machine group scheduling in laser cutting job shop to minimize the makespan". In: *Manufacturing Science and Technology* 383 (2012), pp. 6236–6241.

[232] Y. Yun and C. Moon. "Genetic algorithm approach for precedence constrained sequencing problems". In: *Journal of Intelligent Manufacturing* 22 (2011), 379—388.

[233] L.J. Zeballos et al. "Multi-period design and planning of closed-loop supply chains with uncertain supply and demand". In: *Computers and Chemical Engineering* 66 (2014), pp. 151–164.

[234] A. Zhang, X. Qi, and G. Li. "Machine scheduling with soft precedence constraints". In: *European Journal of Operational Research* 282 (2020), pp. 491–505.

[235] T. Zhang, S. Xie, and O. Rose. "Real-time job shop scheduling based on simulation and Markov decision processes". In: *2017 Winter Simulation Conference (WSC)*. IEEE. 2017, pp. 3899–3907. DOI: 10.1109/WSC.2017.8248100.

[236] X. Zhang, L. Sun, and J. Wang. "Single machine scheduling with autonomous learning and induced learning". In: *Computers and Industrial Engineering* 66 (2013), pp. 918–924.

[237] Y. Zhang, X. Wu, and X. Zhou. "Stochastic scheduling problems with general position-based learning effects and stochastic breakdowns". In: *Journal of Scheduling* 16 (2013), pp. 331–336.

[238] C.L. Zhao, W.L. Zhang, and H.Y. Tang. "Machine scheduling problems with a learning effect". In: *Dynamics of Continuous, Discrete and Impulsive Systems, Series A: Mathematical Analysis* 11 (2004), pp. 741–750.

[239] I. Zulj et al. "Picker routing and storage-assignment strategies for precedence-constrained order picking," in: *Journal of Computers and Industrial Engineering* 123 (2018), pp. 338–347.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.